Searching Musical Audio Datasets by a Batch of Multi-Variant Tracks

Yi Yu
Department of Information and Computer Sciences, Nara Women's University, Japan
yuyi@ics.nara-wu.ac.jp

J. Stephen Downie
Graduate School of Library and Information Science
UIUC, USA
jdownie@uiuc.edu

Lei Chen
Department of Computer Science,Hong Kong University of Science and Technology
leichen@cs.ust.hk

Vincent Oria
Department of Computer Science,New Jersey Institute of Technology, USA
oria@njit.edu

Kazuki Joe
Department of Information and Computer Sciences, Nara Women's University, Japan
joe@ics.nara-wu.ac.jp

ABSTRACT

Multi-variant music tracks are those audio tracks of a particular song which are sung and recorded by different people (i.e., cover songs). As music social clubs grow on the Internet, more and more people like to upload music recordings onto such music social sites to share their own home-produced albums and participate in Internet singing contests. Therefore it is very important to explore a computer-assisted evaluation tool to detect these audio-based multi-variant tracks. In this paper we investigate such a task: the original track of a song is embedded in datasets, with a batch of multi-variant audio tracks of this song as input, our retrieval system returns an ordered list by similarity and indicates the position of relevant audio track. To help process multi-variant audio tracks, we suggest a semantic indexing framework and propose the Federated Features (FF) scheme to generate the semantic summarization of audio feature sequences. The conjunction of federated features with three typical similarity searching schemes, K-Nearest Neighbor (KNN), Locality Sensitive Hashing (LSH), and Exact Euclidian LSH (E^2LSH), is evaluated. From these findings, a computer-assisted evaluation tool for searching multi-variant audio tracks was developed to search over large musical audio datasets.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval; H.5.5 [**Information Systems**]: Sound and Music Computing; J.5 [**Arts Humanities**]: Music

General Terms

Algorithms, Performance, Experimentation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MIR'08, October 30–31, 2008, Vancouver, British Columbia, Canada.
Copyright 2008 ACM 978-1-60558-312-9/08/10 ...$5.00.

Keywords

Content-based audio retrieval, cover songs, musical audio sequences summarization, hash-based indexing

1. INTRODUCTION

The World Wide Web (WWW) has become much more lively musically speaking since we can upload our own audio or video records onto the Internet and share them with the world on such popular web sites as http://www.midomi.com, http://www.yyfc.com and http://www.last.fm. Often, these music social clubs provide an online forum for Internet singing contests to attract people to vote for their favorite songs. Besides people's subjective comments, the computer-assisted locating of a song version is also of great importance to us. When a particular song is sung and recorded by different people, "cover songs" or "multi-variant audio tracks" for the song are produced. As a branch of query-by-content audio search and retrieval, locating and evaluating multi-variant audio tracks is an interesting task [1]. Some interesting examples of multi-variant audio tracks can be found at http://www.e.ics.nara-wu.ac.jp/∼yuyi/AudioExamples.htm.

Most of recent work [2, 3, 4] related to multi-variant audio track detection focus on musical audio sequences comparison to achieve a perfect matching. However, it takes much time to match two audio sequences since the feature sets (Chroma [3], Pitch [5], MFCC [6]) used for audio data have very high dimensionality. In this paper we review recent research works on the popular query-by-content audio retrieval techniques in Figure 1. It consists of two main parts: 1) audio feature selection; and, 2) audio sequence matching approaches. As shown in Figure 1 Dynamic Programming (DP) [3, 5] is a very important matching method, that performs exhaustive audio sequence comparisons which obtain good exactness. However, DP lacks scalability and results in slower retrieval speeds as databases get larger.

We do not adopt DP in our scheme. Instead a novel weighting scheme–Federated Features (FF)–is proposed to generate the semantic summarization of audio sequence. We also use musicminer [7] to generate more potentially useful audio features as candidates for the FF. According to the importance of features used in audio sequence comparisons, we are interested in the combination between frequently-used

Audio Features	Similarity Matching
Fourier transform [8]	Locality Sensitive Hash [8]
MFCC [6]	Locality Sensitive Hash [6]
Timbre, Rhythm and Pitch [9]	K-Nearest Neighbor [9]
MFCC [10]	K-Nearest Neighbor [10]
STFT [4]	Locality Sensitive Hash [4]
Chroma [3]	Dynamic Programming [3]
Pitch [5]	Dynamic Programming [5]

Figure 1: Existing audio retrieval techniques.

and musicminer-generated audio features. Through a large audio corpus we evaluate the proposed audio feature training scheme and show the weighted audio feature summarization can effectively represent melody-based lower-level music information.

