Discriminating Speech from Music

Pablo Martin Garcia
martingarcia@umail.ucsb.edu

Antonio Javier Samaniego Jurado
samaniegojurado@umail.ucsb.edu

I. Introduction

In this project we will be implementing a method for discriminating speech from music using MATLAB. What we are going to do is train and learn from a set of data that is given to us so as to establish thresholds for future comparisons and to be able to discriminate speech from music. Then an audio file in .wav format will be read, which will contain either a recorded speech audio or music audio. Having established the different thresholds with the training data, the algorithm will determine automatically if the input audio is either speech or music.

II. Extracting Features from Training Data

A. *Reading Training Data*

Firstly, to train the machine and establish the thresholds we are interested in so as to distinguish and discriminate speech from music, our algorithm will read and process all of the files, both speech and music files, that were given to us in the challenge description. We are going to create two cells structures, one with the speech files and the other with the music files (39 files on each), C_speech{39,1} and C_music{39,1}. On a note we should also include that the algorithm also stores the sample frequency components in the cell.

In these two cells structures, we will read the audio files in .wav format as well as the sample frequencies of each one of them with the in-built function [x,fs]=audioread(input_audio) from MATLAB. Our algorithm will then work with these cells to calculate parameters to discriminate speech from music. The parameters of the training data that we will calculate, for both speech and music, will be: (in a **time domain analysis**) the silence ratio, the variance, the zero-crossing, and the autocorrelation; (and also in a **frequency domain analysis**) the RMS (Root Mean Squared) and the spectral centroid.

With these parameters we will be able to distinguish between a speech input file and a music input file.

B. *Silence Ratio – Time Domain Analysis*

The purpose calculating this ratio is that a person speaking or giving a speech is more likely to have pauses and moments where the amplitude of the audio is close to zero or at least much smaller than where the crucial information of the signal is, which is basically when the person is speaking. On the other hand, music is much more likely not to have quiet moments all through the recording. That is why our algorithm will execute the silence ratio of every signal, both speech audio inputs and also music audio inputs, and do an average so as to establish this threshold.

The threshold is established by separating the input audio of the training data into five different parts and executing the maximum value of each part. After this, we calculate the average of the five maximum values to get an idea of what the maximum amplitude of the data is. With each of the maximum average values of the amplitudes of each of the signals in the training data, we can determine which amplitudes are less than percentages of that maximum values, for example ten percent. We can consider all of these amplitudes as silence. Summing up the all of the samples where the amplitudes are less than the ten percent of the maximum value of the signal and then dividing by the length of the signal, we get the ratio of silence of the input signals.

Every signal in the input training data varies a little bit. However, this part of the algorithm is quite accurate. To have threshold values, the algorithm calculates the average of the ratios of both music and speech silence ratios.

To illustrate this better, we have plotted the proportion of silence given a certain threshold that is established in our algorithm, which can be seen below (Fig. 1).

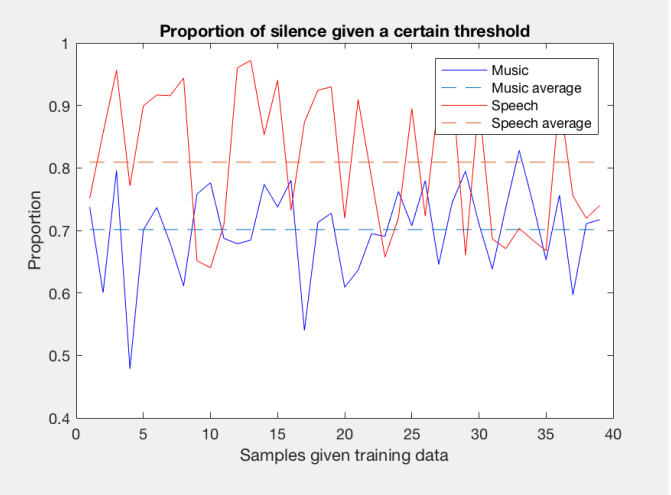

Fig.1. Ratio of silence of both music and speech input training data, given by the audio samples of the training data. The blue line represents the music silence ratios, while the dashed blue line represents the music silence ratio average. The red line represents the speech silence rations, while the dashed red line represents the speech silence ratio average.

