ECE 420 Final Project Report

Speaker Recognition

Extra Credit Video Link: https://youtu.be/jKwpYE7E61I

Team Members:

Hiraal Doshi (hiraald2)

Richard Nai (rnai2)

Francis Roxas (roxas2)

Introduction

Speaker recognition systems have a wide range of applications such as authentication for security purposes, forensic applications amongst many others. The project that this paper will describe is a speaker recognition system. The system identifies the name of the speaker if they have been registered in the codebook, otherwise the output is that the speaker was not recognized. In order to perform the recognition of the speaker, different components of the sound are utilized, and a real-time speaker recognition algorithm was implemented.

Literature Review

In order to implement the real-time speaker recognition, the system required the use of an MFCC processor, an LBG algorithm, and a nearest neighbor search. The basic implementations of these algorithms were derived from the work of Kinnunen, Karpov, and Franti [5] and the paper written by Pradeep [6].

The work of Kinnunen, Karpov, and Franti describes a functionality of all of the components required for this project. The methods they describe also focus on optimizing the time for recognition, since it is highly dependant on the amount of speakers on the codebook. Given a large amount of speakers, there are even more feature vectors to compare in order to find the best match, if it exists, in the codebook. In order to implement the optimizations they described, they reduced the amount of test vectors by removing unlikely speakers during pre-quantization. Afterwards, the best variants found during the pre-quantization are modeled using a Gaussian mixture model. In order to extract the features needed for the training and recognition, a MFCC processor is used.

During the training stage for the recognition model, a clustering approach is used by Kinnunen, Karpov, and Franti. The algorithm they use to implement the clustering of the training vectors is the LBG algorithm mentioned before due to its efficiency. The result of the clustering is called a codebook (a set of M clustered vectors).

To implement the classification (recognition) of speakers, Kinnunen, Karpov, and Franti compute the average quantization distortion is computed, and the resulting vector with the smallest distortion is classified as the best match. A vantage-point tree is used to perform the nearest neighbor search in an optimized time-frame.
The other reference used, Pradeep's paper, uses a very similar approach to that of Kinnunen, Karpov, and Franti. He describes the implementation of a system that incorporates the use of a MFCC processor and the LBG vector quantization algorithm, which are the same main components of the other source.

Pradeep elaborates on the functionality of the MFCCs. His system makes use of an MFCC because it follows the human's ear response to sound. This makes it applicable to voice recognition, and MFCCs can also be used to distinguish between an recording of a voice and the actual voice input. That is to say, a MFCC for the recording would differ from that of the actual voice, which makes recognition more accurate and secure. Pradeep notes that the model is limited by a single coefficient having a very large vector quantization, but it can be mitigated by making use of high quality audio devices in a noise-free environment.

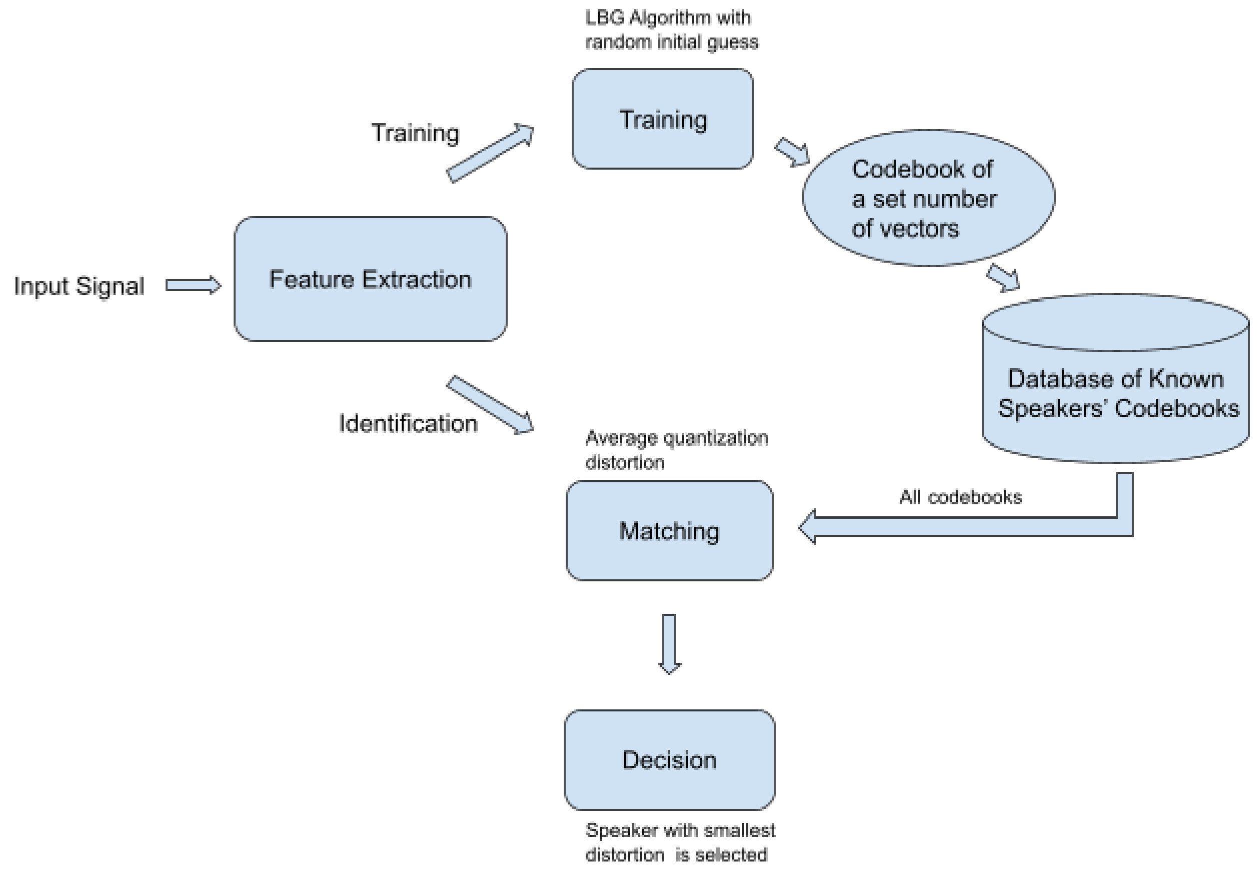
Pradeep's implementation clustering method works on the basis of clustering a speaker's acoustic dimensions into a vector, known as a codeword, and storing it in the codebook. To get the best match, he simply uses the euclidean distance from an unknown (input) speaker. The smallest euclidean distance is then referred to as the the vector quantization distortion, as mentioned previously. He states that he made use of the Euclidean distance due to its simplicity and ease of implementation, despite the tradeoffs that may result regarding the time complexity.

