Spectral Analysis:

Using Averaged Spectrograms to Detect Voice Signatures

Nicholas Ferry

TJ Mosteller

Veronica Mcleod

University of South Carolina Beaufort

Abstract

Spectrogram analysis is a vital component in the development of signal processing. A spectrogram is a graphical representation of the spectrum of frequencies of a signal as it varies with time. The experiment aimed to analyze voice signals in order to compute a matrix of averages correlating to the specific word they represent. With this matrix, the basis of instantaneous voice signal processing can be made. MATLAB® is a powerful software that contains an extensive signal processing library of tools and algorithms. This report aims to outline how MATLAB® can be used to process and analyze the sentence, “We found a simple sentence with about ten words”. The output is a matrix of averages that can be used to detect the same respective word from any one person.

Table of Contents

Speech Analysis:
Using Averaged Spectrograms to Detect Voice Signatures

Introduction

A spectrogram is a graphical representation of the spectrum of frequencies of a signal as it varies with time. An analysis of this data allows for a greater understanding of signals, primarily what separates one signal from another. In order to better understand how these analyses of spectrograms can be of use in the real world, our group was tasked with choosing a simple sentence to analyze. MATLAB® is a powerful program that contains a vast library of sound analysis tools and algorithms. The software is hosted and supported by Mathworks®.

The objective of this report is to demonstrate how MATLAB® can be used to analyze each individual's sentence by separating the various words into segments of data. These segments of data can be analyzed in spectrogram plots produced by MATLAB® as well as the original sound wave plots. The average of each individual’s respective segment/plot data can then be calculated to form a matrix of averages for that specific word. It is expected that this matrix can then be used to formulate a general standard for the specific word the matrix represents. What this means is that this matrix of averages can be used to detect the same word no matter who speaks the word. The objective of this report is to demonstrate how MATLAB® can be used to process and analyze voice signals in the interest of creating a methodology for the basis of instantaneous voice signal processing. In the real-world, this experiment formulates the basis for voice detection systems such as Alexa ® and Siri ®.

The group members that participated in the experiment were: Nicholas Ferry, TJ Mosteller and Veronica Mcleod who are all computational science majors at the University of South Carolina Beaufort. The sentence the group chose to analyze for the experiment is: “We found a simple sentence with about ten words”. The group chose this sentence due to its simplistic nature and ease of vocality. In order to record each group member's voice signals of this sentence, the Voice Memos® app on the iPhone® was utilized. In order to make the spectrogram analysis easier, during each individual recording session, the group members allowed for a one-second interval of silence between each word to reduce noise in the words signal. After capturing these recordings within the app, we were able to send them to an email of our choice so that the voice recording files would be easily accessible. This method of signal capturing was utilized for every signal used in this experiment.

Methodology

Procedure.

The original audio files were read with the “audioread” function which returns a matrix of the audio data and the rate in frames per second (or Hertz). The matrix of audio data was split for each word by masking the matrix for each interval that was individually found. The “spectrogram” function was used to create a spectrogram of the audio data. This is a 3-dimensional graphical representation of the frequencies over time. Graphs were aligned for comparison using the subplot function. Since each of the audio data for the same word had different lengths from different time intervals, the data were normalized to the same length using interpolation and specifically the “interp1” function. The averaged spectrum was found by calculating the average of the audio data. This was done first by concatenating the three audio files for each word along the third dimension using the “cat” function. Then the average was calculated using the “mean” function along these dimensions. This data is used for comparison with the test audio files.

The start- and end-points for each word segment were calculated using the formula time*frequency where time was found utilizing the built-in MATLAB® graphical/plot analysis tool. By plotting the sound file and simply clicking on the locations of the visual start and end times of the word oscillations, the x or time value can be determined. The frequency rate is determined upon reading the audio data through the inclusion of a second return parameter to represent the frequency rate.

Assumptions.

Assumptions made in this project are about the data used and the methods used to complete the objectives. One assumption is that the original three audio files are enough to calculate an average baseline for comparison. Larger sample size would increase the accuracy of the average. There are other factors that were not considered like voice pitch, annunciation and accents, and talking speed. Other assumptions were made in the implementation of the program. Comparisons of the data were made visually, which can lead to human error. The time intervals for each word were manually found and may also affect the results. The data toward the end of each word may also be inaccurate as it was interpolated to make the intervals the same size.

Pseudo-code.

1. Step 1
    a. First, the audio files must be read in. With these audio files, we can create a time and length variable to represent the sizes of the different axes we want. We then want to plot this data on a spectrogram plot for further analysis.
2. Step 2
    a. Next, we want to split the audio file into segments based on the time that each word is spoken. Since there are nine words, nine separate segments will be created. To ensure that the proper amount of data for each word is being captured, we will write the segmented audio data to a new file so that we can listen for the word the file should represent
3. Step 3
    a. We then want to normalize the data so as to ensure that when calculating the average of the 3 separate data segments, we are calculating the average based on data with the same independent variables and parameters.
4. Step 4
    a. After we normalize each individual array data, we then want to concatenate these 3 arrays together to form one array.
5. Step 5
    a. We then calculate the average of this concatenated array to create a matrix of average values to represent the word being analyzed. We then write this data to a new audio file for further analysis.
6. Step 6

   a. Plot all files and spectrograms for further analysis and visual representation of the sound data being analyzed.

7. Optional Steps:
   1. Repeat steps 1-5 for an audio file from a different student as well as an audio file of a different sentence.
   2. Compare averages of Step 1 (optional) matrix to step 5 matrix of averages.

Results

Findings.