There is a noticeable set of ratios where both speech and music silence ratios can match. This is why we established different weights for discriminating speech from music:

- If the ratio is less than 0.6, the algorithm gives a score of 1.5 for music.
- If the ratio is between 0.6 and 0.65, the algorithm gives a score of 1 for music.
- If the ratio is between 0.65 and 0.7, the algorithm gives a score of 0.25 for music.
- If the ratio is between 0.8 and 0.85, the algorithm gives a score of 1 for speech.
- If the ratio is bigger than 0.85, the algorithm gives a score of 2 for speech.

As you can see, the algorithm directly rejects ratios between 0.7 and 0.8, where both speech and music silence ratios can be the same and results in confusion.

C. Variance – Time Domain Analysis

The variance, in statistics, is defined as the expectation of the squared deviation of a random variable from its mean. To explain why we used this parameter for our algorithm, we can think of the amplitude of an input signal which is varying in time. A speech audio signal would not be vary considerably with respect to its mean compared to a music audio signal which will vary more noticeable due to the constant change on amplitude caused by all the instruments and different components of the signal.

The variance is given by the following formula:

$$Var(x) = E[(X - \mu)^2]$$

where E is the function of the expected value, X is the signal, and μ is the mean.

The algorithm implements the variance automatically calculating it with the in-built function var(input_audio) on MATLAB. In this case, the input audio would be the cell C_speech{i}(:) for speech audio files and C_music{i}(:) for music audio files.

So in theory, there should be a higher variance for music signals and a lower variance for speech signals. Having trained our machine with the training data, this is how we established a threshold for the variance.

To illustrate the difference in variance between speech and music files from the training data, we plotted the variances for speech and music based on the 39 signals on speech and the 39 signals on music, as well as the average values for both variances, speech and music. This can be seen below (Fig. 2).

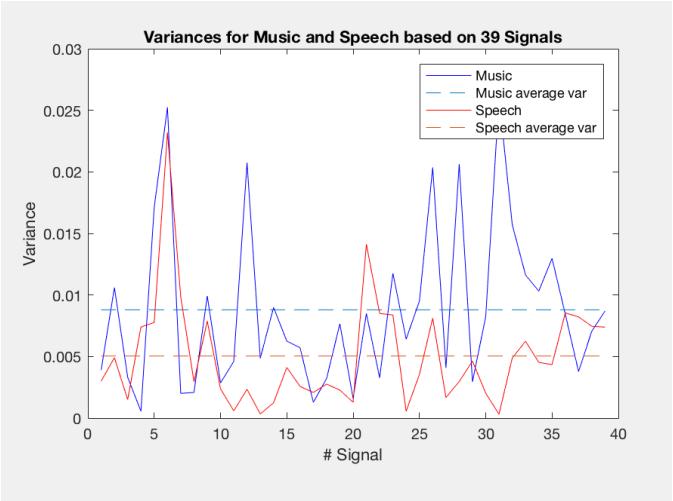

Fig.2. Variances for music and speech, given by the audio samples of the training data. The blue line represents the music variances, while the dashed blue line represents the music average variance.

However, this parameter is not very well distinguished between speech average variance and music average variance, so this is why we gave the algorithm a low score in between this gap. The weights over and under this gap is established this way:

- If the variance is greater than the music average variance, the algorithm gives a score of 1 for music.
- If the variance is less than the speech average variance, the algorithm gives a score of 1.5 for music.

As you can see, the algorithm rejects the variances that are in the gap between both average variances.

D. Zero-Crossing – Time Domain Analysis

The Zero-Crossing is a parameter that counts how many time the signal crosses the zero value, either from positive to negative or from negative to positive, as the name says. To clarify, to calculate this parameter the algorithm should count the number of times the amplitude of the signal changes sign.

The parameter is defined by:

$$ZCR = \frac{1}{T-1}\sum_{t=1}^{T-1} func\{s_t s_{t-1} < 0\}$$

where s is the sound signal of length T measured in time and the function inside the summation equals 1 if the comparison is true and 0 otherwise.

The algorithm plots a zero-crossing rate for all 39 signals both for speech and music, which can be seen below (Fig. 3).

