Speech and Music Discriminator

Vicki Chen, Ben Liu
ECE 160
UCSB
vchen@umail.ucsb.edu , bliu00@umail.ucsb.edu

***Abstract*—To produce an accurate speech and music discriminator program, we must explore many concepts of sound signals such as the amplitude, frequency, zero crossing rate, and other aspects that make speech signals unique from music signals. Although humans can easily distinguish between music and speech, in order for a program to do this, it must make use of these different factors in order to make an accurate decision. This program utilizes percent empty, zero crossing, and root mean to read a folder of audio files and precisely distinguish whether it is a speech file or a music file.**

I. INTRODUCTION

There are many different factors we must look into when it comes to creating a program that needs to analyse audio files and differentiate certain audio files from others. Since audio signals are very complex in itself, we needed to look deeper into more complicated ways to breaking down these signals so that we can accurately compare them with others.

In this challenge, we are told to create a speech and music discriminator that should read a list of audio files and output a cell array that provides the name as well as the binary value, “0” if it is a music file and ”1“ if it is a speech file. To create this program we created 3 different functions that reads specific values from the signals and adds either 0 or 1 into the final array based on the decision the program makes.

II. METHOD

We came up with three methods to determine whether our samples were speech or music.

A. Percent empty

In most speech audio files, the signals often contain pauses in between words or sentences. This leaves many gaps in the audio signals where the amplitude is close to zero. However, in music files, if one instrument fades, usually another instrument will play in its place. Because of this music signals will usually have fewer gaps than speech signals. With this knowledge, one way to differentiate between speech and music is to see what percent of the audio is considered empty or have an amplitude close to zero.

In order to find the amplitude, a loop goes through each point in the signal to check if it is a local maximum. This local maximum can be considered the amplitude of a certain region. A point is a local maximum if the point has a bigger absolute value than the absolute values of the points immediately before and after.

To check for local maximum , we used the code:

```
if(((x(i)>x(i+1))&&(x(i)>x(i-1))||(x(i)<x(i-1)&&(x(i
)<x(i+1)))))
        totalMax = totalMax + 1;                    (1)
```

After determining a point is a local maximum, we check whether or not it is small enough to be considered empty. We used a threshold that was specific to each file by having the threshold be a percentage of the max value in the signal.

To check if the point was below the threshold, we used the code:

```
maxP = .05*max(x);
if((abs(x(i)))<maxP)
   lessMax = lessMax + 1;
End                                                 (2)
```

By collecting how many times the local maximum would go under the threshold and dividing that number by the total number of local maximums, we can get a percentage of how much of the entire file can be considered empty. By testing two different sample folder, one filled with music and the other filled with speech, we learned that the general percent empty average was about 15-25% higher for speech than music. With this knowledge we used a number in between the two averages and programed the function to respond 1 if it was about the number and 0 if it was below.

B. Zero Crossing

The second method we used is to calculate the zero crossings of a signal input. A zero crossing is a point where the sign of a signal changes from positive to negative. In order for this method to be effective, we measured the short term variance in the zero crossing rate. To obtain an accurate measurement, we compared the distribution of zero crossings for every 20ms frame of a signal and the zero crossing for every second of the data.

Using the equation below

```
for i = 2:length(fdata)

zcr(y) = zcr(y) + abs(sign(fdata(i)) - sign(fdata(i-1)));

end

zcr(y) = zcr(y)/(2*flength);

y = y + 1;

End
```

(3)

We can sum up the zero crossings across the frame. The local variance is then calculated over each second of the signal. The final value of the zero crossing rate for the audio signal is the mean of the local variance.

C. Root Mean Square of Power

The second method we used to determine speech and music is calculating the root mean square of power over the length of the input signal. This feature is useful is determination whether an audio signal is music or speech because speech typically has more quiet frames and pauses than music. By calculating the power over the length of the sample, we can find out when we encounter low levels of power which indicates periods of silence.

