AUTOMATIC LABELING OF TRAINING DATA FOR SINGING VOICE DETECTION IN MUSICAL AUDIO

Kyogu Lee
Media Technology Lab, Gracenote
2000 Powell Street, Emeryville, CA94608, USA
klee@gracenote.com

Markus Cremer
Media Technology Lab, Gracenote
2000 Powell Street, Emeryville, CA94608, USA
mcremer@gracenote.com

ABSTRACT

We present a novel approach to labeling a large amount of training data for vocal/non-vocal discrimination in musical audio with the minimum amount of human labor. To this end, we use MIDI files for which vocal lines are encoded on a separate channel and synthesize them to create audio files. We then align synthesized audio with real recordings using dynamic time warping (DTW) algorithm. Note onset/offset information encoded in vocal lines in MIDI files provides precise vocal/non-vocal boundaries and we obtain from the minimum-cost alignment path the corresponding boundaries in actual recordings. This near labor-free labeling process allows us to acquire a large training data set, and the experiments show promising results when tested on an independent test set, using hidden Markov models as a classifier. We also demonstrate that the data generated by the proposed system is good data by showing that the overall performance increases with more training data.

KEY WORDS

singing voice detection, supervised learning, automatic labeling, MIDI, dynamic time warping

1 Introduction

In most popular music, the singing voice section captures one of the most important characteristics, and thus detection of the singing voice has attracted a number of researchers in a music information retrieval (MIR) community for many years. Detection of singing voice or vocal/non-vocal discrimination can be used in many other related applications in MIR such as singer identification, singing voice separation, query-by-humming (QBH), structural analysis of music or lyrics-audio alignment, and so on. For instance, the vocal/non-vocal boundaries often coincide with the structural segmentation boundaries in most popular music and thus knowing the former helps to detect the latter.

Another closely-related problem is speech/music discrimination [13]. Although this appears to be a more general task, vocal/non-vocal discrimination in musical audio is more challenging because most vocal sections in popular music are accompanied by musical instruments as well, making vocal sections less distinguishable from non-vocal sections.

Many researchers view vocal/non-vocal discrimination as a classification problem and take machine-learning approaches to solve it [1, 15, 9, 12]. That is, given an unseen input, we classify it as vocal or non-vocal based on the properties *learned* from vocal/non-vocal data which is pre-labeled. Even though it is a binary classification problem with just two classes — *i.e.* vocal vs. non-vocal class — the acoustical variance within each class is so large that it is imperative to have sufficient amount of training data to avoid overfitting. For instance, a vocal section may include a male or female singer, or both. It may also contain a chorus or group singing. There is a great diversity in a non-vocal or instrumental section too, because different artists use a different set of instruments: some may use electric guitars or others may prefer acoustic ones; some may use drums or others may not. In addition to these variances in voice/instrumentation, we must also consider differences in musical attributes such as tempo, melody, harmony and/or dynamics.

Such diversities present in musical audio as is mentioned above lead to large variances in acoustic features, and it may cause a serious overfitting problem if the models are trained on insufficient data, failing to perform well on an unseen input. The importance of large amount of training data in supervised learning is proved by a great success in automatic speech recognition (ASR), owing to gigantic databases accumulated over decades. However, it is hard to find such data sets for music applications, especially for vocal/non-vocal discrimination because it requires a great deal of human labor to manually annotate vocal/non-vocal boundaries as he/she listens to a number of audio files.

In this paper, we propose a solution to this bottleneck problem of acquiring a large, labeled training set for vocal/non-vocal discrimination while keeping human-involved labor at the minimum level. The key idea in our approach is to use symbolic music files like MIDI files as a tool to find vocal/non-vocal boundaries in real recordings. If vocal lines are written on a separate channel, which is often the case, we can extract the precise vocal/non-vocal boundaries from note onset/offset information in a MIDI file. We then synthesize audio from the MIDI file and time-align it with a real recording using a dynamic time warping (DTW) algorithm. The minimum-cost alignment path found by the DTW algorithm gives us vocal/non-vocal boundaries in real audio, and then we can generate labeled

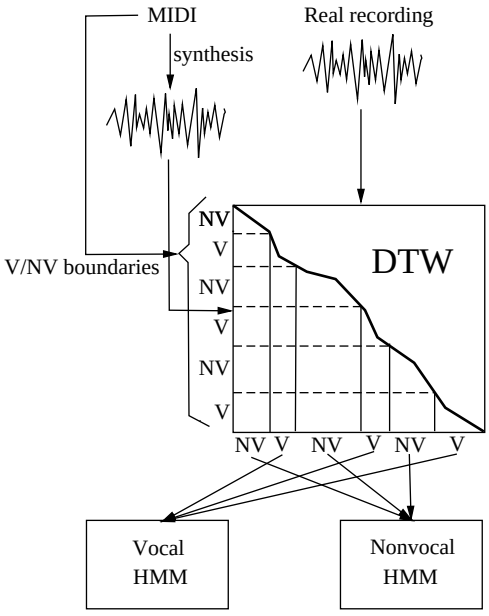


Figure 1. System overview. Vocal/non-vocal boundaries extracted from a MIDI file are used to find the boundaries in a real recording by time-aligning MIDI-synthesized audio with a real recording.

data to build the models. The overall process is illustrated in Figure 1.

