THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Vocal Detection: An evaluation between general versus focused models

Yi-Na Tsai

This thesis is submitted in partial fulfillment of the requirements for the Degree of Master of Science at the University of Waikato.

April 2011

Abstract

This thesis focuses on presenting a technique on improving current vocal detection methods. One of the most popular methods employs some type of statistical approach where vocal signals can be distinguished automatically by first training a model on both vocal and non-vocal example data, then using this model to classify audio signals into vocals or non-vocals. There is one problem with this method which is that the model that has been trained is typically very general and does its best at classifying various different types of data. Since the audio signals containing vocals that we care about are songs, we propose to improve vocal detection accuracies by creating focused models targeted at predicting vocal segments according to song artist and artist gender. Such useful information like artist name are often overlooked, this restricts opportunities in processing songs more specific to its type and hinders its potential success. Experiment results with several models built according to artist and artist gender reveal improvements of up to 17% when compared to using the general approach. With such improvements, applications such as automatic lyric synchronization to vocal segments in real-time may become more achievable with greater accuracy.

Acknowledgements

I would like to thank several people who helped me with my thesis. These people include my supervisor Associate Professor Steve Jones who was always enthusiastic and guided me through my research. I would also like to thank Rodney Macfarlane and Gabriel Engel for the sponsorship of the project and Rob Scovell who was an invaluable mentor.

Apart from people who constructively contributed to my thesis I would also like to thank my friends who shall remain anonymous for providing a healthy dose of distraction when needed.

Finally of course I want to thank my parents for supporting me, especially my caring mother for always preparing delicious meals to keep me energised throughout the day.

Table of Contents

Figures

Tables

1 Introduction

Nowadays mobile technology is advancing at an unprecedented rate and as the market has grown, demand for mobile applications has dramatically increased in the past few years. Since such a high percentage of the population now use mid to high end devices capable of performing many tasks that a desktop computer can perform, more people now rely on their mobile devices over their desktop computer than ever before. Approximately 70% of the world's population now carries a mobile phone and multimedia content sites such as YouTube have had reports in 2010 of approximately 200 million hits via mobile devices each day. Since mobile devices are on such high demand, mobile applications such as games, entertainment and multimedia have become highly sought after. Multimedia content such as music have become almost an essential part of the daily lives of a large majority of people hence many new audio processing techniques for various different applications such as music information retrieval have become increasing important.

The research component of this thesis is based on one of the functional requirements proposed for a commercial grade mobile application specifically targeted at mobile devices running the Android operating system. The application falls into the multimedia and entertainment category, more specifically audio entertainment. There are three main components to this project where each component corresponds to a functional requirement proposed for the mobile application. The third component is the research component and develops into the bulk of this thesis.

The system being developed requires the automatic synchronization of textual lyrics to its corresponding vocal segments within a song. Achieving the highest possible accuracy in synchronization is crucial due to the application being commercially focused and quality user experience is essential. In order for lyrics to be aligned to vocals, first vocal segments are required to be located. Existing vocal detection solutions are limited in their accuracy as they do not make use of information about the song being analyzed such as artist name or gender of artist.

In addition the end product is projected to run on mobile devices, therefore the solution must be balanced between complexity and precision.

Our approach focuses on taking advantage of available information about the song being analyzed as we believe knowing what type of data is being processed provides opportunity to process data in a different and more specific way. Since the typical approach is to build general purpose statistical models for classifying various types of segments of audio into vocal or non-vocal, our approach attempts to improve on this technique by building statistical models that classify segments of audio specific to artist and gender. We believe that there are certain vocal attributes that are unique to certain types of songs. Therefore, we aim to investigate the potential improvements of using specific statistical models over general purpose models. Since statistical models are also relatively lightweight, improving on current techniques using this method satisfies the complexity issue for mobile devices in some regard.

Test results reveal that using more focused statistical models improved classification accuracies significantly.

Structure of this thesis

This thesis is organized into eight main parts. The first part being the introduction, the second part introduces related work and useful concepts in the field of audio processing. The third part discusses the mobile platform used and the functional requirements of the mobile application which was developed in parallel as part of the research project. The fourth part describes a well proven technique for detecting vocals within audio tracks and the fifth part describes our approach to improving on that technique. The sixth part presents evaluations of our system and its practicability while the remaining two parts discuss conclusions and future work.

Parts one, two, six, seven and eight are rather self-explanatory; Parts three, four and five are described with slightly further detail in the following.

Part three, the functional requirements of the mobile application, provides some technical background which describes the nature of the mobile application and its

requirements. There are three major functional requirements, one is known to be a difficult problem and is the research component.

Part four, describing a well proven technique for detecting vocals within audio tracks, provides background knowledge on a widely accepted technique for vocal detection. This background information helps create a starting point for the research problem described in part three and will remain the main focus throughout this thesis.

Part five, describes our approach on improving on the technique mentioned in part four and also discusses the reasons for requiring improvement and why this technique is suitable for our application.

Note that due to the nature of project and the mobile application being developed, parts seven and eight discuss the feasibility of our approach and ideas of how the application's requirements can be fully fulfilled through the use of techniques learned.

2 Related Work

2.1 Background

There have been various studies in the field of audio processing, many of which are of much interest to us especially research in the fields of speech recognition, vocal detection within audio signals, song structure identification as well as automatic alignment between lyrics and songs. These techniques combined can be invaluable to our ultimate goal.

Vocal detection techniques in short, identify sections of an audio track containing singing voice. Knowing where vocals occur in a song can be especially helpful when attempting to align lyrics to vocals of a song.

Speech recognition techniques deal with transcribing sections of speech into text. Having prior knowledge of vocal detection, one would be quick to conclude that putting the two together would be helpful in automatically transcribing songs, however it is generally not reliable enough to transcribe human singing voice into text via conventional speech recognition, simply due to the fact that regular speech and singing have obvious different properties such as harmonics, frequency ranges, pronunciation and duration of syllables. In addition to such differences, singing voice in most cases are accompanied by background instrumental noise which conventional speech recognition does not account for, therefore while the basic idea may be to combine the two techniques it is not a very clever idea to attempt to obtain a transcription of a song simply by applying speech recognition directly on a song containing vocals. There are also complex techniques which do almost the opposite of vocal transcription; that is it takes a transcription and attempts to match it to its corresponding audio equivalent. This can be achieved by adapting a phone model for regular speech to a singing model [1]. This would be useful to us for obvious reasons; however it is not a very lightweight solution and can easily extend beyond our scope.

Forgetting about speech recognition (as it cannot be easily applied to singing voice) and lyric-to-song alignment, identifying sections of vocals within an audio

track (song) has its own difficulties. Most existing songs with vocals are accompanied by various background instrumentals. Fujihara *et al.* [2] propose a complex technique that reduce background accompaniments while others propose building classification models that learn the differences between vocals (with accompaniment) and accompaniment [3] [4].

Once we have obtained information regarding start and end points of vocal sections of a song, it could be useful for us to know the structure of the song i.e. which sections correspond to the verse and which sections correspond to the chorus. There have been several studies regarding song structure detection; the technique that stood out most to us was the use of chroma feature vectors to identify chorus sections of a song [5] [6] [7]. Chroma vectors are features of audio that can represent the intensity of the 12 most dominant pitches of a tiny segment of audio all arranged in chronological order, then a brief structure of the whole song can be visualized by plotting the intensity of each of the 12 pitches on its respective row of pitches in the time scale and then using self similarity techniques a high level structure of a song can be produced. Goto maintains that the most repeated section of a song is most likely to be the chorus [5]; therefore the use of self similarity to detect repeated patterns on the chroma vectors can give indications on where chorus sections are likely located within a song. Figure 1 illustrates a visualization of a chroma vector of a song where the dominant pitch classes are darker and the less dominant pitches are lighter in colour.

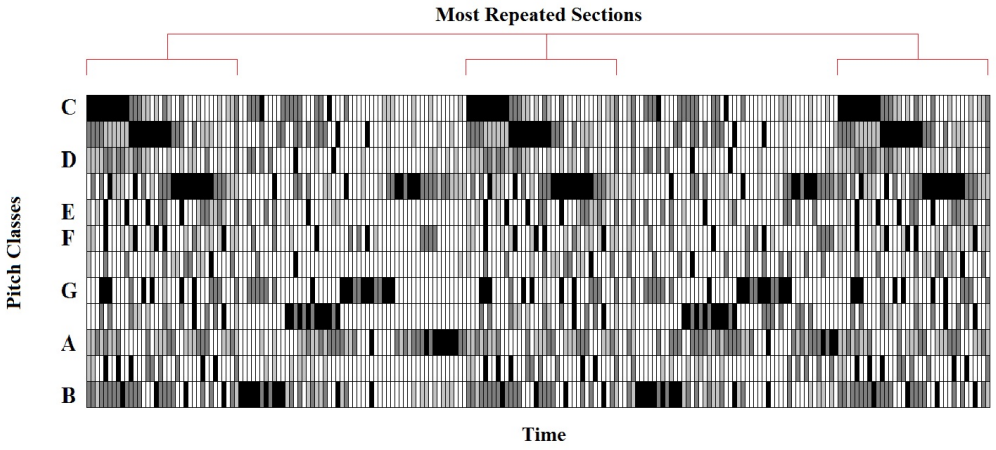


Figure 1 Chroma vector with repeated pattern

All these works established a good foundation for our own research goal, however, we found that previous work in the area of vocal detection only detect vocals in a broad sense where one general model is built for detecting vocals for songs of many different types e.g. pop, rap, classical, acoustic and different artists. Statistical vocal detection models to this day do not specifically take into account the vast amount of different types of songs. The basic idea is to have a model built using a wide enough range of songs so that for any given song it hopes to classify it accurately. However, in order for a model to be really general enough it can very quickly become a model too enormous to classify songs efficiently. This type of approach does not make use of information (song name, artist name etc) about songs being classified and most studies use relatively small corpuses for building statistical models. In addition we have never come across any studies that focus on applying low complexity vocal detection techniques onto mobile devices with limited resources.

2.2 Concepts of vocal detection techniques

In this section we will look at a few vocal detection techniques reviewed in slightly more detail. The first is Accompaniment sound reduction which summarizes how it can be helpful in vocal detection applications, the remaining are several variants of vocal detection by means of statistical models.

Accompaniment Sound Reduction

Since songs with singing voice are often accompanied by background music, it seems quite logical to attempt to reduce the accompaniment background music of a song to leave only the singing voice behind before attempting to locate sections of voice and transcribe or align lyrics to those sections of vocals. However, reducing background accompaniment is a complex problem. In the perfect case, a song's background accompaniment can be completely cancelled out if somehow we can obtain an audio sample of Vocal plus background accompaniment and an audio sample of the exact same background accompaniment without vocals by subtracting the raw audio signal of the latter from the former. The perfect case is almost impossible to come by as music production studios almost never produce

songs in a way that can be easily manipulated in this manner. The method in [2] describes a technique of sound reduction by extracting harmonic structure of the melody from the raw audio signals, and then, using a sinusoidal model re-synthesizes it. This technique in summary uses Goto's PreFEst [8] method to estimate the most predominant fundamental frequency ranges of an audio track and extracts it, then using an audio re-synthesis technique the melody is then re-synthesized resulting in a noise-reduced audio track [9]. The main issue with this technique though is that accompaniment noise cannot be completely removed easily, while this may be helpful for certain applications, using this to accurately identify start and end points of vocals of a song may require further steps.

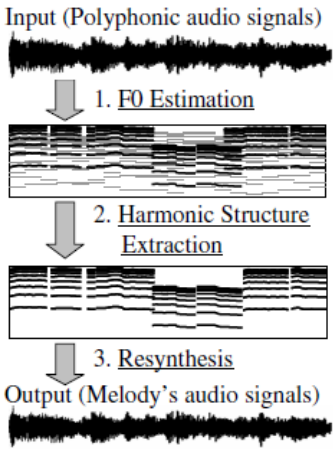


Figure 2 Picture taken from* [2] *illustrates the process of fundamental frequency estimation, harmonic structure extraction and re-synthesis

Audio feature Classification via Statistical Model

There are a number of variations of this technique, some vary in extraction of audio features and some vary in audio pre-processing techniques. The general concept behind audio feature classification is to build a statistical model that learns the difference between audio features that correspond to vocals and audio features that correspond to non-vocals. Such a statistical model can be powerful enough to classify any song's audio features into vocals and non-vocals and consequently allows for detection of vocal segments. An audio feature is a piece of information that can be derived from some amount of audio; it could be basic features such as the amplitude, frequency or any value that can be derived using those basic features. Since audio features are most commonly some numerical

value, they can represent some amount of audio of X length, therefore, intuitively one can deduct that audio features extracted for every sample in an audio stream in succession can represent the entire track of audio in some numerical form. The following are some existing approaches to vocal detection via statistical models that employ slightly different feature extraction techniques:

Fujihara *et al.* [1] proposed the extraction of LPC-derived mel cepstral coefficients (LPMCCs) for statistical model training as they claim that in the context of singing, LPMCCs are a better representation of vocal characteristics compared to MFCCs, an audio feature commonly used for music modelling.

Berenzweig *et al.* [4] proposed the use of several different audio features for classification, they then analysed the effectiveness of each. The main audio feature they extracted was the Posterior Probability Features (PPF) acquired from the acoustic classifier of speech recognizer. They claim that from that, they were able to derive a series of statistical models that could detect vocals with approximately 80% accuracy.

Nwe *et al.* [3] proposed the use of LFPC (Log Frequency Power Coefficients) as the bases for audio feature extraction. The claim is that LFPC audio features represent the energy distribution among sub-bands, and since vocal signals tend to contain higher energy levels than instrumentals their idea is to measure energy level of an entire song using LFPC values and mark where energy levels make a significant increase or decrease and conclude those changes in energy are the beginning and end points of vocals.

All the above have similar concepts in that some audio feature is extracted, that audio feature is then associated with vocals or non-vocals and then used in some statistical model for training and subsequently the model is used to classify some newly unseen audio into vocal or non-vocal segments.

3 Technical Background

This research on vocal detection and textual alignment to vocals is based on the functional requirements of an application targeted for mobile devices running the Android platform owned by Google Inc. The Android mobile platform was first founded in 2003 and was later acquired by Google Inc in 2005 and has become increasingly popular in recent times and is said to be one of the main competitors against another major mobile platform, the iOS, which stands for iPhone OS (operating system) and of course is run on the iPhone hardware developed by Apple Computer Inc.

Note that there is no favoritism toward a particular platform; the sole purpose of this introduction is to provide some idea of the scale in which the target platform operates and provide some brief history about the platform.

3.1 Android platform synopsis

According to the Android developer website [10] "Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language". Simply put, the Android platform is a combination of tools that can be incorporated into certain IDEs (Integrated Development Environment) which allows developers to write, compile and deploy applications all within the set of provided tools. The platform is based on the Java programming language, a popular and relatively easy to learn language. Applications are written in Java and the platform itself provides an extensive set of libraries for the developer. The platform is not completely limited to Java, if required it also provides ways of running C or C++ code through the use of native library calls. The Android platform is dubbed as an open development platform as many features used by the Android operating system itself such as accessing device hardware and setting system alarms is also available for developers to take advantage of.

3.2 Existing Mobile Application Overview

The mobile application that was developed in parallel with this research project originated from an application that was developed for the iPhone platform. Effectively, the application was ported from the iPhone platform to the Android platform with the intention of improving the application further. The nature of the application is based on the basic idea of a karaoke machine where songs stored on the mobile device can be played back with or without vocals along with some textual display of the lyrics to the song being played.

The main feature of the application is vocal removal from songs, this allows for simulation of a karaoke machine where songs can be played back with only instrumentals and users can sing along to the song minus the original artist's voice. The universal algorithm for vocal removal is to simply take the audio signal from the left channel and deduct it from the right channel (or the right from the left), the result is an audio signal with vocals reduced or completely removed in the perfect case. The requirement for this algorithm to work as desired is that the vocal signal in the left channel should be same as the vocal signal in the right channel and all other audio signals such as instrumentals can be different between both channels, this then allows the direct subtraction of one vocal signal from the other leaving only the instrumentals.

Other features included in the mobile application are vocal effects. These vocals effects are real-time transformations of audio signals captured through the device's microphone, once the audio signal is transformed it is output to the device's speakers with as little latency as possible giving the end user a sense of time-time vocal transformation. Effects such as echo, reverb and even auto-tune can be chosen by the user, this gives a somewhat simulated karaoke experience.

3.3 Mobile Application Functional Requirements

The nature of the application as mentioned is similar to that of a karaoke machine, therefore the functional requirements that follows are in accordance to the nature of the application. The following are the brief goals set out for the application:

- Port existing iPhone application to Android

- Implement vocal effects for sound input via microphone
- Investigate and implement synchronization of audio playback and lyric tracking

Port existing iPhone application to Android

An existing iPhone application was ported to the Android platform; it performs vocal removal from audio files and plays back the audio file with vocals removed. The application also displays lyrics (if it exists) for the particular song that is being played back and allows users to use the microphone to amplify their vocals through the device's speakers. The most basic goal was to port these functionalities to Android first. There were restrictions on the iPhone where developers had very limited access to audio files stored on the device and the only form of access to audio files was to use the API for audio playback. For this reason, audio manipulation such as vocal removal could not be easily applied directly to audio files on the device; some form of third party file transferring application was required. With the Android platform such restriction does not exist and audio files are accessible without issues, the only requirement is that the application should declare in some metadata that it requires access to manipulate files stored on disk. Porting the application to Android was relatively straightforward, the only major issue at the time of development was that low latency audio buffers for real-time audio processing were not available and vocals captured by the microphone could not be played back within a tolerable amount of delay.