The conjunction of the proposed Federated Feature (FF) with three typical searching schemes, K-Nearest Neighbor (KNN), Locality Sensitive Hash (LSH), and Exact Euclidian LSH (E^2LSH) [11], is evaluated. We also extend and summarize our work as an evaluation tool to help search musical datasets with a batch of multi-variant audio tracks.

2. BACKGROUND

Audio feature descriptors of musical audio sequences have very high dimensionality, which makes it increasingly difficult to quickly detect audio documents that closely resemble a given input query as the number of audio tracks in the database increase. Solving this hard problem is mainly related to two points: 1) refining music representation to improve the accuracy of musical semantic similarity (pitch [5], Mel-Frequency Cepstral Coefficient (MFCC) [10], Chroma [2, 3]); and, 2) organizing music documents in the way that helps speed up music similarity searching (LSH [4, 6, 8], E^2LSH [4, 16], tree structure [9, 15]).

2.1 Related Work

Chroma and DP are applied in [2, 3] to perform cover song detection. They guarantee the retrieval accuracy, however, at the cost of long sequence comparison time. In [4] LSH and E^2LSH are used to accelerate sequence comparisons in query-by-content music retrieval. Yang used random subsets of spectral features derived from a Short Time Fourier Transform (STFT) to calculate hash values for the parallel LSH hash instances in [8]. With a query as input, the relevant features are matched from hash tables. To resolve bucket conflicts, a Hough transformation is performed on these matching pairs to detect the similarity between the query and each reference song by the linearity filtering.

In [9] a composite feature tree (semantic features, such as timbre, rhythm, pitch, e.g.) was proposed to facilitate KNN search. Principle Component Analysis (PCA) was used to transform the extracted feature sequence into a new space sorted by the importance of acoustic features. In [15] the extracted features (Discrete Fourier Transform) are grouped by Minimum Bounding Rectangles (MBR) and compared with an R*-tree. Though the number of features can be reduced, sometimes the summarized (grouped) features may not sufficiently discriminate two different signals. In [16] application of LSH in large-scale music retrieval is evaluated. Shingles are created by concatenating consecutive frames and used as high-dimensional features. E^2LSH is then adopted to compare a query with references.

2.2 LSH and E²LSH

LSH [11] is an index-based data organization structure proposed to find all similar pairs of a query point in the Euclidean space. It has been used in such applications as the well-known solution to determine whether any pair of documents are similar or not (image [12], audio [6, 8], video [13], etc.). Features are extracted from documents and projected with a group of hash functions to hash values (index point). Feature vectors are regarded as similar to one another if they are projected to the same hash value.

If two features (V_q, V_i) are very similar they will have a small distance $\|V_q - V_i\|$, hash to the same value and fall into the same bucket with a high probability. If they are quite different they will collide with a small probability. A function family $H = \{\mathrm{h} : \mathrm{S} \rightarrow \mathrm{U}\}$, each h mapping one point from domain S to U, is called locality sensitive, if for any features V_q and V_i , the probability

$$Prob(t) = P_{rH}\left[h(V_q) = h(V_i) : \|V_q - V_i\| = t\right] \qquad (1)$$

is a strictly decreasing function of t. That is, the collision probability of features V_q and V_i is diminishing as their distance increases. The family H is further called (R, cR, p_1, p_2) $(c > 1,\ p_2 < p_1)$ sensitive if for any $V_q, V_i \in S$

$$\begin{aligned} &if\ ||V_q - V_i|| < R,\ \ P_{rH}[h(V_q) = h(V_i)] \geq p_1 \\ &if\ ||V_q - V_i|| > cR,\ P_{rH}[h(V_q) = h(V_i)] \leq p_2 \end{aligned} \qquad (2)$$

A good family of hash functions will try to amplify the gap between p_1 and p_2.

In E^2LSH the locality sensitive dimension reduction can be applied on a vector X whose each dimension follows the same P-stable distribution D. $f_{\bar{y}_k}(X)$, the inner product between X and a real vector $\bar{y}_k$, linearly combines all dimensions of X and generates a single output. With the matrix $Y = (\bar{y}_1, \bar{y}_2, ..., \bar{y}_m)$ an m-dimension vector $f_Y(X) = (f_{\bar{y}_1}(X), f_{\bar{y}_2}(X), ..., f_{\bar{y}_m}(X))^T$ can be obtained, each dimension of which also follows the distribution D. When each dimension of V_q and V_i follows a P-stable distribution, each dimension of $f_Y(V_q)$ and $f_Y(V_i)$ also follows the same distribution. Then V_q and V_i can be replaced by $f_Y(V_q)$ and $f_Y(V_i)$ respectively in Eq.(1-2).