This paper continues with a review on related work in Section 2. In Section 3, the method of labeling the training data and the details of HMM classifier are explained, along with the acoustic features. We present the experimental results with discussions in Section 4. Finally, we draw conclusions and suggest the directions for future work in Section 5.

2 Related Work

Scheirer and Slaney proposed a speech/music discrimination system in multi-dimensional classification frameworks [13]. In their work, they use 13 different acoustic features to capture distinct properties in speech and/or music signals and build four different classifiers. Of 40 minutes of hand-labeled audio, they used 90% (36 minutes) to train the models and 10% (4 minutes) for evaluation. Using the "best 3" features — 4 Hz energy, variance of spectral flux, and pulse metric — and a k-d spatial classifier, they obtained the best results of 5.8% error on a frame-by-frame basis, and 1.4% error using a longer (2.4 seconds) analysis window. Although their work tackles a somewhat different problem from vocal/non-vocal discrimination, it lays an important background in the choice of the feature set and by providing a complete evaluation framework.

Berenzweig and Ellis presented a system for locating singing voice segments within musical signals using the acoustic classifier of a speech recognizer as a detector [1]. They used posterior probability features (PPFs) generated from a multi-layer perceptron neural network trained on speech database. Segmentation and classification are accomplished simultaneously from the optimal state path from a simple, two-state HMM. Using 101 short (5.5 seconds on average) fragments of audio (61 for training and 40 for testing), their system obtained about 80% accuracy.

Kim and Whitman proposed a method for vocal region detection in their singer identification system [6]. Based on the observation that the singing voice is highly harmonic, they came up with a measure of harmonicity using an inverse comb filter. Using a harmonicity value of 2.0 as a threshold for vocal classification, they achieved performance of about 55%, using 20 songs as a test set. Zhang also described in his singer identification system a technique for automatically detecting a starting point of the singing voice based on a set of pre-defined thresholds [17]. He then used the first 25 seconds from the beginning point of the detected singing voice for singer identification. No quantitative evaluation is performed on the singing voice detection, however.

Tzanetakis addressed the two main problems in supervised learning approaches on the singing voice detection — *i.e.*, 1) the variability of singer and instrumentation characteristics and 2) the difficulty of obtaining ground truth annotations — and proposed a bootstrapping method to solve these problems [15]. In his song-specific bootstrapping, a small random collection of snippets are used to train a classifier that is subsequently used to segment/classify the entire song. He used eight feature sets and six classification algorithms for experiments. Using 10 Jazz songs as a test set, he obtained the best results of 75% accuracy, using 16 2-second long snippets (32 seconds total) for bootstrapping. Although he tried to avoid time-consuming human annotation using a bootstrapping technique, it still requires a significant amount of time to manually annotate 16 clips for *every* song.

Nwe *et al.* take one step further to utilize song structure information in a multi-model hidden Markov model (MM-HMM) classifier, using a bootstrapping technique for song-adaptive classification [9]. They observe that popular songs in general have a certain structure and each section has different characteristics. They also take into account tempo and loudness to explain inter-song variation. Based on three parameters — song structure, tempo and loudness — they build 40 models – 20 each for vocal and non-vocal class. They then use a bootstrapping method from the initial results from an MM-HMM classifier, and performed song-adaptive classification to further improve performance. Using six songs for training and 14 songs for testing, they obtained the best results of 86.7% with bootstrapped HMMs, using Harmonic-Attenuated Log Frequency Power Coefficients (HA-LFPCs). The high performance is somewhat shadowed by strong assumptions they made about the structure (intro-verse-chorus-bridge-outro) and meter (4/4) of a song.

Recently, Rocamora and Herrera compared several audio descriptors for singing voice detection by conducting a series of experiments [12]. They developed three independent data sets for training, validation and testing, all of which are manually annotated. In addition to various feature sets and different classifiers, they also experimented with the duration of an input fragment to examine the effect of long-term characteristics of feature frames. Using 46 songs in different styles for testing, they achieved the accuracy of 78.5%, using Support Vector Machines (SVMs) and MFCC feature.

As addressed above by others [6, 15], one of the biggest problems in machine-learning approaches for vocal/non-vocal discrimination is lack of training data to train the classifiers due to a great deal of human labor and time it takes to manually mark the vocal/non-vocal boundaries for a number of audio files. Some avoid this by just using a small collection of hand-labeled audio segments [13, 1], or others tackles this problem via bootstrapping [15, 9]. Or others just make a hard decision based on a pre-defined, heuristic threshold [6, 17].

A solution to this bottleneck problem is proposed in the following sections.

3 System

Our system is composed of two stages. In the first stage, we automatically generate labels for training data by aligning real recordings with MIDI-synthesized audio, which contains vocal/non-vocal boundary information. In the second stage, we extract from the training data acoustic features appropriate for vocal/non-vocal discrimination, and build HMMs. These steps are described in more detail in the following sections.