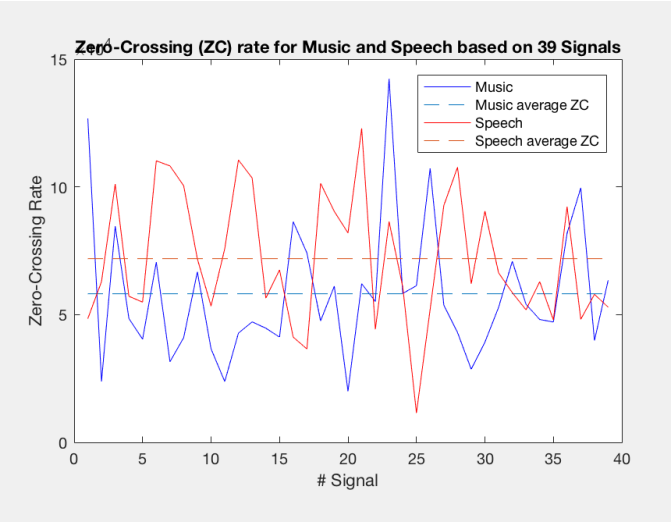

Fig. 3. Zero-Crossing rates for speech and music based on the training data given by the speech and music audio signals. The blue line represents the music ZCR, while the dashed blue line represents the music average ZCR value. The red line represents the speech ZCR, while the dashed red line represents the average speech ZCR value.

We can see how overall the zero-crossing rate for this particular set of audio files (speech and music) is higher for speech than for music. However, we can also observe a lower variance in music zero-crossing rate than for speech, which denotes a higher uniformity for music.

It can also be noticed that when the zero-crossing rate is beyond the average for speech, it is more likely to be a speech signal, whereas if it is below the average for music, it will be likely to be a music signal. Based on such observations, we have set the thresholds for this feature such that:

- If the zero-crossing rate is higher than the average for speech, the algorithm gives a score of 2 for speech.
- If the zero-crossing rate is lower than the average for music, the algorithm gives a score of 1 for music.

E. Autocorrelation – Time Domain Analysis

The process of autocorrelation is defined by the correlation of a signal with a delayed copy of itself. To clarify this, it is the similarity between observations as a function of the time lag between them. The purpose of analyzing the autocorrelation of a signal is useful for finding repeating patterns, such as the presence of a periodic signal obscured by noise, or identifying the missing fundamental frequency.

This is useful for certain types of music input files, because it will give a higher autocorrelation for files with periodic patterns, which it is common with music songs.

The autocorrelation of a signal X is given by the following formula:

$$R(\tau) = \frac{E[(X_t - \mu)(X_{t+\tau} - \mu)]}{\sigma^2}$$

where τ is the time lag (or time difference), μ is the mean and σ is the variance.

The autocorrelation is automatically calculated with the function xcorr(input_audio) on MATLAB.

To calculate the autocorrelation, the code creates a cell with its first components (first row of the cell) as vectors of all autocorrelations of the different training data and audio signals, both speech input signals and music input signals, and then second components (second row of the cell) with the mean of each autocorrelation. After this, the algorithm would compute the mean of every music autocorrelation averages and also the mean of every speech autocorrelation averages.

To illustrate the results, we have plotted the average autocorrelation for music and speech based on the 39 signals of the training data for each of the different audio signals. The plot can be seen below (Fig. 4).

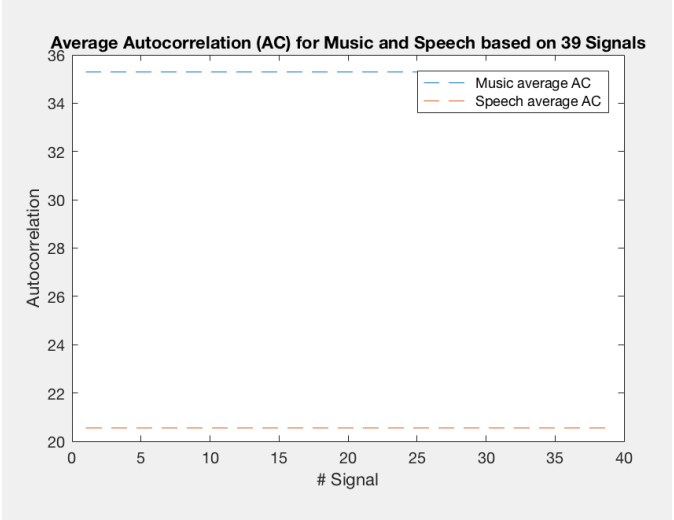

Fig. 4. Average autocorrelation for speech and music based on the training data given by the audio signals (39 for speech and 39 for music). The dashed blue line represents the music average autocorrelation and the dashed red line the speech average autocorrelation.

