SPEAKER RECOGNITION SYSTEM

El Tirado

ABSTRACT

Speaker recognition is a popular yet challenging task. In the recent past, an amelioration of work related to audio processing became prolific, especially with the burgeoning popularity of machine learning and deep learning techniques combined with effortless access to large datasets. In this project, we implement a voice recognition system, which is very accurate and efficient, but a simple approach uses no machine learning or deep learning techniques. In fact, we show that our implementation yields 100% recognition accuracy in both training and test sets.

1 INTRODUCTION

Speaker recognition is the process of identifying the user based on extracted information from a given speech sample. This technique makes it possible to use the speaker's voice to verify their identity for authentication purposes. In this project, we have implemented a speaker recognition system using Digital Signal Processing (DSP) techniques that can learn from a set of speakers and identify if there is a known speaker in a given voice sample.

Specifically, our system can be broken down into two main parts: speech feature extraction and speech feature matching. We first adopt Mel-frequency Cepstrum Coefficients (MFCCs) [1] to extract a robust representation of a given voice sample (i.e. features). Then, the extracted features are used to generate codewords through the well-known Linde, Buzo, and Gray (LBG) Algorithm [2]. Finally, based on the concept of Vector Quantization (VQ), we match the MFCCs obtained from a given voice sample with the resulting codebook to classify and identify the corresponding speaker.

The content of this report is organized as follows. Firstly, Section 2 presents the design details and our methodology for the speaker recognition system, including feature extraction and feature matching. Afterward, Section 3 evaluates and shows the performance of our system by conducting experiments on several data sets. Section 4 discusses and provides more insight into our evaluation result. Finally, Section 5 summarizes and closes this work with conclusions.

2 METHODOLOGY

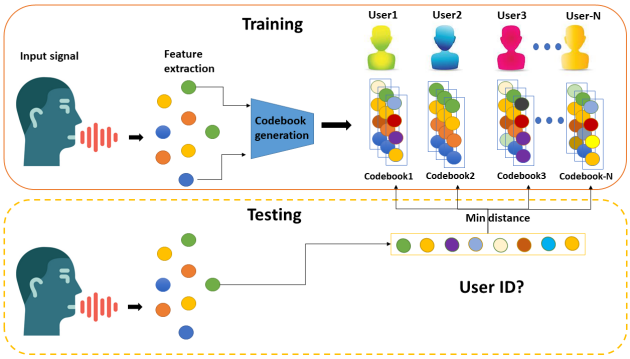
The implementation of a speaker recognition system follows two essential phases: the training phase and the testing phase. In the training phase, we have sample voice signals extracted from each user that we need to identify. In the testing phase, we test each incoming voice signal against the pre-stored users' features and try to guess the identity of the user.

Figure 1 illustrates the holistic view of such system. We model the problem as a classification problem of classifying a given voice segment to one of the N stored users. We utilize the training phase to extract the most crucial features from each user and keep that feature mapping for future use. Such feature mapping is known as a codebook.

Team El Tirado: Achintha Wijesinghe, Chih-Ho Hsu.

EEC 201 Project,
2022.

Figure 1: Overview of the speaker recognition system



The derivation of a codebook follows many steps. For clarity, we present each step as a block. As figure 1 depicts, extract the input from the user. These inputs are pre-extracted and stored under two datasets: a training set and a test set. The given training set consists of eleven users pronouncing the word zero, and a second sample from selected eight users is provided as the test set. Therefore, we read audio files from the corresponding directory depending on the phase. Once the audio file is accessible, we move to the feature extraction block from the voice files. The main problem we face in the feature extraction process is the differences in the duration of the voice signals. The situation can be exacerbated at the testing phase, and robust implementation against such variations is paramount. Similar to many voice recognition work in literature, such as [1], we utilize Mel Frequency Cepstral Coefficient (MFCC), for feature extraction.