3.1 Automatic Labeling

Unlike a PCM waveform that contains digitized audio samples, symbolic music documents such as MIDI (Musical Instrument Digital Interface) contain a set of event messages like pitch, velocity and note duration, along with clock signals from which we can synthesize audio samples. In addition, many MIDI files use separate channels to encode each musical instrument, including vocal lines. Therefore, if we know which channel contains vocal melody lines in MIDI files, it is straightforward to segment MIDI-synthesized audio into vocal and non-vocal sections because we can extract precise note onset/offset information from MIDI files.

However, we cannot directly use MIDI-synthesized audio to train our models because it contains no real human voice and it is critical to have it for our purpose. Therefore, we must use real recordings with human singing voice that correspond to MIDI files we use to obtain vocal/non-vocal boundaries. However, it is impossible to simply apply vocal/non-vocal boundaries in MIDI-synthesized audio to real recordings because the two audio files are likely to be of different length or there may be changes in tempo in either or in both audio. Therefore, we need to dynamically align the two audio in order to find precise vocal/non-vocal boundaries in real recordings.

We explain the feature vector of our choice for the alignment purpose in the next section.

3.1.1 Chroma Feature

In order to align MIDI-synthesized audio with real recordings, we need to find acoustic features that, when extracted from the two audio signals, should be close to each other. MFCCs (Mel-Frequency Cepstral Coefficients), which are widely used in many speech and music applications, do not appear as a good choice because the sonic quality in the two different audio will be quite different, particularly due to other instrument replacing the vocal lines in MIDI-synthesized audio. On the other hand, pitch information would be perfect because it is mostly identical in both audio, but a drawback is that it is very difficult to estimate accurate pitches from real recordings.

Chroma feature appears ideal considering these difficulties. Chroma feature is widely accepted to describe tonality in musical audio, and is extensively used in key extraction and/or chord recognition systems. A chroma vector is often a 12-dimensional vector, each dimension representing spectral energy in a pitch class in a chromatic scale, and is obtained by collapsing the whole spectrum into an octave [4]. Therefore, if the pitch information in the two audio signals is the same, the resulting chroma vectors will not be significantly different, no matter what kind of instruments are playing the notes.

However, we must consider situations where there is mis-tuning in audio or the two audio are in different keys because it will affect the distance measure on which our alignment algorithm is based. To avoid this problem, we obtain the global chroma vectors by averaging the whole chromagrams and compute the correlation coefficients between the two global chroma vectors as we rotate a chroma vector. When we find the maximum correlation, we rotate a chromagram so that the two chromagrams are aligned around the same tonal center. Serra and Gomez successfully used this technique to find cover versions in a different key from that of the original song [14]. Figure 2 shows an example where a chromagram from MIDI-synthesized audio and one from a real recording have two different tonal centers, along with the global chroma vectors below.

If we carefully look at the first two chromagrams in Figure 2, we can see one is off by two semitones from the other. It becomes more obvious when we look at the global chroma vectors. It is clear that a real recording has a tonal center of C, while MIDI-synthesized audio is centered around D, yielding a negative correlation of -0.1. As mentioned above, this discrepancy in tonal centers significantly affect the alignment algorithm based on the distance between feature vectors, resulting in incorrect vocal/non-vocal boundaries. Therefore, we first rotate-shift the sec-