2.3 This Work

This present work mainly aims to retrieve cover songs with a batch of multi-variant audio tracks. For this purpose, we introduce a semantic indexing framework for content-based music information and propose a novel semantic feature regression model based on fundamental similarity rules. Two groups of audio feature sets (frequently-used features in Figure 1 and musicminer-generated features [7]) are respectively trained via the proposed regression model. Our new Federated Features (FF) set results from the combination of the trained set of frequently-used features and the trained set of musicminer-generated features. A group of more meaningful weights are assigned to FF. In this way the musical audio sequence can be summarized as the especially interesting semantic audio representation. Searching schemes, KNN, LSH and E^2LSH are implemented to evaluate FF. Moreover, we develop an audio detection system to batch multi-variant

audio queries. The experimental results show that our FF is also very helpful for the large datasets.

3. APPROACHES

In this section we discuss the similarity searching problems associated with the musical audio content and give the definition of musical audio datasets searching by a batch of multi-variant tracks. To solve this retrieval problem, we propose a new semantic-based audio similarity principle and explain how to train a group of meaningful weights for our FF set and how to map the semantic audio representation to the indexable hash values.

3.1 Problem Description

Content-based music similarity searching strives to capture relevant music content/semantic-description (which may be a song [8, 9], or an established category related to genre [7], emotion [17], etc.) in the database with a query example, where the query example (from some defined description of music) is similar [8, 9] or categorized [7, 17] to the reference/provided content/attribute according to some similarity criteria. Content-based music similarity can be defined over a wide continuum since it is related to search intention, musical culture, personal opinion and emotion, etc. This work takes into account a similarity retrieval mechanism as follows: 1) taking a batch of multi-variant tracks as audio inputs (which are originally from the same popular song but recorded by different people); 2) performing LSH/E^{2}LSH and KNN searches; 3) returning the ordered audio tracks list; and, 4) evaluating the retrieval results by relevance. For each group/batch of multi-variant audio tracks, there is only one relevant audio track in the test datasets. As introduced in Section 1, this is an essential component for browsing, searching and evaluating the musical audio-based content information on the Internet (or personal digital media player).

To maintain notional consistency throughout the paper, the general description of key symbols in this work is given here. Given a collection of songs $R = \{r_{i,j} : r_{i,j} \in R_i,\ 1 \leq i \leq |R|, 1 \leq j \leq |R_i|\}$, ($r_{i,j}$ is the j^{th} spectral feature of the i^{th} song R_i), the feature sequence $r_{i,j}$ of the i^{th} song R_i is summarized to V_i, an n-dimension feature vector in the Euclidean space ($V_i \in \Re^n$). The summarized feature V_i instead of the feature sequence $r_{i,j}$ can then be utilized in the retrieval stage. To further accelerate the retrieval speed, hash-based indexing is also adopted. Each hash function $h_k(\cdot)$ maps V_i to a single value. Then the group of N independent hash functions $h_1(\cdot),\ h_2(\cdot), ..., h_N(\cdot)$ generate a vector of hash values $H(V_i) = [h_1(V_i), h_2(V_i), ..., h_N(V_i)]^T$. Inside a hash instance each hash vector is assigned to a bucket and two summarized features with the same hash vector fall into the same bucket. L parallel hash instances are constructed to support a high recall. Given a query song Q (with the summarized feature V_q) and a similarity function $\rho(\cdot,\cdot)$, we would like to compute the similarity degree $\rho(V_q, V_i)$ between the query Q and each of the songs R_i in the database. They are similar if $\rho(V_q, V_i)$ is above the predefined similarity threshold β. The similarity between two feature vectors can be computed by several similarity measures, Euclidean distance, cosine measure, etc. In this paper we feed our semantic musical audio features into Euclidean space, $d(V_q, V_i) = \|V_q - V_i\|_2 / (\|V_q\|_2 \cdot \|V_i\|_2)$.