There are quite a few ways to compare words using spectrogram analysis in MATLAB®. However, one technique cannot properly identify two images as similar by itself. Instead, all aspects of an image should be taken into account before an educated guess on whether two words match should be made. In general, the following guidelines should be looked at when comparing two different words

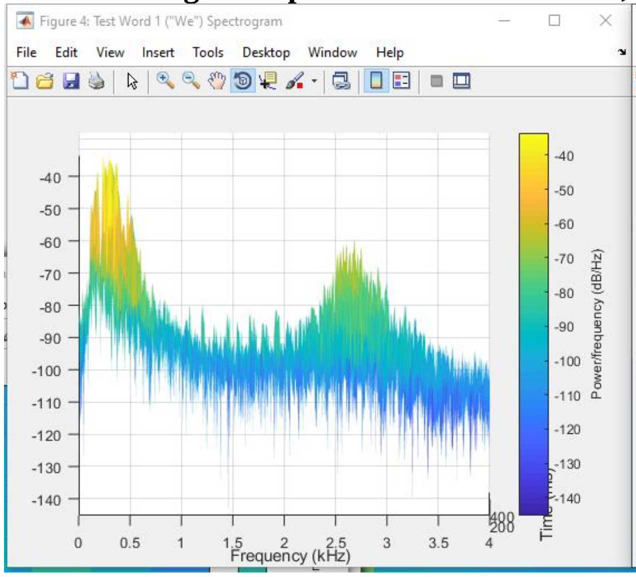
- Looking for max frequency points inside of the graph, and comparing the data between the two points.
  - Ex: “Group Average Vs. Different Word – Dramatic Dips” (Sentence vs. Found), one can see that the frequency drops dramatically after the maximum value, then rises slightly to the second maximum value in both images. This is a strong indication that two words match
    - It can be seen in the graph found under “Group Average Vs. Different Word – Max Value Test (This vs. Ten)” found in the data section

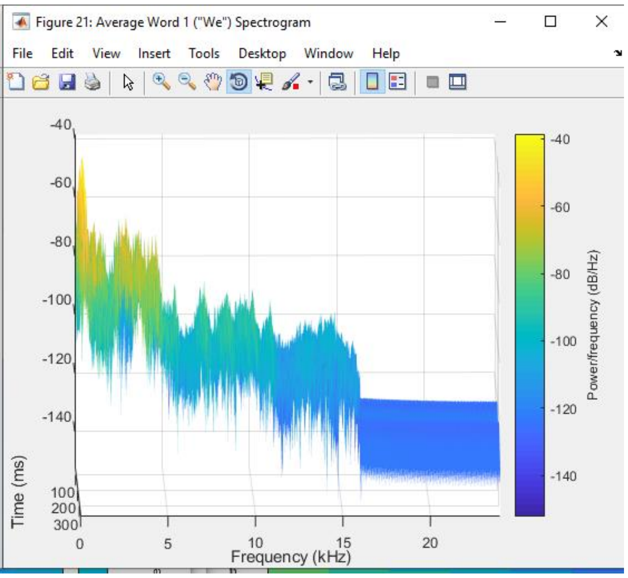
- Checking to see if the shape of the graph is similar.
    - Checking to see if the graph is skewed left, or skewed right
        - An example can be found under “Group Average Vs. Different Word – Different Skew (Is vs. Ten)” found in the Data section
- Compare dramatic changes in frequency between the two graphs
    - For example, if one spectrogram changes in frequency in any direction over a short period of time, check to see if the other spectrogram exhibits a similarly dramatic change in frequency at a similar point in time on the graph.
- Checking the range of the power/frequency of the graph, and comparing to see if the rate between the power and frequency are similar

Data.

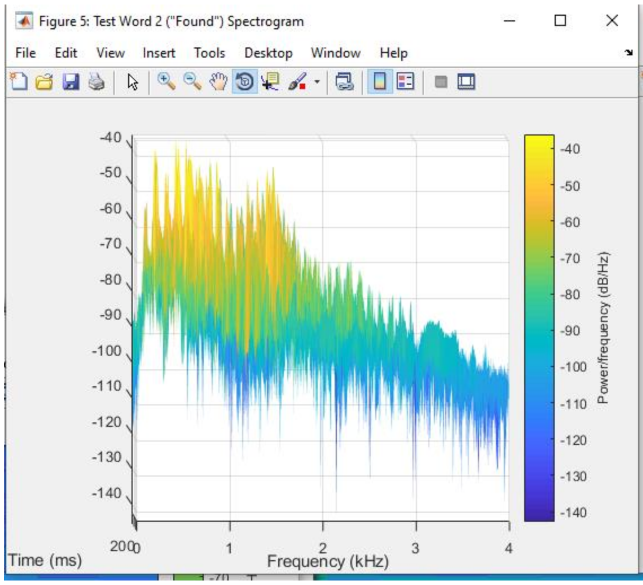
Each following graph is a side-by comparison of all words between one of the group members, and the averaged recordings of the group members. In order (“We found a simple sentence with about ten words”)