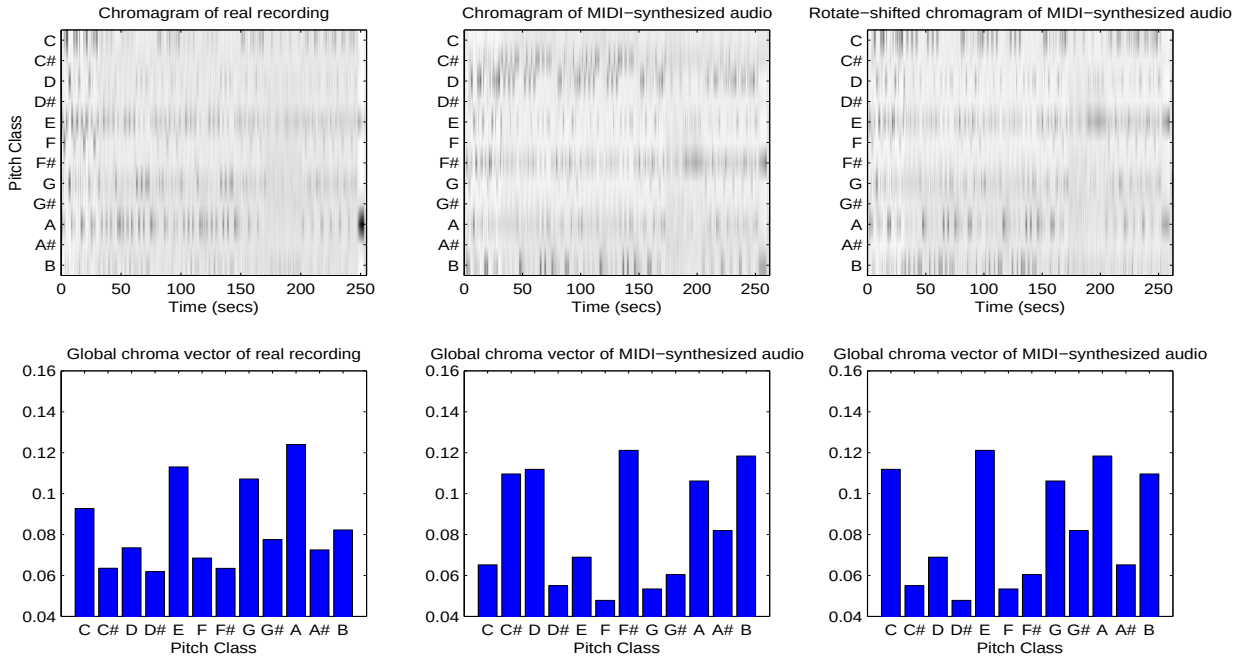


Figure 2. Chromagrams (top) and global chroma vectors (bottom) of a real recording (left) and MIDI-synthesized audio before (middle) and after shift-rotation (right). *Otherside* by Red Hot Chili Peppers.

ond chromagram by two semitones so that two chromagrams have the same tonal center, as shown in the rightmost plot in Figure 2. After rotation, the two global chroma vectors yields a high positive correlation of 0.9.

Once the two chromagrams are arranged around the same tonal center , we use them as an input to an alignment algorithm, which is described in the following section.

3.1.2 Audio Alignment

Although the note onset/offset time of the vocal lines in MIDI files gives us precise vocal/non-vocal boundaries, it does not guarantee that these boundaries will correspond to those in real recordings because the two audio files are usually different in length and there may be changes in tempo in one or both audio. Therefore, we need to find a way to *dynamically* align the two audio files, using the chromagrams as a front end.

We use the dynamic time warping (DTW) algorithm to this end. DTW is widely used to efficiently time-align the two sequences of different length using dynamic programming (DP) [7]. Given a pair of feature sequences, a distance matrix is first computed using the Euclidean distance, and then the DTW finds the minimum-cost alignment path based on pre-defined transition cost function. Therefore, as explained in Section 3.1.1, rotating the chromagrams so that they have the same referential point is critical to compute a meaningful distance matrix and for the DTW algorithm to correctly find the optimal path. Figure 3 shows the vocal/non-vocal boundaries we obtained using the method described above.

As shown in Figure 3, the automatically generated boundaries are very close to the human-annotated boundaries. Although there are a few sources of errors such as backing vocals which are not encoded on vocal lines, we believe that these errors are negligible compared with a even more amount of data which is correctly labeled.

In the next section, we describe a vocal/non-vocal classifier using hidden Markov models.

3.2 HMM Classifier

Hidden Markov models (HMMs) are widely used in many classification problems, particularly in speech/audio applications because of its capability to explain temporal dynamics present in signals. We do not intend to provide in this paper mathematical backgrounds of HMMs, but interested readers are encouraged to see Rabiner [11].

3.2.1 Acoustic Features

Chroma feature proves very successful for an alignment purpose as demonstrated in Section 3.1.1. However, it is not suitable for vocal/non-vocal discrimination because it focuses only on pitch relations or tonality instead of capturing timbral characteristics, which is critical for our task. For example, when in a non-vocal section a musical instrument plays the same melody line as vocal in a vocal sec-

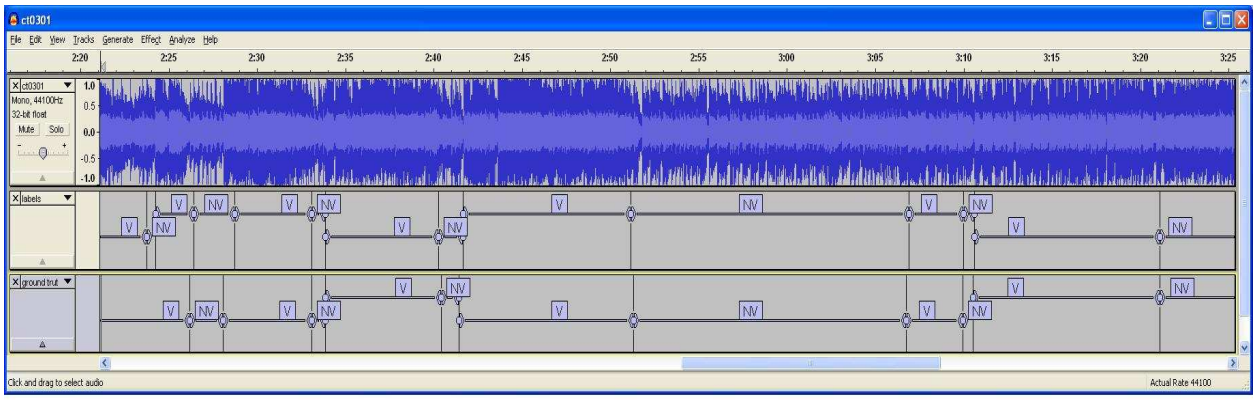


Figure 3. Automatically generated vocal/non-vocal boundaries for *Otherside* by Red Hot Chili Peppers. 'V' and 'NV' denote vocal and non-vocal segments, respectively. At the top is shown the PCM waveform. At the middle are shown the ground-truth boundaries. At the bottom are shown the boundaries automatically obtained using the alignment method described in the text.

tion, the resulting chroma features from the two sections will look very similar, and thus it will be difficult to tell one from the other.