Implement vocal effects for sound input via microphone

This function of real-time vocal effects was not implemented in the original iPhone version of the application; this feature is one of the improvements made on top of the iPhone version. Vocal effects provide a richer experience for the user as it allows users to transform their voice in real-time and can prove to be exceptionally entertaining. The level of difficulty to implement this feature on the iPhone is unknown since this feature was never part of the iPhone version,

however, on the Android platform this feature also proved to be rather straightforward to implement. Algorithms and libraries for vocal transformation are readily available. The most difficult component to implementing this feature was converting audio samples in such a way that make it usable for certain libraries, and in some cases modification of library code was required. The other challenging aspect to implementing this feature was dealing with latency issues since the Java programming language typically is not as lightweight as other native programming languages such as C or C++ and computing transformations of real-time vocal input and output required more than simply applying algorithms to audio samples, therefore the JNI (Java Native Interface) was used to make native library calls from the Java layer which consequently allowed faster audio transformation to take place. Vocal effects implemented included but not limited to echo, chorus, reverb and pitch-shift.

Investigate and implement synchronization of audio playback and lyric tracking

From a functionality stand point, the goal was to develop a method for taking a song containing vocals and its corresponding lyric file (in plain text without any timing data) and display the lyrics in synchronization with the vocals from the song. In other words the perfect end result would be to be able to display lyrics along with some form if indicator pointing to the current word of a song that is supposed to be sung similar to that of karaoke machines.

From a research stand point, the goal was to find effective ways of locating vocal segments within a song. Combined with audio structure and lyrical structure analysis, it may be feasible to utilise such techniques to align lyrics with detected vocal segments. Few ideas contributing to this goal during the early stages of the project included voice recognition techniques for detecting words in the song and aligning the words from the lyrics word by word, however, after some consideration it was clear that voice recognition would not serve our purposes. Bear in mind that techniques considered in this project must be sufficiently lightweight since the platform it is to be run on is targeted at mobile devices therefore the solution must be well balanced between complexity and speed.

The approach we took focuses on detecting the location of vocal segments given that knowing where vocals occur during a song provides opportunities for anchoring lines of text to start points of vocals and an initial high level alignment can possibly be achieved. The more accurately vocal locations can be detected the more accurate lyric alignment can become. Using other techniques such as song structure detection with chroma vector analysis and lyric structure detection using self similarity analysis, the possibility of aligning lyrics with vocals more accurately may increase.

4 Vocal Detection

4.1 Vocal Detection via Statistical Model

Classification of a song with statistical models is a relatively inexpensive way of detecting segments containing vocals or non-vocals. The catch is that in order for classification to be effective enough across a wide range of different types of songs, a relatively large corpus containing many different types of songs is required, each of which is manually annotated where segments of a song are marked ‘vocal’ or ‘non-vocal’ (from here on we will refer to ‘vocal’ as human singing voice which may be accompanied by background instrumentals or not, and ‘non-vocals’ shall be referred as only instrumentals or silence) for a statistical model to be trained on. This means that for any given song to be segmented, the statistical model should have a wide enough range of data to be able to predict whether any given segment of a song is vocal or non-vocal. To get quality results the model is really dependent on the amount and precision of manually annotated data, therefore the more quality data available the better. However, it should be noted that due to time constraints manually annotating large sets of songs in real time does not always produce precise training data because of many factors including human reaction time, ignoring slight pauses or breaths between words being sung and the simple fact that one human alone cannot know the start and end times of vocal sections of every song which means mistakes are bound to be made.

In order to train a model, audio features for each frame of a song from the corpus needs to be calculated. Audio features are properties of a song that can be extracted or formulated, for example a 10 millisecond frame of audio can have basic audio features such as frequency range or amplitude.

In our case we used LFPC (Log Frequency Power Coefficients) audio features, which in basic terms give indication of energy distribution among sub-bands. It splits a frame of audio into a specified amount of sub-bands with ranges spaced logarithmically (Figure 3) between the lowest and highest frequency limits where the average human singing vocal frequencies fall. In other words, some audio data

(perhaps 0.21 seconds to 0.41 seconds) between the frequency ranges of say 4 kHz to 6 kHz is isolated (by using a technique known as band-pass filtering or sub-band filtering) and its energy level is analyzed, then the same piece of audio (0.21 seconds to 0.41 seconds) is isolated between frequencies of say 6 kHz and 8 kHz and is analyzed the same way until all sub-bands have been analyzed. We end up with a set of 12 (although this number can be adjusted we shall use 12 as an example) LFPC values for each small frame of audio (Figure 4) which can be imagined as something structurally similar to a 2-dimentional array.

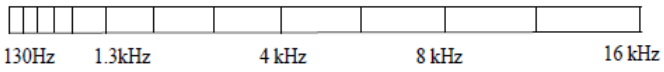


Figure 3 Image taken from* [3] *shows frequency ranges spaced logarithmically where the number of sub-bands is 12

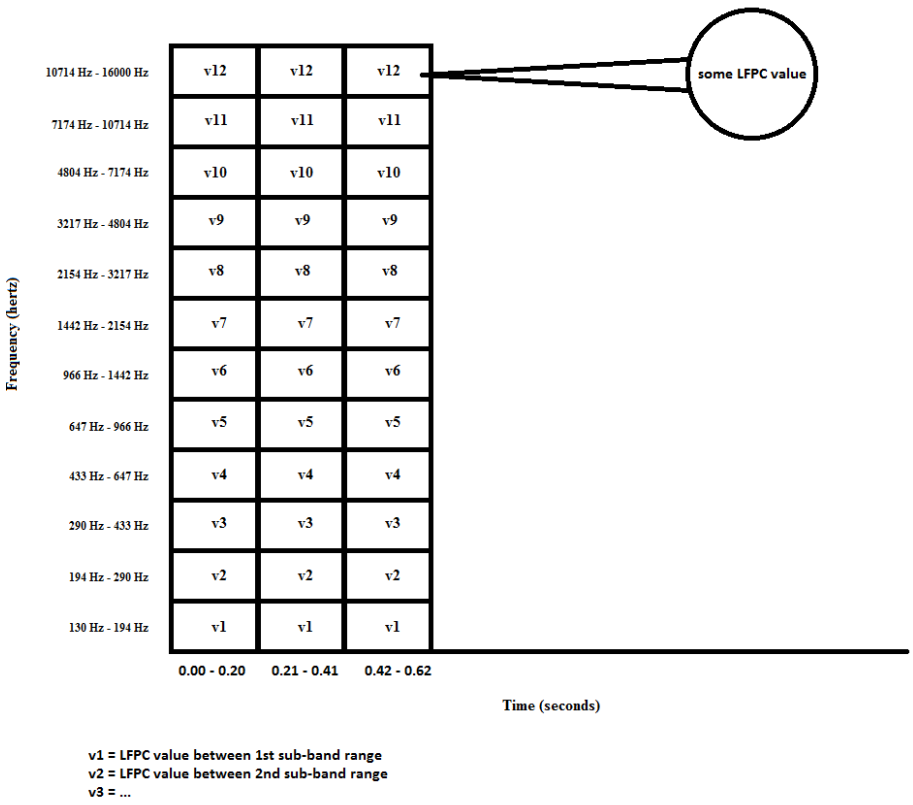


Figure 4 An illustration of small segments of audio each having 12 LFPC values

4.2 Calculating LFPC Values

To begin with, the audio data can be processed with various frame and window hop sizes. A frame of audio is the piece of audio data that is to be analysed, it could be for example 20 ms or 200 ms, however once the frame size and window hop size has been decided on it shall remain fixed throughout the entire analysis process of a song. The window hop size can be thought of as the amount of milliseconds we slide along the long queue of audio data before capturing the next frame of audio data for analysis. The process for calculating LFPC feature values does not read audio data one frame at a time where each frame contains completely unseen audio data, but rather it reads audio data with some overlap where each frame of audio read contains some audio already seen from the previous frame. Figure 5 illustrates the difference between reading audio data with a windowed hop versus reading completely new unseen audio data.

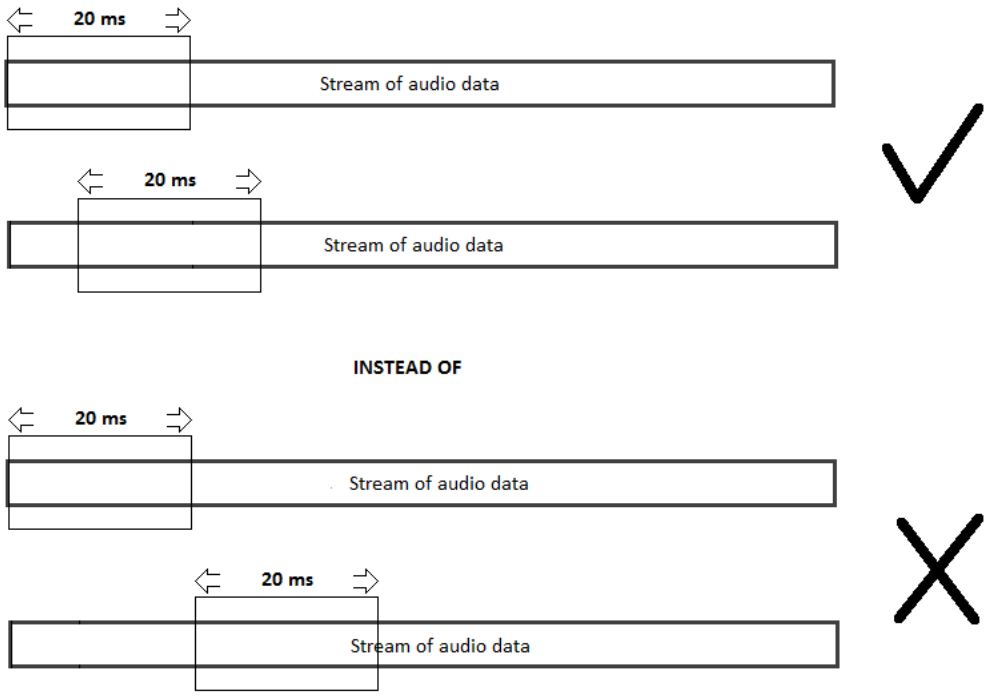


Figure 5 Reading 20 ms of audio with some overlap versus reading without overlap

Let us take for example, for a frame size of 20 ms and a window hop size of 13 ms we would read the first three frames at the specified time positions like so:

Frame 1: 1-20 ms

Frame 2: 8-27 ms

Frame 3: 21-40ms

..

Each frame of audio that is read is then multiplied by a Hamming Window function, which in simple terms transforms a frame of audio slightly so that values that are close to zero are raised up slight but maintains its overall trend. This helps minimise discontinuities at the end of each frame (see Figure 6**Error! Reference source not found.**). The formula of a Hamming Window function is as follows:

$$Window(k) = 0.54 - 0.46 \cos(2 \ PI \ k / (n - 1)) \quad \textbf{Equation 1}$$

Where n = number of input values and k = 0...n-1

For a 20 ms frame example, n would be 20 and k would go from 0 to 19. In other words, for a 20 ms frame each n^{th} millisecond is multiplied by the above function with variable k depending on which millisecond out of the 20 it is looking at.

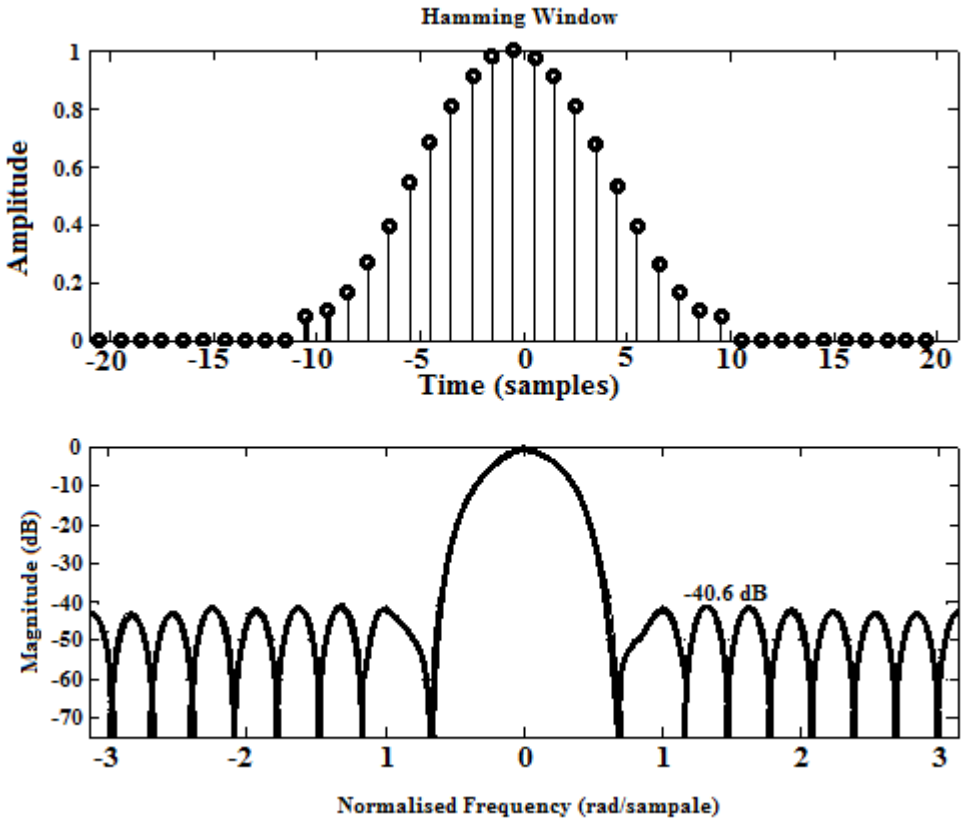


Figure 6 Original samples transformed with a Hamming Window-function which results in values towards zero being raised up while maintaining trend

For each frame of audio that has been multiplied by the Hamming Window function, a FFT (Fast Fourier Transform) is then applied. FFTs are commonly used in audio signal processing for analysing the frequencies contained in a sampled signal. The original signal can be thought of as amplitude on the y-axis and time on the x-axis (Figure 7) and FFT output is simply a transformation to magnitude on the y-axis and signal frequencies (Hz/kHz) on the x-axis (Figure 8). There are many implementations of FFT and each has its own advantages and disadvantages; in our case we used the org.apache.commons.math.transform java library.

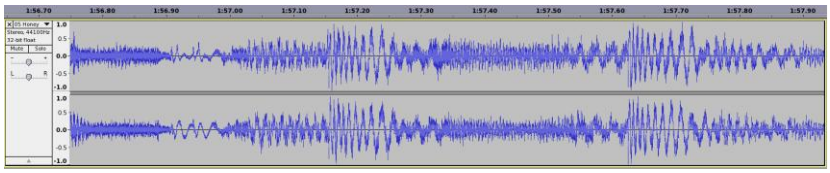


Figure 7 Amplitude versus Time

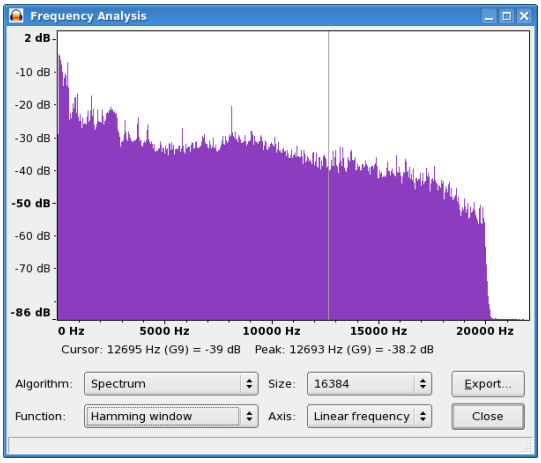


Figure 8 Magnitude versus Frequency

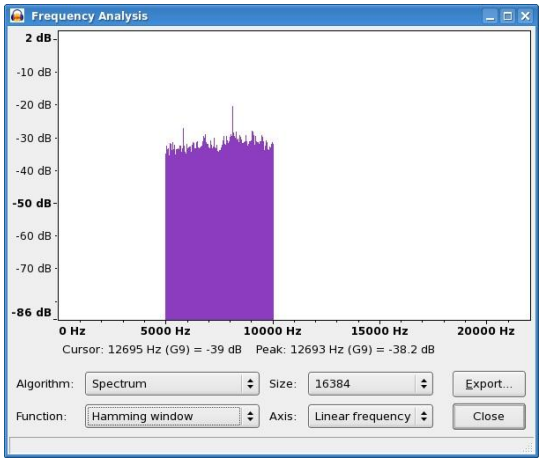


Figure 9 Band-pass filter ranging from 5000 Hz to 10000 Hz

Now we have a representation of each frame with a Hamming Window function then a FFT applied resulting in magnitude (dB) versus frequency (Hz). Each of these resulting frames are then passed into a set of 12 band-pass filters where the first filter begins at approximately 300 Hz and the last filter ends at 16 kHz with the width (frequency range) of each filter not constant but rather spaced logarithmically (Figure 3). Let us pretend that one band-pass filter has the frequency range from 5000 Hz to 10000 Hz, imagine taking out an individual slice (Figure 9) between those boundaries out of the 'Frequency Analysis' diagram (Figure 8), that would be one band-pass output value. We will end up with 12 output values for each frame of audio where each output effectively represents a magnitude value for its corresponding frequency range according to the 12 logarithmically spaced frequency ranges. The formula for calculating LFPC values are presented by the following two equations [3]:

$$S_t(m) = \sum_{k=f_m-\frac{b_m}{2}}^{f_m+\frac{b_m}{2}} \left(X_t(k)\right)^2 , \quad m = 1,2,\ldots,12$$

Equation 2

Where $X_t(k)$ is the k^{th} spectral component of the hamming windowed signal, t is the frame number, $S_t(m)$ is the output of the m^{th} sub-band, and f_m and b_m are the centre frequency and bandwidth of the m^{th} sub-band, respectively.

$$LFPC_t(m) = 10\log_{10}\left[\frac{S_t(m)}{N_m}\right]$$

Equation 3

Where N_m is the number of spectral components in the m^{th} sub-band.
For each frame, 12 LPFCs are obtained

Equation 2 results in 12 output values for each frame as is expected, 1 from each band-pass filter which are the sub-bands represented by *m*. Therefore, for instance for frame number 1 we shall compute $S_t(m)$ where $t = 1$, resulting in a series represented by:

$$S_1(1), S_1(2), \ldots, S_1(12)$$

And for frame number 2 where $t = 2$

$$S_2(1), S_2(2), \ldots, S_2(12)$$

And so on...