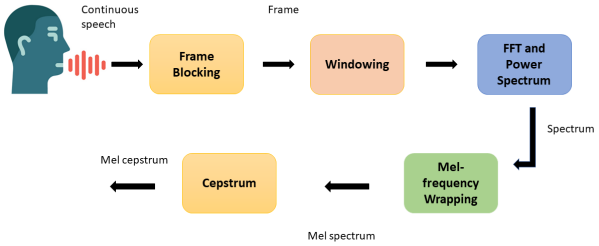
Next, we generate codebooks for extracted features using the clustering-based vector quantization method. In particular, we use LBG algorithm [2]. Once we have a codebook for each user, we move to the testing phase. We elaborate on each block in the rest of the subsections.

2.1 Feature extraction

As we discussed in the previous section, we use MFCC for feature extraction. The figure 2 portrays the functional blocks of a typical MFCC processor. In an incoming voice signal, the frequency changes over time. In many applications related to voice, processing after the arrival of the complete signal carries no importance. Therefore, in the beginning, we split the signal into overlapping frames, as it is reasonable to assume that for a given short time period, frequencies of a signal have a stationary property. It is empirically proven that such a method can accurately approximate the Fourier representation of the signal. As in any typical setup, we also represent one frame by (N) 256 samples and the number of overlapping samples to (M) 100.

Next, we apply a window on top of each frame. The usage of a window helps to mitigate any spectral leakage. There is a copious

Figure 2: Block diagram of a typical MFCC processor



amount of choices for a window, but we select the most used Hamming window for our implementation. The Hamming window is as follows,

$$w[n] = 0.54 - 0.46cos(2\pi n/(N-1)) \tag{1}$$

After applying the window, we take the Fourier Transform of the result. We can use the Fast Fourier Transform (FFT) as it is readily available in many languages or computer programming platforms. We deviate a little from the traditional approaches and compute the power spectral density (PSD) as it provides better and deterministic characteristics than FFT [3]. The PSD is calculated using the following equation.

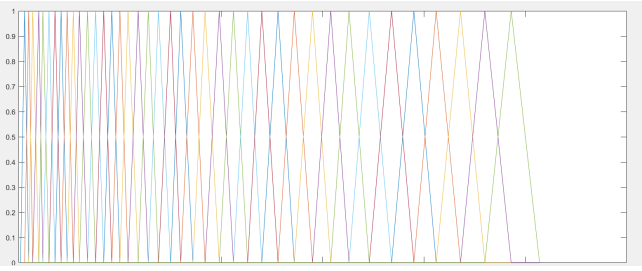
$$PSD = \frac{|FFT(frame_i)|^2}{N} \tag{2}$$

Where, $frame_i$ is the i^{th} frame of the split signal.

Next, we exploit the property of human ear sensitivity for our model. In fact, the human ear has a non-linear perception and this can be modeled using filter banks in the Mel domain (figure 3). For this, we use the following map to convert the frequency range of the signal into the Mel frequency domain and extract corresponding filter banks and send PSD frames through filter banks.

$$m_f = 2595 \log(1 + \frac{f}{700}) \tag{3}$$

Figure 3: Mel frequency filter banks



As per the final step, we apply discrete cosine transformation on the extracted filter banks. We omit the first coefficient and extract the rest. Furthermore, it is known that for automatic speech recognition, the most crucial information is embedded in cepstral coefficients 2 to 13. The fine details contained in other coefficients have no (or less) role in our problem and we can safely discard them.

Apart from these archetypal blocks, we perform some pre-processing at the input of the signal [1] and normalization at the output of the feature extraction block. The holistic view of the overall process is illustrated in figure 4. The 'Pre-Emphasis' block at the beginning amplifies the high frequencies of the input signal. Implementation of this block provides a good signal-to-noise ratio, stabilizes the numerical calculations of the Fourier Transform, and balances the frequency spectrum. The implementation of the 'Pre-Emphasis' is as follows,

$$y(t) = x(t) - \alpha x(t-1) \tag{4}$$

where α is a constant and $x(t)$ is the input signal. The traditional choice of α is 0.95. But, in our implementation we use $\alpha = 0.99$.

