Using Sequential Patterns as Short-Term and Long-Term Features for Audio Genre Classification

Jia-Min Ren **Zhi-Sheng Chen** **Jyh-Shing Roger Jang**
Department of Computer Science, National Tsing-Hua University, Taiwan
{jmzen0921, bobon, jang}@mirlab.org

ABSTRACT

This extended abstract describes a submission to the Music Information Retrieval Evaluation eXchange (MIREX) 2009 for the audio genre classification task. We improved the approach [1] by mining sequential patterns from audio transcriptions according to each genre. Moreover, we consider these patterns as either short-term features or long-term features for further classifiers design.

1. SYSTEM DESCRIPTION

The system consists of four stages, as diagrammed in Figure 1. The first stage is based on [2], which aims to tokenize a song into a string of hidden Markov model (HMM) indices (these models are called acoustic segment models, ASM). Then we mine sequential patterns for each genre in the next stage. The third stage attempts to generate two unit-song matrices which denote the occurring frequency of the specific combinations of ASM units for each song; the first unit-song matrix is the same as [2], and we will describe how to generate the second one in the following sections. In the final stage, these two matrices are normalized and utilized for the classifiers design. We submit four versions of algorithms to evaluate our performance.

1.1 Tokenizing Audio as Transcription

The tokenization stage which can be found in [2] tokenizes a song into a string of ASM indices. The idea is analogous to tokenizing an utterance into a string of words or phonemes in the field of automatic speech recognition (ASR). First, we perform a maximum-likelihood segmentation algorithm [3] to tokenize the training corpus into acoustic segments. Next, by quantizing the segment centroids, we build a vocabulary of ASMs, which consists of small acoustic tokens. Then, a string of symbols (e.g., the ASM indices that best represent segments of a song) can represent a temporal transcript for a song. Besides, we need to refine these acoustic segment models to capture more exact spectral information. Specifically, these initial transcripts will be considered as a reference to re-train the acoustic models via Baum-Welch estimation, and then Viterbi decoding utilizes the updated ASMs to create new transcriptions. This iterative process is repeated until convergence.

In this system, we model each ASM as a 3-state HMM (8-component GMM with a diagonal covariance matrix in each state). In the generation of initial transcriptions, we only use the first 8 MFCCs (frame size is 30ms and no overlap) to capture the slowly changing spectral shape of songs. In the procedure of re-estimating acoustic models and creating new transcriptions, we use 39 MFCCs (frame size is 30ms, and overlap is 10ms) instead.

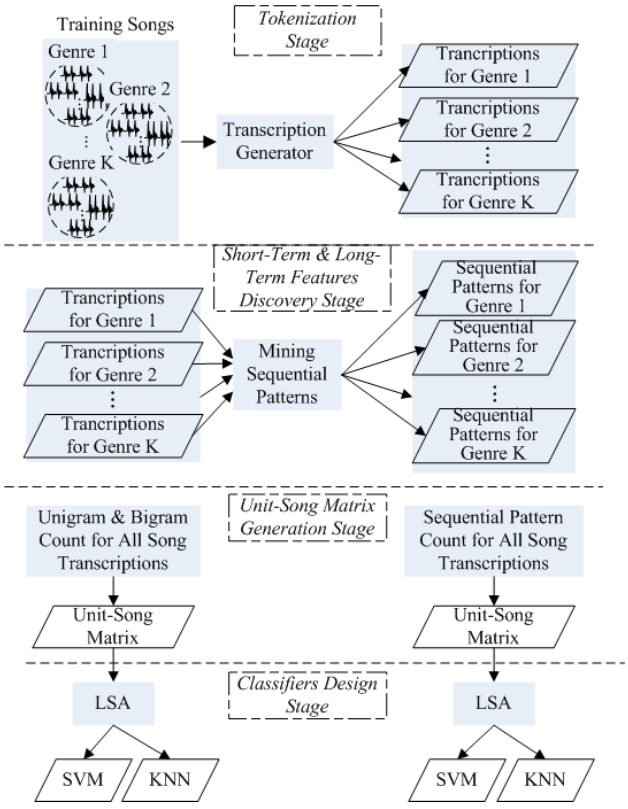

Figure 1. Diagram of our system.

1.2 Mining Sequential Patterns from Each Genre

Given a database of music transcriptions and a *minimum support* (*minSup*) threshold which is used to determine whether a mined pattern is frequent or not, we can apply *AprioriAll* [4] algorithm to find the sequential patterns. Table 1 shows an example of music transcriptions.

Song Id	Transcription
1	(12, 23, 45, 34, 40, 12)
2	(12, 34, 23, 34, 10)
3	(23, 45, 25, 23)
4	(45, 20, 23, 43, 34, 25)
5	(12, 23, 34, 25)

Table 1. An example of transcriptions within the same audio genre (number denotes the ASM index).