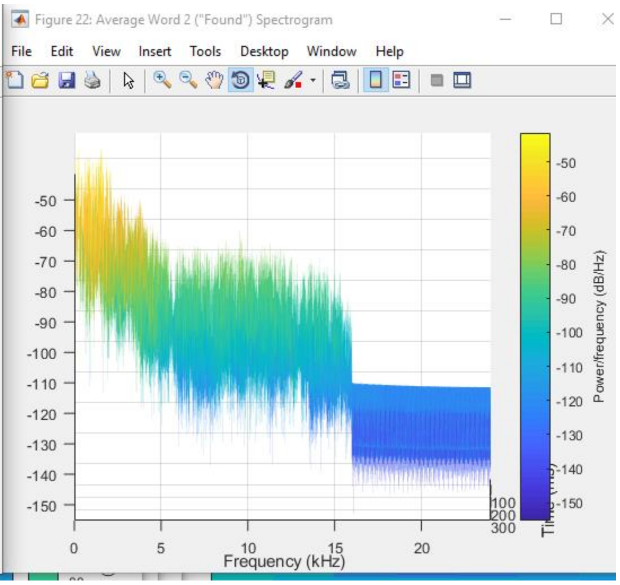
Average Comparison Vs. Test Word 1, We



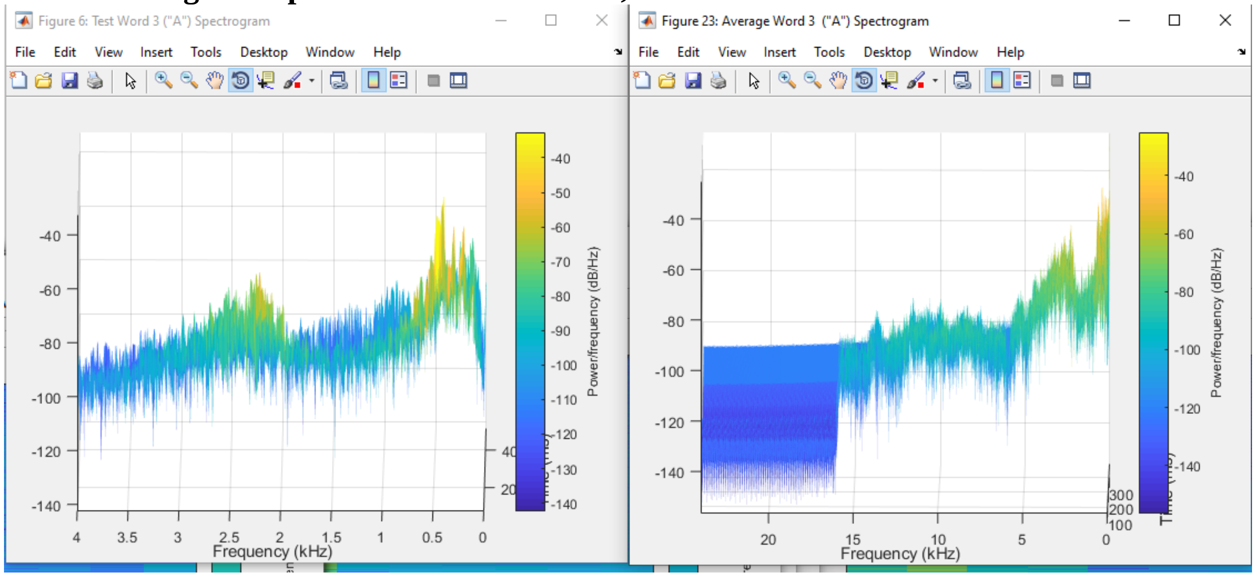


Average Comparison Vs. Test Word 2, Found

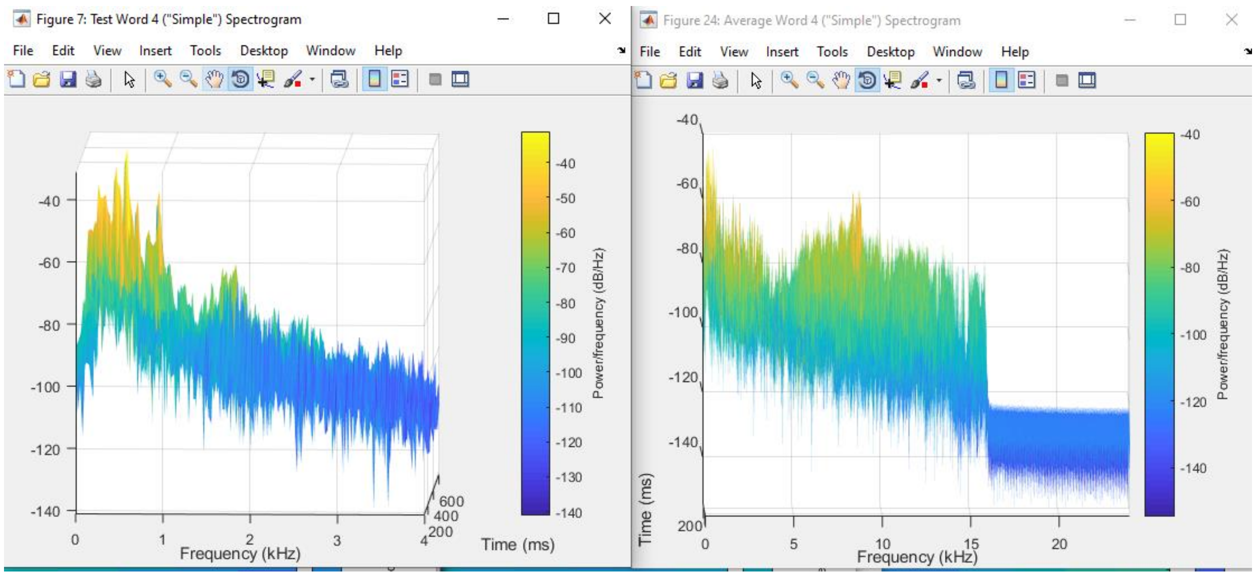




Average Comparison Vs. Test Word 3, A



Average Comparison Vs. Test Word 4, Simple



Average Comparison Vs. Test Word 5, Sentence

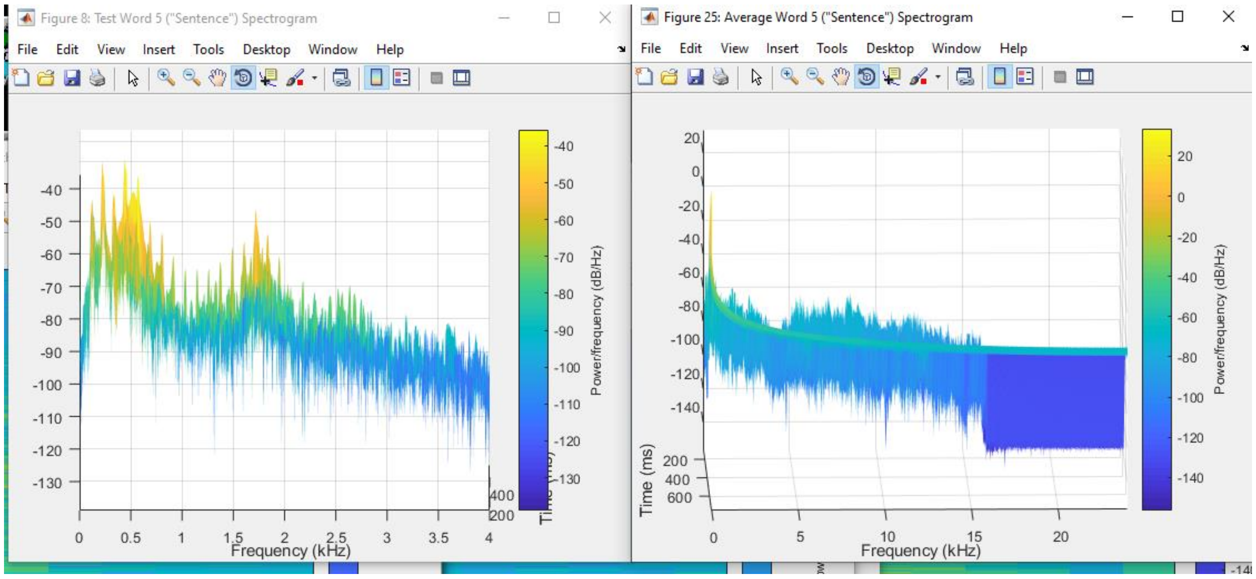


Figure 8: Test Word 5 ("Sentence") Spectrogram
Figure 25: Average Word 5 ("Sentence") Spectrogram
File Edit View Insert Tools Desktop Window Help
Frequency (kHz)
Power/frequency (dB/Hz)
Time (ms)