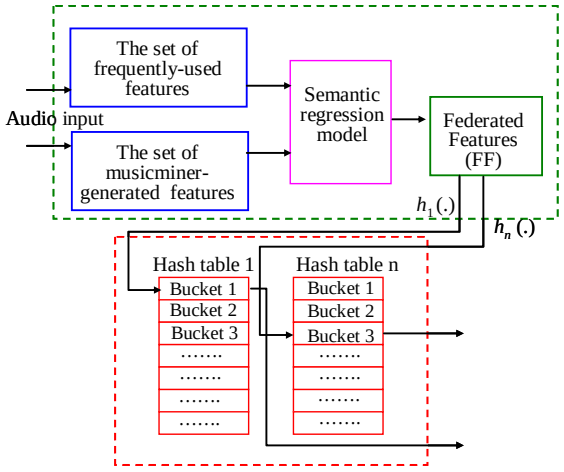

Figure 2: Federated features based indexing framework.

3.2 Semantic Indexing Framework

The indexing framework for semantic musical audio representation includes two major parts: 1) effectively generating semantic feature summarization; and, 2) accurately calculating hash values. Figure 2 demonstrates a semantic indexing framework based on federated features. Audio feature representation is a very important component for designing the content-based audio retrieval engine, especially when there are multi-variant audio queries. To achieve a better semantic audio representation, we select the set of frequently-used audio features introduced in Section 3.3 and the set of musicminer-generated audio features [7] as candidates to train a group of weights by our semantic regression model. For each audio track the summarized semantic FF is calculated and mapped to hash values. The FF features of audio tracks in the datasets are organized by LSH/E^{2}LSH in hash tables. The FF of an audio query can be calculated and mapped to one fixed hash value (or directly perform exhaustive KNN searching). If hash values of two FF collide into the same bucket, it would mean that they are very similar with a high probability.

3.3 Federated Features

Audio documents can be described as time-varying feature sequences. Directly computing the distance between two audio documents (matching audio documents) is one important task in implementing query-by-content music retrieval. DP can be used in matching two audio feature sequences and is an essentially exhaustive search approach (which offers high accuracy). However it lacks scalability and results in slower retrieval speeds as databases grow larger.

To quicken the audio feature sequences comparison and obtain the scalable retrieval, semantic features are extracted from the audio structure or based on the extracted semantic features the weight for each feature is determined by regression method [9]. Unfortunately, the semantic feature extraction (high-level) used in [9] is originally and mainly proposed to summarize the audio feature sequence for musical genre classification [18]. These semantic feature summarizations cannot effectively represent melody-based lower-level music information. We can find the most popular query-by-content

Feature	Short time dimension	function	Long time dimension
MFCC	13	Mean(), Std(), Skew(), Kurt(), Mean(|Δ|), Std(|Δ|), Skew(|Δ|), Kurb(|Δ|)	13*8
Mel-Magnitudes	40		40*8
Chroma	12		12*8
Pitch	1	histogram	88
Total		25 groups	608

Figure 3: Dimension of features.

audio retrieval techniques in Figure 1 in terms of audio sequence representations. Different features represent different aspects of audio signals and were proposed for different purposes. For example, Pitch [5] is extracted from the sung melody to generate a note sequence while semitone-based Chroma [3] is used to tolerate differences in instrumentation and general musical style.

In [7] the large scale generation of long-term audio features is used to obtain concise and interpretable features summarizing a complete song and indicate the probability of this song belonging to a certain group. Frequently-used features in Figure 1 and musicminer-generated features [7] encourage us to combine them by using multivariable regression to train a group of features that are summarized as a simple descriptor to represent the characteristics of one audio sequence as comprehensively and effectively as possible. The dimensionality of final musical audio features is kept low through the summarization created by our proposed regression model. The goal is to avoid the heavy computation of feature sequence comparisons and make query-by-content music retrieval for large databases possible.

The feature sets used in training the regression model are given in Figure 3. They are used in audio content retrieval. For a description of the feature extraction methods the interested reader is referred to [3, 5, 7, 10, 19, 20].

3.3.1 Similarity-Invariance of Summarization

Two questions occur in the summarizing stage: 1) how to summarize the high dimensional features? and, 2) how to guarantee that a summarized feature descriptor reflects the melody information? As for the first question, there are several summarizing methods such as calculating the mean and standard deviation of all the features, PCA, etc. As for the second question the summarizing procedure should exhibit substantive characteristics of similarity-invariance, i.e., similar melodies lead to similar summaries, non-similar melodies lead to non-similar summaries. To solve the above issues a basic melody-based summarization principle is considered as follows. $r_{i,j}$ is the j^{th} spectral feature of the i^{th} reference song. The sequence similarity between i^{th} song and k^{th} song is $\varphi_{ik}(\{r_{ij}\},\{r_{kj}\})$. The i^{th} audio feature sequence $\{r_{i,j}\}$ is summarized to V_i. The similarity between the i^{th} and k^{th} feature summary is $\psi_{ik}(V_i,V_k)$. A similarity threshold θ or θ' would state how close between any two feature sequences or summarized features. With a good summarization we could expect that

$$\varphi_{ik}(\{r_{ij}\},\{r_{kj}\}) > \theta \Leftrightarrow \psi_{ik}(V_i,V_k) > \theta' \quad (3)$$

In this sense the summarization is similarity-invariant.