For each $S_t(m)$ we shall compute the sum of $(X_t(k))^2$ for the values of k in the given range. $X_t(k)$ is the magnitude for frequency k of frame t.

The minimum and maximum values of k are respectively given by:

$$f_m - (b_m/2) \qquad \text{and} \qquad f_m + (b_m/2)$$

Where f_m is the centre of sub-band m and b_m is the width of sub-band m. Therefore, if the sub-band were 4000 Hz to 6000 Hz then f_m would be 5000 Hz and $b_m/2$ would be 1000 Hz. Hence values of k ranges from 4000 to 6000 in this case.

Now in order to obtain the 12 LFPC values for each frame t we use the 12 output values from **Equation 2** to substitute the variable $S_t(m)$ in **Equation 3**. Here, m = 1, 2, ..., 12 as in **Equation 2**.

4.3 Training data preparation

For each frame of LFPC values in the time scale (ordered in chronological order) a 'vocal' or 'non-vocal' tag can be associated with it by using the manually annotated data already prepared earlier. Since the manually annotated data should contain information which describes when vocals begin and end accurate to milliseconds, it is then possible to associate for example the first 5000 milliseconds of LFPC values of a song with multiple vocal or non-vocal tags by automatically searching the manually annotated data of that song and checking whether the first 5000 milliseconds of that song contains vocals or non-vocals or a mixture of both and at what times. Lets imagine that each set of 12 LFPC values below represent a 500 millisecond frame and the manually annotated data (Table 1) states that for some given song from 0 milliseconds to 1000 milliseconds there were non-vocals and from 1001 milliseconds to 1500 milliseconds there were vocals, then the associated values would look similar to Table 2. In reality the tables would be far larger in size for an average song length.

0 – 1000	Non-vocals
1001 – 1500	Vocals

Table 1 Manually annotated data

0ms – 500 ms	$LFPC_1(1)$, $LFPC_1(2)$, $LFPC_1(3)$, $LFPC_1(4)$, $LFPC_1(5)$, $LFPC_1(6)$, $LFPC_1(7)$, $LFPC_1(8)$, $LFPC_1(9)$, $LFPC_1(10)$, $LFPC_1(11)$, $LFPC_1(12)$	Non-vocals
501ms – 1000 ms	$LFPC_2(1)$, $LFPC_2(2)$, $LFPC_2(3)$, $LFPC_2(4)$, $LFPC_2(5)$, $LFPC_2(6)$, $LFPC_2(7)$, $LFPC_2(8)$, $LFPC_2(9)$, $LFPC_2(10)$, $LFPC_2(11)$, $LFPC_2(12)$	Non-vocals
1001ms – 1500 ms	$LFPC_3(1)$, $LFPC_3(2)$, $LFPC_3(3)$, $LFPC_3(4)$, $LFPC_3(5)$, $LFPC_3(6)$, $LFPC_3(7)$, $LFPC_3(8)$, $LFPC_3(9)$, $LFPC_3(10)$, $LFPC_3(11)$, $LFPC_3(12)$	Vocals

Table 2 LFPC vocal/non-vocal tag association

According to Zhang [11] the idea is that if only instrumentals (accompaniment) was happening and some amount of time later vocals was heard, one would see a sudden increase in energy levels across the LFPC values [3]**.** Therefore, sections that were marked as vocals should have higher associated LFPC values whereas sections marked as non-vocals have should lower associated LFPC values. All the LFPC values and their associated tags are then bundled together to create one file (which can contain LFPC values for multiple songs) which will be used for training the statistical model.

4.4 Training

Once all necessary training data has been prepared the next step is to train a model using this data. To train or build a model a statistical model builder is required, we will not discuss how one can be obtained, however in a later chapter will describe our approach and the exact tool that we used to train models. The basic idea behind training statistical models is that a set of pre-classified examples/instances are passed to a statistical model builder, then the statistical model builder attempts to learn which values correspond to which class. In terms of the vocal detection technique we are employing, one example/instance of training data would contain a set of feature values (LFPCs) and along with that set of values a class is associated with it, in this case it would be either vocals or non-vocals. Typically statistical models are trained with relatively large amounts of examples/instances, it begins to differentiate classes with higher accuracy since it has seen so many example values that correspond to vocals and so many that correspond to non-vocals. Typically the more it trains the more accurate it becomes at differentiating classes.

4.5 Classification

Once a model has been built, segments of a randomly selected song can be automatically classified into vocals or non-vocals, first by calculating LFPC values for each window of the song and then running those values through a classifier using the model already built to determine whether segments of the selected song are possible vocals or non-vocals (with varying degrees of accuracy). Typically the classification process can classify segments that already

contain actual class values (also known as training data) or segments that have never been seen before. The advantage of classifying training data is that accuracy of classification can be measured by comparing predicted classes against actual classes. The classifier classifies instances in no particular order, once it has classified all the instances it has been given it can produce an accuracy result. Note that although obtaining an indication of accuracy of a particular model's classifier is helpful for evaluating its quality, it does not however give indication of the location of vocals. The accuracy of a model's classifier will remain the same if the same series of instances it performs accuracy tests on is completely shuffled and tested again. Figure 10 illustrates how accuracy can be the same when instances are classified in chronological order compared to instances classified in random order.

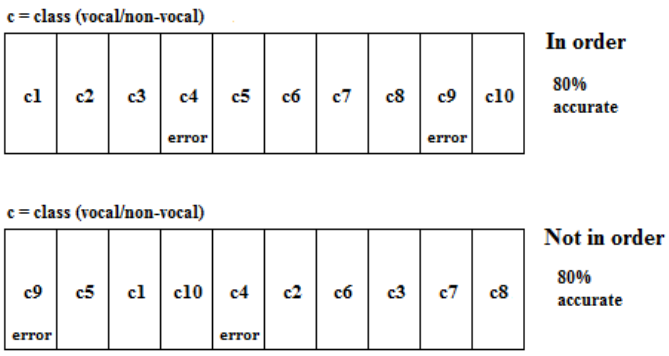


Figure 10 classification output in order versus not in order

What we really are concerned about are the class predictions of unseen instances in the chronological order in which they were calculated. Let us look at Figure 11 for instance, frames are read from the audio stream in sequential order and for each frame of audio LFPC values are extracted while maintaining the same sequential order, once all LFPC values have been calculated for each frame they are then classified while still maintaining the order in which the frames were read. Since all classes were classified in chronological order, each class value can represent the exact same time segment as its corresponding frame of audio. In other words, if the fifth frame (f5) in the audio stream represents a piece of audio that goes from 200 milliseconds to 250 milliseconds then the fifth class (c5) value

represents the predicted class (vocals/non-vocals) that goes from 200 milliseconds to 250 milliseconds.

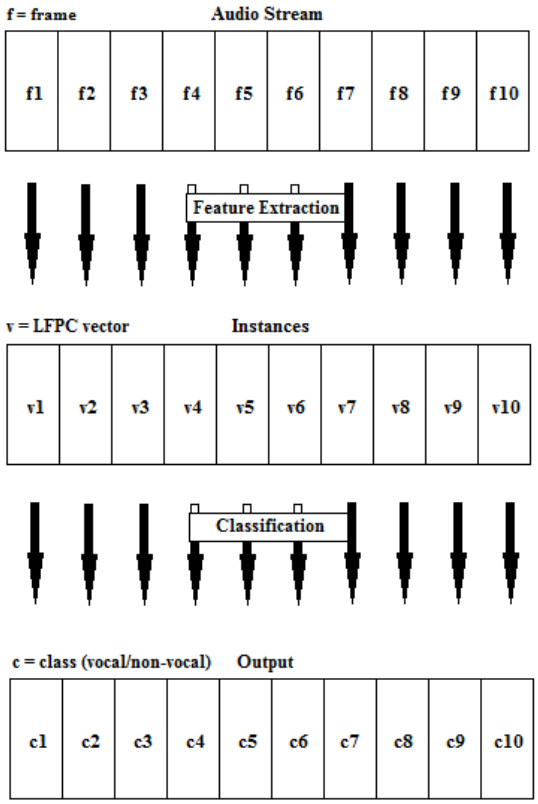


Figure 11 Flow from feature extraction through to classification

If we can get a result of perhaps 80% correct classification of vocals for a 2 second segment of a song we can conclude that those 2 seconds is indeed overall 'vocals'. There will be misclassifications among the 2 second segment, thus the 80% classification accuracy. Any incorrectly classified frames are likely to be spread throughout the 2 second segment as in Table 3 instead of all being clustered together as in Table 4. Since the frames that were classified as N (non-vocals) are spread throughout the 2 second segment as illustrated in Table 3 it is most likely that they are misclassifications, and can be tolerated as long as the overall trend point towards a clear trend. Table 3 illustrates that for a 2 second segment of audio the overall trend is 80% V (vocals).

However, if the classification process produces a result as illustrated by Table 4 where the classifier predicts classes with a cluster of N (non-vocals) in between

other clusters of vocals it may require further analysis to determine whether they are misclassifications or a legitimate segment of non-vocals even though the entire 2 second segment retains a vocal trend of 80%.

Although the overall trend can provide indication of vocals or non-vocals it can be difficult to justify how large the overall trend space should be. If we analyzed the trend for 2 second audio segments where each frame is 1 second long, every now and then it would be impossible to find the trend if 1 frame is classified as vocals and the other frame is classified as non-vocals. It would require some balancing between the size of the trend space and the size of frames. Perhaps increasing the trend space and lose precision or decreasing the frame size and possibly produce more misclassifications.

In a later chapter we will discuss a simple approach for resolving possible misclassifications.

V	V	V	V	V	V	N	V	V	N	V	V	V	N	V	V	V	N	V	V

Table 3 Theoretical classification output in chronological order with possible misclassifications spread apart

V	V	V	V	V	V	N	N	N	N	V	V	V	V	V	V	V	V	V	V

Table 4 Possible misclassifications clustered together

4.6 LFPC Parameters

There are several variables involved when calculating LFPC values, they are:

- Frequency range
- Frame size
- Window hop size

Frequency range

Typically the frequency ranges are set to include the majority of the human singing frequency range since the majority of observed energy changes are a

result of human sing voice being present. We can set the highest and lowest frequency bounds of the logarithmically spaced sub-bands to any value, however, the impact of different sets of frequency bands will depend upon how different the magnitude values for the audio are across the frequency range. If we load a track into an advanced audio application and plot the spectrum displaying with it a log frequency axis we can observe slight differences between the different frequency bands, however the differences are not extreme. Take for example the following figures where Figure 12 represents a snippet of audio containing only music and Figure 13 contains vocals and music, in both figures the difference in magnitude between different sub-bands are subtle across all sub-bands, thus frequency range has almost minimal impact. There is however a noticeable difference between the two figures where the overall magnitude levels across all the bands of vocals is higher than non-vocals; this is some indication of vocal presence.

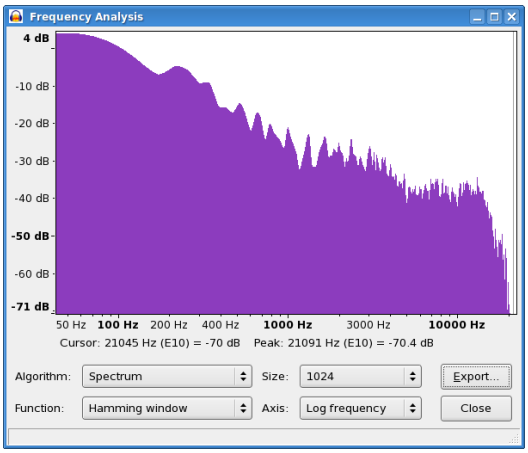


Figure 12 Frequency analysis of non-vocals

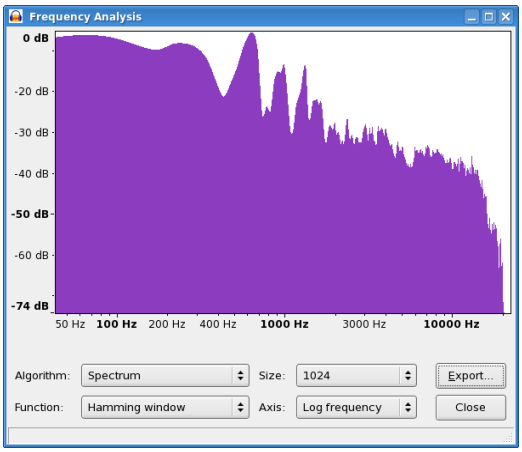


Figure 13 Frequency analysis of vocals

What is of interest to us is the difference in energy distribution between vocals and non-vocals across sub-bands. Figure 14 illustrates the differences in energy distribution between vocals and non-vocals across 12 sub-bands where blue samples represent non-vocals and red samples represent vocals. By nature there are significantly less non-vocal samples due to the fact that songs usually contain more segments with vocals present, however in any case there are obvious differences in energy distribution between vocals and non-vocals where vocal

samples have visibly higher energy distribution. The larger the difference between the two classes the higher the chances of the statistical model learning the difference, thus classifying samples of any given song into vocals or non-vocals with higher accuracy.

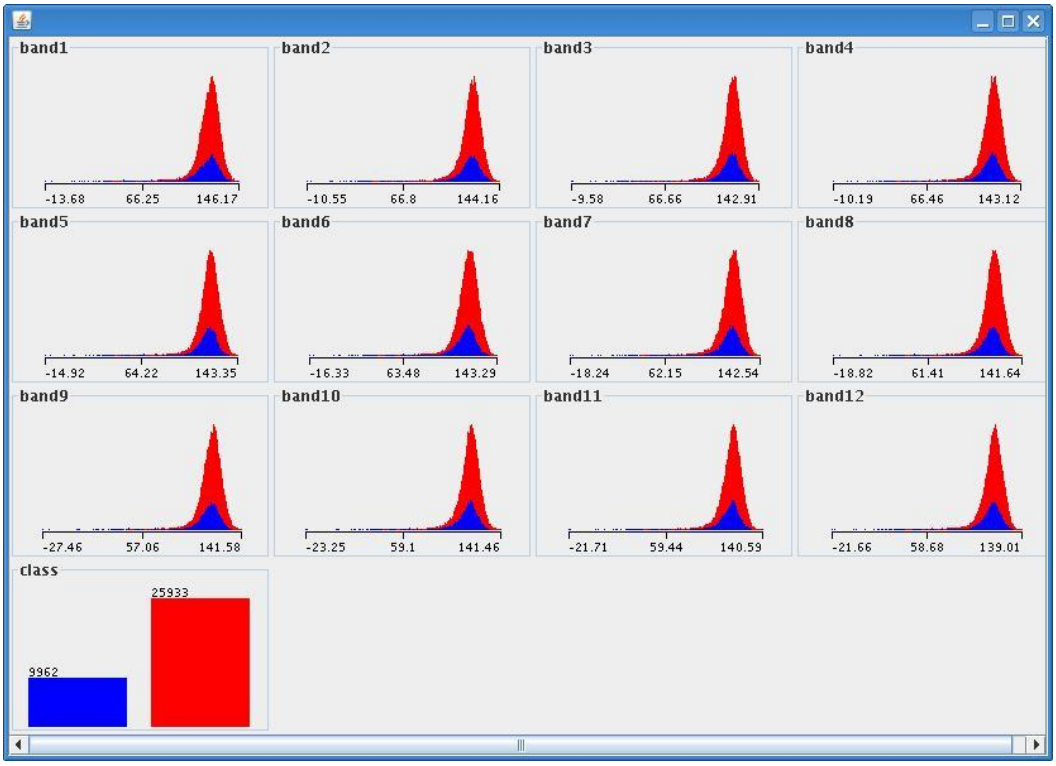


Figure 14 Distribution of LFPC values across 12 sub-bands

Frame size

The frame size of an audio sample can in fact be any size. It could be 5 milliseconds or 5 seconds, however the size does have affect on the accuracy of classification. This is true due to the fact that manual annotation is not 100% accurate and can be inaccurate up to 500 milliseconds. Take for example a manual annotation situation, the user attempts to annotate a 1500 millisecond song containing 500 milliseconds of vocals at the beginning of the song as illustrated by Figure 15 A. Figure 15 B shows the user committing an error annotating the vocals segment resulting in tagging the first 600 ms as vocals instead of 500 ms and tagging the remaining 900 ms as non-vocals instead of 1000 ms.