Average Comparison Vs. Test Word 6, With

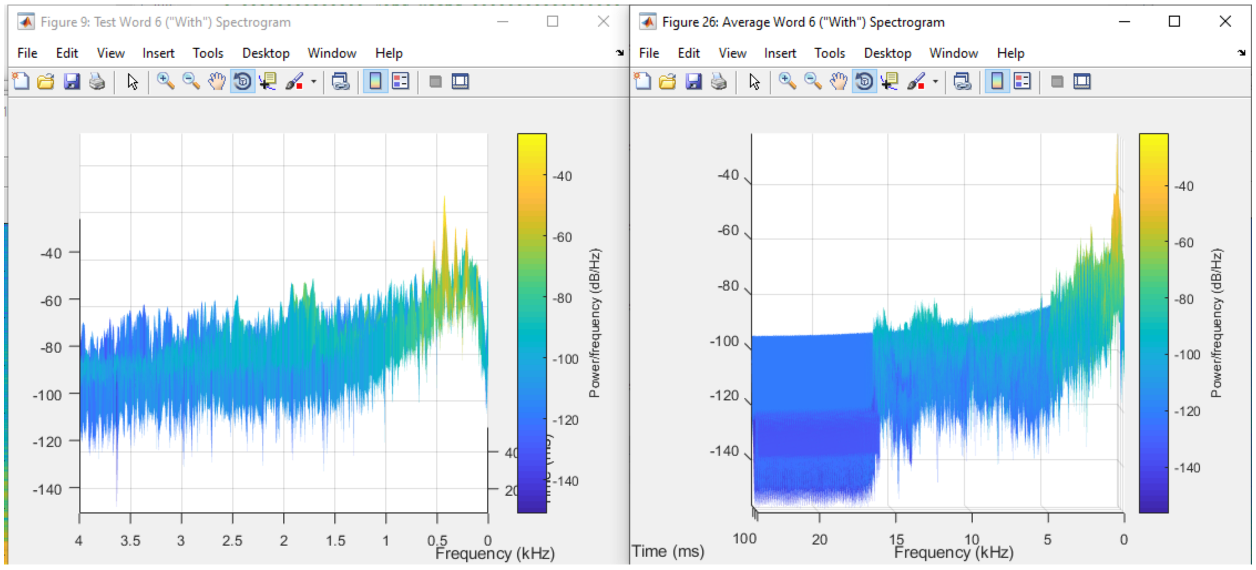
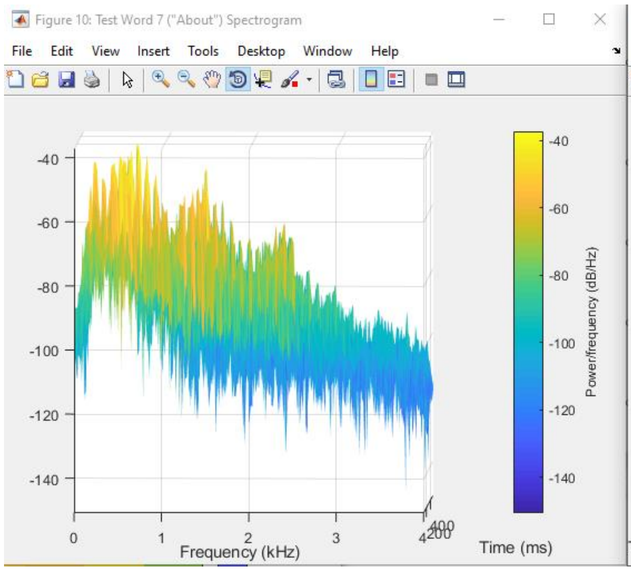
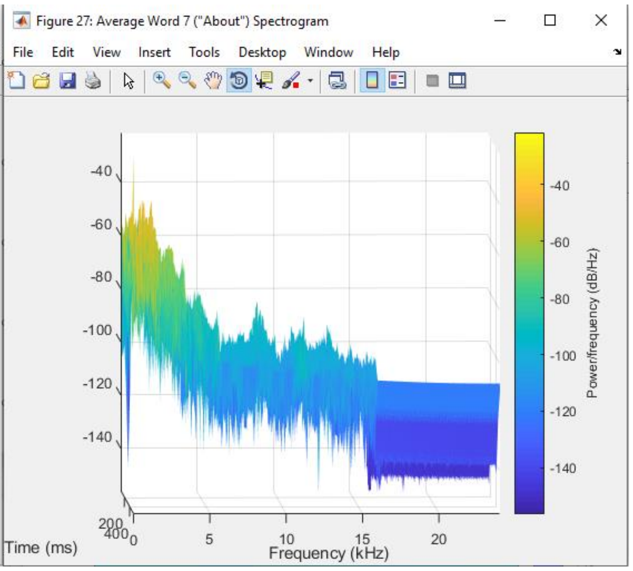


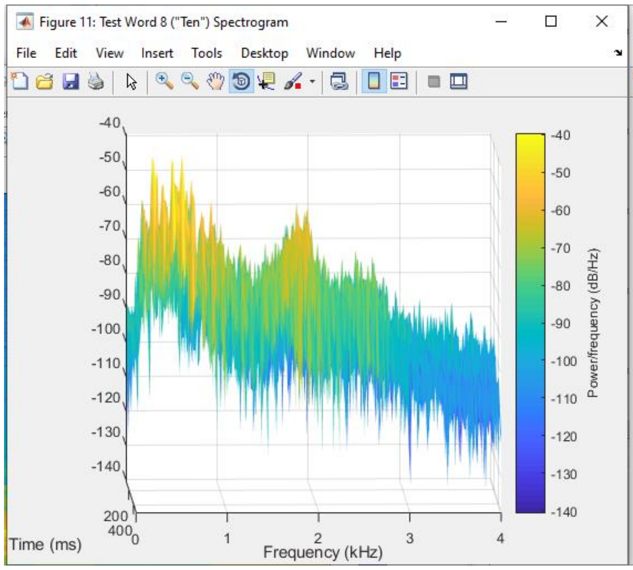
Figure 9: Test Word 6 ("With") Spectrogram
Figure 26: Average Word 6 ("With") Spectrogram
File Edit View Insert Tools Desktop Window Help
Frequency (kHz)
Power/frequency (dB/Hz)
Time (ms)