3.3.2 Regression model

A single feature can not well summarize a song. Multiple features must be combined to represent a song. These features play different roles in the query and must be weighed by different weights. We introduce a scheme based on multivariable regression to determine the weight for each feature. The goal of our approach is to apply linear and non-parametric regression models to investigate the correlation.

In the model we use K (K=25 according to Figure 3) groups of features. Let the groups of features of i^{th} song be $v_{i1}, v_{i2}, ..., v_{iK}$. With different weight α_k assigned to each feature group, the total summary vector is

$$V_i = [\alpha_1 v_{i1}, \alpha_2 v_{i2}, ..., \alpha_K v_{iK}]^T \quad (4)$$

The Euclidean distance between two features V_i and V_j is

$$d(V_i,V_j) = d(\sum_{k=1}^{K}\alpha_k v_{ik}, \sum_{k=1}^{K}\alpha_k v_{jk}) = \sum_{k=1}^{K}\alpha_k^2 d(v_{ik},v_{jk}) \quad (5)$$

To determine the weight of the above equation, we apply multivariable regression process. For training purpose, we select from the database M pairs of songs, which contain both similar pairs and non-similar pairs. By these pairs we obtain the sequences of Chroma similar to [3], and then calculate M pairs of sequence similarity $D(R_i,R_j)$.

We will choose the weight in Eq.(4) so that the similarity calculated by the summary is as near to the sequence similarity as possible, i.e., we hope the melody information is contained in the summary. After we determine the similarity between the pairs of training data, we get a $M*K$ matrix D_V and the M-dimensional column vector D_{DP}. The i^{th} row of D_V has K distance values calculated from independent features $d(v_{ik},v_{jk}), k=1,2,...,K$ and the i^{th} element of D_{DP} is the normalized distance between the two feature sequences $D(R_i,R_j)\cdot K/(|R_i|\cdot|R_j|)$. Let $A=[\alpha_1^2,\alpha_2^2,\cdots,\alpha_K^2]^T$. Then $A=(D_V^T D_V)^{-1}D_V^T D_{DP}$ and we obtain the weight α. We are only interested in the absolute value of α. The feature set defined in Eq.(4) is called the Federated Features (FF) set.

3.4 KNN and LSH/E^{2}LSH Searching

After training a group of weights for federated features, we obtain the combined independent high dimensional feature set for each audio sequence. It can be used together with KNN and LSH/E^{2}LSH.

Here we first introduce the indexing structure of datasets with LSH/E^{2}LSH. According to Figure 2, the federated features of the songs in the datasets are stored in the hash tables in terms of their hash values. When LSH is used, the hash values are directly calculated from the FF and each hash instance has its own hash functions. On the other hand, when E^{2}LSH is used, the FF is first projected to a lower dimension vector in each hash instance. Then hash values are calculated. In Figure 2 the total effect of the hash function is represented as $H_k(.)$ for the k^{th} hash instance.

Consider a query V_q and its relevant song V_i in the dataset. With KNN, V_q is compared against each of the song in the dataset and the one with least distance is regarded as the relevant song. When LSH/E^{2}LSH is used, instead of performing an exhaustive search, from the query V_q, its hash value $H_k(V_q)$ is calculated for the k^{th} hash instance. The features located in the bucket of $H_k(V_q)$ are then found and V_q is only compared against these features. With locality

sensitive mapping, V_q and its relevant song V_i have the same hash values $H_k(V_q) = H_k(V_i)$ in at least one of the hash instances with a high probability. Therefore V_i will probably be located in the buckets indexed by $H_k(V_q)$ and be found.

4. EXPERIMENTS

In this section we introduce our experimental setup, show some evaluation results over a large audio datasets, and demonstrate a novel content-based music information detection system with multi-variant audio (i.e., cover song) queries.