Figure 4: Overall view of the feature extraction process before feature mapping

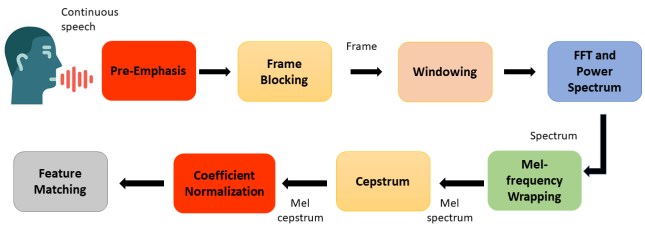


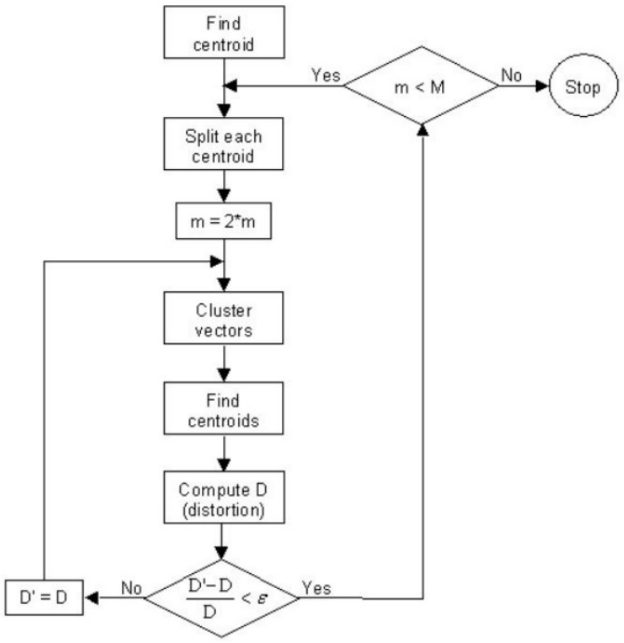
Figure 4 portrays the overall pipeline of the speaker recognition system. In the training and test phases, we follow the discussed process (up to coefficient normalization block on figure 4) for each training and test sample separately. We can then use these coefficients to represent the input signal as represented in figure 4. In the following subsection, we elaborate on the process of the derivation of codebooks and speaker recognition using these features.

2.2 Feature Matching

After extracting MFCCs from the input voice sample, we should then design a way to recognize the speaker based on these voice features. In this project, we adopt the VQ technique to address the problem owing to its effectiveness, efficiency and robustness.

VQ is a process of mapping vectors from a large vector space to a finite number of regions in that space. Each region is called a cluster and each cluster can be represented by its centroid, or codeword. This process can be accomplished by the well-know LBG algorithm [2], which cluster a set of L training vectors into a set

Figure 5: Flow diagram of the LBG algorithm



of M codebook vectors. The algorithm is formally implemented by the following recursive procedure as presented in figure 5.

To begin with, the algorithm will initialize a single-vector codebook representing the centroid of all training vectors. Then, it will double the size of the codebook by splitting each current codebook cb_n according to the following rule

$$cb_n^+ = cb_n(1+\epsilon) \tag{5}$$

$$cb_n^- = cb_n(1-\epsilon) \tag{6}$$

where ϵ is a splitting parameter and be set to 0.01 in this project. Afterward, we will update the centroid of each training vector to create a new set of codebooks. We can see that in each iteration, the size of the codebook will be doubled. The algorithm would repeat this process until the number of codebooks meet our requirement. That is, $M = 8$ in this project.

Since the obtained features (i.e. MCFFs) is a high dimension vector, it's very challenging to illustrate the data with the plain figure. Thus, in order to better present the algorithm process, we first reduce the dimension of obtained feature into four. Then, pick up two from these four dimensions as the x-axis and y-axis respectively. Finally, we pick up one user from the testing set as an example.

Figure 6 depicts the visualization of the LBG algorithm's executing process, where the 192 black dots represent the features extracted from the voice sample we selected of 192 frames, and green dots represent the centroids (i.e. codebook) from the LBG algorithm. From the subfigure (a) to (h), we can see the evolution of codebook in 1^{th}, 2^{th}, 3^{th} and 4^{th} iterations respectively. We can clearly observe the number of centroids split and evolve from 1 to 8 and then the iteration is terminated.

3 RESULTS

In this section, we present the performance of our implementation under several experiments. We have used several datasets for evaluation purposes. The original dataset contains eleven training samples representing eleven users, and the testing set have eight samples from eight selected users as mentioned above. In each sample a single user pronounces the word " zero ". We extend the dataset by including our own voices and in some cases with test samples contains pronunciation of other words than zero.