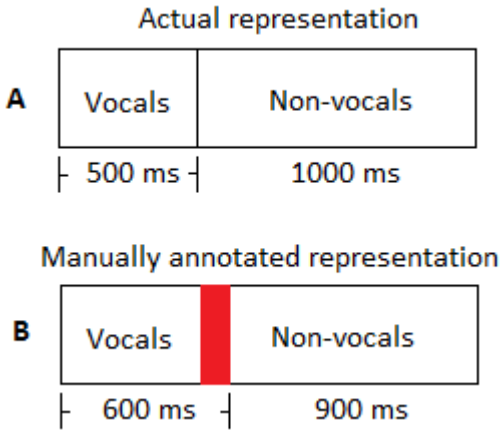


Figure 15 Actual vocal/non-vocal versus manually annotated vocal/non-vocal

As mentioned earlier, mistakes are bound to be made due to imperfect human reaction times; ignoring slight pauses or breaths between words being sung and the simple fact that one human alone cannot know the start and end times of vocal sections of every song. Now let us take for example a frame size of 10 ms (forgetting about window hops for a moment) for analyzing LFPC values of the 1500 ms of audio, we would end up with 150 audio frames each being 10 ms long. Due to the 100 ms of error committed while manually annotating we end up with 10 frames of audio being incorrectly annotated, this could potentially give undesirable results. The classifier now "thinks" that it has seen 60 examples of vocals and 90 examples of non-vocals instead of 50 examples of vocals and 100 examples of non-vocals; this means 16.7% (10/60) of the examples seen for vocals are false and classification errors will be inevitable.

Now let us suppose the frame size for LFPC values is 250 ms, we would obtain 6 frames of LFPC values to cover the 1500 ms. The first 2 frames (500 ms) shall be correctly tagged as vocals, the third frame (which is in fact non-vocals from 501 ms to 750 ms) is said to contain 100 ms of "apparent" vocals and 150 ms of non-vocals and the remaining 3 frames will all contain non-vocals. The third frame is made up of 100 ms vocals and 150 ms non-vocals, however due to the fact that a frame of 250 ms can only have one tag between vocal or non-vocal, the dominant

annotation prevails resulting in the third frame being tagged as non-vocals. This time the classifier "thinks" that is has seen 2 examples of vocals from the first 500 ms and 4 examples of non-vocals from the remaining 1000 ms, which is in fact exactly how the 1500 ms song is composed. This demonstrates how different sized frames can affect the accurateness of training data generation.

However, although in that example having a larger frame size appears to generate more accurate training data it is only advantageous to a certain point. Take for example a frame size of 2000 ms, we may very well generate reliable training data with this frame size, however we can lose a great deal of precision when it finally comes time to segment songs into vocals and non-vocals. Precision is lost due to such a large frame size, anything that was manually annotated ends up being rounded to the nearest 2000 ms during the training data preparation stage. This especially undesirable when for example a section of vocals goes from 0 ms to 1600 ms, however due to a 2000 ms frame size results in the predicted vocal segment to begin at 0 ms and end at 2000 ms. Figure 16 illustrates the actual begin and end intervals for various segments of a song, along with its predicted begin and end intervals. We can see that it is obviously lacking precision as it only predicts segments to the nearest 2000 ms, therefore when choosing a frame size for LFPC analysis the desired degree of precision of begin and end points should be taken into consideration. The model could be 100% accurate at predicting, however having songs segmented into 2 second chunks is almost of no use.

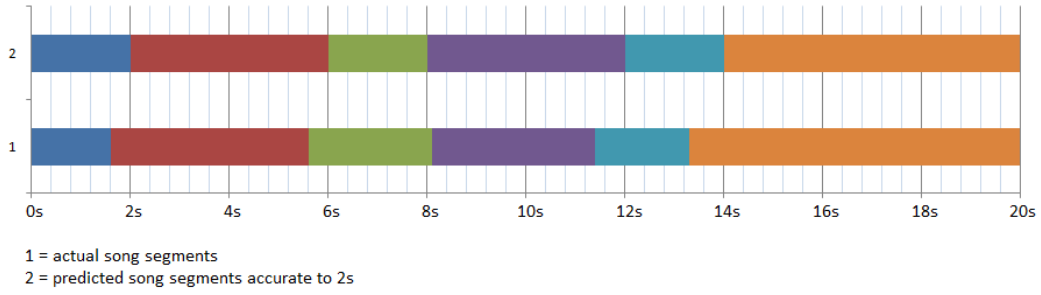


Figure 16 Actual song segments versus Predicted song segments with 2s frame size

Window hop size

In actual fact the window hop size can be of any size, however in order to achieve sensible audio analysis, a window hop size should be smaller than the frame size. The window hop size is the amount to "hop" across the stream of audio before capturing the next frame of audio (refer to section 4.2 Calculating LFPC Values). If the window hop size is equal to the frame size we would effectively be reading through audio samples one after the other each being newly unseen audio samples (see Figure 5), if the window hop size is larger than the frame size we would end up omitting audio samples at fixed intervals (see Figure 17).

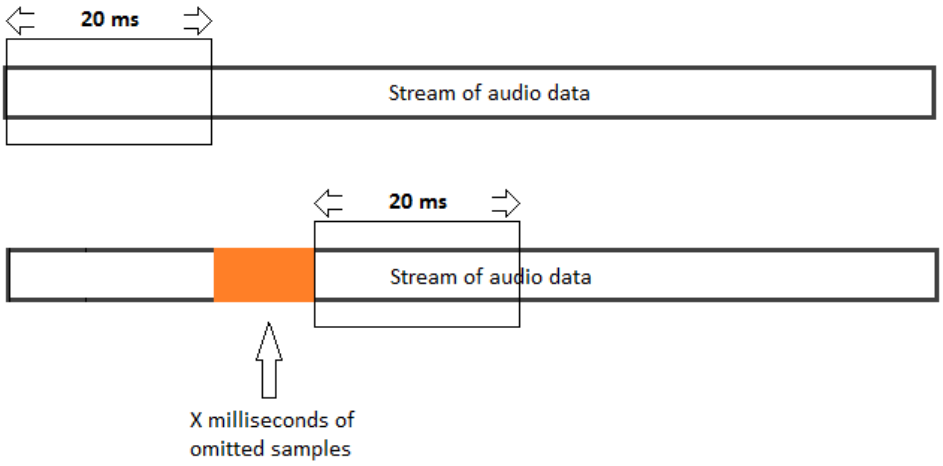


Figure 17 Excessive window hop resulting skipped samples

Take for example a 20 ms frame and 13 ms window hop, we would read 20 ms frames at 13 ms intervals and in effect creating a 7 ms overlap (where overlap is equal to frame size minus window hop size) between frames. The first 3 frames would be read at the following times:

1 ms – 20 ms

13 ms – 33ms

26 ms – 46 ms

Analyzing audio using a windowed approach has a couple of advantages, the first being that we obtain more samples per song hence obtaining more training data, the second and perhaps the most important advantage is so that "interesting" changes in the audio that would otherwise be split by frame boundaries are captured. For instance, if something interesting happens at 17 ms – 24 ms, having no overlap would mean that part of the interesting frame was in frame 1 ms – 20 ms and part of it was in frame 21 ms – 41 ms. However, by reading frames with some sensible window hop or overlap means that the interesting frame at 17 ms – 24 ms is fully captured in frame 13 ms – 33 ms.

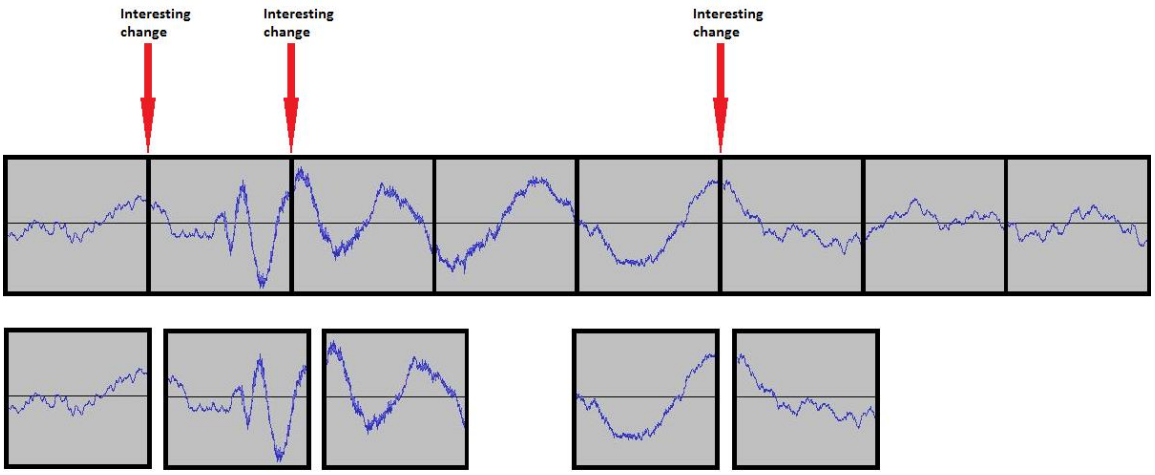


Figure 18 Un-windowed sample reading

Let us first look at an example of a un-windowed approach to reading audio samples. Figure 18 illustrates a stream of audio being captured one by one (un-windowed) where each square represents one frame of audio captured. As the arrows suggest, there are several interesting peaks in the audio that fall between the boundaries of frames being read. Each separate square on the bottom of the diagram illustrate how each frame is seen by the audio analyzer, it does not have any intuition about the peaks between those frame boundaries or the fact that they are even supposedly joined. This can be troublesome especially when those interesting peaks actually represent some energy spike, perhaps some indication

of vocals being sung, however due to the peak being split in half it loses its impact and potentially becomes merely a slight curve in another frame and is perhaps overshadowed by some other interesting peak or plunge.

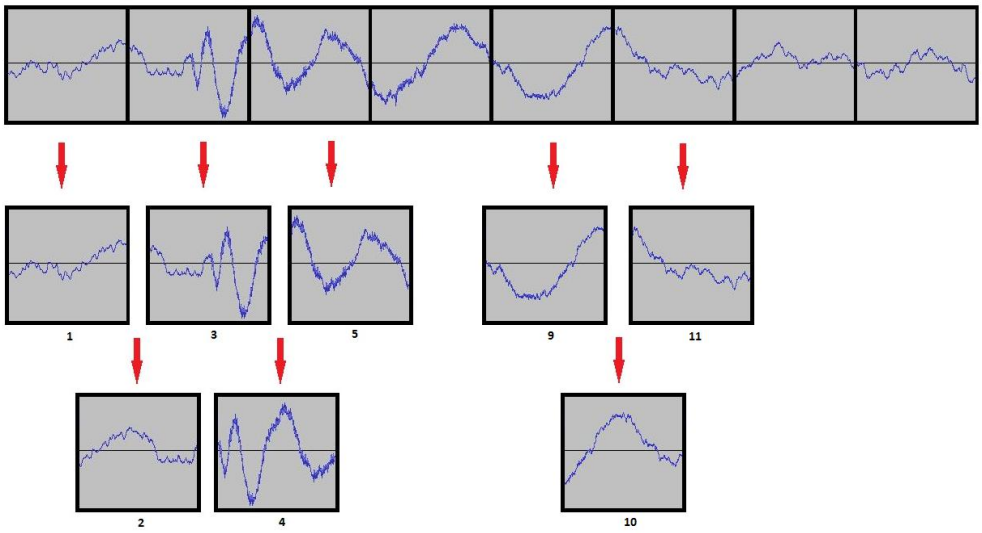


Figure 19 Capturing interesting points with window hops

Now here is an example of a windowed approach to reading audio samples. In Figure 19 the top row of squares illustrate how the original frame would have been read without any windowing, no different from Figure 18. The second row of squares is also no different from the squares in Figure 18, as it illustrates how frames appear when they are split as before. The third row of squares however represents frames that overlap; the numbers under each square indicate the order in which frames are read implying that even numbered frames indicate overlapped frames. Let us look closely at the first 3 frames of the audio in Figure 19, frame number 1 and frame number 3 are supposedly related to each other in that frame number 3 follows from frame number 1 in terms of chronological order in the audio stream. However, since we are now reading frames with an overlap we capture an extra frame in between frame number 1 and frame number 3, and that is frame number 2. Frame number 2 contains some overlapped samples since half of it is captured in frame number 1 and half of is captured in frame number 3. Originally the audio analyzer would not have known anything of interest might

have been happening right in between frame 1 and frame 3, however now that we have obtained frame 2 which overlaps frame 1 and frame 3 it becomes clear that there was in fact some interesting peak right in between frame 1 and frame 3 which would have never been uncovered without employing this windowed approach.

4.7 Audio Corpus

An audio corpus is a collection of audio files that can be used for tasks such as audio analysis. In our case we use an audio corpus to extract audio features from every song in the collection and use the extracted feature values to build statistical models. The audio corpus/corpuses required for vocal detection should contain audio files of both vocals and non-vocals so that a statistical model can be built to recognize both; the ideal would be to include songs of various types for a greater coverage as this will increase the likelihood of a randomly chosen song to be segmented correctly.

4.8 Issues

This method of segmenting songs is relatively lightweight in that most of the hard work is already manually pre-prepared offline, while for the end user the only work required on their device is the one-off calculation of LFPC values and classification of those values. However, as with most solutions there are always some downfalls or areas that prevent it from being a perfect solution. The following are several known issues for detecting vocal segments using statistical models.

Lack of Song coverage

The main issue with detecting vocal segments of songs containing accompaniment noise through the use of statistical models is that different songs can have different background accompaniment (perhaps different energy distribution). Therefore, if in the extreme case where a classification model trains mainly on songs with heavy background drum instruments and attempts to classify a song with only violin background instruments it is likely to classify with lower

accuracy. Also the same can be said about vocals where songs are trained on strong screaming vocals and attempts to classify a song with weak whispering vocals. The ideal situation would be to have coverage for all songs that ever existed, however that would be impractical to achieve and can also hinder classification efficiency as the model grows beyond proportion.

Human error

As previously discussed human error is almost unpreventable when it comes to manual annotation of audio segments. The manual annotation stage of the whole vocal detection process is perhaps one of the most important steps as without this type of data model training cannot take place, however, due to human's being error prone for several reasons this step is also the one with the most critical drawbacks. Providing erroneous training data to the classifier inevitably produces erroneous results. However, it is impossible for humans to be perfect especially when reacting to degrees of precision in milliseconds, therefore providing imperfect training data via manual annotations of audio remain a critical disadvantage factor.

Not enough non-vocal instances

Similar to the issue of the lack of song coverage, it is sometimes overlooked as an issue. Typically there are higher proportions of vocals compared to proportions of non-vocals due to the nature of song composition; this means that naturally there are significantly less non-vocal examples for a classifier to train on. Less training data typically translates to less accurate classification and less accurate classification is obviously undesired. One approach to solving this problem is to include songs composed of only instrumentals; however, obtaining these types of audio files are not always as straightforward.

5 Our Approach

There have been various studies in the field of vocal detection in songs using statistical models. However none have specifically taken advantage of information available about the song being analyzed such as the artist name and the gender of the artist. Most adopt the “one-model-predicts-all” approach and do not consider refining the prediction model to more specialized models. Knowing the artist name of a song provides possibilities of predicting segments of a song into vocals or non-vocal using a model specifically built to predict songs by that particular artist. Similarly, if the gender of the artist is known through knowing the name of the artist a model built to predict a specific gender can be used in the case that an artist specific model is not available. Obviously if a specialized model is not available or the information provided by the song name is not clear about the artist or gender we can always fall back onto a general model that has been trained on various types of songs, effectively performing vocal detection the conventional “one-model-predicts-all” way. The aim however is to investigate the advantages of using several specialized statistical models for prediction rather than using simply one general model. If we can get special built models to classify segments of songs according its artist or artist gender more accurately than with a general built model, we will have higher probabilities of aligning textual lyrics of a song to its vocal segments with more accurate timing.

In this chapter we will discuss the process of vocal detection through firsthand experiences including the corpus that was used for baseline evaluation, training data preparation, model training, classification of song segments, and the interpretation of classification results.

5.1 The baseline Corpus

For our statistical model training purposes a fairly large corpus was compiled, this corpus included many songs used in a study done by Ewald Peiszer on the topic of

"Automatic Audio Segmentation". This was one of the larger publically published corpuses we have seen and we also made many additions to the list making it an even larger corpus. It consists of

- 17 songs by *The Beatles*
- 4 songs by *Belle and Sebastian*
- 4 songs by Bettie Serveert
- 4 songs by *Teenage Fanclub*
- 2 songs by *ABBA*
- 2 songs by *Anthony and the Johnsons*
- 2 songs by *Arctic Monkeys*
- 2 songs by *Boyz II Men*
- 2 songs by *Badly Drawn Boy*
- 2 songs by *Baby Face*
- 2 songs by *The Jesus and Mary Chain*
- 2 songs by *The Roots*
- 2 songs by *The Stone Roses*
- 57 songs by various different artists

(Refer to the Appendix for a complete list of song names)

However the corpus being large has no value to us unless they contain information regarding the whereabouts of vocals and non-vocals for every song. Each song in the corpus was manually annotated to indicate where vocals/non-vocals occurs. This corpus will be used as the baseline corpus, in other words it is the corpus that will be used to train a general model for predicting a wide range of songs. In a later chapter we will introduce several specially compiled corpuses that will be used for evaluations against our propositions.

5.2 Manual Annotation

This process of is one of the most important steps to vocal detection via statistical models. It provides example audio feature values of segments of audio that are definitely vocals and segments that are definitely non-vocals. Having many such examples allows a model to train and consequently predict with higher accuracy. There is no automatic way to provide examples other than to prepare them manually. Below is a snapshot (Figure 20) of an application which demonstrates how any given song can be manually annotated to give indicators of where vocals and non-vocals occur.