4.1 Experimental Setup

Our music collection includes 4121 songs that fall into four non-overlapping datasets. Trains80 is collected from our personal collections and www.yyfc.com (a non-commercial amusement website where users can sing her/his favorite songs, make records online, and share them with friends or the yyfc community). It consists of 40 popular songs each represented in two versions and 80 single-version songs. These 160 tracks are used to train the weights of regression model proposed in section 3.3.2. Covers79 is also collected from www.yyfc.com and consists of 79 popular Chinese songs each represented in different versions (sung by different people with possibly similar background music). Each song on average has 13.5 versions, resulting in a total of 1072 audio tracks. The corpora RADIO (1431) and ISMIR (1458) are used as noise background datasets and respectively collected from the public websites www.shoutcast.com and http://ismir2004.ismir.net/genre_contest/index.htm.

Each song is 30s long in mono-channel wave format, 16bit per sample and the sampling rate is 22.05KHz. The audio data is normalized and then divided into overlapped frames. Each frame contains 1024 samples and the adjacent frames have 50% overlap. Each frame is weighed by a hamming window and further appended with 1024 zeros to fit the length of FFT (2048 point). From FFT result the instantaneous frequencies are extracted and Chroma is calculated. From the amplitude spectrum pitch, MFCC and Mel-magnitude are calculated. Then the summary is calculated from all frames.

The ground truth is set up according to human perception. We have listened to all the songs and manually labeled them so that retrieval results of our algorithms correspond to human perception to support practical application. Trains80 and Covers79 datasets were divided into groups according to their verse (the main theme represented by the song lyrics) to judge whether songs belong to the same group or not (one group represents one song and different versions of one song are members in this group). The 30s segments in these two datasets are extracted from verse sections of tracks.

The experiments were run on a PC with an Intel Core2 CPU (2GHz) and 1GB DRAM under Microsoft Windows XP Professional OS.

4.2 Performance Evaluation

Our task is to detect the musical audio content with multi-variant tracks. In this section we provide some experimental evaluation results. The whole evaluation was run over 2968 (79+1431+1458) audio tracks, where the original track of each song in Cover79 were put in the dataset. The rest (1072-79=993) tracks in Cover79 were used as queries. Mean reciprocal rank (MRR) of the ground truth were calculated as the evaluation metric. The top 20 retrieved songs were analyzed by default.

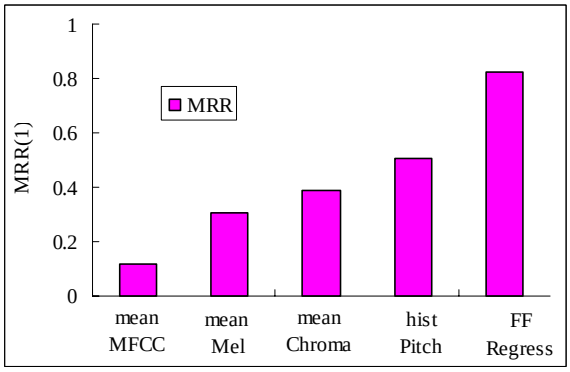

Figure 4: MRR achieved by different features.

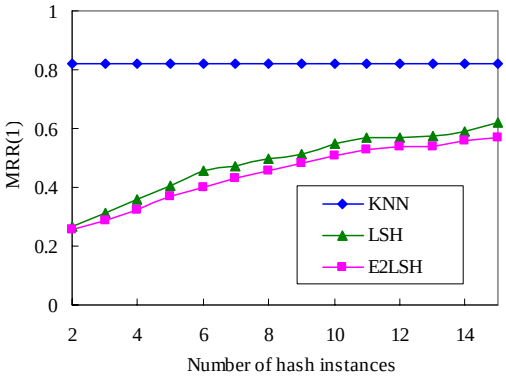

Figure 5: Mean reciprocal rank at different number of hash instances.

First we trained the weights used in the FF sets by using the Trains80 dataset as the ground truth. We selected 40 pairs of similar songs (each pair includes two versions of the same song) and 40 pairs of non-similar songs (one pair includes two different songs). We selected some long term features as the final semantic audio representation according to their importance. Figure 4 compares MRR(1) achieved by different features utilizing exhaustive KNN search. With each feature, 993 queries were performed against 2968 tracks. FF has the best performance in comparison with other separate competitive feature sets. Moreover it also reveals that FF set can represent human perception more effectively. However, MRR(1) of FF is only 0.823. It can not reach 1 even by the exhaustive search due to the following factor: some of the perceptually similar tracks have quite different features and result in quite large distances in feature space.