3.1 Evaluation on the original training set

The training accuracy of the model is calculated after extracting codebooks for each user. In the process of training set evaluation, we simply load training samples and calculate their MFCC. The calculated MFCC is then compared with the pre-stored user codebooks and find the distance from each sample to the saved codebooks.

Table 1 illustrates the distance calculated from each training sample to such user codebooks. We use Euclidean distance as the metric to evaluate the similarity. From table 1, we see that all the diagonal elements hold the minimum value for each row. This implies that our speaker recognition system can identify all of the speakers in the training samples accurately (i.e. 100% training accuracy).

3.2 Evaluation on the original testing set

After having desired performance on the training set, we evaluate our model on the testing set. The provided testing set is imbalance and has only eight data points representing the first eight users.

Table 2 summarizes the overall performance of the model against the testing set. We see similar behavior to the training set evaluation and our speaker recognition system correctly identifies all the testing users (i.e. 100% testing accuracy). This result preliminarily verifies the effectiveness of our speaker recognition system.

3.3 Evaluation on introducing new users

In this subsection, we further analyze the effect of introducing new users to our speaker recognition system by adding voice samples from two group members to the original training and testing set. Specifically, s12 and s13 are voice samples from Achintha and Chih-ho respectively.

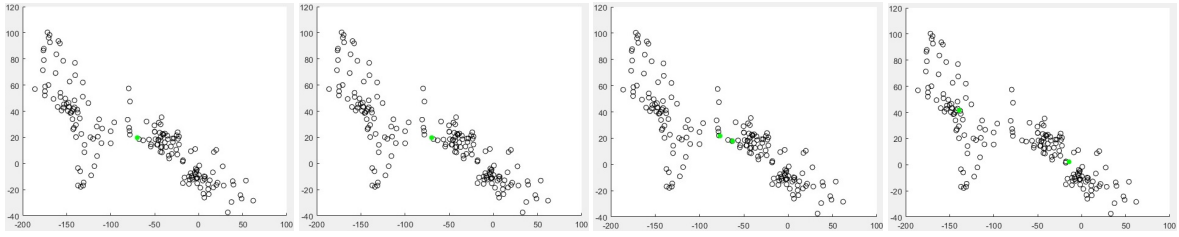
The evaluation result is illustrated in Table 3. We can see that our system successfully recognizes all of the users in the testing set. However, since the number of new users is limited in our experiment, we can only conclude that the performance of our system can preserve robustness to new users when the number of total users in the data set remains low. Hence, the testing accuracy remains at 100%.

3.4 Evaluation on the effect of different phrases

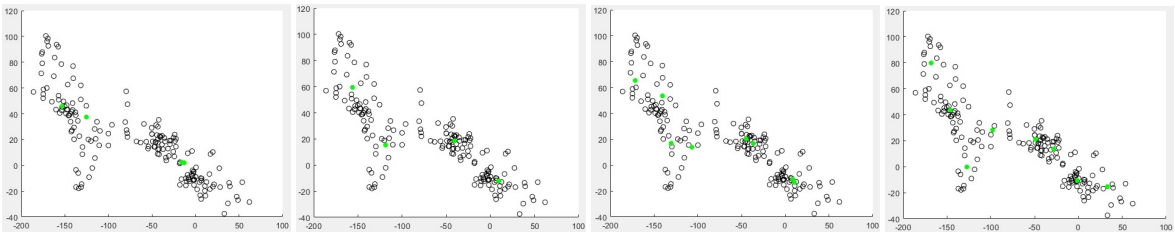
Next, we turn to test the ability of our speaker recognition when voice samples that contain different phrase are used. In this experiment, without changing other settings, we replace the test data of s13 with the voice from the same speaker saying a different phrase.