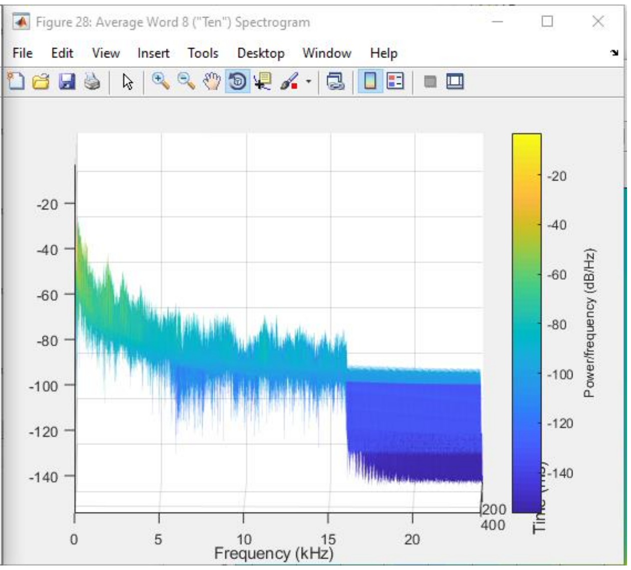
Average Comparison Vs. Test Word 7, About



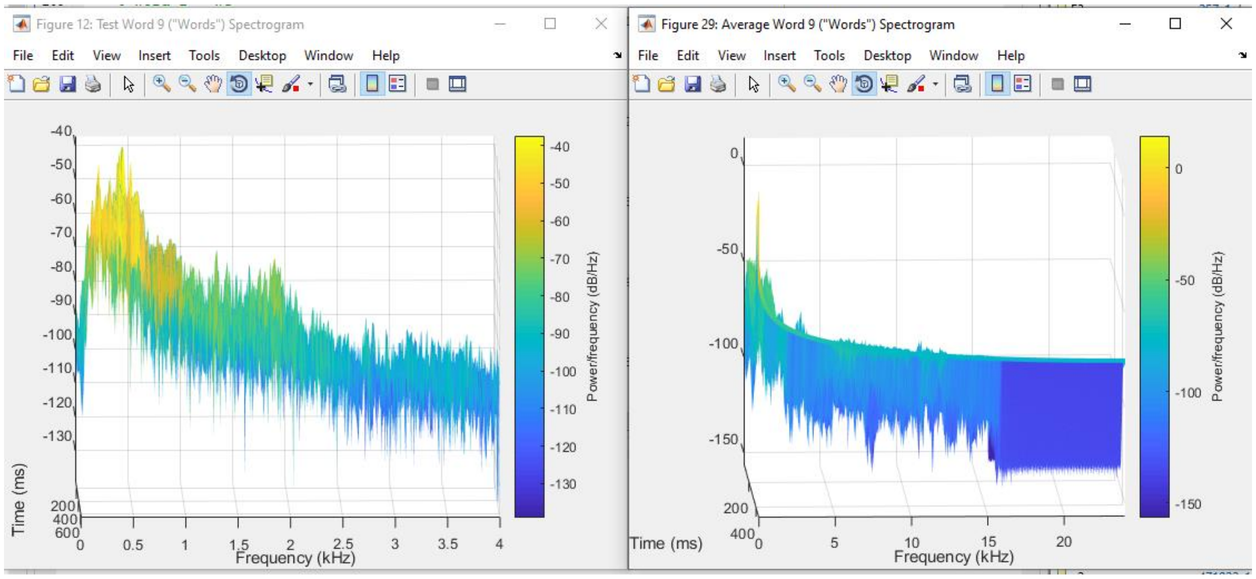


Average Comparison Vs. Test Word 8, Ten



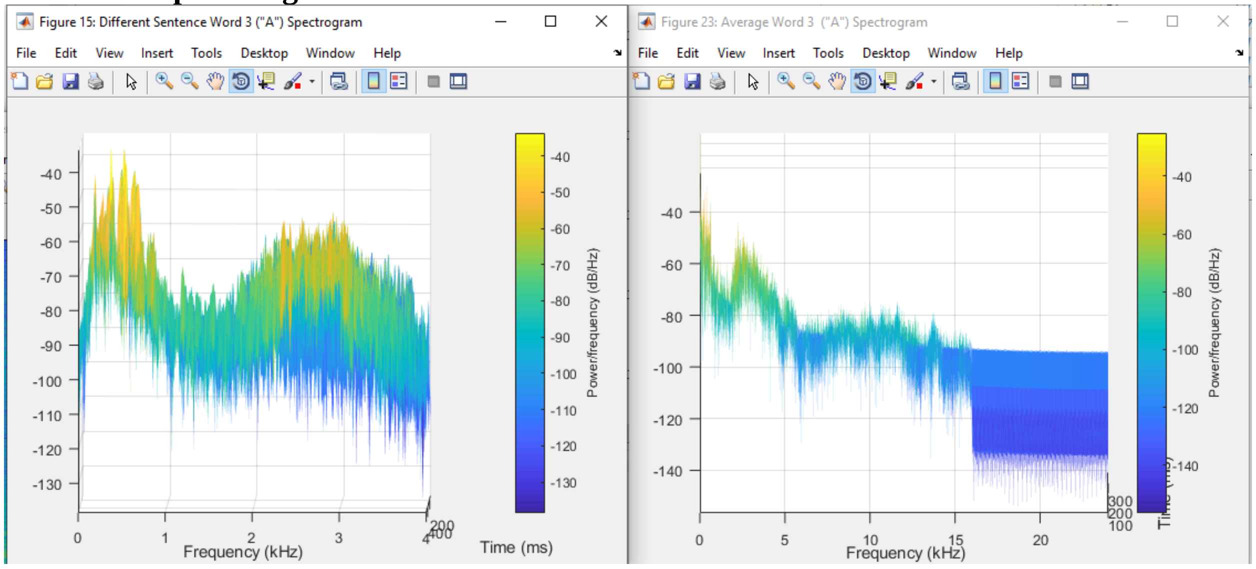


Average Comparison Vs. Group Member Test Word 9, Word

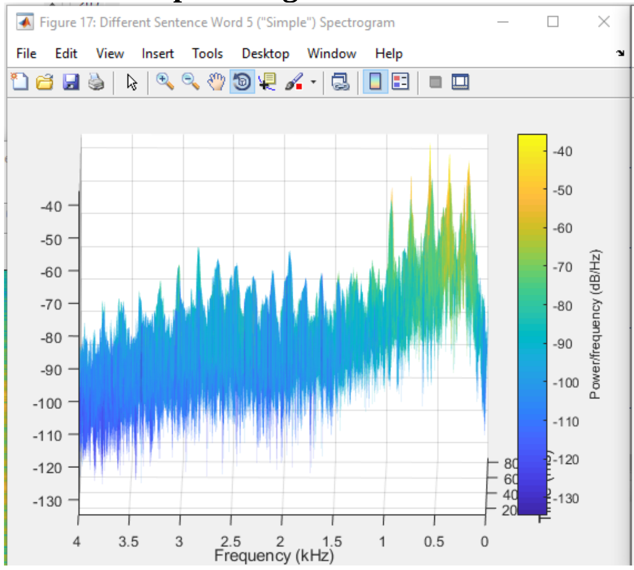


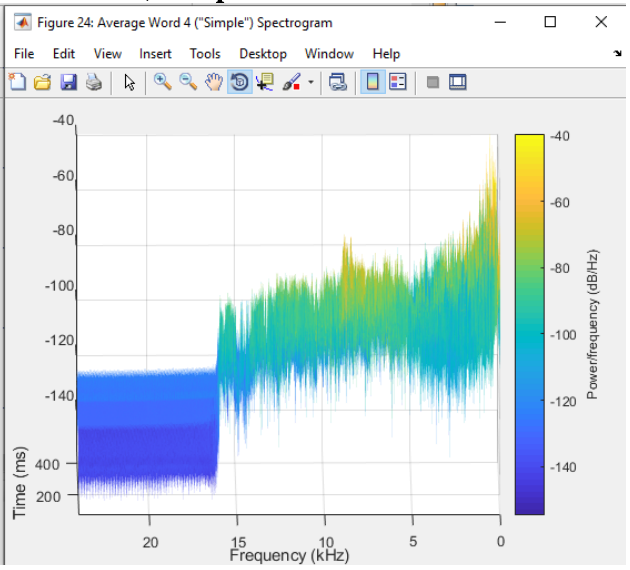