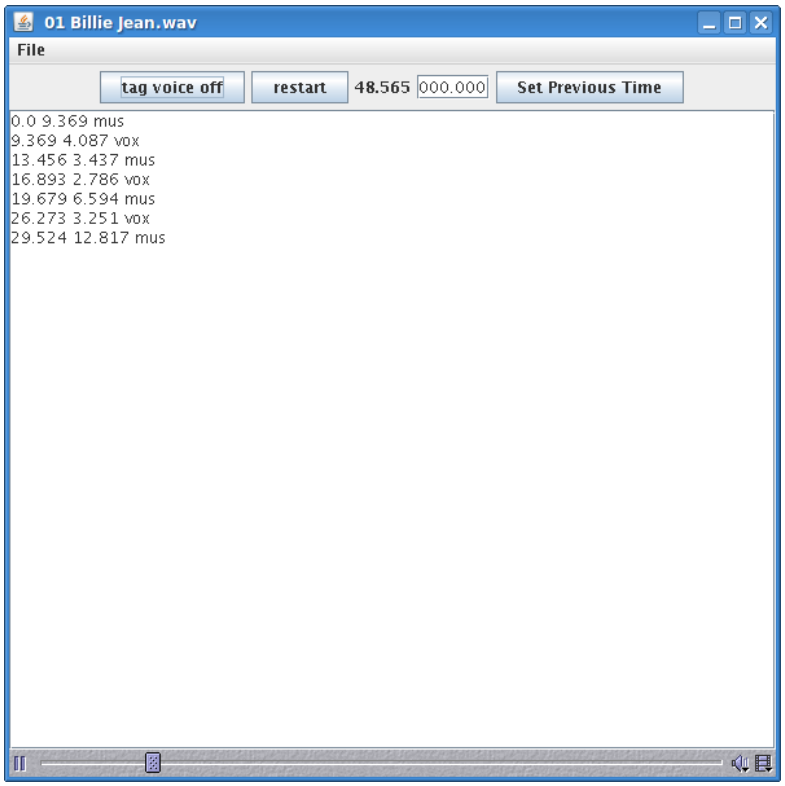


Figure 20 Manual song annotator

The application was specially developed to help ourselves with manual annotation. It contains a File menu which allows users to browse and open wav files. Once a wav file has been opened successfully it will immediately begin playback. The user then has several options; either, click the 'tag voice on' button if the beginning of vocals has been heard or click the 'restart' button which will

replay the song from the beginning and clear any output seen on screen. The 'restart' button is especially useful when the first vocal segment had been missed and a quick restart is required. Assuming the 'tag voice on' button had been clicked, the label on the button will change to display 'tag voice off' which for obviously reason allows the user to click the button to end tagging of a segment of vocals if in fact that segment of vocals had ended. There are other less important options such as dragging the playback bar for fast-forwarding or rewinding. Figure 20 illustrates that:

- The song "01 Billie Jean.wav" has been opened
- From 0 seconds non-vocals was tagged (denoted by 'mus') which lasts for 9.369 seconds
- From 9.369 seconds vocals was tagged (denoted by 'vox') which lasts for 4.087 seconds
- The current playback position is at 48.565 seconds
- Currently vocals are happening (application is waiting for a 'tag voice off' button press)

If the 'tag voice on' button is clicked, it will output some values to the text window regarding the amount of non-vocals it had just annotated. For example, from the absolute beginning of the song the user waited for 9.369 seconds before clicking the 'tag voice on' button, this resulted in the output of "0.0 9.369 mus" which in simple terms means from 0.0 seconds some non-vocals lasted for 9.369 seconds. At this point the user waited another 4.087 seconds before clicking on the 'tag voice off' button which then outputs the values "9.369 4.087 vox". If for some reason the user does not react by clicking anything during a vocal/non-vocal transition, they can simply use the playback bar to rewind. If the user makes a mistake by either clicking the 'tag voice on' or 'tag voice off' button too early or too late they have the option to delete the incorrect output and use the 'set previous time' button coupled with the text box to the left of it to set the time of the last correct annotation, then use the playback bar to rewind and do over. Once

a song has been fully annotated the File menu allows for saving to a *.lab file which contains the exact information displayed in the output text area.

5.3 Generating Training Data

In order to generate training data for a statistical model to begin training, LFPC values are required to be calculated. With all the LFPC values calculated the next step is to associate each LFPC value with a vocal (vox) or non-vocal (mus) tag. To accomplish this task, a separate [1]application was developed which first calculates LFPC values for each song in the corpus then searches for its corresponding manually annotated data (*.lab) file to create another file which contains LFPC values and its corresponding mus/vox tag.

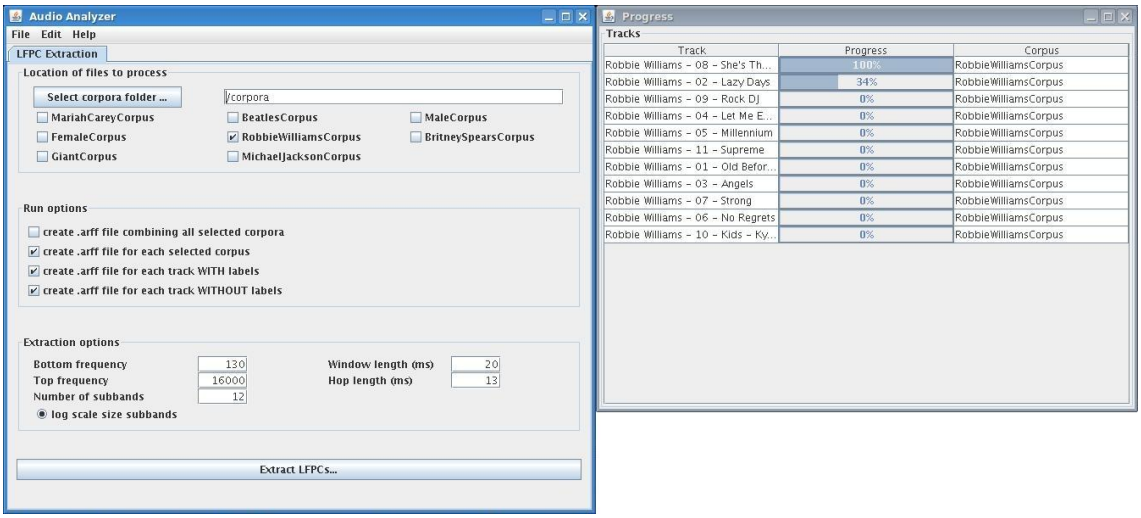


Figure 21 LFPC/training data generator

Figure 21 illustrates the application calculating LFPC values for the song named "Lazy Days". What the user does not see happening though is that after calculating the LFPC values of a song it also searches for the corresponding *.lab file and associates the LFPC values with mus/vox tags. The result of calculating LFPC values of the songs in the entire chosen folder is one *.arff file containing

[1] Assoc Prof Steve Jones was the developer of the entire application

all the LFPC values and their mus/vox associations. Below is a sample of the contents that can be found within the automatically generated *.arff file.

127.706,118.030,118.474,110.820,128.848,110.864,112.030,135.242,109.386,126.381,107.864,129.477,vox

% /home/user/Corpus/training/audio/Desmond Dekker – Lazy Days

This would have been derived from first calculating the first frame (assuming a 100 ms frame size) of 12 LFPC values and then looking into the *.lab file to find for example that from 0 seconds to 7.639 seconds (see below for sample annotation) there were vocals, therefore resulting in the first frame (0 seconds to 0.1 seconds) of audio analysed being associated with vocals (vox).The above training data sample is one of hundreds or even thousands of lines contained in an *.arff file, it is simply constructed by 12 LFPC values and followed by either a vox or mus tag followed by the song's full path (which is commented out by the preceding %). This is sufficient information for statistical models to use for training.

Sample annotation:

0.0 7.639 vox
7.639 16.417 mus
24.056 0.650 vox
24.706 3.030 mus
27.736 1.428 vox
29.164 1.765 mus
30.929 2.368 vox
33.297 17.763 mus
51.061 1.022 vox

5.4 Training

Once relevant data has been generated, the next step is to use that data to begin training a statistical model. This trained model will be used to predict and classify any LFPC values that it is given, in other words if a whole series of LFPC values calculated for a song were given to the model it shall be responsible for sorting each of those values into either vocals or non-vocals in chronological order. For this training process we relied heavily on "WEKA", an application developed by several senior professors at the University Of Waikato in New Zealand [12]. This

application is a compilation of algorithms developed for machine learning and data mining tasks and can be used for both training models and classifying raw data . Figure 22 is a snapshot of the interface of WEKA, the ‘Pre-process’ and the ‘Classify’ tabs contain all functions that is required for our purposes.

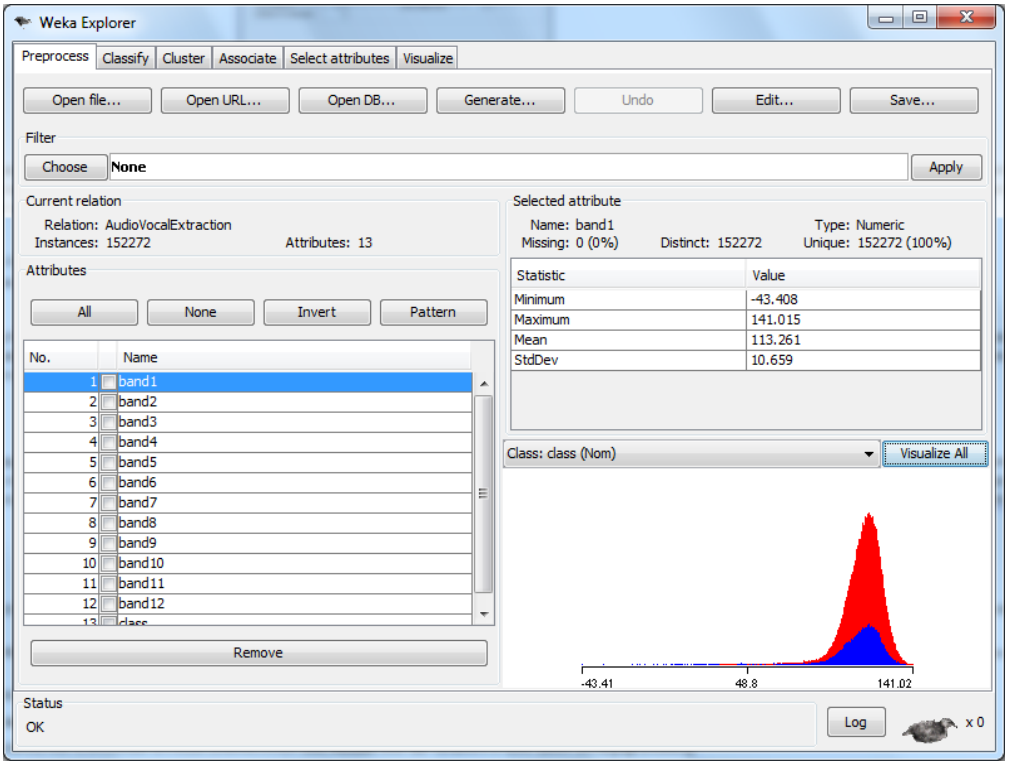


Figure 22 WEKA, data mining application

The ‘Pre-process’ tab allows users to load training data (*.arff file) similar to the ones discussed in the previous section 5.3 Generating Training Data, once a file is loaded on the same tab several types of basic information is displayed including basic statistics regarding the data’s distribution and the number of instances the training data contains. As a word of advice, even though generally the more instances there are the higher the chances a wider range of values is covered it is not always a smarter idea to increase that number is much as possible due to the fact that more data implies more processing power required which implies less efficiency as well will see later. Assuming we have loaded some training data into WEKA, the next step is to train a model using that data. To do this, we use the ‘Classify’ tab. Figure 23 illustrates the end result of a model being built/trained.

The percentages 74.4867% and 25.5133% represent the correctly and incorrectly classified instances respectively; these numbers are derived by using a 10 fold cross-validation technique where the training data is divided into 10 equal parts, 9 out of the 10 equal parts are used for training and the remaining 1 out of 10 equal parts is used for testing, this is then alternated and repeated until each of the 10 parts has had the chance to be part of the training data as well as being the testing data set and the overall accuracy is recorded.

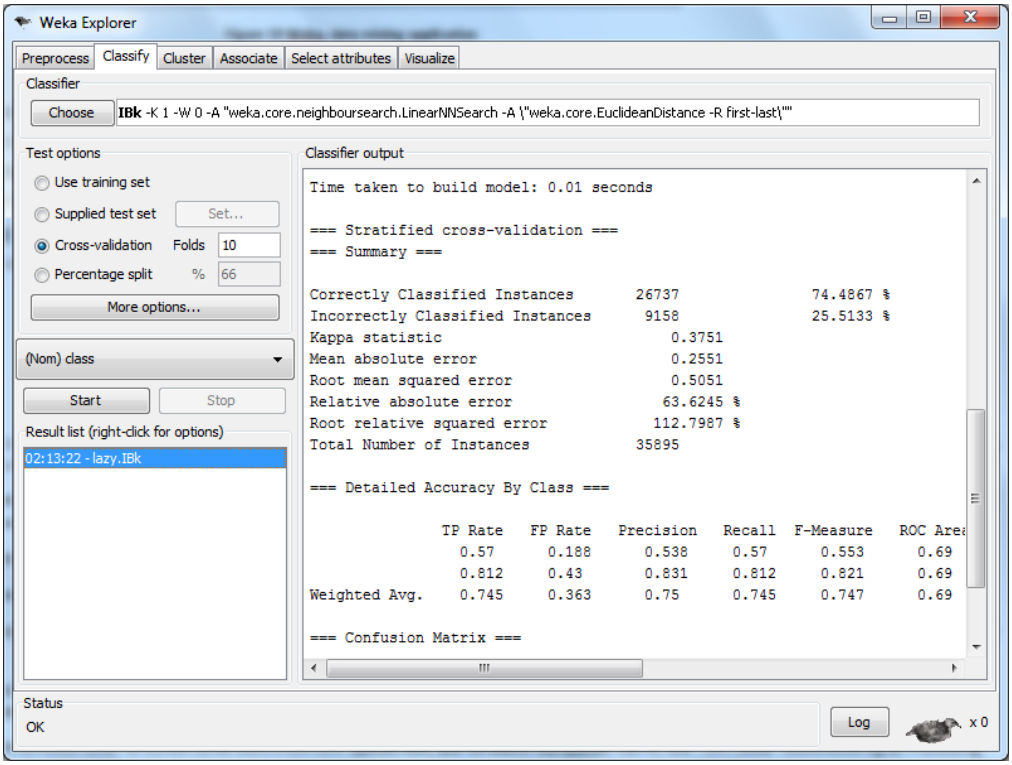


Figure 23 WEKA, classification tab

How accurate a model is at classifying instances can be known by verifying with the "answers" in the test data. In other words, think of the LFPC values in the training data as questions for a test and the two classes 'mus' and 'vox' are the possible answers to those questions and the test itself is the classification process, when we give the model a test to answer it can temporarily cover up the answers and attempt to answer the questions without looking at the answers, once it has finished answering it will verify its results with the correct answers and obtain a accuracy figure. Take the following example for instance:

15:mus

18:mus

22:vox

25:vox

15:mus

20:vox

The numbers here shall represent LFPC values, next to them are their associated classes or tags. The first four numbers shall be the training data. The first two numbers have ‘mus’ associated with them, the following pair of numbers have ‘vox’ associated with them. The remaining two numbers shall become the test set, it covers up the “answers” as described previously and they become:

15:?

20:?

Now testing begins. Since the model has already seen that the number 15 is associated with ‘mus’ it gives the following correct prediction for the first test instance:

15:mus

Now it looks at the number 20 and attempts to give it an answer, however because it has never seen examples of 20 anywhere in the training data so it takes a guess and predicts its class incorrectly as:

20:mus

Once all predictions have been completed the system verifies its predictions by uncovering answers to give an accuracy score. There were two instances tested, since the prediction of the first instance was correct and the prediction of the second instance was incorrect it concludes that this model is approximately 50% accurate. In reality it is more complicated, however this example provides the basic idea of what happens when the training data is divided for testing. The accuracy of a model can be increased (or decreased) by using different classifiers, each classifier has a different algorithm for classifying instances, and

consequently this means that some algorithms are better than others in different situations depending on the type of data it is presented with.

5.5 Classification

The ultimate goal is to classify instances into vocals or non-vocals; therefore the classification stage is perhaps one of or if not the most important stage. With instances classified, it is possible to begin observing its accuracy on various types of unseen data and ultimately allows for segments of vocals of a song to be located (with varying degrees of accuracy and precision).

WEKA has the capability of allowing users to load a trained model and provide to it an external source of instances (LFPC values of some given song) for classification. This external source of instances can either have classes already defined or undefined. The advantage of using instances with classes already defined for classification is that it provides immediate feedback of its accurateness by comparing between predicted classes and actual defined classes and this can be helpful for understanding which types of models classify more accurately with which types of songs (LFPC values), the downfall is that only gives indication of how accurate the model is at predicting classes of a particular song which already has known classes (segments of vocals already known) and does not provide any further assistance toward vocal detection of unknown songs. Figure 24 illustrates the classification process in Weka where an externally supplied test set is used for classification instead of cross-validating among the original data supplied for training.

