# Speech Recognition

# Data Extraction and Cleaning

Phillip Spratling

## Introduction

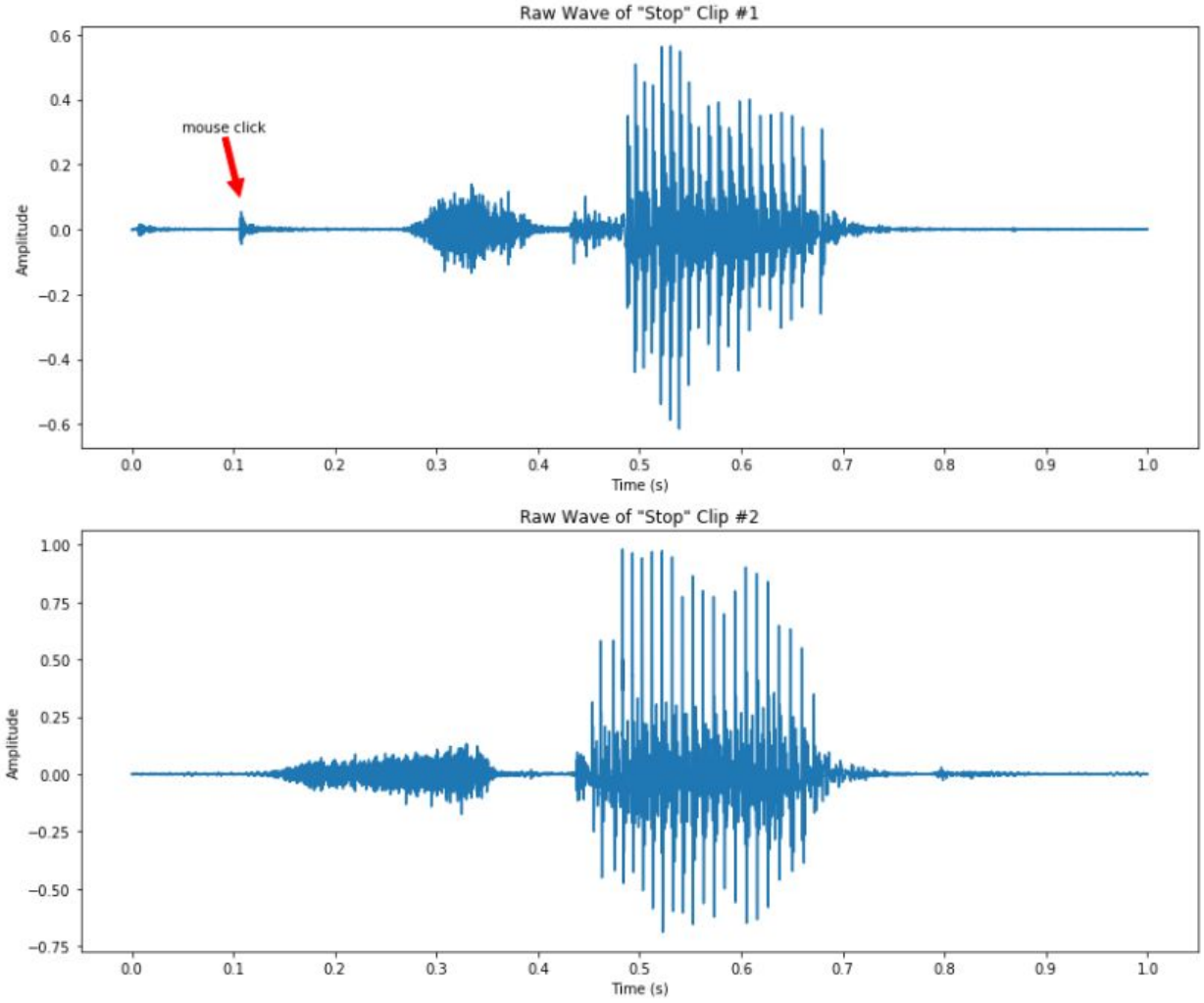
Speech recognition technology has come a long way in the last couple decades. Until recently, it was predominantly performed using methods such as hidden Markov models, but now deep learning neural networks have proven to be much more effective. Because of the improvement in performance, speech recognition in devices is begging to show up everywhere. Previously used mostly in military applications and telephone call processing, you can now find sophisticated speech recognition being used in smartphones, cars, televisions, and even refrigerators.

In this project, I will be using the Speech Commands Dataset from TensorFlow to build a speech recognition algorithm that understands 10 command words and will understand whether a sound clip is silent as well. It will also know to classify sounds that aren't one of these 11 categories as "unknown". These words were chosen to be used as a speech interface, so they are mostly commands such as "yes," "no," "stop," "go," etc.  The goal is to build an algorithm that could be used in a simple speech recognition application, or for the foundation of a more sophisticated application.

## Data Cleaning and EDA

The raw speech data was gathered from the TensorFlow website. The data consists of 35 words, each with around around 500 samples. Each sample is approximately 1 second long and was sampled at a sample rate of 16,000. The data was originally in WAV files, so I needed to do some preprocessing before they could be fed into the end models. Additionally, since the clips are not exactly the same length, I had to pad or delete the ends of all the clips so that they are exactly 1 second long. Once the samples were ready and separated into training, validation, and testing sets, I could begin to process the audio data.