As you can see there is quite a difference in average autocorrelation between music and speech. That is why we established different scores to discriminate between audio and speech:

- If the correlation calculated is closer to the average speech autocorrelation, the algorithm gives a score of 0.1 to speech.
- If the correlation calculated is closer to the average music autocorrelation, the algorithm gives a score of 0.1 to music.

This is how the algorithm sets its scores in relation to the autocorrelation, which as you can see, they are not very determinant when the overall score of the algorithm.

F. RMS (Root Mean Square) – Frequency Domain Analysis

The Root Mean Square can be defined as the square root of the arithmetic mean of the squares of a set of numbers, in this case of an input signal. The RMS is calculated for both cells of speech and music.

This parameter is calculated with the following formula:

$$RMS = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}}$$

where n is the number of samples and x is the value of it.

Moreover, the algorithm calculates this feature for multiple frequency intervals, 6 in total, and equal in length, dividing the spectrum in 6. This way, we have 6 references of this feature for all 39 signals, computing an average of the values.

Graphic description of the process is shown below (Fig. 5).

Signal 1	Signal 2	...	Signal 39	
RMS1 - for interval 1	RMS1 - for interval 1	...	RMS1 - for interval 1	→ Average
RMS2 - for interval 2	RMS2 - for interval 2	...	RMS2 - for interval 2	→ Average
...	...	...	...	
RMS3 - for interval 6	RMS3 - for interval 6	...	RMS3 - for interval 6	→ Average

Fig. 5. How RMS is calculated and averaged over 6 intervals, for all 39 signals, both in speech samples and music samples of the training data.

By using the RMS in this way, the algorithm can measure the absolute value of the Fast Fourier Transform values calculated for every signal, in a certain analysis window.

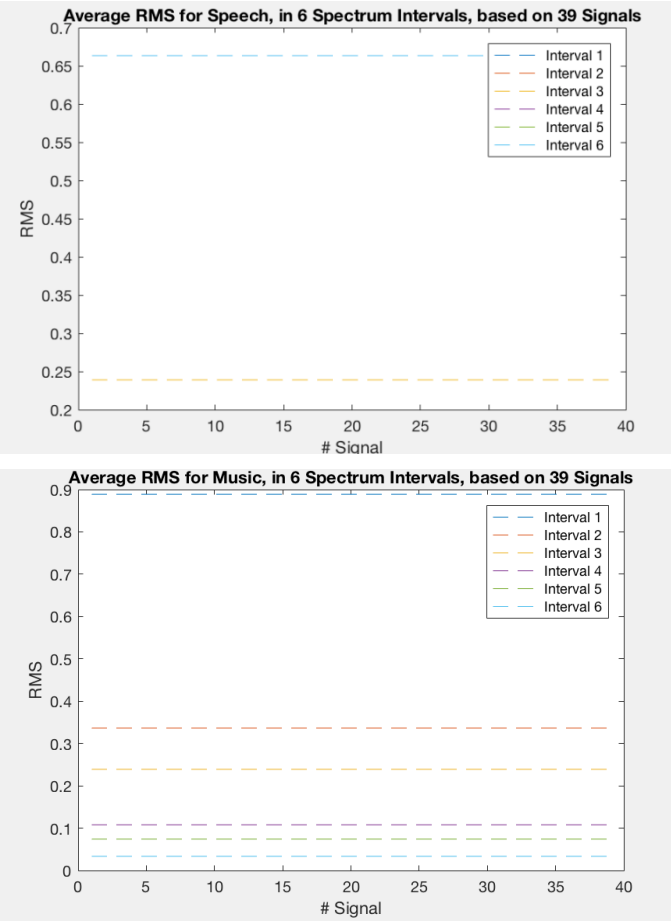

Fig. 6. Average RMS values for 6 frequency spectrum intervals.