The project described in this paper makes use of both the basic, detailed model developed by Pradeep and the optimizations developed by Kinnunen, Karpov, and Franti.

Technical Description Project

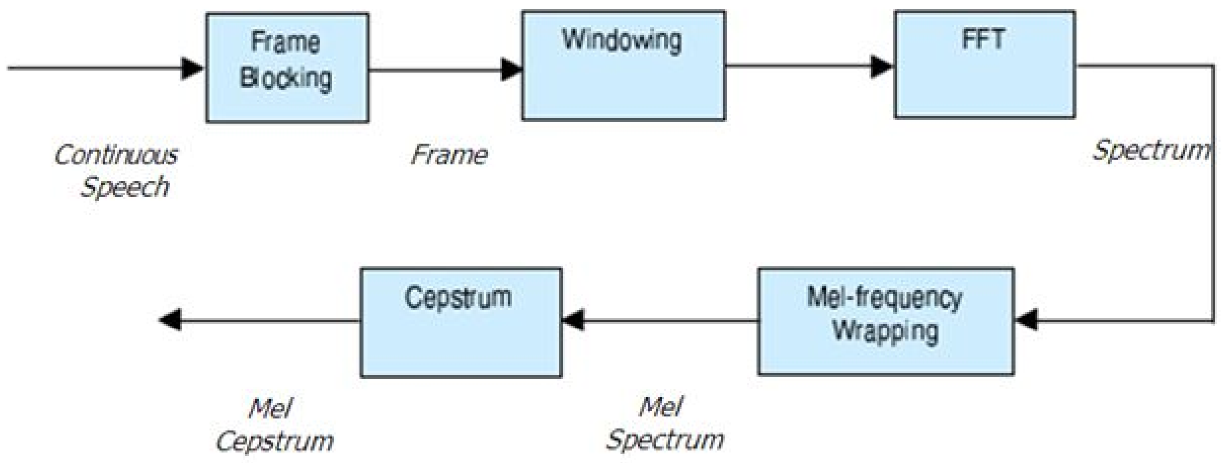
Below we will outline the algorithms we used to complete our final project.

Block Diagram for Speaker Verification and Identification


LBG Algorithm with random initial guess
Training
Training
Codebook of a set number of vectors
Input Signal
Feature Extraction
Database of Known Speakers' Codebooks
Identification
Average quantization distortion
All codebooks
Matching
Decision
Speaker with smallest distortion is selected

Feature Extraction: Mel-frequency cepstrum coefficients processor (MFCC Processor)

Implementing the MFCC Processor was done in 5 separate parts which are shown in the block diagram below and further elaborated upon underneath the block diagram.



Frame Blocking:
In the frame blocking step, we took our continuous speech signal in real time and divided it into frames of sample size 1024 samples. If the sample power spectrum was below a certain threshold, we labelled the sample as unvoiced and went to sleep for the next sample.