We began by finding the rms value for every 20ms frame of the input signal using

```
rms(y)=sqrt(sum(fdata.^2)/length(fdata));
```

(4)

The local mean is then calculated over the last second of the signal. For each frame after the first, the frame was counted as low-energy if it fell below 50% of the local mean.

```
if (total_frames > fsec)
 for i = fsec+1:num_frames
 rms_mean(i) = mean(rms(i-fsec:i));
 if (rms(i) < 0.5*rms_mean(i));
 low = low + 1;
```

(5)

The number of low energy frames in a sample divided by the total number of frames processed becomes the sample's data value. The amount of low energy frames will signify the percentage of power falling below the 50% threshold.

```
rms = low/(total_frames-fsec);
```

(6)

III. RESULTS

I. PERCENT EMPTY

Determining Empty Threshold

Threshold Percent	Average Percent Empty Music	Average Percent Empty Speech	Difference In Average Percent Empty
20%	79.12%	84.75%	5.63%
15%	68.88%	80.189%	11.31%
10%	53.74%	73.24%	19.50%
5%	31.61%	58.04%	26.43%

To accurately classify whether a local maximum is empty or not we compared the value to a percentage of its maximum. We tested different percentage thresholds and looked for the threshold where the difference in average percent empty for speech and music was the greatest. Since 5% threshold had the greatest difference, we this to plot the scatter graphs.

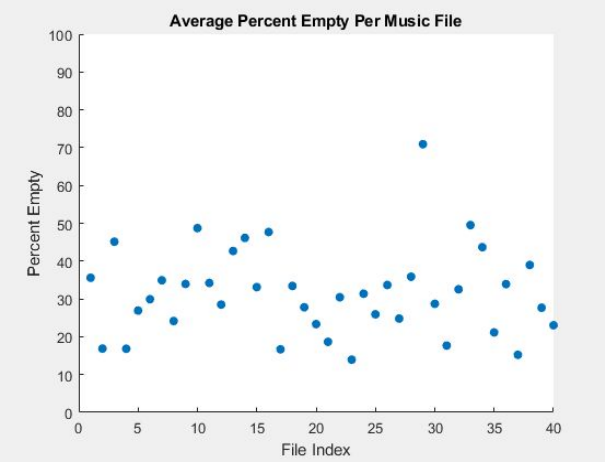

Fig. 1. Graph of Average Percent Empty For Music Files

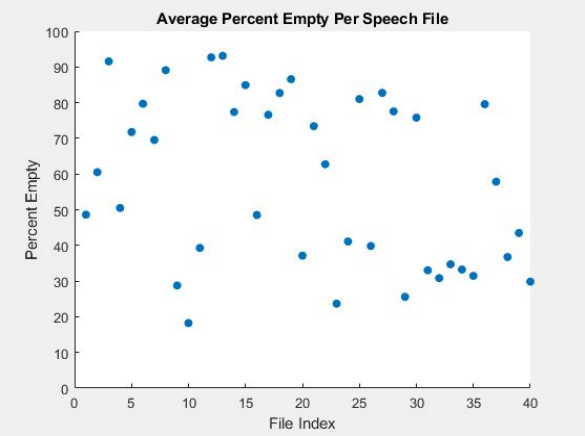

Fig. 2. Graph of Average Power Empty For Speech Files

From the two figures, we can see that the points for the music files generally had a percent empty that stayed below 50% while for the speech files the percent empty stayed above 40%. Speech files had an average of 58.04% and music had an average of 31.61%. From this average, we tested several percentages in between 58.04% and 31.61% to use as a cutoff percentage to see which gave us the greatest % accuracy for both music guessed and speech guessed. By going through this we determined that the program had the most accuracy when we used 36% as the cutoff percentage. Using this number, if the mean percentage was above 36%, it would conclude the file is a speech file and if the mean percentage was below 36%, it would conclude the file is a music file.

TABLE I. PERCENT EMPTY RESULTS

	Music % Guessed	**Speech %Guessed**	**Total Samples**
Test all speech	75	25	40
Test all music	77.5	22.5	40

II. ZERO CROSSING

Using the histogram function in matlab, we plotted the zero crossing rate of a speech file and a music file to analyze the difference. Compared to the speech file, music has a fairly normal distribution of frames with lower and higher zero crossing rates. On the other hand the Speech file displayed a much more skewed distribution with long periods of low zero crossing rate and very distinct period of transition to a much higher zero-crossing rate. As a result of these characteristics, the average variance in the zero-crossing rate of speech signals tends to be higher than that of music signals.