If *minSup* is set to 0.6 (which denotes a mined sequential pattern needs to exist in at least 3 times out of these 5 transcriptions), we can mine sequential patterns as (45), (23, 25) and (12, 23, 34). In our experiments, the *minSup* was set empirically to 0.8. Note that the items in sequential patterns need not to be consecutive but the order must be preserved. Moreover, a pattern is said to be a sequential pattern if it is not contained in any other pattern, i.e., pattern $(a_1 a_2 ... a_n)$ is said to be contained in pattern $(b_1 b_2 ... b_n)$ if there exists integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 = b_{i_1}$, $a_2 = b_{i_2}$, …, $a_n = b_{i_n}$. Thus, (12), (23), (25), (34), (12, 23), (12, 34) and (23, 34) are not sequential patterns, since they are contained in those three mined sequential patterns. Please refer to [4] for more details about the mining procedure.

After mining sequential patterns from each music genre, these patterns can be considered as temporal features to facilitate the classification task, since the long-term temporal structures of the music are explored where most music of the same genre have such kind of patterns.

1.3 Generating Unit-Song Matrix

A unigram count is the occurrence of an individual ASM and a bigram count is the occurrence of an ordered ASM pair [2]. For instance, if a transcription of a song is (2, 12, 2), then we will get a vector which has 2 in the location of ASM unit 2, and a 1 in the location for ASM units 12, (2, 12) and (12, 2), but 0 in the locations for the other ASM units. In our system, 128 ASMs are trained. So the resulting vector for a song has a dimension of 128+128*128=16512.

Once we mine sequential patterns for each genre, then we can count how many times this pattern appears in a transcription as follows. Given a sequential pattern $s = (s_1 s_2 ... s_n)$ with length *n* and a transcription of ASMs $t = (t_1 t_2 ... t_m)$ with length *m*, the count for *s* is defined as,

Unit_count = 0;
while (1) {
  If s is not contained in t, then break;
  Find the minimal index j such that s is contained in $(t_1 t_2 ... t_j)$;
  If no such j exists, then break;
  Remove $(t_1 t_2 ... t_j)$ *from t*;
  unit_count = unit_count + 1; }

After counting how many times these patterns appear in each transcription, we can get another unit-song matrix, where a unit refers to a sequential pattern.

1.4 Training Classifiers

Before training classifiers, we applied latent semantic analysis (LSA) to convert unit-song matrices into weighed unit-song matrices. This procedure is also applied in [2]. Next, we utilized support vector machine (SVM) with linear kernel [5] and K-nearest neighbor (k = 3) as classifiers in our system. For each unit-song matrix (where each vector is a feature of a song), we can train two classifiers. So totally four classifiers are in our submission, and we evaluate the performance of these classifiers separately.

Algorithm 1: Feature: uni & bigram unit-song matrix
Classifier: KNN (K=3)
Algorithm 2: Feature: uni & bigram unit-song matrix
Classifier: SVM (linear kernel function)
Algorithm 3: Feature: sequential pattern unit-song matrix
Classifier: KNN (K=3)
Algorithm 4: Feature: sequential pattern unit-song matrix
Classifier: SVM (linear kernel function)

2. RESULTS

Table 1 shows the result of our approaches (where Algorithm 2 in the mixed set ran out of memory). The best recognition rates of this task are about 73% and 74% in the mixed and Latin sets, respectively. Although the proposed methods perform worse than the most of other submissions, it is worth noting that when sequential pattern counts are used as features, they perform better than the use of uni & bigram counts as features. This implies that such kind of feature does really capture the long-term structure information to facilitate the classification task. In the future, we will try to use some timbral features, e.g. flux, centroid, skew and kurtosis, to do this task.

Set / Algo.	Mixed	Latin
Algorithm 1	32.50%	38.93%
Algorithm 2	*OOM*	52.43%
Algorithm 3	37.71%	46.78%
Algorithm 4	50.99%	55.22%

Table 2. Recognition results of the proposed methods.

3. REFERENCES

[1] J. Reed and C.-H. Lee: "A study on music genre classification based on universal acoustic models," *Proceedings of the International Symposium on Music Information Retrieval*, pp. 89-94, 2006.

[2] J. Reed and C.-H. Lee: "On the importance of modeling temporal information in music tag annotation," *Proceedings of International Conference on Acoustic, Speech and Signal Processing*, pp. 1873-1876, 2009.

[3] T. Svendsen and F. Soong: "On the automatic segmentation of speech signals," *Proceedings of*

International Conference on Acoustic, Speech and Signal Processing, pp. 77-80, 1987.

[4] R. Agrawal and R. Srikant: “Mining Sequential Patterns,” *Proceedings of 1995 International Conference on Data Engineering*, pp. 3–14, 1995.

[5] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.