Each following graph is a side-by comparison of the same word between a student from outside of the group and the averaged recordings of the group members.

Group Average Vs. Outside Student Test Word 1: A

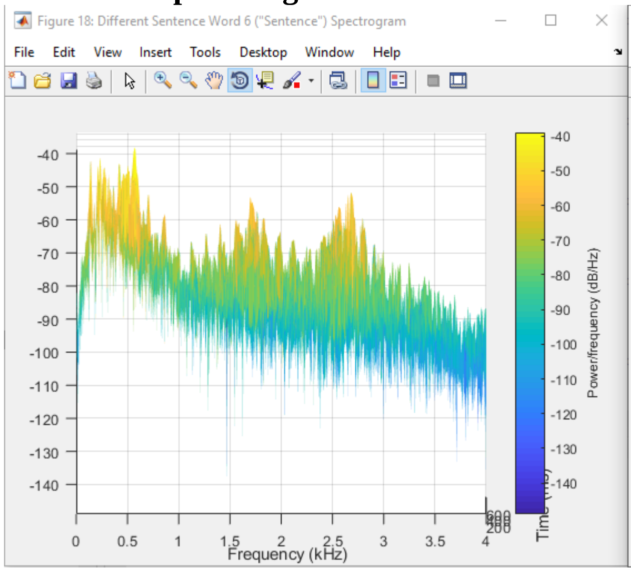


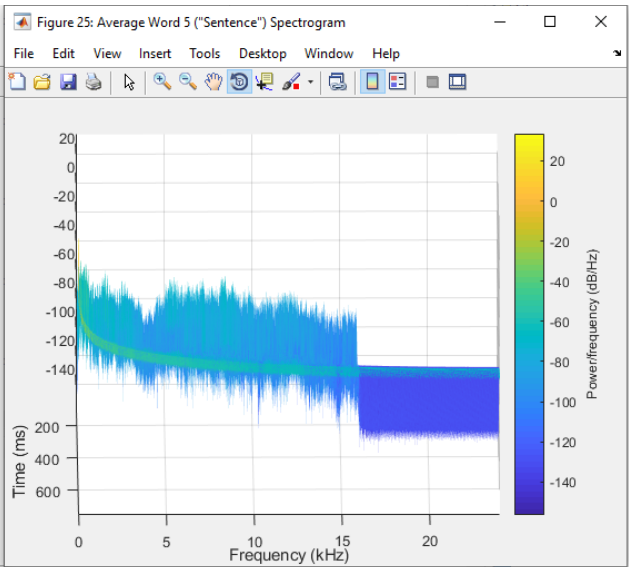
Group Average Vs. Outside Student Test Word 2, Simple





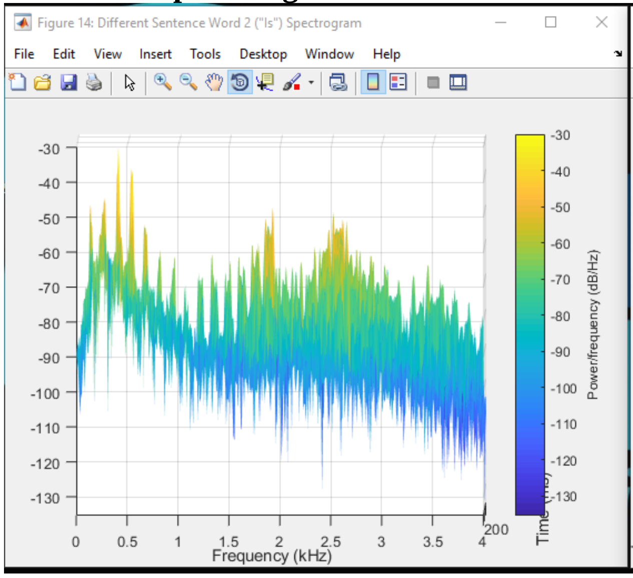
Group Average Vs. Outside Student Test Word 3: Sentence