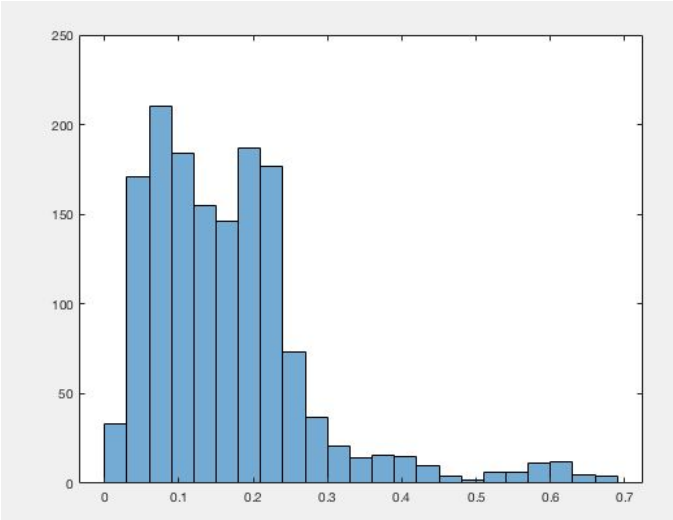

Figure 3. Histogram of Zero Crossing Rate of Speech File 'China.wav"

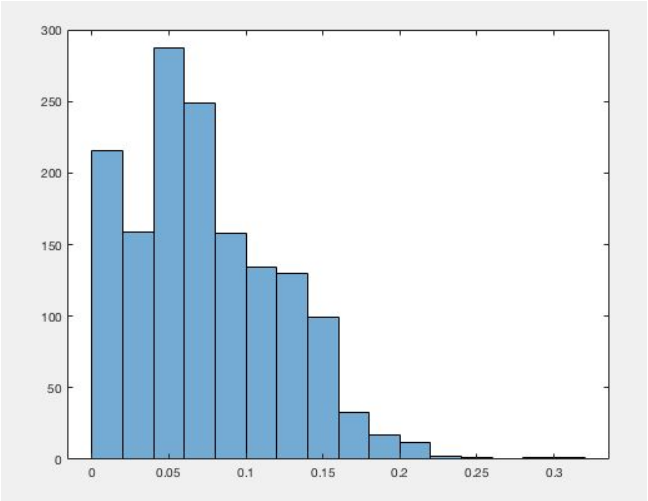

Figure 4. Histogram of Zero Crossing Rate of Music File "classical.wav"

To test the samples using zero crossing, we trained our program with 5 music files and 5 speech files and saved the zero crossing values as our trained datasets. Then, we went into each directory containing either all music or all speech to test the accuracy of our method.

K- Nearest Neighbors

To accurately classify our input signals as either music or speech, we compare our datasets with the K-nearest neighbor algorithm[2].

For each sample we're testing, we found the k=3 nearest neighbors based on the Euclidean distance between the value in the trained sample and our testing sample. The normalized Euclidean distance metric is generally :

$$d(x,y) = \sqrt{\sum_{j=1}^{d} wj^2(xi - yi)^2} \quad (7)$$

The distance between a new data point and every training point is compared among the 3 closest neighbors to determine whether the new data sample belongs in speech or music. To record the results of this classification method, we created a empty matrix that will store the name and the output. If two or more neighbors classify as music, we would give the audio signal an output of 0 and otherwise, a value of 1.

TABLE II. ZERO CROSSING RESULTS

	Music % Guessed	**Speech %Guessed**	**Total Samples**
Test all speech	20%	80%	35
Test all music	91.4286 %	8.5714 %	35

III. ROOT MEAN SQUARE OF POWER

Since music is more continuous than speech due to background instruments and vocals, it has fewer cases of quiet frames which fall under the 50% of the average root mean square. To visually see the difference in a speech file and music file, we plotted the rms and its 50% threshold on a separate graphs.

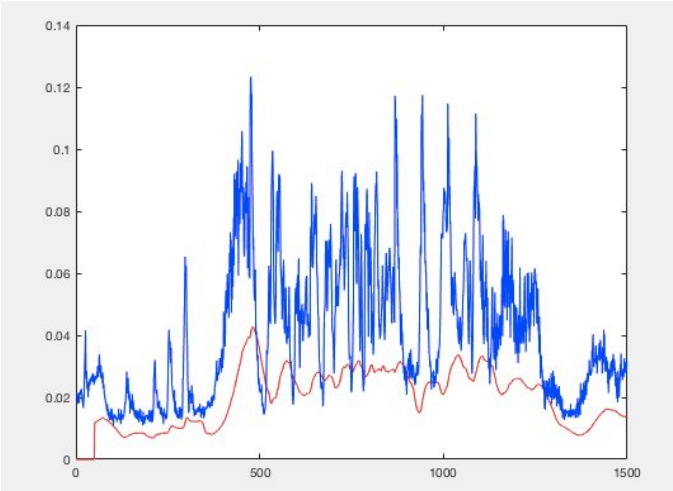

Figure 5. RMS Power of a Music File "Classical.wav"