Figure 5 shows MRR(1) of the three schemes with respect to different numbers of hash instances. KNN always performs best and E^2LSH always has the worst performance. When there are only two hash instances, LSH performs as poorly as E^2LSH. However LSH outperforms E^2LSH when the number of hash instances is greater than 3. MRR(1) increases as the number of hash instances does. However this increase is not linear. Their performances start approaching steadiness when the number of hash instances increases to 10. Further increase in hash instances results in diminishing

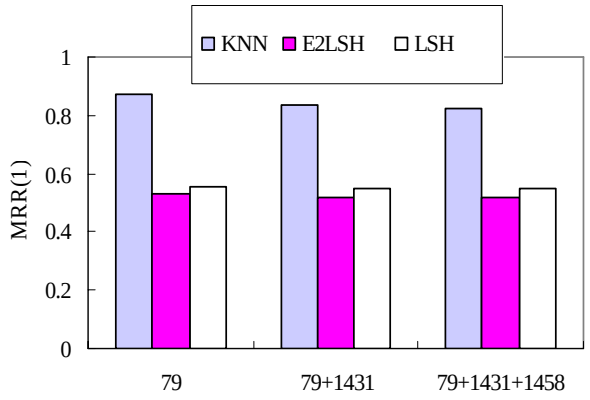

Figure 6: MRR at different database sizes.

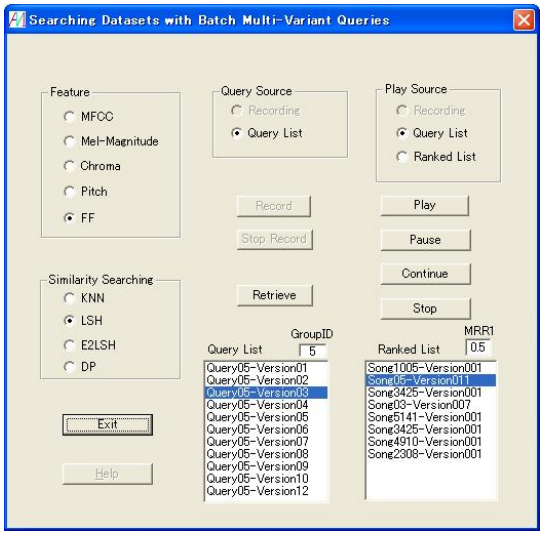

Figure 7: Searching with multi-variant audio content.

returns. This is because feature summarization results in information loss which can not be salvaged by the increase of hash instances. Therefore the number of hash table is set to 10 in the following experiment.

Figure 6 shows the effect of different database sizes. The three databases respectively contain Cover79 (79), Cover79 & RADIO (79+1431), Cover79 & RADIO & ISMIR (79+1431 +1458). It is obvious that the increase of database size has very little effect on the MRR(1), which confirms that the FF set can effectively distinguish the (newly added) non-similar tracks.

4.3 Searching Datasets with Multi-Variant Audio Tracks

Figure 7 illustrates our content-based music information retrieval demo system developed in C++. From the left side, the features and similarity searching methods can be selected. In this demo system, the query set consists of 79 groups with 1072-79=993 audio tracks (the original track of each song is in the datasets). Any group ID can be freely specified to show the query audio tracks. With any audio track selected as query, the system can give a ranked list of the retrieval result. The first eight relevant audio tracks are reported automatically with the best similarity audio track as the first. Both queries and retrieval results can be played to confirm the correctness of searching.

This system not only retrieves relevant audio tracks from the datasets, but also evaluates the performance of our approaches. In the current setting in Figure 7 Feature "FF", Similarity Searching "LSH" and Query Group ID "5" are set up respectively. In "Ranked List", the retrieval results are ordered and the first eight relevant audio tracks are given in the list. The corresponding evaluation result is also given at the right side. We can easily see that with the third variant of the song in Group 5 as query input after searching over the datasets its relevant audio track appears in the second position (MRR(1) is 0.5).

5. CONCLUSION

With more and more personal music recordings available via the WWW, there is an increasing demand in developing tools for detecting multi-variant audio music recordings. The goal of such retrieval tools is to rank a collection of music recordings according to their similarity. To help process multi-variant audio tracks, musical audio feature extraction plays an important role in audio content retrieval and searching. A good audio feature not only needs to effectively represent musical sequence characteristics but also is easily mapped to an indexable format. In this paper we introduced an index-based semantic framework for speeding up the content-based audio retrieval. We proposed a semantic regression model to train a meaningful semantic audio summarization called Federated Features (FF). Three similarity searching schemes were adopted to test our proposed approach to musical audio representation. The experimental results demonstrate that our weighting scheme is useful for audio content detection over the large databases. We also developed a retrieval system to not only retrieve audio content with a batch of multi-variant audio tracks, but also evaluate the performance of searching schemes.

