Reversing Word Order in Spoken Word

Nathan Guan
ECE 160
May 17, 2019

***Abstract*—In this challenge, we were presented with an audio file and were tasked to segment the file into the individual spoken words. This was done by extracting features from the audio file to identify when each of the words start and end. Once all the words had been identified, we would then reverse the order of the words such that if the original file was "One, two, three", we would output "Three, two, one". However, we were only able to produce the wanted outcome when the input audio signal is slow.**

I. INTRODUCTION

In this challenge, we were tasked with reversing the order of words spoken in any given audio file. To complete this challenge, we must identify the words that are in the audio file. To do so, we must extract features from the audio file and using those features, try to segment the audio file into the words. One possible way to do this is to use block-based analysis on the signal and extract different features from the signal as we analyze each specified block of the signal. This approached was used as it made analyzing the signal much easier as we are only looking at a small part of the signal at a time. This allows for a more overall accurate result as we were able to obtain data we would not have otherwise obtained.

II. METHODS

A. Low-Pass Filter

Before any analysis was done, we first used a low pass filter on the input audio signal. This was done such that we can filter out the higher frequencies of the signal. We know that the word is most prominent at higher frequency signals, but we do not necessarily know when the word ends at lower frequency signals. This is especially true if the words are spoken quickly. By applying a low-pass filter, we can concentrate on the lower frequency signals to find the start and the end of words in the audio file.

B. Block-based Analysis

Instead of analyzing the entire signal, we implemented block-based analysis. This was done by first deciding what the block and hop lengths were going to be. With those variables set, we can find the number of blocks needed to cover the entire input audio signal. The number of blocks could be found by flooring the quotient of the length of the audio file and the hop length and then subtracting it with the ceiling of the quotient of the block length and the hop length. Once the number of blocks needed to cover the signal, given the block and hops length, is found, we can apply each block by giving it a start and end position. The start position for each block can be found by the current block number, n, minus one multiplied with the hop length plus one. The end position for each block can be found by looking at the minimum between the amount of audio left or the start position plus the block length plus one. From the calculations, we will have successive blocks that overlap with each other slightly. This allows for there to be some sort of continuity between the blocks that have been analyzed.

C. Features

From each block, we will attempt to extract different features to use them to differentiate when words start and end. The first of the features used is average energy. The average energy of a block of the audio signal is found by taking the mean value of the data in that block and squaring it. From this, we can plot the resulting array of data points extracted from the input and the result is a graph that somewhat represents a half wave rectified version of the input audio file. From here, we can go one step further to make the data points more pronounce such that we can see the breaks between syllables and words more clearly. This is done by adding 0.00001 to the average of the block and taking the logarithm of it. Now if we were to graph the logarithm of the energy, we would have a similar graph to that of energy but with more exaggerated peaks and troughs. Next, we calculate the spectral flux, which is the measure of how fast the power of a signal is changing. This is calculated by comparing the discrete Fournier transform of the current block with that of the previous block. This is done by taking the sum of the absolute differences of the values of the DFT of the current block with that of the previous block, raised to the p, where p is the norm type. With the summation of all those values, we will raise that to the 1/p and our result will be the spectral flux for that block.

$$\text{flux}(t) = \left(\sum_{k=b_1}^{b_2} |s_k(t) - s_k(t-1)|^P \right)^{1/P}$$

D. Segmentation

To segment the audio file into separate words, check to see if the thresholds that we set are met. These thresholds represent whether we believe the word has either started or ended. The thresholds are determined by looking at all the features and carefully finding values that most accurately represent the start and end to a word.

III. Results

From the above method, we were able to come up a multitude of results. Listed below are the two main categories in which the results classified themselves into:

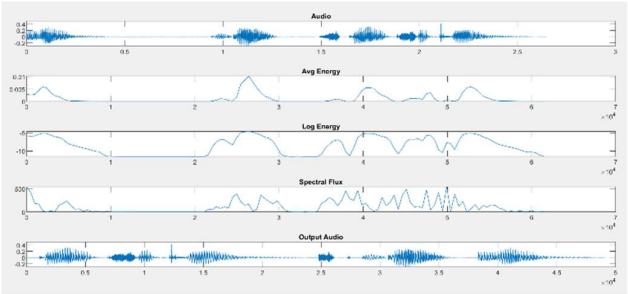

Figure 1: Fast audio signal "One death, strike a king(?)"

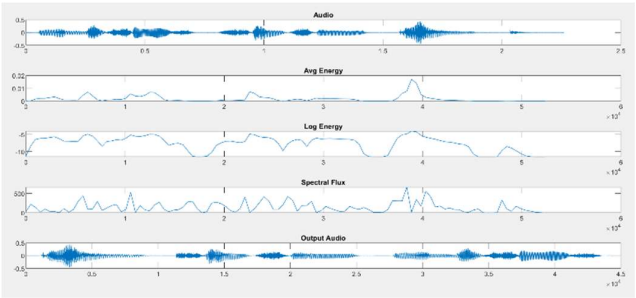

Figure 2: Slow audio signal "Ulysses Simpson Grant"

The algorithm mentioned above works extremely well when the words in the audio signal is slow and well enunciated. Due to the nature in which the words are spoken, the algorithm can easily use the previously established threshold values and mark when words start and end. This creates a seamless reversal of the words spoken in the audio file.

When the speech is faster, however, it is harder to find the difference between two words. The algorithm assumes that the two (or more) words are just one long word due to the threshold values never being reached. As a result, a segment may potentially contain a phrase instead of just individual words.

IV. Conclusion

From the resulting data, we can conclude that the algorithm works the best with audio input that have slower and enunciated speech patterns. If the audio file contains a phrase that is too fast or is slurred together, it will not be able to detect the differences between the words. This may potentially be fixed with the use of more features in the segmentation portion of the algorithm. With more features and a weighting system, the threshold hold values will be much more accurate at pinpointing when the words start or end. All in all, the current methods listed provides a way for the words in a audio file to be reversed but only when the words in the audio file are well enunciated and are spoken with pauses.

V. Acknowledgements

Nathan Guan wrote the code, analyzed the results, and wrote the report