Windowing: We applied a hamming window on the input speech frames we collected in the previous step to get a cleaner FFT spectrum.

FFT: After applying the hamming window to each of the frames, the speech signals were converted to frequency domain using the FFT.

Mel- Frequency Wrapping: In this step, we converted the frequency domain signal found in the previous step into the mel frequency domain. By converting the frequency domain signal into a mel domain signal, we are logarithmically “stretching” out the information in the signal. Doing this allows us to linearly apply a filter bank to the signal in order to keep only the most vital information of the speech for classification.

Cepstrum: A special filter bank with triangle functions at frequencies that the human ear is sensitive will be used on the resultant FFT spectrum. This extracts more information from the signal. In the assigned lab we had previously done this step by collecting 14 mel frequency components from the result of the multiplication with the first 14 output coefficient vectors. However, we found that our tablet worked better when we collected 20 coefficient vectors since the distinguishing factors of human speech between speakers are found in the higher frequencies.

Classification of Pre-Recorded Speech Data

The training portion of our project was based almost entirely on a KNN search. In our algorithm we used a set our coefficient vectors that were obtained from the Cepstrum part as our training vectors. The algorithm we implemented for classification of pre recorded speech data is outlined below:

1. A vector codebook was designed. This vector codebook would be empty at first but would append MFCC coefficient training vectors to itself as data was recorded. As the training data was read by the codebook, we would manually label the training data using the user generated label on the tablet. This label is also known as the name that the we assigned to that sample of the training data

2. After this, we created a nearest neighbor classifier using OpenCV using 1 neighbor. For each training vector, we found the codeword that was closest to it in the current codebook. This vector will be assigned to the corresponding cell (associated with the closest codeword).
    a. The measure of closeness between two different vectors will be determined by the L2 norm of the two vectors.

3. If the distance between the input training vector and the nearest neighbor were too large, the tablet would output “unrecognized speaker” instead of the nearest neighbor.

4. We repeat the above steps for both real time input speech signals and preprocessed input speech signals.

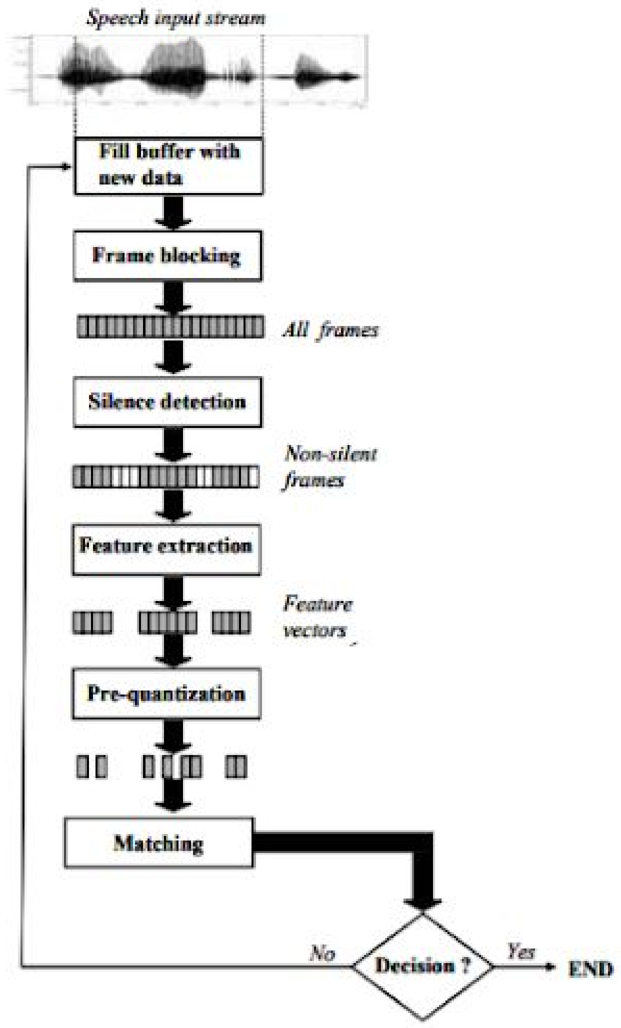
Real-Time Speaker Identification and Verification

The real-time speaker identification and verification algorithm was very similar to the MFCC Processor algorithm. We will outline this algorithm below:

- The algorithm begins by creating a buffer to contain newly found speech data from a microphone.
- Afterwards, the data from the buffer will be broken up into overlapping frames of size 1024.
- We then went through all of the frames and determine which frames are “voiced” frames and which are “unvoiced” frames.
    - The distinction between a “voiced” frame and an “unvoiced” frame will be determined by comparing the total energy of a given frame to a threshold. If the energy exceeds the threshold, then the frame is voiced. If not, it is unvoiced.