6. ACKNOWLEDGMENTS

We thank Initiative Project of Nara Women's Unveirsity for offering an opportunity to study abroad. This work was partly discussed and done when Yi visited International Music Information Retrieval System Evaluation Laboratory (IM IRSEL) in summer, 2007. The second author is a founder of IMIRSEL at UIUC, and was supported by the Andrew W. Mellon and national Science Foundation (NSF) under Nos. IIS-0340597 IIS-0327371. The fourth author was partially supported by a grant from DoD-ARL through the KIMCOE center of Excellence.

7. REFERENCES

[1] J. S. Downie. The Music Information Retrieval Evaluation eXchange (MIREX). In D-Lib Magazine 12, 2006. http://dlib.org/dlib/december06/downie/12downie.html.

[2] J. P. Bello. Audio-based Cover Song Retrieval Using Approximate Chord Sequences: Testing Shifts, Gaps, Swaps and Beats. ISMIR'07, pp.239-244, 2007.

[3] D. Ellis and G. Poliner. Identifying cover songs with chroma features and dynamic programming beat tracking. ICASSP'07, 2007.

[4] Y. Yu, K. Joe, and J. S. Downie. Efficient Query-by-Content Audio Retrieval by Locality Sensitive Hashing and Partial Sequence Comparison. IEICE Transaction on Information and System, Vol.E91-D, No.6, pp1730-1739, 2008.

[5] Y. Yu, J. S. Downie, and K. Joe. An Evaluation of Feature Extraction for Query-by-Content Audio Information Retrieval. Ninth IEEE International Symposium on Multimedia Workshops (ISMW), pp. 297-302, 2007.

[6] Y. Yu, M. Takata, and K. Joe. Index-Based Similarity Searching with Partial Sequence Comparison for Query-by-Content Audio Retrieval. Workshop on Learning Semantics of Audio Signals (LSAS'06), pp.76-86, 2006.

[7] F.Moerchen, I. Mierswa, and A. Ultsch. Understandable Models of Music Collection based on Exhaustive Feature Generation with Temporal Statistics. KDD'06, pp.882-891, 2006.

[8] C.Yang. Efficient Acoustic Index for Music Retrieval with Various Degrees of Similarity. ACM Multimedia, pp.584-591, 2002.

[9] B. Cui, J.L. Shen, G. Cong, H.T. Shen, and C. Yu. Exploring Composite Acoustic Features for Efficient Music Similarity Query. ACM MM'06, pp.634-642, 2006.

[10] T. Pohle, M. Schedl, P. Knees, and G. Widmer. Automatically Adapting the Structure of Audio Similarity Spaces. Workshop on Learning Semantics of Audio Signals (LSAS'06), pp.66-75, 2006.

[11] LSH Algorithm and Implementation (E^2LSH) http://web.mit.edu/andoni/www/LSH/index.html.

[12] P. Indyk and N. Thaper. Fast color image retrieval via embeddings. Workshop on Statistical and Computational Theories of Vision (ICCV), 2003.

[13] S.Y. Hu. Efficient Video Retrieval by Locality Sensitive Hashing. ICASSP'05, pp.449-452, 2005.

[14] J. Reiss, J. J. Aucouturier, and M. Sandler. Efficient multi dimensional searching routines for music information retrieval. ISMIR'01, 2001.

[15] I. Karydis, A. Nanopoulos, A. N. Papadopoulos and Y. Manolopoulos. Audio Indexing for Efficient Music Information Retrieval. MMM'05, pp.22-29, 2005.

[16] M. Casey and M. Slaney. Song Intersection by Approximate Nearest Neighbor Search. ISMIR'06, pp.144-149, 2006.

[17] M. Lesaffre and M. Leman. Using Fuzzy to Handle Semantic Descriptions of Music in a Content-based Retrieval System. Workshop on Learning Semantics of Audio Signals (LSAS'06), pp.43-5, 2006.

[18] G. Tzanetakis and P. Cook. Musical Genre Classification of Audio Signals. IEEE Transactions on Speech and Audio Processing, Vol.10, No.5, pp. 293-302, 2002.

[19] R. Miotto and N. Orio. A Methodology for the Segmentation and Identification of Music Works. ISMIR'07, pp.239-244, 2007.

[20] L.Rabiner and B.-H. Juang. Fundamentals of Speech Recognition. Prentice-Hall, 1993.