Therefore, we use different feature vectors more appropriate for discrimination purposes. Based on the other related works on the same problem, we choose three different features — Mel-Frequency Cepstral Coefficients (MFCCs), Perceptual Linear Prediction Coefficients (PLPCs) and Log Frequency Power Coefficients (LFPCs). More details on computing these features can be found in Davis and Mermelstein [2], Hermansky [5] and in Nwe *et al.* [8], respectively. We use Matlab implementations by Daniel Ellis for computing MFCCs and PLPCs.[1]

The original audio signals in the training set are in a stereo format with a sampling rate of 44.1 kHz. Before extracting the spectral features from this raw audio, it is downmixed to mono signals, and is bandlimited from 130 Hz to 16kHz because most spectral energies are distributed in this frequency range. A mono, bandlimited audio signal is then processed every 10 ms using a Hamming window of 25 ms, resulting in 100 feature frames per second.

3.2.2 Models

Using the labeling process described in Section 3.1, we generate training data from 163 audio files that are more than 11 hours of audio or 3.5 million feature frames for each feature. This data set covers wide range of music in different styles — pop, rock, jazz, blues, country, etc. — of male and female singers. Of all the data, about 60% are labeled as vocal, and 40% are labeled as non-vocal. 10% of each class is reserved for model validation, and we use only 90% of the total data set to train the models.

For each feature set, we build a vocal HMM and a non-vocal HMM, resulting in 6 HMMs in total, using Gaussian distribution to model the observation probability distribution. Each model has $Q = 4$ hidden states and $M = 2$ mixtures per state. We use a HMM toolbox by Kevin Murphy for HMM implementation.[2]

In order to examine how the amount of data affects the models' performance, we build the models trained on different amount of data. That is, we build five models for each class using 20%, 40%, 60%, 80% and 100% of the training set. We believe that more data allows the models to be more generalized, resulting in better performance. In the next section, we present the experimental results.

4 Results

4.1 Evaluation

Prior to evaluating the models' performance on the singing voice detection problem, we first evaluated our automatic labeling algorithm to examine the integrity of the labels. We randomly selected 100 audio fragments from each class, and listened to them for verification. Four out of 100 vocal-labeled clips were actually non-vocal, resulting in 96% accuracy. The accuracy for non-vocal clips dropped to 93%, seven of 100 containing the singing voice parts. The slightly worse accuracy for non-vocal labeling is mostly due to backing vocals, some of which are not explicitly written in the MIDI files.

We performed several quantitative evaluations to test our models. As explained in Section 3.2.2, we used 10% of the data as a model validation set. This set includes about 67 minutes of audio or 400,000 feature samples, of which 58% are vocal and 42% are non-vocal. In addition to the validation set, we developed an independent test set for the following reasons. First, even though none in the validation set is used to train the models, it has the same origins as the training set and thus the statistics of the data will be similar. Therefore, for a more complete evaluation, we

[1] `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`

[2] `http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`

Table 1. Titles of 10 songs in the test set

Title	Artist	Genre
No Reply	The Beatles (M[a])	Rock
Here Comes My Baby	Cat Stevens (M)	Folk Pop
Anticipation	Carly Simon (F)	Pop
A Thousand Days	Clay Aiken (M)	Pop
Lean On Me	Bill Withers (M)	Deep Soul
Smile	Lily Allen (F)	Electronic Pop
When I'm Gone	3 Doors Down (M)	Alternative Rock
Wishes	Out Of The Grey (F)	Rock
Breathe	Faith Hill (F)	Country
Everyday I Have The Blues	B.B. King (M)	Blues

[a]M[F] denotes a Male[Female] singer.

need a statistically independent test set. Second, the data in the validation set is not annotated by humans, but is labeled automatically using the method described in Section 3.1, which may not be 100% true because of a few sources of possible errors in a labeling process, such as backing vocals not encoded on vocal lines. We do not believe that these small errors have significant effect on building statistical models, but we cannot ignore them to evaluate the models. Hence, we manually annotated a small set of songs for testing, which was carefully selected to include as wide range of songs as possible. The test set is composed of 10 songs, which contain approximately 37 minutes of audio or 218,000 feature frames. 65% of the test bed are vocal and 35% are non-vocal. The songs used for testing are listed in Table 4.1.

Although the feature vectors are processed at a frame rate of 100 frames/second, vocal or non-vocal segment seldom has such a short duration of a single frame (10 ms) but contains many consecutive frames. In addition, it is important to examine temporal evolution of feature vectors over a longer time period in order to make a more statistically reliable decision. Therefore, based on the findings of others' experiments, we choose one second as our analysis window length for evaluation, with 50% overlap [10, 12]. The score in % is computed by dividing the duration of correctly classified fragments by the total duration.