- Next, we will go into a feature extraction block which takes the frames of data to generate 20 dimensional feature vectors which will be used for classification.
- We will then match the feature vectors to the correct speaker using a KNN classifier.

<u>Block Diagram for the Real Time Speaker Verification Algorithm:</u>



Results
Testing done:

C++ Testbench:
We created a C++ testbench that compiled and ran on our development machine. It was a main function with access to the library functions we were using in our tablet code, and took in a predefined 1024 block of input and returned the computed MFCC coefficients using our method. The source code was symlinked to the Android project's source directory, and an openCV library compiled for the development machine (x86) was used in place of an openCV library compiled for the tablet's ARM. Thanks to this testbench, we were able to find all the bugs in our MFCC calculation algorithm, although the final MFCC terms were slightly different due to differences in DCT implementation between the .

Classification of Correct Speakers:
We tested our algorithm using our own respective voices as training data and using speakers that we found on YouTube. When recording our own voices, we at first repeated a bunch of key phrases (i.e. "the quick brown fox jumps over the lazy dog") to minimize the distance of the training and testing vectors used by the KNN for testing. Afterwards, we attempted to train the algorithm using stories that we read online and tested it using our normal conversations. Testing this was relatively simple since we were able to see what the results of the classification were almost immediately.

Classification of Unrecognized Speakers and Recognition of Unvoiced Frames
This was done in a similar manner to the above however when we wished to recognize an unrecognized speaker, we played a sample of a speaker on YouTube talking. We also had the tablet print out the average distances to what it was classified as for multiple YouTube speakers and then took the standard deviation of all of our data. With this data we would shift the thresholds for both of these outputs and repeat the process over again. Testing here was mostly trial and error.

Problems Encountered:

In our first milestone, which was to use precompiled data to generate a codebook upon startup to identify the speakers, we had poor identification accuracy, despite good accuracy in our Python assigned lab. We had recorded training data on a mobile phone, and then generated an identification codebook in Python. We then validated our identification accuracy against test data. Although the initial identification was noisy, we were able to clean the results up by averaging the classification within a small time window.

However, when we transitioned the algorithm to the tablet and began doing classification in real time, the classification accuracy was poor, even with a similar averaging technique to stabilize the reading. We observed that the mobile phone we used to gather training and test audio samples had a sampling rate of 44kHz, while the tablet had a sampling rate of 48kHz. In addition, our Discrete Cosine Transform implementation computed slightly different numbers than the Python implementation for the same inputs.

When we began our second milestone, which was to be able to add new speakers in real time, we found identification accuracy to have improved drastically. We attribute a number of factors to this improvement. First off, we were using the same hardware and implementation to record training data and test data, which improved consistency. In addition, we increased the number of MFCC coefficients used from 14 to 20, to capture more information in the higher frequencies.

Further Work / Extensions

Modifications

Future work on our algorithm can involve two major modifications that would increase the speed at which the training is accomplished and increase how well the tablet classifies unrecognized speakers. The first modification to our algorithm would be the blocking our input speech data into groups of threes or fours and averaging them. The idea is that we want to minimize the amount of deviance in our input training data to get a better classification of our speakers and also use less training data to recognize our speakers which would speed up the real time classification. The second modification to our algorithm would involve pruning vectors. This would involve implementing an upper bound on how many vectors we could read in for any one speaker and removing the training vectors that are farthest away from the centroids of each of the clusters in our codebook. This would allow us to get a better representation of our current speakers and make it easier to classify unrecognized speakers since the distance of the unrecognized speakers to the nearest neighbors will go up on average.

<u>References</u>:
[1] Do, M. (n.d.). DSP Mini-Project: An Automatic Speaker Recognition System. [online] Available at: http://minhdo.ece.illinois.edu/teaching/speaker_recognition/speaker_recognition.html. [Accessed 3 Mar. 2019].

[2] N. Singh, A. Agrawal and R. Khan, "Principle and Applications of Speaker Recognition Security System.", in Proceedings of CAIRIA, Lucknow, India, 2018, pp. 16-20.

[3] S. Tirumala, S. Shahamiri, A. Garhwal and R. Wang, "Speaker identification features extraction methods: A systematic review", 2019.

[4] M. Jin and C. D. Yoo, "Speaker Verification and Identification," in Behavioral Biometrics for Human Identification, IGI Global, pp. 264–289.

[5] T. Kinnunen, E. Karpov and P. Franti, "Real-time speaker identification and verification," in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 277-288, Jan. 2006

[6] P. CH, "TEXT DEPENDENT SPEAKER RECOGNITION USING MFCC AND LBG VQ." [Online]. Available: http://ethesis.nitrkl.ac.in/4398/1/i.pdf. [Accessed: 17-Mar-2019].