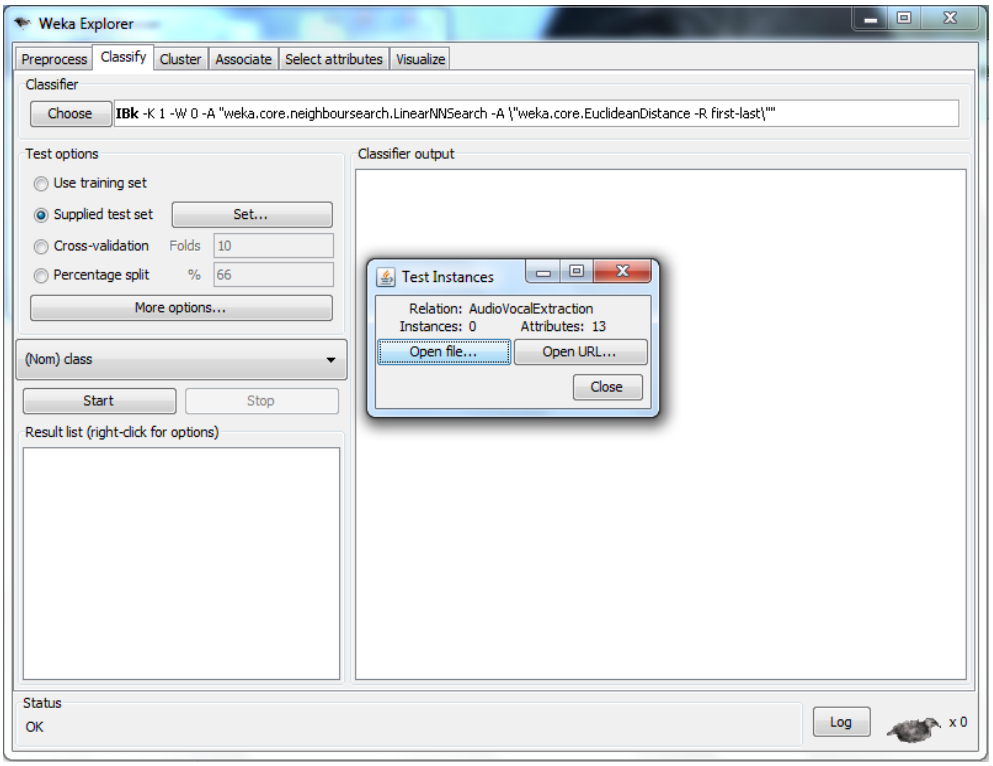


Figure 24 Classification via supplied test set

What is more interesting to know however, is how accurate a model can classify instances of a song that has had no manual annotations (defined classes) attached, in other words how accurate can the model segment a given song that it has never been seen before into vocals and non-vocals. How accurate a model can classify raw instances that do not have classes defined is however somewhat up to human perception as there exists no "answers" to compare to and the only means of knowing is to visualise the predictions on a chronological time scale while playing back the song and verifying that the cues of start and end points of vocals segments are in fact in sync with the playback.

The steps required to obtain a visualisation of the predicted instances are as follows:

1. Build a model with some training data
2. Classify new instances into vocals or non-vocals
3. Transform the classification output to replace all occurrences of vocals by the number 1 and all occurrences of non-vocals by -1
4. Plot a histogram of all the values of 1s and -1s

By replacing all occurrences of vocals by 1 and non-vocals by -1 we are able to gain a numerical representation of the two different classes in their extremities, these numerical values will then be used to visualise the boundaries between vocals and non-vocals. The following is a sample output (in table format) of the first 10 instances classified obtained from classifying instances of a song through WEKA:

=== Predictions on test data ===				
inst#	actual	predicted	error	prediction
1	1:mus	1:mus		1
2	1:mus	1:mus		1
3	1:mus	1:mus		1
4	2:vox	2:vox		1
5	2:vox	2:vox		1
6	1:mus	2:vox	+	1
7	2:vox	2:vox		1
8	2:vox	2:vox		1
9	2:vox	2:vox		1
10	1:mus	2:vox	+	1

Table 5 Results of predicting the first 10 instances

Table 5 shows that the first 3 instances were predicted as mus (non-vocals) followed by 2 instances predicted as vox (vocals) followed by 1 mus 3 vox then finally 1 mus. Table 6 illustrates how each of the class values are transformed into numerical values where - 1 represents non-vocals and 1 represents vocals.

instance#	actual class	predicted class	actual class value	predicted class value
1	mus	mus	-1	-1
2	mus	mus	-1	-1
3	mus	mus	-1	-1
4	musvox	musvox	1	1
5	musvox	musvox	1	1
6	musvox	mus	1	-1
7	musvox	musvox	1	1
8	musvox	musvox	1	1
9	musvox	musvox	1	1
10	musvox	mus	1	-1

Table 6 Predicted class values transformed into numerical values

Figure 25 is a histogram of instance classes plotted in the song's chronological order. Note this example only represents the first 10 instances (first few seconds) of a song and already we can see some sections of vocals present. Assuming that each bar represents 1 second of audio, we can conclude that this diagram tells us that the model has predicted the first 3 seconds of the song as being non-vocals followed by 2 seconds of vocals and so on. We can see that vocals began at the 4 second mark and ends at the 6 second mark, in addition to that section of vocals another can be observed that begins at the 7 second mark and ends at the 10 second mark. With this visual, we are gain a rough sense of where vocals occur within a song.

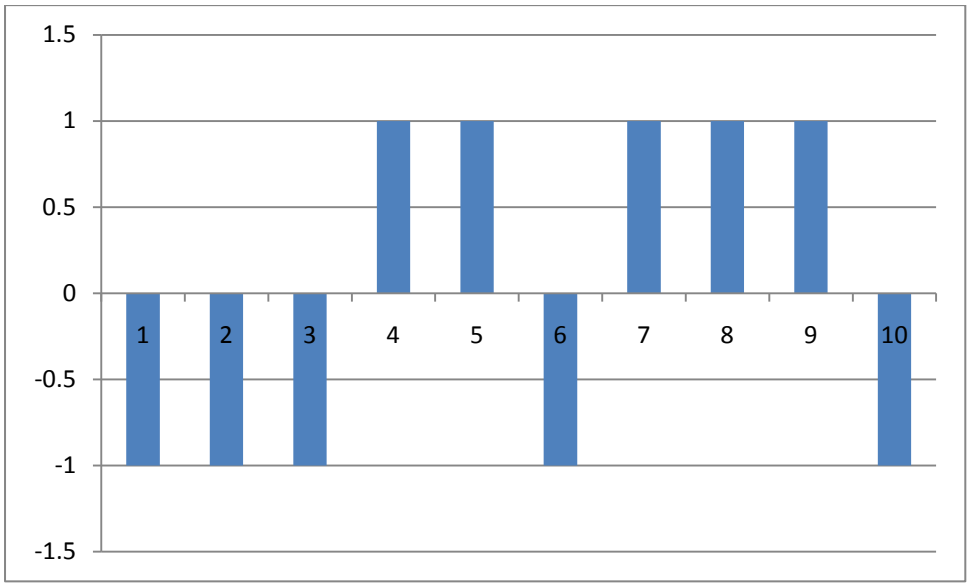


Figure 25 Visualisation of predicted classes (histogram)

Vocal start and end points can be defined as the change from one extreme to the other or more specifically the change in value from 1 to -1 or vice versa. Figure 25 shows that the first 3 values remained the same until the fourth where the value changed from -1 to 1, this indicates the beginning of a vocal section, the change from 1 to -1 at the sixth value indicates the end of a vocal section. Therefore the first section of vocals begins at 4 seconds and ends at 6 seconds as expected. Although this technique can be useful for locating vocals, it must be noted that not all changes in values indicate presence of vocals or non-vocals. Bear in mind that the classification process is almost never 100% accurate, therefore it is highly likely that there are incorrectly predicted values among some section of vocals or non-vocals. Take Figure 25 for example, at first sight it may seem as if vocals begin at 4 seconds and end at 6 seconds, however looking at the bigger picture it can appear as if the section of vocals actually begin at 4 seconds and end at 10 seconds with one misclassification in between at 6 seconds. The likelihood of one instance with a different class among a series of instances of the opposite class being correctly classified is rather low. Note that each bar in Figure 25 could just as easily represent 100 milliseconds of audio in which case it would mean that the figure is suggesting that some vocals lasted for 200 milliseconds followed by 100 milliseconds of non-vocals followed by 300 milliseconds of vocals, intuitively

this does not give the impression that it predicted the one non-vocal segment correctly as it would appear more likely that it was actually a 600 millisecond segment of vocals without a minuscule pause.

5.6 Moving Average

To reduce the issue of misclassifications, we can simply adopt a moving average technique where values between a certain range is averaged to give an overall trend and eliminate short term fluctuations, this then gives us a overview of a longer term trend of the data set. The basic idea behind the moving average technique is to select a window size, average all the values within that window and move to the next window and average those values until it has iterated through the whole data set. Take Figure 25 once again as an example; if we chose a window size of 3, we would look 1 value behind and 1 value ahead along with the current value and average the total to replace the current value by the average. The following is a table similar to that of Table 6 with the addition of moving average values (window value):

instance#	actual class	predicted class	actual class value	predicted class value	window value
1	mus	mus	-1	-1	
2	mus	mus	-1	-1	-1.00
3	mus	mus	-1	-1	-0.33
4	musvox	musvox	1	1	0.33
5	musvox	musvox	1	1	0.33
6	musvox	mus	1	-1	0.33
7	musvox	musvox	1	1	0.33
8	musvox	musvox	1	1	1.00
9	musvox	musvox	1	1	0.33
10	musvox	mus	1	-1	

Table 7 Predicted class values with moving average

If the chosen window size is 3, we would begin calculating from the second value since this is first value that has values before and after it and this allows it to be

averaged by the 3 combined values (including the current value). Let us take the predicted class values from Table 7 for example, the predicted class values of the first 3 instances are all -1, therefore we begin with the second -1 and sum it with both values before and after it resulting in -3. We then average this value to give a value of -1; this value then replaces the current value which happens to be -1 anyway. We then move onto the next predicted class value in the series, this time the current value is -1 with a -1 value before it and a +1 value after which results in an average of -0.33. The process continues until no more values remain. Immediately we can see (Figure 26) that the sixth value that seemed to be incorrectly classified has been smoothed out and became a positive value instead of being a negative.

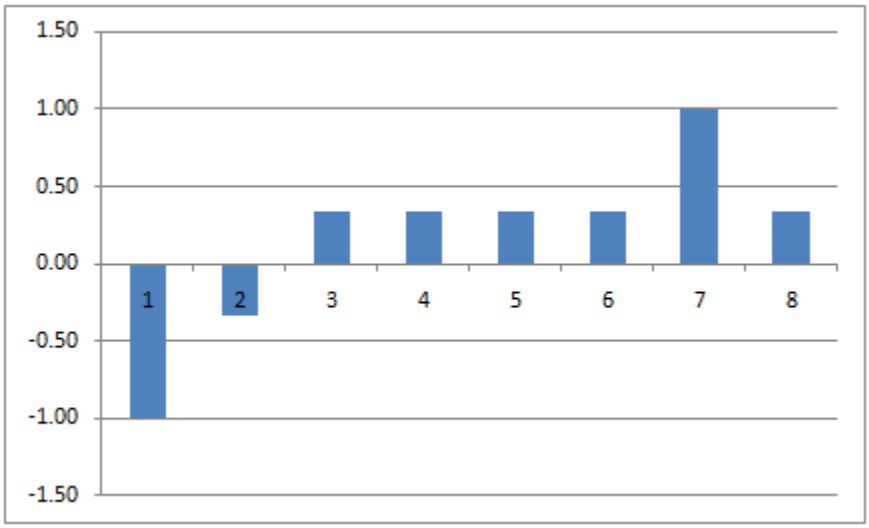


Figure 26 Corrected class values using a simple moving average

Bear in mind that the above examples demonstrate basic idea behind the simple moving average technique, in reality there are several variations in calculating a moving average. The formula for a simple moving average is as follows:

$$SMA = \frac{p_M + p_{M-1} + \cdots + p_{M-9}}{10}$$

Where $p_M, p_{M-1}, \cdots, p_{M-9}$ are the observed values, i.e. the positive and negative values that were transformed from mus and vox. The number 10 represents the window size and can be substituted by any number. *M* is the window size

There are also moving averages that take into account the current value's importance, in other words giving more weight to the current value. Typically some form of normal distribution or linear decay is applied to all included values in the window where the current value is the middle value and the further away the neighboring values are the less they weigh against the average. Table 8 is an example of a weighted moving average where the window size is 3 and both values before and after the current value are multiplied by some ratio while the current value is simply multiplied by 1.

instance#	actual class	predicted class	actual class value	predicted class value	window value
1	mus	mus	-1	-1	
2	mus	mus	-1	-1	-0.50536
3	mus	mus	-1	-1	-0.50536
4	musvox	musvox	1	1	0.49364
5	musvox	musvox	1	1	0.50536
6	musvox	mus	1	-1	-0.49364
7	musvox	musvox	1	1	0.49364
8	musvox	musvox	1	1	0.50536
9	musvox	musvox	1	1	0.50536

Table 8 Moving average with weighted values

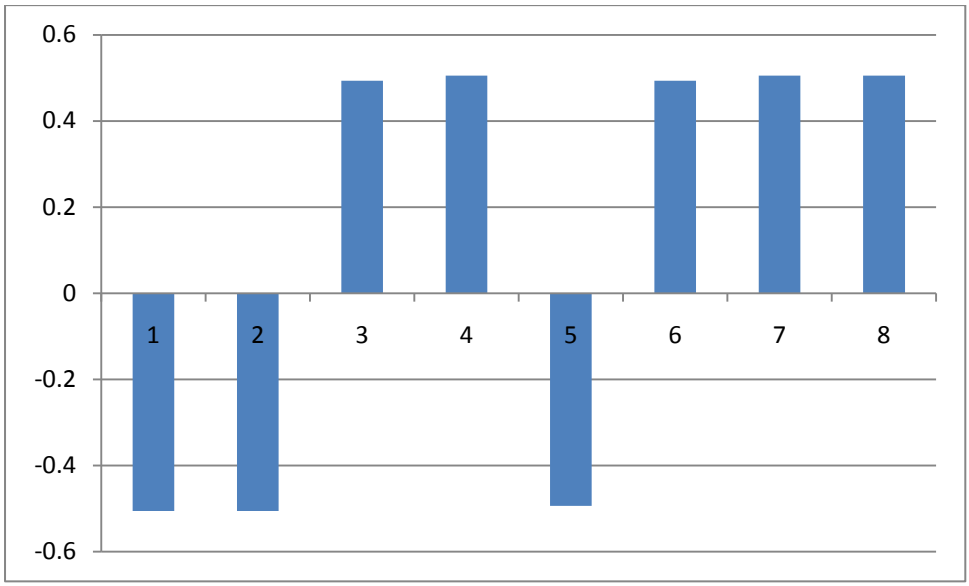


Figure 27 Corrected class values using a weighted moving average

6 Evaluations

As discussed at the beginning of the previous chapter the aim is to investigate the advantages of using several specialized statistical models for prediction rather than simply using one general model. If we can get special built models to classify segments of songs according its artist or artist gender more accurately than with a general built model, we will have higher probabilities of aligning textual lyrics of a song to its vocal segments correctly. The proposition is that similar audio feature values can be found between songs that are similar, perhaps songs with the same artist or songs that contain dominant vocals of the same gender. Since different artists have relatively different energy levels in their vocals and certainly vocals between males and females generally have noticeable differences we expect that prediction will be more accurate with the use of models specifically built for vocals of a particular gender or artist. This approach allows the field to be narrowed down significantly meaning potentially redundant data is not considered in the classification process, in other words predicting a song containing only female vocals with a model built for predicting songs of both male and female means that half the data (males) in that model is potentially useless. The remaining of this chapter will cover the experimental process through which we investigate whether vocal detection using special built models depending on the song type provide supported evidence of improved classification accuracy

6.1 Experimental Setup

In order to investigate the validity of the proposition, a series of tests was devised to help support this. This series of tests included compiling several corpuses in addition to the large corpus already introduced in chapter 5.1 The baseline Corpus) for feature extraction, model training and data classification. The following is an outline of the experimental process:

1. Compile a general corpus containing various types of songs (as seen in chapter 5.1 The baseline Corpus)
2. Compile additional corpuses for:

I. male only songs
II. female only songs
III. two different male artists
IV. two different female artist
V. one group or band

3. Prepare training data
4. Build models for all corpuses
5. Cross-validate instances from all corpuses against each other's models
6. Visualize predictions
7. Analyze results

Compile a general corpus containing various types of songs

This corpus will serve as a baseline for testing. The general corpus contains many different types of songs across several genres, artists and genders. Results that are obtained for tests obtained from other models should be compared to results from testing with the model of the general corpus. For example, if a Mariah Carey song is tested on a Mariah Carey model the result should be compared to testing the same Mariah Carey song on the general model. This shall then provide some indication of any advantages of using specialized built models. The entire list of songs compiled for the general corpus can be found in the appendix section.