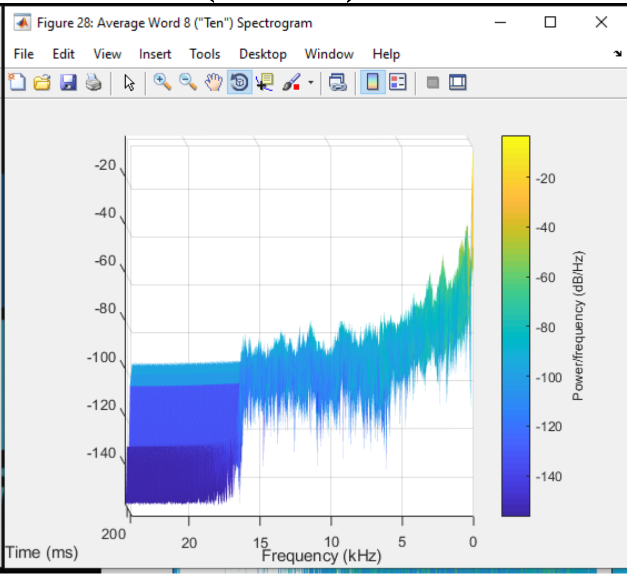




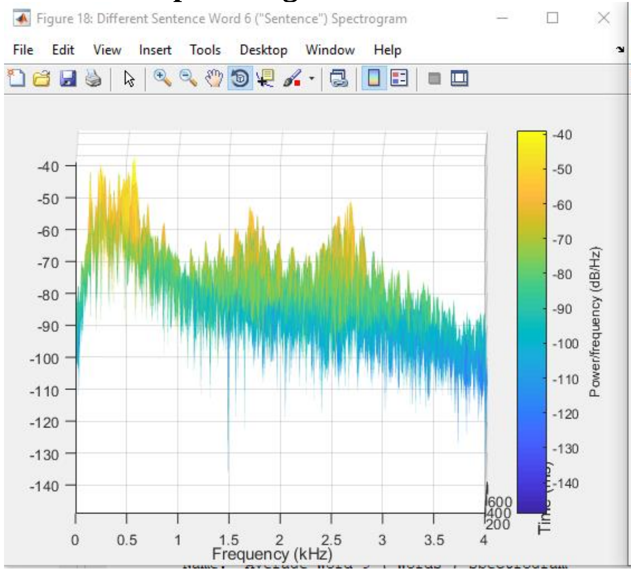
Each following graph is a side-by comparison of two different words between a student from outside of the group and the averaged recordings of the group members.

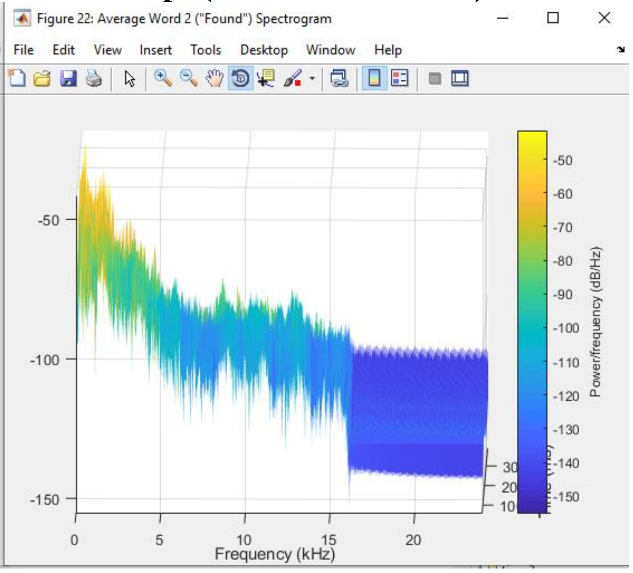
Group Average Vs. Different Word – Different Skew (Is vs. Ten)



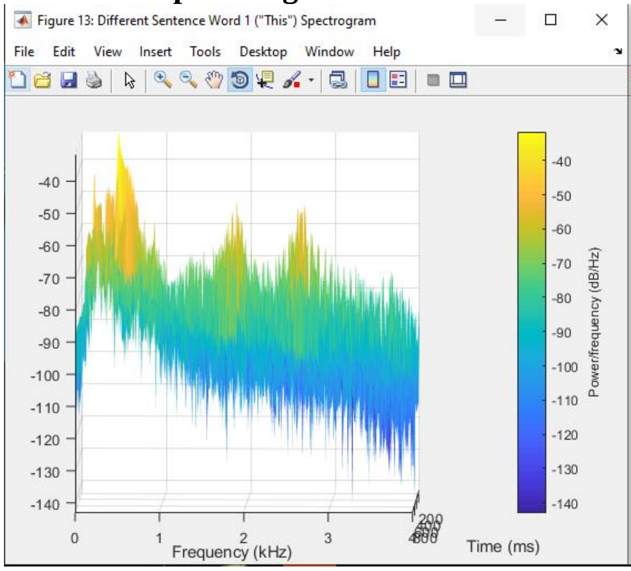


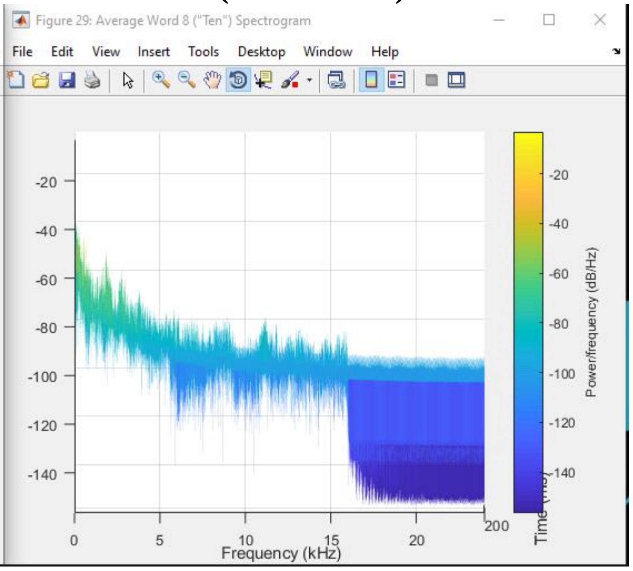
Group Average Vs. Different Word – Dramatic Dips (Sentence vs. Found)