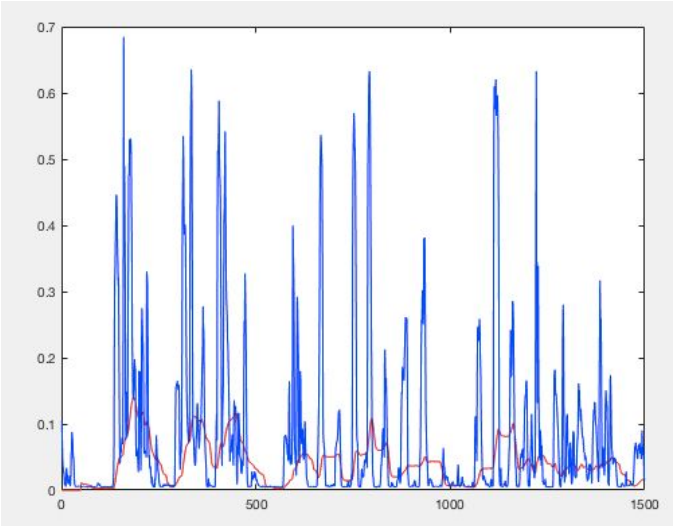

Figure 6. RMS Power of a Speech File "China.wav"

After analyzing the graphs, we noticed that the percentage of low energy frames is significantly higher for the speech files. This proves the fact that speech indeed will have higher rates of power falling below the 50% threshold due to more pauses and quiet frames.

Using the same method we used to test the zero crossing rate, we again compared newly tested samples with our trained data set which now contains a set of average low energy percentage for speech and music. After calculating the percentage of low energy for the new signals, we used K nearest neighbor once again to determine whether the signal will be classified as either speech or music. This gave us the following results which indicate a fairly successful performance.

TABLE III. RMS RESULTS

	Music % Guessed	**Speech %Guessed**	**Total Samples**
Test all speech	0%	100%	35
Test all music	91.4286 %	8.5714 %	35

IV. CONCLUSION

To measure the accuracy of each method, we used the equation:

$$Accuracy = \frac{\#\ of\ correctly\ classified\ examples}{\#\ of\ examples} \times 100 \quad (8)$$

TABLE IV. COMPARISON OF RESULTS

Accuracy	Zero Crossing	RMS power	Percent Empty
Speech	80%	100%	75%
Music	91.42%	91.42%	77.5%

In the end we had 3 different methods that each discriminate between speech and music in different ways. Amplitude or percent error had the lowest accuracy out of the three. We believe this is due to the fact that some audio files had a lot of background noise like static or laughter. This background noise removed the empty gaps in the audio signal making it difficult to tell when the person isn't talking. Zero crossing and root mean square however, didn't have these issues and had a much higher accuracy percentage. In conclusion, we would think the root mean square method in finding power values below a 50% threshold would be the best in discriminating between speech and music because it resulted in 100 percent accuracy when tested on all speech files and is very accurate with it's predictions.

For this project, we worked together to construct all three methods. Each doing our own research and suggesting different ideas. We contributed equally to put the methods together and debug everything.
Some things we noticed during testing out the code for percent empty was that the threshold value wasn't giving proper percentages because it included all points of the signal. To fix this, we set up an if statement that looked for all the local maximums and only checked if these maximums were over the threshold.

For zero crossing and root mean square, we saved the code we used for testing as separate files. After making sure our methods produced accurate results, we went ahead and loaded all the speech files and music files that we didn't initially use as training files, as part of the new trained data sets and saved them as respective matrices. methodzcr.m and methodrms.m loads these trained data sets in the beginning upon execution. This will allow the program to produce more accurate results since we now have a bigger dataset to perform calculations with. The result is then saved in a matrix called array with one column listing the name of the file and the second column indicating whether the signal is classified as speech or music.

REFERENCES

[1] Mathworks.com(2018).*NearestNeighborsclassification.*[Online].Available:https://www.mathworks.com/help/stats/classification-nearest-neighbors.html

[2] "K-Nearest Neighbors," *geeksforGekks,* 09-Feb-2018.[Online]. Available: https://www.geeksforgeeks.org/k-nearest-neighbours/