4.2 Results and Discussion

As described in Section 3.2.2, we compared the models trained on different amount of data to investigate the effect of the data size on performance. Figure 4 shows the accuracy in percentage for each feature using the data size as a variable, with two mixtures per state ($M = 2$).

We observe in Figure 4 that performance increases as the data size increases, although not monotonously for all three features, which supports our hypothesis that more data wins. No significant difference in performance is seen among the various features. Having observed that the models perform best with the full training set, we did another experiment using the number of Gaussian mixtures per state as a variable. The results are shown in Figure 5.

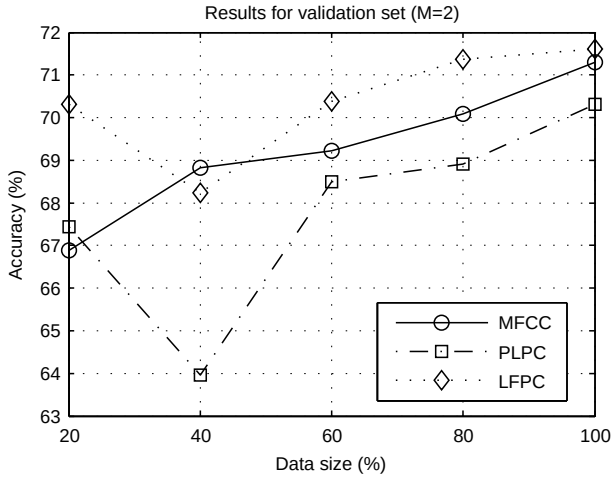


Figure 4. Performance of the models on the validation set as a function of data size (M=2).

Table 2. Performance comparison among MFCC, PLPC and LFPC features on the validation set (% accuracy; trained on the full training set, $M = 10$)

Feature	Vocal	Non-Vocal	Average
MFCC	72.86	69.14	71.30
PLPC	65.82	79.04	71.37
LFPC	74.42	74.56	74.48

As shown in Figure 5, as the number of mixtures per state increases, overall performance also increases, although not monotonously, except for the MFCC feature. This suggests that we have enough data to build richer models, yielding improved performance. We can also see the LFPC feature outperforms the other two features.

Table 2 summarizes performance of three features using the models trained on the full training set with $M = 10$. We can see from Table 2 that the MFCC feature is better in classifying vocal correctly while it is opposite with the PLPC feature. The LFPC feature works equally well for both classes.

As described in Section 4.1, we used an independent test set consisting of 10 songs in different styles for a more complete evaluation. The results on this test set are displayed in Figure 6 using the data size as a variable, with 10 mixtures per state ($M = 10$).

Figure 6 shows a similar pattern as is seen in Figure 4 in that for all three features, performance increases in general as the data size increases, which again supports the fact that more data improves the model performance. Also, no significant difference is found among the features. However, we notice that the overall performance has improved by about 7% on average compared with the results on the validation set. This is surprising considering that the validation set has the same origins as the training set while

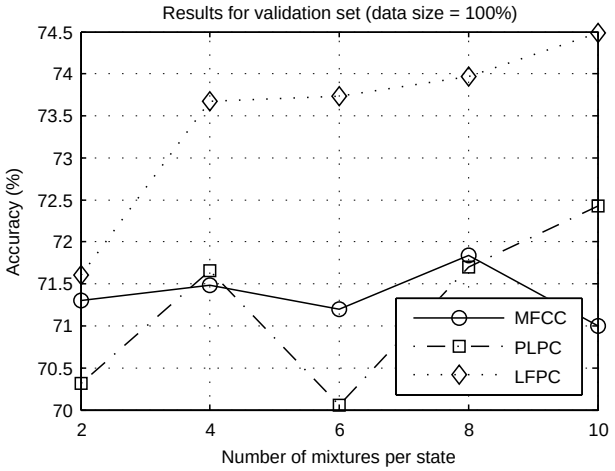


Figure 5. Performance of the models on the validation set as a function of the number of mixtures per state (data size = 100%).

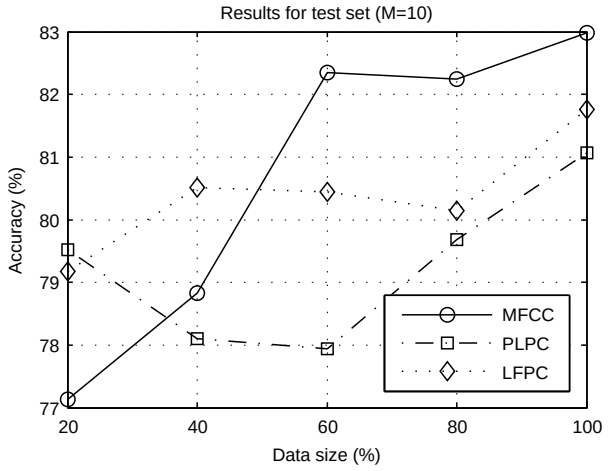


Figure 6. Performance of the models on the test set as a function of data size (M=10).

the test set does not. The possible explanation for this can be drawn from the fact that part of the validation set was mis-labeled due to errors in the automatic labeling process, as explained in Section 4.1. This causes false positive or false negative even for correct classification when computing the scores. Therefore, these errors come from the scoring process, not from classification. However, the test set was manually annotated by the authors and no errors come from mis-labeling.