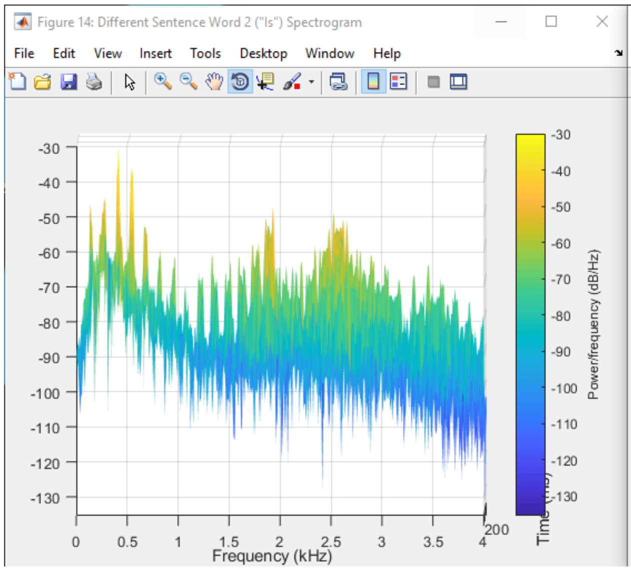


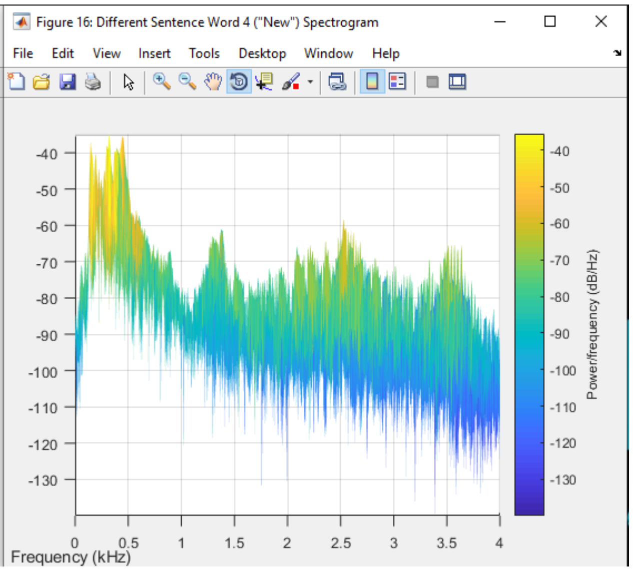
Group Average Vs. Different Word – Max Value Test (This vs. Ten)





Different Word "Is" Vs. Different Word "New" – False Positive





Errors.

As we can see in the image “Different Word “Is” Vs. Different Word “New” – False Positive”, words can have similar frequency/power patterns that can induce false positives. When looking at this image, we can see that the maximum value test, the power/frequency range test, and the direction/shape test all come back as true. However, the two images represent the frequency sequence of two separate words. The biggest issue regarding accuracy was the distortions in frequency between two similar voice signals. This distortion is the margin of error in the accuracy between the average voice signal and the similar voice signal being compared.

Discussion

Explanation of Results.

An important point is that the data being used to evaluate the different words is very limited. Only three separate samples were used for each value, and the data was not properly balanced (ex: two male voices, but only one female voice). With more audio recordings for each word, the data would be a lot more accurate. For this reason, this program cannot be used for the general public in its current state.

Therefore, the method and program being used can only be used to identify similarities between words. This concept is a fundamental idea in understanding the basis of what voice analysis is and does. The data shows how the different spectrograms display crucial information relevant in understanding a voice signature. This experiment shows how the average of several variations of a similar word can be used to begin identifying that word from the general public.

Future Improvements.

The best way to improve this method of identifying words based on the frequency rate would be to add more sample data to work with. This way a more accurate overall average of voices could be reached so that new data can be tested by more reliable information. Another improvement would be to incorporate this technique into a much larger technique so as to accomplish the task of voice analysis with a minimum margin of error. One idea for this would be to somehow get smoother data that encompasses more potential variables corresponding to the voice signal source.

Outside Source Analysis.

Through the formulation of these matrices of averages, a standard spectrum is formulated for that specific signal. This standard spectrum can then be used to represent a base representation of what the standard spectrogram of that word would be. This base can then be used to identify similar vocal signals from other sources. To test this, the average spectrum for the sentence used in this study was compared to a spectrum of the same sentence, but from a different signal source. Through visual analysis, the comparison test between the average and actual spectrogram lead to a logical match, with some margin of error. This margin of error can be seen by the slight variations in the magnitude of the power/frequency between the data. Essentially, while the technique of using the matrix of averages is a good start in voice analysis, this large margin of error is too variable between words from different vocal sources, making it unpractical as a standalone solution. This variation can also cause false positives, as can be seen when testing the standard spectrum against a different sentence from a different voice signal. A false positive is when the data appears to look the same but the voice signals are in fact not the same.

Conclusion

Spectrogram analysis is a vital component in signal analysis. This experiment has demonstrated a technique that utilizes spectrogram analysis to make predictions of a specific voice signal/word. MATLAB® is a powerful software that has an extensive library of signal and spectrogram analysis tools and algorithms. The technique starts by splitting three voice samples that contain the same voice signals, into individual signals. The three voice signals are then combined to form one signal of which the average is then computed of this combined signal. Utilizing this average, voice signals can be recognized as long as the average is similar to this signal as well. This technique formulated within the experiment helps to form a basis of voice recognition. While this technique is not the most accurate representation of voice analysis, the technique still helps to formulate a base for working towards a fully capable voice recognition system.

Appendices

A. MATLAB Functions Used

   a. round()
      i. Round to nearest decimal or integer
   b. audioread()
      i. Read audio file
   c. spectrogram()
      i. Spectrogram using short-time Fourier transform
   d. length()
      i. Length of largest array dimension
   e. linspace()
      i. Generate linearly spaced vector
   f. figure()
      i. Create a figure window
   g. audiowrite()
      i. Write audio file
   h. max()
      i. Maximum elements of an array
   i. interp1()
      i. 1-D data interpolation (table lookup)
   j. cat()

      i. Concatenate arrays

   k. mean()

      i. Average or mean value of an array

   l. subplot()

      i. Create axes in tiled positions

   m. title()

      i. Add title