Additional corpuses

These additional corpuses will be used for verifying the validity of the proposition that special built statistical models should outperform a general model. Models will be built out of each of these corpuses for examination. The addition corpuses consist the following:

Male corpus, Female corpus, Mariah Carey corpus, Britney Spears, Michael Jackson corpus, Robbie Williams and Beatles corpus

Male Corpus

Artist	Song name
Bow-wow (feat Omarion)	Let me hold you
Boyz II Men	I'll make love to you
Aerosmith	I don't want to miss a thing
Beatles	You really got a hold on me
Blink 182	What's my age again
Coolio	The devil is dope
David Bowie	Kooks
Eminem (feat Dido)	Stan
Immortal Technique	Dance with the devil
K-Ci and JoJo	Crazy
Kc & The Sunshine Band	That's the way I like it
Michael Jackson	Billie Jean
Oasis	Songbird
Prince	Kiss
Red Hot Chili Peppers	Parallel Universe

Female Corpus

Artist	Song name
Leona Lewis	Bleeding love
Alicia Keys	No one
Alanis Morissette	Head over feet
Alanis Morissette	Thank you
Bjork	It's oh so quite
Britney Spears	Hit me baby one more time
Britney Spears	Oops I did it again
Cassie	Me and you
Gloria Gayner	I will survive
Madonna	Into the groove
Madonna	Like a virgin
Mariah Carey	One sweet day
Monica	Angel of mine
Norah Jones	Lonestar
Salt-N-Pepa	Whatta Man

Mariah Carey Corpus (female artist)

Song name
Sweetheart
When you believe
Whenever you call
My all
Always be my baby

One sweet day
Fantasy
Hero
Deamlover
I'll be there
Someday
Love takes time
I still believe
Without you
Do you know where you're going to

Michael Jackson Corpus (male artist)

Song name
Billie Jean
Scream
The way you make me feel
They don't care about us
Black or white
Stranger in Moscow
Rock with you
This time around
Bad
D.S
Man in the mirror
You are not alone
Beat it
Bad
Heal the world

Beatles Corpus (group)

Song name
A day in the life
All I've got to do
All my loving
Anna
Don't bother me
I saw her standing there
It won't be long
I wanna be your man
Lucy in the sky with diamonds
Misery
Money
Please Mr. Postman
Please please me
Till there was you

Prepare training data

To begin with we must manually annotate each song in each corpus. This process of manual annotation is tedious yet rewarding. The entire training and classification stages of vocal detection rely on it, therefore producing quality annotations is important. In this section we will not be discussing this process further; every single song listed in the previous section was manually annotated using the method discussed in a previous chapter on manual annotation.

Using data from the manual annotations, we then generate training data by calculating LFPC feature values for all songs and associate class values (vocals/non-vocals) to every feature value. This will result in a *.arff file containing training data.

Build models for all corpuses

We will build statistical models for each of the listed corpuses for evaluation. Each corpus contains approximately 10 to 15 songs and one *.arff file is created for each corpus containing the necessary training data. This *.arff file is then loaded into WEKA for training and its overall accuracy is evaluated using automatic cross-validation. There are two classifiers that we consider the best for our aim and they are as follows:

- IBk
- RandomForest

Both classifiers produce very similar results, the IBk classifier was selected for our evaluation. The IBk classifier is based on the nearest neighbor approach where the test instance is classified by considering the classes of the nearest k training instances. This is especially suitable since high or low LFPC values determine the class of an instance; if the class value of a certain LFPC value is unknown then it looks to see what the classes are of the nearest k LFPC values, if

the nearest k LFPC values are high and mostly vocal classes then it is most sensible to classify the unknown instance as vocals.

To begin with, a random song was taken from the general corpus and its actual vocal sections were plotted using frame sizes and window hop of 20-13, 200-130, 400-260, 600-390, and 1200-780 milliseconds. The reason for this is to first settle on a frame size that can provide a balance between accuracy, precision and efficiency and to use this frame size for the remaining evaluations. As discussed earlier, the frame size can affect the accuracy and precision of vocal segments, here we will demonstrate this visually in the following diagram:

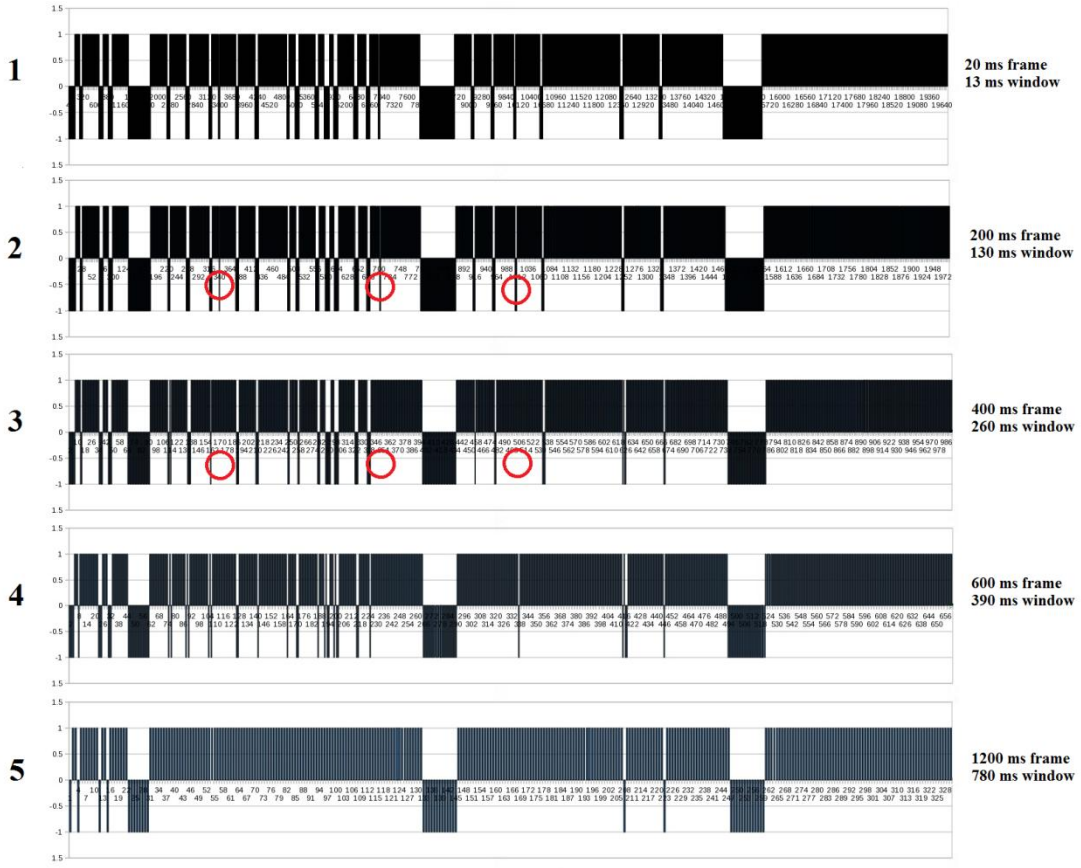
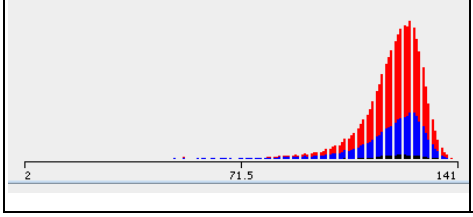
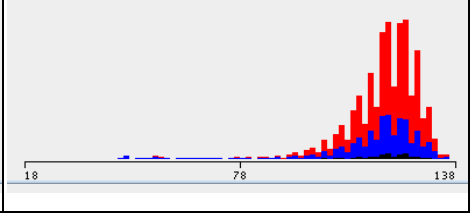


Figure 28 Plot of actual vocal segments of a song with different frame sizes

We first extracted LFPC values of the chosen song using the mentioned frame sizes and eventually producing an *.arff training data file for each frame size, then models were built using each of those *.arff files. The method for obtaining the actual (not predicted) plots is by simply "predicting" the song's LFPC values

contained in the *.arff training data file using the model built with the same *.arff training data file. This way the model will theoretically predict every instance 100% correct since every instance it attempts to predict is are also the training instances. We then transform each predicted class into positive or negative numerical values to obtain the plot of vocal segments in the time scale as seen in Figure 28. Now examining the diagram, we shall take the 1st plot which has frame sizes of 20 milliseconds as a baseline since it will be the most precise plot out of them all due to its small frame size. Here the 2nd plot compared to the 1st is very similar where all sections of vocals and non-vocals practically match exactly except that a few of the smaller non-vocal segments appear to be slightly different as if it is almost disappearing. If we look at the 3rd plot we can actually see that very small sections of non-vocals have disappeared, this is the affect that frame sizes have on precision as discussed in a previous chapter. It becomes obvious that precision is lost as frame sizes increase as illustrated by the 4th and 5th plots. From this analysis the conclusion is to use frame sizes of 200 milliseconds since it will be more efficient compared to 20 millisecond frames and it does not appear to lose a great deal precision, therefore the remaining evaluations will use frame sizes of 200 milliseconds.

Since all models that we build using WEKA will have information about its distribution of each of the two classes between vocals or non-vocals we shall represent each model with the visualization along with its cross-validation results. The following are snap shots of each model's LFPC value distribution followed by cross-validation results and basic statistics taken from WEKA representing each model:

General model	Beatles model
	

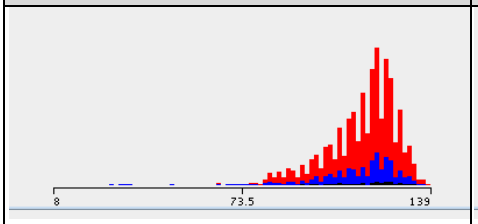
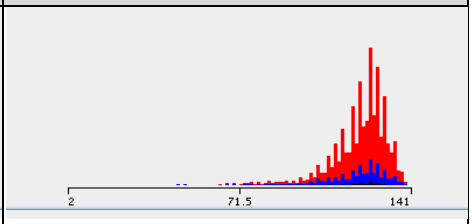
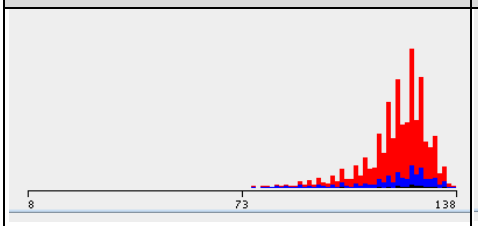
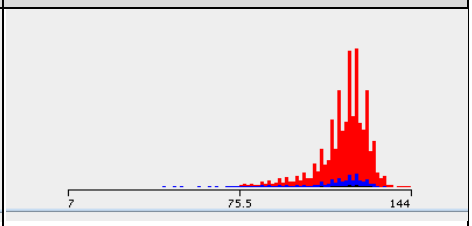
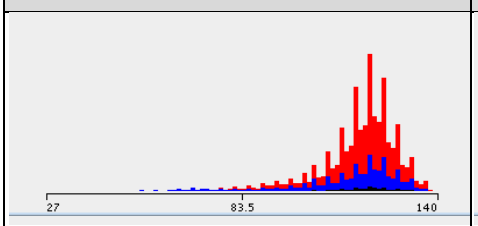
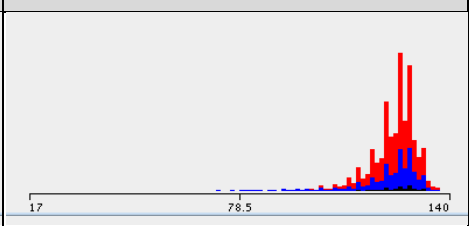
Female model	Male model
	
Britney Spears model	**Mariah Carey model**
	
Michael Jackson model	**Robbie Williams model**
	

Table 9 LFPC Distribution of every model

The percentage values in the following table represent the accuracy of cross-validating models against itself giving indication of how accurate the model is at classifying data that are similar. For example, the Britney Spears model is approximately 83.9015% accurate at classifying songs by Britney Spears. The remaining columns are self explanatory as they provide additional information regarding each model's LFPC value distribution.

	Cross-validation accuracy	Minimum LFCP value	Maximum LFPC value	Mean LFPC value	Standard deviation
General	73.8818%	2	141	119.781	11.508
Beatles	76.2083%	18	138	117.228	12.534
Britney Spears	83.9015%	8	138	119.457	11.129
Female	83.3127%	8	139	114.579	13.847
Male	84.2806%	2	141	119.742	12.683

Mariah Carey	91.612%	7	144	116.563	12.354
Michael Jackson	74.1698%	27	140	116.786	12.46
Robbie Williams	76.126%	17	140	122.465	10.91

Table 10 Cross-validation accuracy and distribution of each model

Cross-validate instances from all corpuses against each other's models

Every LFPC value from each corpus (except the general corpus) on the horizontal axis is tested against each model on the y axis and the accuracy is recorded. This will ensure that each corpus is tested against the general model in addition to testing each corpus against other focused models. Corpuses are not tested against their own corresponding models since theoretically it will always give 100%.

The percentage values represent how good a particular model is at classifying instances produced by some artist or by some artist of a certain gender. For example the General model is approximately 73.5705% accurate at classifying songs by Britney Spears or is approximately 67.1741% accurate at classifying songs by females.

	Beatles	**Britney Spears**	**Female**	**Male**	**Mariah Carey**	**Michael Jackson**	**Robbie Williams**
General	62.1313 %	73.5705 %	67.1741 %	67.4924 %	77.9397 %	63.2638 %	65.7742 %
Beatles		68.8231 %	65.2345 %	67.4386 %	69.0601 %	61.8875	63.7058 %
Britney Spears	58.4167 %		73.5762	67.9696 %	81.7949 %	66.4772 %	67.5188 %
Female	59.7656 %	78.9699 %		69.2939 %	81.0786 %	66.4588 %	67.5575 %
Male	69.0842 %	73.3516 %	71.1073 %		75.4608 %	66.8182 %	66.3541 %
Mariah Carey	63.7638 %	80.2588 %	76.1919 %	74.9908 %		70.4494 %	67.5575 %
Michael Jackson	57.5978 %	74.0781 %	67.7652 %	66.8403 %	78.0029 %		65.1218 %

Robbie Williams	56.4417 %	64.633 %	61.7418 %	61.0997 %	65.3313 %	59.5711 %	

Table 11 Cross-validation among all models

Visualize predictions

To visualize predicted vocal segments, one song from one corpus is used for illustration. First the song's actual vocal segments are visualized, then vocal segments are predicted using the general model in addition to using the song's own specialized model to illustrate any differences between actual vocal segments and predicted vocal segments. By doing so, we can observe the differences between using general models and specific models.

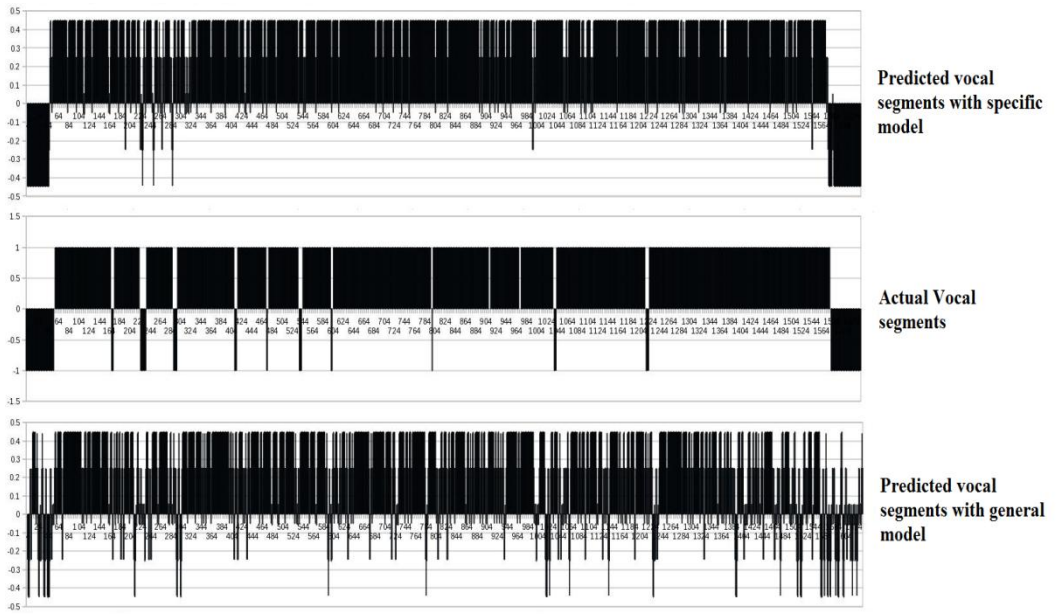


Figure 29 Actual vocal segments versus predicted vocal segments

Here in Figure 29 we can see obvious differences between actual vocal segments and predicted vocal segments. The plot toward the top of the diagram corresponds to predictions using a specific model while the plot toward the bottom corresponds to predictions using a general model. The plot in the centre is the baseline (actual vocal segments). Both predictions were processed using a weighted moving average size of 5 instances. The vocal detection precision and

accuracy of the specific model is significantly much better compared to the general model. The specific model used for this test was the Mariah Carey model and the song being tested was “Without you” by Mariah Carey. The general model classified 78.1875% of instances correctly where as the specific model classified 91.5625% of instances correctly. A difference of 13.375% does not seem to have much impact however when the predictions are visualised the impact is tremendous.

Analysis

Let us begin analysis by combining some crucial results from Table 10 with results from Table 11. The combined table allows us to compare the differences in accuracy between classifying songs with a general model and classifying songs with a specific model.

	Cross-validation accuracy	Accuracy against General model
General	73.8818%	100%
Beatles	76.2083%	62.1313%
Britney Spears	83.9015%	73.5705%
Female	83.3127%	67.1741%
Male	84.2806%	67.4924%
Mariah Carey	91.612%	77.9397%
Michael Jackson	74.1698%	63.2638%
Robbie Williams	76.126%	65.7742%

Table 12 General model versus specific model

Here we can see that the General model is on average 73.8818% accurate at classifying songs chosen by random since the general corpus mainly contains songs of many different types. Now comparing the Beatles model to the General model we can see that the Beatles model is on average 76.2083% accurate at classifying songs by the Beatles, however when using the General model to classify Beatles songs only a 62.131% accuracy was obtained. The Beatles model is significantly more accurate at classifying Beatles songs compared to a General

model attempting to classify Beatles songs, the difference is approximately14.077%.

If we compare how accurate a Britney Spears model is at classifying Britney Spears songs against how accurate a General model can classify the same Britney Spears songs, the General model again falls short this time by approximately 10.331%.

Iterating down the table we begin to see a trend, which is, every special built model is at least 10% more accurate at classifying songs of its own type compared to using a General model where the highest difference in advantage peaks at approximately 17%.

The following graph shows the accuracy between testing specific songs against the General model and testing specific songs against its own specific model.