Table 4 shows the result when s13 speaks "Yes" instead of "zero" in testing data. We can see that the speaker s13 can still be successfully recognized with a distance value of 0.0283 even if the training

Figure 6: Visualization of LBG clustering algorithm. Black dots represent the features extracted from the voice sample. Green dots represent the centroids (i.e. codebook)



(a) Initialize with 1 codebook (b) Update centroid (c) Double the codebook by spliting (d) Update centroids



(e) Double the codebook by spliting (f) Update centroids (g) Double the codebook by spliting (h) Update centroids

True user	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	Prediction
s1	**0.0726**	0.0876	0.0960	0.0841	0.0837	0.0873	0.0818	0.0848	0.0886	0.1136	0.1198	s1
s2	0.0824	**0.0666**	0.0867	0.0810	0.0945	0.0989	0.0910	0.0974	0.1014	0.0972	0.1028	s2
s3	0.0917	0.0875	**0.0635**	0.0873	0.1022	0.0897	0.0859	0.0964	0.1017	0.0959	0.0982	s3
s4	0.0826	0.0837	0.0933	**0.0705**	0.0917	0.0949	0.0847	0.0852	0.0960	0.1032	0.1143	s4
s5	0.0779	0.0959	0.1022	0.0898	**0.0664**	0.0862	0.0878	0.0816	0.0808	0.1136	0.1179	s5
s6	0.0848	0.1032	0.0952	0.0925	0.0865	**0.0729**	0.0799	0.0801	0.0846	0.1095	0.1108	s6
s7	0.0816	0.0995	0.0950	0.0855	0.0921	0.0825	**0.0690**	0.0797	0.0875	0.1094	0.1180	s7
s8	0.0834	0.1029	0.1002	0.0863	0.0841	0.0839	0.0794	**0.0719**	0.0840	0.1101	0.1138	s8
s9	0.0893	0.0985	0.1074	0.0965	0.0829	0.0951	0.0892	0.0845	**0.0709**	0.0939	0.0918	s9
s10	0.0982	0.0922	0.1013	0.0969	0.0870	0.1093	0.0951	0.0939	0.0853	**0.0664**	0.0743	s10
s11	0.0991	0.0911	0.0923	0.0968	0.0925	0.1074	0.0994	0.0968	0.0894	0.0738	**0.0613**	s11

Table 1: Performance of the speaker recognition system against the training set. Each row illustrates the distance from the training sample to each training user. If we concentrate on the diagonal elements we get the minimum distance for that row. The column corresponding to the minimum distance in each row is re-coded as the predicted user. Here the distance is the L_2 norm.

data and testing data contain a different phrase. However, the recognition ability of other speakers has been affected accordingly. In this case, user s5 is falsely recognized as user s7 while user s1 and user s8 are on the edge of being falsely recognized as user s5 and user s9 respectively.

We have also tried saying different languages in the voice sample and observed similar results. Specifically, the distance in the table 4 (s13, s13) will become **0.0289** and **0.0362** for saying "*Go me na sai*" (i.e. sorry in Japanese) and "*wèi shén me*" (i.e. why in Mandarin) respectively. Thus, we conclude that the recognition can be valid for samples with different phrases when sufficient features can be extracted from the sample.

3.5 Ablation study

In this subsection, we perform an ablation study to show the importance of the " pre-emphasis " block. As we can see from the table 5, the model miss classifies user 1 as user 4. Furthermore, if we move along the elements in the rows s6 and s7 in table 5 and compare with the corresponding entries in table 2, the distance gaps in the entries of table 5 are small. That is, we can strengthen the gaps (i.e. confident levels of the predictions) using the pre-emphasis block.

3.6 Evaluation on the filtered test data

In this experiment, we derive a new testing set using the original testing set (i.e. eight users case) and evaluate the robustness of

True user	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	Prediction
s1	**0.0737**	0.0901	0.1020	0.0831	0.0789	0.0881	0.0799	0.0800	0.0881	0.1145	0.1215	s1
s2	0.0837	**0.0726**	0.0858	0.0812	0.0949	0.0976	0.0893	0.0951	0.0993	0.0966	0.1040	s2
s3	0.0893	0.0885	**0.0799**	0.0883	0.0929	0.0953	0.0859	0.0913	0.0913	0.0956	0.1019	s3
s4	0.0813	0.0880	0.0903	**0.0795**	0.0891	0.0924	0.0825	0.0839	0.0931	0.1064	0.1134	s4
s5	0.0800	0.0951	0.0943	0.0880	**0.0754**	0.0814	0.0854	0.0836	0.0845	0.1139	0.1183	s5
s6	0.0810	0.0993	0.0912	0.0920	0.0882	**0.0708**	0.0789	0.0817	0.0884	0.1152	0.1133	s6
s7	0.0816	0.0955	0.0942	0.0838	0.0872	0.0864	**0.0742**	0.0788	0.0876	0.1115	0.1191	s7
s8	0.0852	0.1010	0.0976	0.0868	0.0826	0.0807	0.0816	**0.0752**	0.0853	0.1103	0.1137	s8