To confirm this assumption, we randomly selected 20 segments for each class from the validation set, and examine if some of them are mis-labeled. Of 20 vocal segments, one segment was silence, which we labeled as non-vocal. Two of 20 non-vocal segments contained backing vocals, and one has a strong vocal line. The source of the first two errors was consistent with our expectation, and the third one turned out that the vocal line was encoded on a different MIDI channel, causing mis-alignment between MIDI-synthesized audio and a real recording.

Table 3 summarizes performance of three features using the models trained on the full training set with $M = 10$. In contrast with the results on the validation set shown in Table 2, MFCC feature works equally well on both vocal and non-vocal classes while LFPC feature performs better on non-vocal class. However, it is noticeable that PLPC feature consistently works far better on non-vocal class than on vocal class.

Figure 7 shows two examples of vocal/non-vocal segmentation results on (a) *No Reply* by The Beatles and (b) *Smile* by Lily Allen. Both results were obtained using the models trained on the full training set and the MFCC feature, with $M = 10$.

We can see from Figure 7 that the vocal/non-vocal boundaries detected by the algorithm are very close to the true boundaries. In fact, often the times it is ambiguous

Table 3. Performance comparison among MFCC, PLPC and LFPC features on the test set (% accuracy; trained on the full training set, $M = 10$)

Feature	Vocal	Non-Vocal	Average
MFCC	82.25	83.73	82.96
PLPC	72.52	89.62	78.50
LFPC	79.02	84.49	80.93

even for humans to mark the clear-cut boundaries between vocal and non-vocal segments. Considering this somewhat ill-defined nature of a ground truth, the results look very promising.

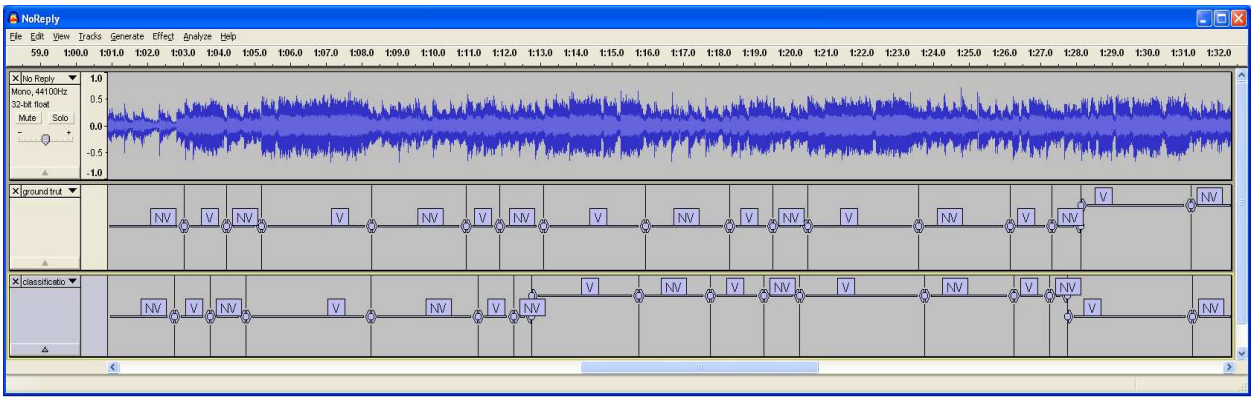
5 Conclusions

In this paper, we presented a novel approach to acquiring a large set of labeled training data in a near labor-free way for vocal/non-vocal classification in musical audio. The main contribution of this paper is to demonstrate that the data generated by the proposed system is good data and the statistical classifier is capable of identifying large quantity of good data. We demonstrated this by showing that the overall performance improves as the training data set gets larger.

In order to find the vocal/non-vocal boundaries in real recordings, we used the MIDI files in which vocal lines are written on a separate channel and synthesized them to create audio files. Although synthesized, MIDI-generated audio contained nearly identical tonal structure as in real recordings, and thus using a chroma feature as a front end, we aligned these two audio files using the dynamic time warping (DTW) algorithm.

In parallel, we extracted the vocal/non-vocal boundaries from note onset/offset information in the MIDI files,

(a) *No Reply* by The Beatles.



(b) *Smile* by Lily Allen.