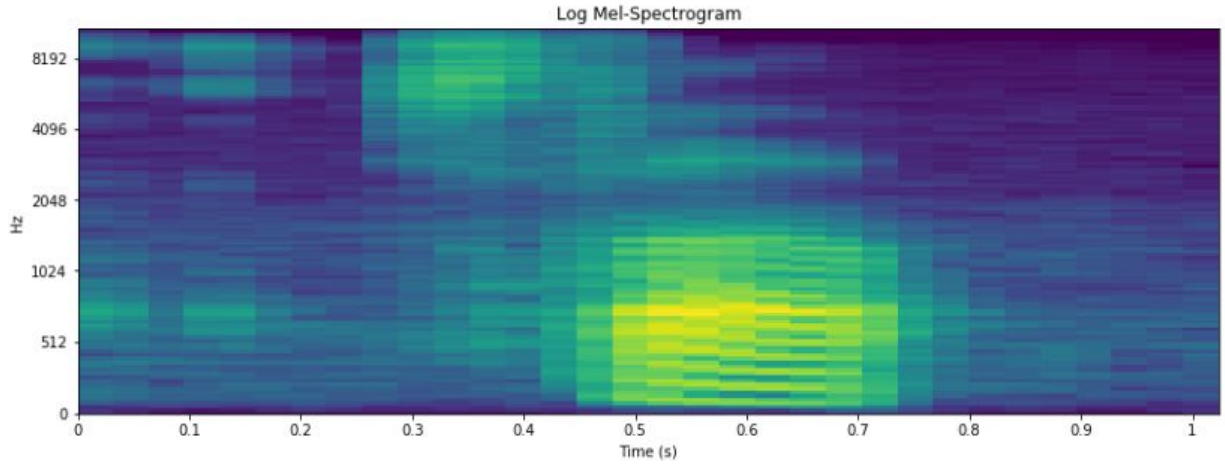
There are a few ways speech data can be processed. The first and the simplest is by extracting the WAV files into the basic wave form frequencies. One sample is then a one-dimensional vector with each data point indicating an amplitude at a certain point in time. This method is visualized below for two sample for the word “stop.”




It is important to note that the amplitude is unitless when extracted by librosa - the tool used to process the audio samples. This method is the simplest, but will not produce as good results as other more sophisticated methods.

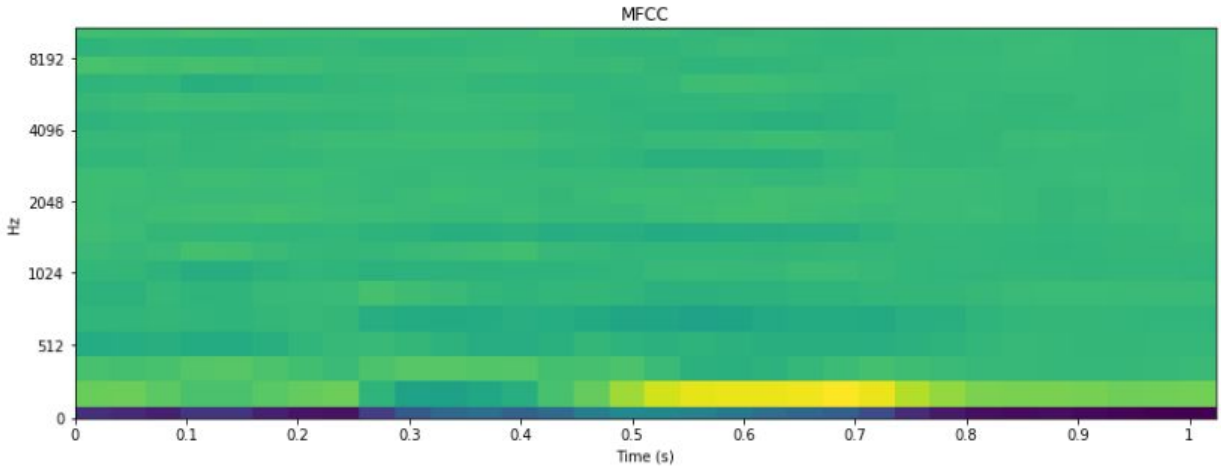
The second method is to extract the samples into their log-spectrograms. This adds a dimension of complexity to the raw wave forms - it is a 2 dimensional array of amplitudes at

different frequencies at different points of time. This method is more complex, but will produce much better results than the raw wave data. This method is visualized below for the first sample of the word “stop”:




The more yellow colors represent a louder noise at that frequency at that point in time. Already we can see the benefit of the added dimension - this spectrogram shows that the “s” in “stop” is in the higher frequencies while the rest of the word is in the lower frequencies. This added information would be lost in the raw wave form data, so with this method we should get better results.

The final method of data extraction used is the Mel Frequency Cepstral Coefficient (MFCC). MFCCs are a feature widely used in speech recognition, and are designed to extract information about linguistic content from audio while ignoring background noise. These are used in most speech recognition systems today, and so we will use these for our final model. One sample of the same word “stop” using this technique is visualized below:




Although we appear to lose some information, these features will probably work the best when it comes to classification with neural networks, while also using less data than the raw spectrograms.

In addition to the speech samples, there were 6 samples of background noises provided to use as “silence” data. These were around a minute long instead of the second long speech samples. To use these as data for the neural networks, I extracted at random 1 second clips from each of these samples, and then multiplied the raw wave data by a random number between 0 and 1 to simulate different noise levels. This was done until I had an equal number of silence samples as the other words. These were then converted to the MFCC form like the speech samples to use in the neural networks.

**Next Steps**

In the upcoming section, I will feed the extracted MFCC data into separate neural networks. I will start with a simple single layer perceptron and a feedforward neural network to get an idea of base level performance. I will then design a more sophisticated convolutional neural network as well as a LSTM network to get the final speech recognition system.