Andrew Case

Mini-Project Write-up

4/20/2015

The project that my partner and I made is a basic speech recognition system that can be used to recognize numeric digits spoken individually. The speech recognition being done by the system is very simple due to its limited dictionary of recognized words. If the system created were to be given more training data and conditioning such that it could deal with a greater pool of users it would be similar to a speech system used by an automated phone service that handles call routing for many services. The system itself was made through a combination of LabVIEW and Matlab. The Matlab Section of the project handles the actual processing of the signal and is used to decide which digit was spoken. The LabVIEW portion of the project handles the majority of audio signal input and conditioning. Additionally, the portions of the project that were handled in Matlab are integrated with the LabVIEW portion through LabVIEW’s Matlab script node. Each step of the speech recognition process is used as a part of a pipeline in LabVIEW that runs in a loop. Using the VI generated by LabVIEW the user can interact with the speech recognition system and see the result through a number pad that lights up the number selected by the pipeline. Using the button labeled “Begin Recording/Start Processing” the user can repeatedly provide input to test the accuracy of the speech recognition system and see an immediate result.

The method we decided to use for speech recognition for this project was to differentiate spoken words based off of their Mel Frequency Cepstral Coefficients (MFCC). Although there are several methods of extracting relevant features from an audio signal, such as the hidden Markov model (HMM), we chose this approach based on its perceived simplicity of implementation. This method of speech processing takes the frequency spectrum and extracts a set of coefficients based on the energy contained in consecutive, overlapping sections that increase in width for sections at higher frequencies. It is based on the values of these coefficients that the closest matching entry in the system’s dictionary is selected. A more detailed implementation of the algorithm follows.

This method begins by taking a periodogram estimate of the signal’s power spectrum. After converting the input signal to the frequency domain we divide the frequency band that contains relevant speech information into different bands. In our implantation we used the band from 0 Hz to approximately 8000 Hz. The positioning of these bands is made so that adjacent bands will overlap and the spacing the bins themselves will be uniformly spaced on a logarithmic scale known as the Mel Scale. The formula to convert between linear frequency and the Mel Scale is M(f) = 1125ln(1 + f/700). After dividing the spectrum in bins, each bin is extracted using a set of hamming windows collectively known as a Mel filter bank. A figure displaying what a typical Mel filter bank would resemble is shown below.

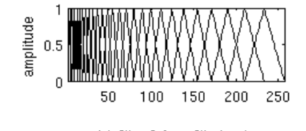


Figure 1: A typical Mel filter bank. Notice that bin size increase for higher frequency bins and that each bin besides the lowest bin begins at the midpoint of the previous bin.

After the filter bank is applied to the original signal the energy contained in each bin is summed up and the DCT of the logarithm of all of the filter bank energies is taken. The DCT is taken in order to eliminate the correlation between successive terms that was present in the sequence of bin energies due to the overlapping nature. The coefficients of the terms of the DCT are used to classify the signal and are the where the MFCCs are produced from. The implementation used by our system only uses the first 12 coefficients as the sources that we used while researching claimed that higher coefficients have been shown to decrease performance in practice.

The implementation that we made was formatted as a pipeline in LabVIEW. When the main VI is started the MFCCs of all ten digits that constitute our dictionary are calculated outside of the main loop. This is done outside of the main for loop because it is only done once and the coefficients will not change between iterations of the loop. Within the main loop the first stage of the pipeline is to take in the audio to be processed. Depending on the settings of the front panel the audio is either taken from a wave file or from a specified microphone input. Selecting the microphone input will allow the user to record 4 seconds of input before processing of the input begins. After the input is taken the input is passed through a lowpass filter with a cutoff frequency of 4 kHz and a stopband frequency of 5 kHz. The filter is a Butterworth filter and is generated before the loop begins using a filter VI (virtual instrument) provided by LabVIEW. After filtering the signal is normalized such that the maximum value of the input signal is 1. The normalized signal is then down-sampled from a sampling rate of 44.1 kHz, the rate used by wave files, to 8820 so that the sampling rate matches the one that we used in the Matlab file. The down-sampled signal is passed from LabVIEW to a Matlab file that computes the MFCCs and the coefficients obtained are compared to the coefficients of the words in our dictionary in order to determine the closest match. Finally the result is displayed on the front panel of the LabVIEW VI and the pipeline begins waiting for the next input.

The final product that was created for this project is a relatively simple speech recognition system. While the final result was what I expected the final project would produce it still leaves much room for improvement. The primary reason that the project can still be improved is because both partners had a limited amount of time to work on the project and the scale of the project had to be revised as the project developed. If there were time to develop the project further I would change the way the speech is processed so that the input is broken up into smaller frames of time. Then based on these frames I would extract the MFCCs to determine what sub-word sounds the input signal is composed of. When these sub-words are combined it would allow the system to more accurately recognize words. Additionally this would allow us to convert the system into one which takes input continuously, take words in succession, as well as increase the size of the dictionary. One other thing that I would do given the time is to increase the amount of conditioning that is done on the input signal and the amount of training done to the recognition system. This would allow the number of potential user to increase due to the decreased reliance on dialect and speaking rate.

The project work was divided such that I handled all of the design done in LabVIEW while my partner, Ted, handled the majority of the project that was done in Matlab. Although this is how the coding work was divided, the research that was done for the project was carried out cooperatively so that the both partners had an idea of what was being done. Additionally the overall design decisions, direction of the project, and time constraints were decided on iteratively through regularly scheduled meetings. A large part of the project was done at these group meetings, while the remainder was done individually and merged using a repository hosted on git hub.