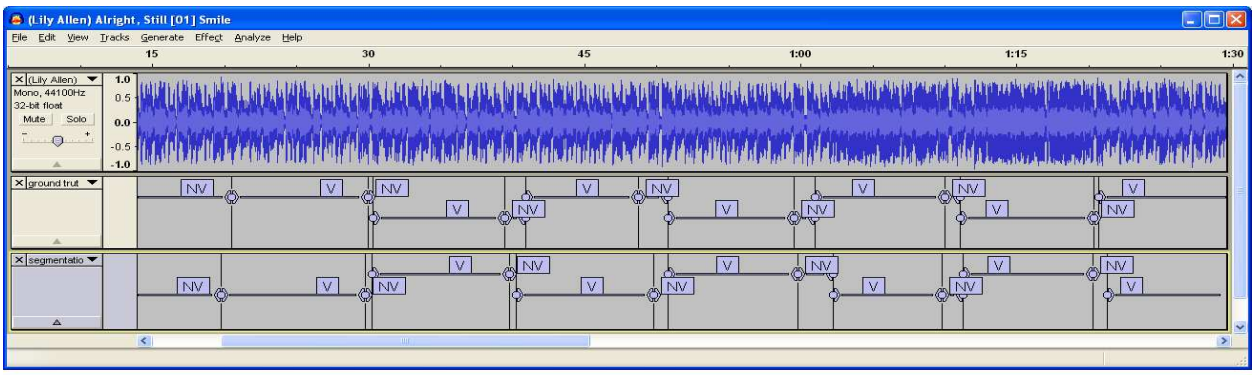


Figure 7. Vocal/non-vocal segmentation results for (a) *No Reply* by The Beatles and (b) *Smile* by Lily Allen. 'V' and 'NV' denote vocal and non-vocal segments, respectively. At the top is shown the PCM waveform. At the middle are shown the true boundaries. At the bottom are shown the boundaries detected by the algorithm.

and by mapping these boundaries to the minimum-cost alignment path found by the DTW algorithm, we obtained the precise vocal/non-vocal boundaries in real recordings. This process allowed to build a large labeled data set, and we built a hidden Markov model for each class. As acoustic features, we compared three different features — MFCCs, PLPCs and LFPCs. To investigate the models' performance on the size of the training data, we built the five different models for each feature and each class that are trained on 20%, 40%, 60%, 80% and 100% of the full training set.

We performed experiments on the two distinct data sets — a validation set and a test set. The validation set was part of the automatically labeled data which was not used in training, and the independent test set was composed of 10 songs in various styles that are manually annotated by the authors. The results on both the validation and the test set showed an increase in performance proportional to the size of the training data, for all three features.

Contrary to our expectations, the models performed better on the unseen, independent test set than on the validation set of the same kinds as the training set. This can be explained by the fact that there were some data in the validation set that were labeled incorrectly, due to errors in the labeling process. However, we believe that these errors are very small compared with the larger, correctly labeled data, and therefore have little or no effect when estimating the model parameters.

It is reported that the combination of information — combination of feature streams, posterior probabilities or hypotheses — results in better performance in classification [3, 16]. We therefore consider information combination in our future work to improve classification accuracy. Also, there are several applications where vocal/non-vocal discrimination is useful as described in Section 1. Among these applications, we consider using the results of our models for lyrics-audio synchronization.

References

[1] A. Berenzweig and D. Ellis. Locating singing voice segments within music signals. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, NY, 2001.

[2] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[3] D. Ellis. Stream combination before and/or after the acoustic model. In *International Computer Science Institute Technical Report*, Berkeley, CA, 2000.

[4] T. Fujishima. Realtime chord recognition of musical sound: A system using Common Lisp Music. In *Proceedings of International Computer Music Conference*, Beijing, 1999.

[5] H. Hermansky. Perceptual linear predictive (plp) analysis of speech. *Journal of Acoustical Society of America*, 87(4):1738–1752, 1990.

[6] Y. Kim and B. Whitman. Singer identification in popular music recordings using voice coding features. In *Proceedings of International Conference on Music Information Retrieval*, pages 164–169, Paris, France, 2002.

[7] C. S. Myers and L. R. Rabiner. A comparative study of several dynamic time-warping algorithms for connected word recognition. *The Bell System Technical Journal*, 60(7):1389–1409, 1981.

[8] T. L. Nwe, S. W. Foo, and L. Silva. Classification of stress in speech using linear and nonlinear features. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Hong Kong, 2003.

[9] T. L. Nwe, A. Sheony, and Y. Wang. Singing voice detection in popular music. In *Proceedings of ACM Conference on Multimedia*, New York, NY, 2004.

[10] T. L. Nwe and Y. Wang. Automatic detection of vocal segments in popular songs. In *Proceedings of International Conference on Music Information Retrieval*, Barcelona, Spain, 2004.

[11] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[12] M. Rocamora and P. Herrera. Comparing audio descriptors for singing voice detection in music audio files. In *Proceedings of Brazilian Symposium on Computer Music*, San Pablo, Brazil, 2007.

[13] E. Scheirer and M. Slaney. Construction and evaluation of a robustmultifeature speech/music discriminator. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Munich, Germany, 1997.

[14] J. Serra and E. Gomez. A cover song identification system based on sequences of tonal descriptors. In *Extended Abstract for MIREX 2007 Task on Audio Coversong Identification*, Vienna, Austria, 2007.

[15] G. Tzanetakis. Song-specific bootstrapping of singing voice structure. In *Proceedings of IEEE International Conference on Multimedia and Expo*, Taipei, Taiwan, 2004.

[16] S. Vembu and S. Baumann. Separation of vocals from polyphonic audio recordings. In *Proceedings of International Conference on Music Information Retrieval*, London, England, 2005.

[17] T. Zhang. Automatic singer identification. In *Proceedings of IEEE International Conference on Multimedia and Expo*, Baltimore, MD, 2003.