In (Fig. 6) we plot the average RMS values for all 39 signals, both for speech and music. This feature in particular is one that gave us a wide range of consistent thresholds in order to determine the discrimination value between music and speech, training the data from the input samples that were given in the challenge description. Based on the results, the algorithm sets some scores for the different values of the RMS for a given input audio file:

- If the RMS is below 0.2 or over 0.8, the algorithm gives a score of 1.5 for music.
- If the ratio is between 0.2 and 0.3, the algorithm gives a score of 2.5 for speech and 0.3 for music.
- If the ratio is between 0.3 and 0.4, the algorithm gives a score of 1.2 for music.
- If the ratio is between 0.4 and 0.8, the algorithm gives a score of 2 for speech.

G. Spectral Centroid – Frequency Domain Analysis

The reason why this parameter is useful is because it is a result of evaluating the center of mass of frequencies making use of the Fourier Transform, in other words, analyzing the frequency domain.

The spectral centroid of X is given by the following formula:

$$Spectral\ Centroid = \frac{\sum_{k=1}^{N} k \times X[k]}{\sum_{k=1}^{N} X[k]}$$

where k is the sample of X in the frequency domain.

Calculating the spectral centroid, we are finding the average frequency weighted by amplitudes, divided by the sum of amplitudes.

We have used it as part of the criteria in the algorithm, setting a threshold based upon the reference or behavior for 39 signals, for music and speech, as well as an average value. Such trend is shown below (Fig. 7).

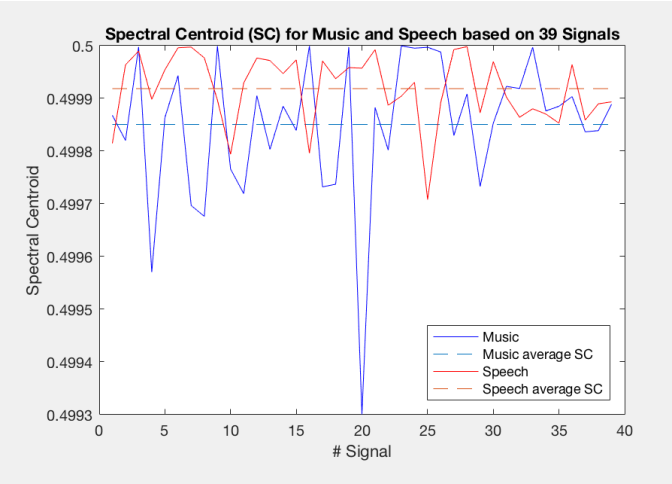

Fig. 7. Spectral Centroid for music and speech, based on the signals of music and speech given by the training data.

The threshold we have based this feature on is a result of the observation that when the average value for speech is surpassed, it corresponds to a speech signal most of the times. Similarly, when the spectral centroid is below the music average value, it is more likely to be a music signal. In consistency with such observation, we have set the following thresholds:

- If the spectral centroid is higher than the average for speech, the algorithm gives a score of 1.5 for speech.
- If the spectral centroid is lower than the average for music, the algorithm gives a score of 1.5 for music.

III. Methods

To discriminate speech from music, after having established certain thresholds and values with the training data that we were given in the challenge description our algorithm will calculate the same ratios and parameters to compare with those training data values with the ones that the algorithm obtains from the input signal. That is how the algorithm will decide if the input audio signal is either a speech input audio or a music input audio.

A. Reading Input Audio

The first thing the code does is read the input audio with the in-built function audioread from MATLAB. This function returns both a vector of the signal and the sample frequency.

B. Calculating Parameters and Ratios

We have to calculate the silence ratio, the variance, the zero-crossing ratio, the autocorrelation, the RMS and the spectral centroid.

C. Comparing with Training Data Values and Scoring

After this, the algorithm will compare with the features extracted from the training data and will weight certain values higher than others in concordance with the accuracy of the methods implemented. All the weights and scores have been established in the "Extracting Features from Training Data" section.

IV. Overall Algorithm

Below is shown a pseudo-code for the algorithm (Fig. 8).