Table 2: Performance of the speaker recognition system against the test set. Each row illustrates the distance from the testing sample to each trained user. If we concentrate on the diagonal elements we get the minimum distance for that row. The column corresponding to the minimum distance in each row is recoded as the predicted user. For example, if the true user is s4 (row), the minimum distance calculated across all the users is 0.0795 and it corresponds to the user s4 (column). Here the distance is the L_2 norm.

User	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	Prediction
s1	**0.0599**	0.0732	0.0829	0.0676	0.0642	0.0716	0.0650	0.0651	0.0716	0.0931	0.0988	0.0956	0.0913	s1
s2	0.0685	**0.0594**	0.0702	0.0664	0.0776	0.0799	0.0730	0.0778	0.0812	0.0790	0.0851	0.0982	0.0836	s2
s3	0.0740	0.0733	**0.0662**	0.0732	0.0770	0.0790	0.0712	0.0757	0.0757	0.0793	0.0845	0.0903	0.0807	s3
s4	0.0663	0.0718	0.0737	**0.0649**	0.0726	0.0754	0.0673	0.0684	0.0759	0.0868	0.0925	0.0930	0.0914	s4
s5	0.0651	0.0774	0.0767	0.0716	**0.0614**	0.0663	0.0695	0.0680	0.0687	0.0927	0.0963	0.0995	0.0869	s5
s6	0.0659	0.0807	0.0741	0.0748	0.0717	**0.0576**	0.0642	0.0664	0.0719	0.0937	0.0921	0.0865	0.1004	s6
s7	0.0664	0.0777	0.0766	0.0682	0.0709	0.0703	**0.0604**	0.0641	0.0713	0.0907	0.0969	0.0997	0.0869	s7
s8	0.0698	0.0827	0.0800	0.0711	0.0677	0.0661	0.0668	**0.0616**	0.0699	0.0904	0.0931	0.0935	0.0873	s8
s12	0.0750	0.0746	0.0818	0.0805	0.0733	0.0820	0.0731	0.0725	0.0738	0.0775	0.0756	**0.0658**	0.0946	s12
s13	0.0826	0.0808	0.0839	0.0780	0.0766	0.0889	0.0796	0.0796	0.0688	0.0700	0.0766	0.0902	**0.0442**	s13

Table 3: Performance of the speaker recognition system against the new test set. We introduce new two voices to both the training set and the test set (Our own voices). These new users are named as s12 and s13. We see our system perform 100% in this scenario as well.

User	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	Prediction
s1	**0.0535**	0.0779	0.0906	0.0655	0.0536	0.0736	0.0594	0.0645	0.0660	0.0941	0.1091	0.0986	0.0936	s1
s2	0.0669	**0.0528**	0.0661	0.0610	0.0738	0.0847	0.0723	0.0807	0.0834	0.0810	0.0901	0.0995	0.0877	s2
s3	0.0802	0.0767	**0.0619**	0.0687	0.0735	0.0872	0.0722	0.0791	0.0762	0.0738	0.0876	0.0901	0.0729	s3
s4	0.0629	0.0712	0.0774	**0.0584**	0.0620	0.0797	0.0602	0.0683	0.0721	0.0894	0.1063	0.0984	0.0937	s4
s5	0.0626	0.0866	0.0792	0.0629	0.0576	0.0649	**0.0540**	0.0614	0.0637	0.0979	0.1169	0.1070	0.0853	s7
s6	0.0627	0.0851	0.0774	0.0732	0.0651	**0.0540**	0.0573	0.0642	0.0651	0.0998	0.1036	0.0855	0.1072	s6
s7	0.0668	0.0822	0.0844	0.0616	0.0574	0.0736	**0.0560**	0.0629	0.0652	0.0918	0.1129	0.1084	0.0769	s7
s8	0.0708	0.0916	0.0868	0.0704	0.0615	0.0659	0.0583	**0.0600**	0.0603	0.0926	0.1070	0.0954	0.0795	s8
s12	0.0757	0.0707	0.0943	0.0754	0.0706	0.0943	0.0700	0.0717	0.0666	0.0750	0.0726	**0.0484**	0.1148	s12
s13	0.0847	0.0883	0.0622	0.0672	0.0705	0.0813	0.0686	0.0760	0.0745	0.0803	0.1051	0.1129	**0.0283**	s13