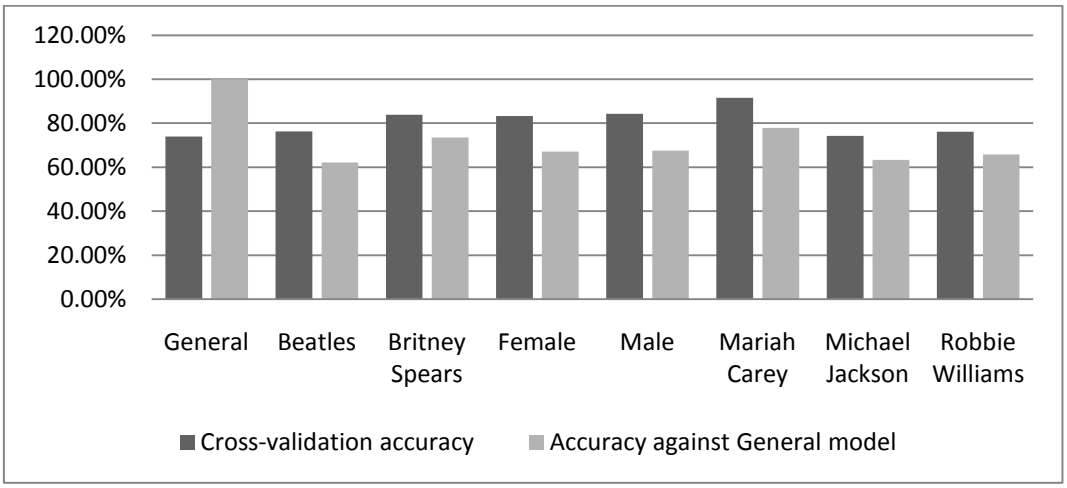


Figure 30 General model versus specific model

Note that the general model classifies 100% correctly since it is classifying the same songs that it used to build its model, therefore accuracy values should be considered from the Beatles model onward. All specific song types tested against the general model give convincingly lower accuracies when compared to testing the same songs against its own specific model. This seems highly promising, since the proposal was that specialised models should classify instances with higher

accuracy than using a general purpose model, this finding undoubtedly supports our assumptions.

From the percentage figures contained in Table 12 it is also interesting to note that the models that gained the largest advantages were the gender specific models. The male model gained an advantage of 16.79% closely followed by the female model with 16.14%. Perhaps this is some indication that separating models into male and female is on average the most effective.

One may naturally wonder why some models cross-validate better than others, the reason is that some models were built with instances that have larger LFPC (energy) differences between vocal and non-vocal segments and some models were simply built with vocal and non-vocal LFPC values closer together. This means that some songs have much higher LFPC values during vocals and much lower LFPC values during non-vocals where as some songs have lower LFPC values during vocals and some have higher LFPC values during non-vocals. In order for classification to be more accurate the larger the difference between LFPC values of vocals and non-vocals the better. Take Table 9 for example, if we examine the LFPC distribution of the General model we see that the energy levels of non-vocals (blue samples) are not significantly different from the energy levels of vocals (red samples). During training, the model attempts to learn the difference between energy levels of vocals and non-vocals, if the differences are not large it may be more difficult for the model to learn the difference in energy levels between the two classes. Now if we examine the LFPC distribution of the Mariah Carey model we can see clearly that the difference in energy levels between vocals and non-vocals are quite significant, hence the higher cross-validation accuracy of 91.612% compared to the General model's of 73.8818%.

Since General models typically contain many different types of songs, the energy levels between vocals and non-vocals become less consistent, this means that there are higher numbers of vocal instances having dissimilar LFPC values which makes it more difficult for the classifier to distinguish what a vocal segment's LFPC value should be.

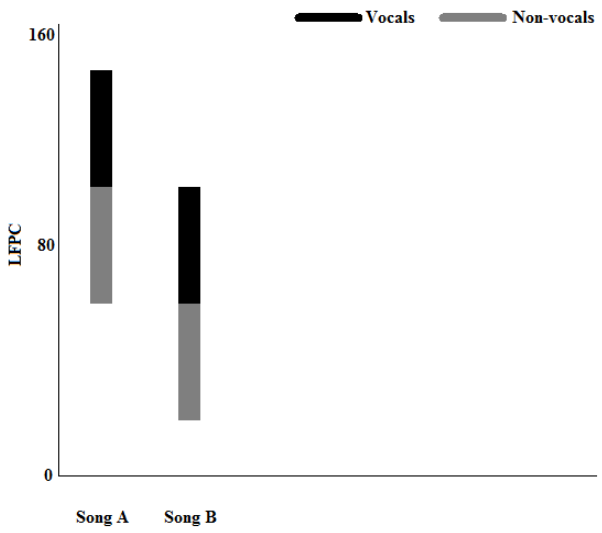


Figure 31 Difference in LFPC values for vocals

For example, a model is built using song A and song B (Figure 31) where song A contains higher LFPC values during vocals and song B contains relatively lower LFPC values during vocals. If an instance of unknown class was to be classified using this model it would be difficult to do it accurately, what would happen if the instance to be classified had a LFPC value of 80 is completely unpredictable. As far as the model knows the LFPC value of 80 is highly like to be non-vocals (according to song A) and is also highly likely to be vocals (according to song B).

The more variation in songs within a corpus the less likely it is to classify instances accurately, this may explain why some models apart from the General model cross-validate much less accurately compared to other models as some artists may have many different song styles throughout their career whereas some maintain a certain style for example.

Subsequently, models that have been trained using a more specific set of songs can benefit due to a larger amount of similarities of LFPC values between the test song and the model, hence the improved classification accuracies compared to general models

7 Conclusions

7.1 Summary

Current vocal detection techniques typically make use of general purpose solutions where one algorithm or one model is used for all audio samples requiring vocal detection. Techniques such as statistical classification of audio segments do not take advantage of available information about the audio when performing classification. Basic information of a typical song such as artist name or artist gender is freely available, usually in the form of plain text appended to the audio file itself. Due to lack of use of such easily obtainable information, general techniques such as statistical classification fall short of its potential.

The use of statistical models for vocal detection remain one of the most popular methods in this field as they are relatively lightweight and usually produce acceptable results. User experience for the mobile application being developed is crucial. Therefore, our goal was to improve accuracy in detecting vocal segments within songs so that attempting to synchronize lyrics to a song can be as accurate as possible. We saw a flaw in current statistical classification techniques in that they only use general purpose built models for classification. In order to improve on reliability and accuracy of vocal detection, our proposed solution was to use focus based statistical models for classifying audio segments instead of the conventional one-model-fits-all approach.

A series of specific corpuses were compiled where each corpus contained songs of a certain type such as all female songs or all Michael Jackson songs. Several specific models were then built using those focused corpuses and their improvements over a general model were investigated.

7.2 Conclusion

After evaluations were completed we were able to gain confidence in saying that classifying song segments using more specific models improved accuracy in locating vocal segments within songs.

The test results reveal that models that were trained on songs that have larger differences in energy levels between vocals and non-vocals typically classify songs of similar types with higher accuracies. For this reason general purpose models typically classify song segments less accurately compared to special built models since general models contain a larger variety of songs thus having a larger range of LFPC values defined as vocals.

The issues that were concerning during research were the lack of non-vocal instances available for training. Due to the nature of songs, a significantly lower percentage of the non-vocals were found during model training stages, this is feared to have caused models to sacrifice accuracies. Since there were so many more vocal instances available for training it could be possible that an instance of unknown class is classified as vocals even though it is actually non-vocals due to the fact that instances for non-vocals are less refined, leaving higher opportunity of a non-vocal instance to be classified as vocals.

Another major concerning issue during research was the process of manually annotating song segments into vocals and non-vocals. As it turns out humans make mistakes, especially when dealing with reaction times down to fractions of a second. What is concerning is that manually annotating songs is arguably the most crucial step in the vocal detection process as well as being the most error prone, if done incorrectly the outcome is bound to have consequences. The tediousness of this process makes it very unpleasant and consequently can cause more errors than expected.

Overall, test results suggest that using specific built models for classifying specific songs produces promising improvements over using general models. Detecting vocals accurate to two hundredth of a second seem to be the most balanced between efficiency, accuracy and precision. This balance is especially

beneficial for applying this technique to mobile devices that have certain processing limitations.

8 Future Work

This thesis presented known effective techniques in the field of vocal detection and improved on it with our approach of using focused models. These improvements provide a better opportunity for the application being developed in parallel to synchronize lyrics with vocals. Areas in which research could progress further are as follows.

8.1 Model genre

During the research, there was one alternative path that we did not take even though we thought it was a good idea to pursue it. The idea was to build specific models according to genre, the reason this was not pursued at the time was because genre classification is very subject to debate. The genre of any one song could be pop or rock or it could even be both or perhaps more, this creates a lot of controversy as to whether a genre specific model built is the genre it claims to be. A song such as “Where is the love” by Black Eyed Peas could be classified as rap or it could also be pop, therefore it can be difficult to completely isolate one genre. However, if possible, improvements could be groundbreaking.

8.2 Combine different techniques for lyric alignment

In order fully satisfy functional requirements of the mobile application being developed in parallel, further research is required. The aim of the application is to have the ability to automatically align lyrics of a song to their corresponding vocal segments. The entry point to solving this problem has been to sufficiently detect vocals segments more accurately. The remaining possible paths to take to fulfill the functional requirement are as follows:

Extract chroma features for song structure analysis

As we have found during research, chroma feature vectors are effective for describing song structure. Each chroma vector describe the twelve dominant pitch values of a segment of a song, when coupled with similarity detection of the

whole song a high level structure can be achieved where sections of the audio that repeat the most is most likely to be the chorus. Knowing where the choruses occur can be particularly useful since it contains timing information of vocal segments, this type of information is scarce and if known is very beneficial.

Analyze plain text lyric files to detect lyric structure

The idea behind textual structure is very similar to chorus detection using chroma vectors. The text is analyzed to detect sections of the most repeated textual sequences; the most repeated section of text is most likely to be the chorus. Knowing which word the first chorus begins with implies that we can align the text of the beginning of the chorus to the time stamp obtained from detecting the first chorus start point in the audio using chroma features.

Combining vocal detection, chorus detection and textual chorus detection we begin to gain alignment possibilities where choruses are aligned and using the time stamps obtained through vocal detection proper alignment can be adjusted. Sections containing verses can possibly be aligned to text by the assumption that most songs begin with the verse, what better way to obtain a time stamp for the beginning of a verse than by using vocal detection.

9 REFERENCES

1. *AUTOMATIC SYNCHRONIZATION BETWEEN LYRICS AND MUSIC CD RECORDINGS BASED ON VITERBI ALIGNMENT OF SEGREGATED VOCAL SIGNALS .* **Hiromasa Fujihara, Masataka Goto, Jun Ogata, Kazunori Komatani, Tetsuya Ogata, Hiroshi G. Okuno.** 2006. Proceedings of the Eighth IEEE International Symposium on Multimedia. pp. 257-264.

2. *SINGER IDENTIFICATION BASED ON ACCOMPANIMENT SOUND REDUCTION AND RELIABLE FRAME SELECTION.* **Hiromasa Fujihara, Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno.** 2005. In Proceedings of the 6th International Conference on Music Information Retrieval.

3. *AUTOMATIC DETECTION OF VOCAL SEGMENTS IN POPULAR SONGS.* **Tin Lay Nwe, Ye Wang.** s.l. : Proceedings of the International Conference on Music Information Retrieval, 2004.

4. *LOCATING SINGING VOICE SEGMENTS WITHIN MUSIC SIGNALS.* **Adam L. Berenzweig , and Daniel P. W. Ellis.** 2001. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.

5. *A CHORUS SECTION DETECTION METHOD FOR MUSICAL AUDIO SIGNALS AND ITS APPLICATION TO A MUSIC LISTENING STATION.* **Goto, Masataka.** 2006. IEEE Transactions on audio, speech, and language processing.

6. *A NOVEL CHROMA REPRESENTATION OF POLYPHONIC MUSIC BASED ON MULTIPLE PITCH TRACKING TECHNIQUES.* **Matthias Varewyck, Johan Pauwels, and Jean-Pierre Martens.** Vancouver, British Columbia, Canada : s.n., 2008. Proceeding of the 16th ACM international conference on Multimedia. pp. 667-670.

7. *DYNAMIC CHROMA FEATURE VECTORS WITH APPLICATIONS TO COVER SONG IDENTIFICATION.* **Samuel Kim, and Shrikanth Narayanan.** Cairns, Qld : s.n., 2008 . Multimedia Signal Processing, 2008 IEEE 10th Workshop on . pp. 984 - 987 . 978-1-4244-2294-4 .

8. *A REAL-TIME MUSIC-SCENE-DESCRIPTION SYSTEM: PREDOMINANT-F0 ESTIMATION FOR DETECTING MELODY AND BASS LINES IN REAL-WORLD AUDIO-SIGNALS.* **Goto, Masataka.** 2004. Speech Communications. pp. 43(4):311-329.

9. *SIGNAL PROCESSING ASPECTS OF COMPUTER MUSIC: A SURVEY.* **Moorer, James Anderson.** 1977. Proceedings of the IEEE. pp. 65(8):1108-1137.

10. Android. [Online] Google Inc. [Cited: April 7, 2011.] http://developer.android.com/guide/basics/what-is-android.html.

11. *SYSTEM AND METHOD FOR AUTOMATIC SINGER IDENTIFICATION.* **Zhang, Tong.** Baltimore : s.n., 2003. IEEE International Conference on Multimedia and Expo.

12. *The WEKA Data Mining Software: An Update.* **Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten.** 1, s.l. : SIGKDD Explorations, 2009, Vol. 11.

10 Appendix

The following is a complete list of songs in the corpus we used which includes various artists and genres.

Artist and song names
ABBA - SOS
ABBA - Waterloo
A-Ha - Take On Me
All 4 One - I Swear
Anthony and the Johnsons - For Today I Am A Boy
Anthony and the Johnsons - What Can I Do
Arctic Monkeys - Red Light Indicates Doors Are Secured
Arctic Monkeys - Riot Van
Artful Dodger feat Craig David - Re-Rewind
Baby Face - Sorry For The Stupid Things
Badly Drawn Boy - Fall In A River
Badly Drawn Boy - Walking Out Of Stride
Beastie Boys - Intergalactic
BEDTIME - BABY FACE & USHER
Belle and Sebastian - A Summer Wasting
Belle and Sebastian - She's Losing It
Belle and Sebastian - Simple Things
Belle and Sebastian - Winter Wooskie
Benny Profane - Devil Laughing
Bettie Serveert - Dust Bunny
Bettie Serveert - Palomine
Bettie Serveert - Rudder
Bettie Serveert - Story in a Nutshell
Black Eyed Peas - Cali to New York
Boyz II Men - On Bended Knee
Boyz 2 Men - The Color Of Love
Brian Mcknight - You're The Only One For Me

Cast - Sandstorm
Chicago - Old Days
Chris Brown - With You
Chumbawumba - Tubthumping
Coolio - See You When Get There
Creedence Clearwater Revival - Have you ever seen the rain
Depeche Mode - It's No Good
Desmond Dekker - You Can Get It If You Really Want
dEUS - Suds and Soda
Dire Straits - Money For Nothing
Dodgy - Whole Lot Easier
Eggman - Out Of My Window
Faith No More - Epic
Jackson 5 - Can You Feel It
K-CI & JO-J0 - All My Life
Kirk Franklin RKelly - Lean On Me
Korn - Got the Life
Lucy Pearl - Don't Mess With My Man
Marilyn Manson - Sweet Dreams
Monaco - Blue
Nick Drake - Northern Sky
Nirvana - Smells Like Teen Spirit
Oasis - Wonderwall
Pet Shop Boys - Always On My Mind
Portishead - Wandering Star
Queen Yahna - Ain't It Time
Radiohead - Creep
Rain - Lemonstone Desired
REM - Drive
R Kelly - I Believe I Can Fly
Saxon - The Great White Buffalo
Scooter - How Much Is The Fish
Simon and Garfunkel - The Sound Of Silence
Simply red - stars
Sinead O'connor - nothing compares to you
Suede - Trash
Supergrass - Alright

Teenage Fanclub - Free Again
Teenage Fanclub - If I Never See You Again
Teenage Fanclub - Radio
Teenage Fanclub - What You Do To Me
The Beatles - Fixing A Hole
The Beatles - Getting Better
The Beatles - Good Morning Good Morning
The Beatles - Help
The Beatles - Hold Me Tight
The Beatles - I'm Happy Just to Dance With You
The Beatles - I Should Have Known Better
The Beatles - Little Child
The Beatles - Lovely Rita
The Beatles - Not a Second Time
The Beatles - Roll Over Beethoven
The Beatles - Sgt Pepper's Lonely Hearts Club Band (Reprise)
The Beatles - Sgt Pepper's Lonely Hearts Club Band
The Beatles - Shes Leaving home
The Beatles - When I'm Sixty-Four
The Beatles - With a Little Help from My Friends
The Beatles - within you without you
The Boo Radleys - Heaven's at the Bottom of this Glass
The Breeders - Do You Love Me Now
The Fratellis - For the Girl
The House of Love - Destroy The Heart
The Jesus and Mary Chain - Come On
The Jesus and Mary Chain - Hole
The Lemonheads - Different Drum
The Libertines - Up the Bracket
The Man From Del Monte - Ascension Day
The Monkees - Words
The Pixies - e Of Mutilation
The Police - Message in a bottle
The Proclaimers - Make My Heart Fly
The Roots & Erica Badu - You Got Me
The Roots - The Next Movement
The Stone Roses - Going Down (Remastered)

The Stone Roses - The Hardest Thing (Remastered)
The Strokes - 1251
The Sundays - I Can't Wait