Input: .wav audio file
Output: Music (=0) or Speech (=1)

Prepare training data
num_files ← Set number of files available
for num_files
  music_samples, Fs ← audioread(music files)
  speech_samples, Fs ← audioread(speech files)

Feature Extraction
%Time domain
Silence Thresholds ← Perform Silence
Variance Thresholds ← Perform Variance
Zero-Crossing Thresholds ← Perform Zero-Crossing
Autocorrelation Thresholds ← Perform Autocorrelation

%Frequency domain
RMS Thresholds ← Perform RMS
Spectral Centroid Thresholds ← Perform Spectral Centroid

Evaluation of Input Audio File
Parameters ← Perform mentioned Time and Frequency domain methods
Compare resulting parameters with extracted feature thresholds
result ← Store Music(0)/Speech(1) / Not clear

Evaluation of Algorithm Accuracy
for each music file
  Parameters ← Perform mentioned Time and Frequency domain methods
  Compare resulting parameters with extracted feature thresholds
  result_music ← Store success(1)/Failure(0)/Not clear(0.5)

for each speech file
  Parameters ← Perform mentioned Time and Frequency domain methods
  Compare resulting parameters with extracted feature thresholds
  result_speech ← Store success(1)/Failure(0)/Not clear(0.5)

accuracy_music= sum(result_music=1)/num_files
accuracy_speech= sum(result_speech=1)/num_files

Graphs
plot of results in time/frequency domain, as well as success rate for both music and speech.

Fig. 8. Overall Algorithm.

V. Results

To analyze the behavior and accuracy of our algorithm, we have run the algorithm with every single audio file from the training data to get a percentage of accuracy based on that training data and on the input file after having trained the data and got the features and thresholds for every calculation.

A. Speech Audio Input Files

To get a sense of the accuracy of the speech discrimination, we set as input every speech audio file. After this, we compare the result we get, either speech or music, with it being speech or music. Note that we already know if the input file is either speech or music.

The accuracy of the speech discrimination is 56.41%

B. Music Audio Input Files

To get a sense of the accuracy of the music discrimination, we set as input every music audio file. After this, we compare the result we get, either speech or music, with it being speech or music. Note that we already know if the input file is either speech or music.

The accuracy of the music discrimination is 79.49%

C. *Overall Accuracy*

Although it was clearly more accurate for music than for speech discrimination, it still provided a decent accuracy rate overall (around 70%).

We can look at the success rate of the algorithm for both speech and music below (Fig. 9).

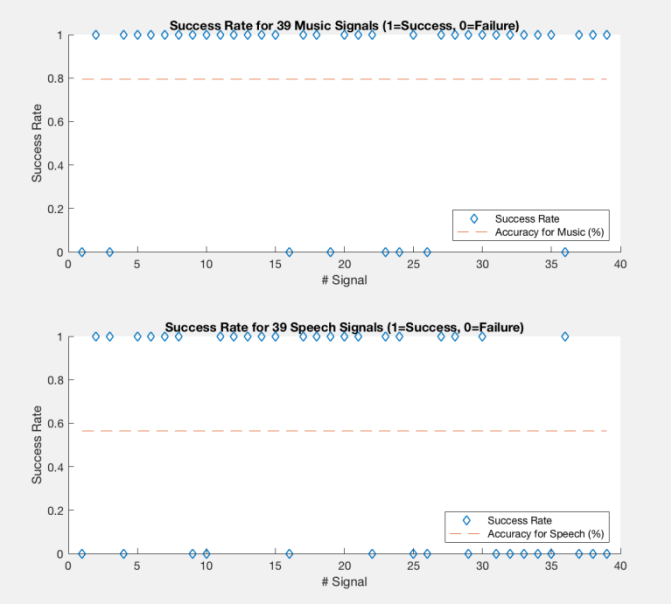

Fig. 9. Success rate for 39 music input signals from the training data, with dashed red line showing the average success rate (top), and success rate for 39 speech signals from the training data, with a dashed red line showing the average success rate (below).

Another representative way to look at the reliability of the algorithm can be seen below in a table with accuracy rates for both music and speech (Fig. 10).

	Evaluation for Music	Evaluation for Music	Overall
# Success Cases	22	31	53
# Failure Cases	17	8	25
Success Rate (Accuracy)	56.5%	79.5%	68%

Fig. 10. Accuracy rates table. Overall accuracy of 68%.

VI. Conclusion

Discriminating speech from music is a challenging task that, when approached with the right tools and strategy, results in such a useful application. Our algorithm uses time and frequency domain parameters and operations, including silence proportions, variance, zero-crossing rate, autocorrelation, RMS and Spectral Centroid, which aim to achieve this goal.

We have seen how the better the thresholds or references based upon features extracted from training data are (39 speech files and 39 music files for this particular case), the better the results, which in our case could have been more reliable for speech, but gave a reasonable overall rate of accuracy, around 70%.