Table 4: Performance of the speaker recognition system when we use different phrases in the testing data of s13, where the data points in blue represent the distance that is very close to the groud-truth value. Predictions that are labeled in red imply false prediction.

our system against the frequency suppression in testing data. In particular, we use a notch filter centered at the frequency 1000 Hz, and vary the width of the notch filter and evaluate the performance. We first select a value for the width (as presented in the table 6) and create a notch filter and apply it on the testing data to create a new testing set to evaluate the performance. As it is evident from the table 6, our system stays robust up to approximately a width of 1250 Hz.

True user	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	Prediction
s1	0.0795	0.0857	0.1004	0.0767	0.0799	0.0948	0.0834	0.0812	0.0932	0.1099	0.1154	s4
s2	0.0812	**0.0756**	0.0827	0.0814	0.0885	0.0943	0.0862	0.0899	0.1036	0.1032	0.1133	s2
s3	0.0846	0.0873	**0.0782**	0.0859	0.0870	0.0911	0.0818	0.0852	0.1022	0.1052	0.1116	s3
s4	0.0818	0.0873	0.0889	**0.0775**	0.0863	0.0937	0.0834	0.0802	0.1000	0.1075	0.1133	s4
s5	0.0822	0.0901	0.0935	0.0833	**0.0783**	0.0884	0.0873	0.0852	0.0897	0.1080	0.1140	s5
s6	0.0785	0.0936	0.0835	0.0877	0.0858	**0.0707**	0.0756	0.0775	0.1046	0.1212	0.1212	s6
s7	0.0832	0.0903	0.0919	0.0800	0.0841	0.0901	**0.0776**	0.0790	0.0952	0.1113	0.1175	s7
s8	0.0870	0.0971	0.0940	0.0828	0.0851	0.0855	0.0831	**0.0771**	0.0931	0.1070	0.1082	s8

Table 5: Performance of the speaker recognition system against the test set without the ‘ Pre-Emphasis’ block. We see a miss classification of the user s1 as user s4.

Notch filter width ($f_s/2$ x)	Identified users (8)
0.1	8
0.2	8
0.3	5
0.4	4
0.5	4

Table 6: Performance of the speaker recognition system against filtered test data. Here we have used a notch filter to filer only test data. We vary the width of the notch filter width (factor x $f_s/2$) and applied it at 1000 Hz.

4 DISCUSSION

One of the key features of a speaker recognition system is to make a binary decision to handle inputs from unknown users. Therefore, we plan to include a binary decision at the beginning of the speaker recognition system in the next phase, as it is vital to detect whether the user already exists. The idea is to alleviate the miss classification of an unknown user with the existing users. For such implementation we expect access to more data to create a separate codebook for negative samples (i.e. unknown users). The empirical results suggest that thresholding on minimum distance cannot solely handle such decisions, but can detect if the user has a very distinctive accent.

5 CONCLUSION

In this project, we explored the concurrent use of Digital Signal Processing and pattern recognition in identifying the user of a given voice segment. Among an ample number of methods, we utilize the use of a simple and very efficient approach of vector quantization and show ideal performance in given datasets. We see such a method can achieve expected accuracy with no leverage on computation complexity, memory usage, and execution time.

REFERENCES

[1] L. Muda, M. Begam, and I. Elamvazuthi, “Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques,” *ArXiv*, vol. abs/1003.4083, 2010.

[2] Y. Linde, A. Buzo, and R. Gray, “An algorithm for vector quantizer design,” *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, 1980.

[3] H. M. Fayek, “Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what’s in-between,” 2016.