Speaker Verification
W2021 214A Project

Sunay Bhat, Pong Chan

University of California, Los Angeles
sunaybhat1@ucla.edu, pongchan222@gmail.com

Abstract

Digital speech processing has been an expansive field of research for many decades that has seen a wealth of new approaches ranging from deep learning techniques to other large scale data driven approaches in the last few years. Modern day technologies generate tremendous amounts of data to guide the design of such systems. At the same time, users are growing more accustomed to having their speech and voice interpreted in real-time with the advent of personal assistants like Siri and Alexa. Speaker verification is an important subset of this field which is specifically focused on identifying the speaker or determining if the speaker is the same given a reference.

In this paper, we explore a version of speech processing in which we train and test a text-dependent classifier which determines if two audio files are spoken by the same person in both clean and babble background noise recordings. We will cover a variety of features we attempted to use, from spectrograms to Cepstrum Coefficients, as well as scoring techniques in which we attempted novel approaches for discriminating between same and different file sets. We will share our final model and results, which utilize Mel Cepstrum Coefficients, Dynamic Time Warping, and Mahalanobis Distances to beat a baseline pitch detector. We will also spend time discussing many techniques we tried that did not succeed, opportunities for improvement, and intuition behind our attempted approaches.

Index Terms: speaker verification, MFCCs, pitch, GMM, HMM, K-means, Mahalanobis, DTW, FPR, FNR. EER, spectrogram, babble

1. Introduction

Speaker identification, or specially in our case, verification, is the task of determining whether a digital audio file is of the same speaker as a reference file. For this project, a sample classifier, training and testing data files, and a project outline were provided along with relevant course materials and instruction. The wav format audio files were of two types, "clean" with little to no background noise, and "babble" which contained conversational background noise, making the task much harder (as opposed to white or other more general noise types). The test data was segmented into two sets, one which was all "clean" data files, and one "multi" set which was 50% clean files and 50% babble files. These data sets were combinations of two audio files with a binary label identifying the pair as either the same (1) or different (0) speakers. The test data was similarly segmented into two sets, except one was clean data and one was only babble data. Labels were similarly provided to verify results after classification and generate relevant False Positive (FPR) and False Negative (FNR) Rates. As a result, there were four baseline train/test combinations: clean/clean, clean/babble, multi/clean, and multi/babble. Results will be shared using the above notation along with the error rate convention FPR%/FNR%.

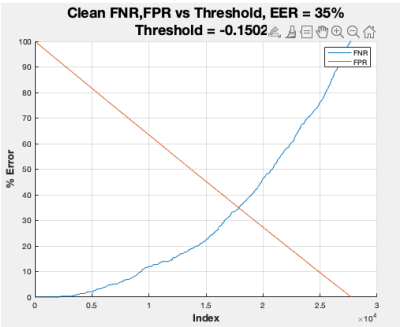
A baseline sample classifier was included with pitch detection. Table 1 below shows the baseline results that we were asked to improve upon to determine the effectiveness of our methods.

Table 1: Baseline FPR/FNR Error Rates

Clean/ Clean	Multi/ Clean	Clean/ Babble	Multi/ Babble
30%/ 36%	46%/ 37%	32%/ 33%	43%/ 42%

The provided scripts also included an Equal Error Rate Computation for scoring mechanisms. When determining a threshold in training, it is often important to balance FPR and FNR, where one can trade between the two error rates by moving the threshold as needed for the application. In general, a good practice is to utilize the threshold in which the two error rates equal each other, as can be seen in the Receiver Operating Curves (ROC) in Figure 1.

Figure 1: EER Receiver Operating Curve



Our methodology will discuss utilization of this Equal Error (EER) rate in more detail for feature selection, but it is important to note that this metric is relevant as our label quantities are highly mismatched. With 293 clean and 293 babble audio files and only 3-5 audio files per speaker, the probability of the speakers being the same in a given combination is quite low. This meant our training data sets consisted of a majority of examples with different speakers.One could get a decent Hamming error by trivially declaring the majority of examples as different with a low

FPR and a high FNR, but this would be a poor metric of the actual quality of the classifier. After making the brief mistake of including the Hamming error, we quickly discarded any consideration for it and focused on the valid metrics of FPR and FNR only for testing.

2. Background and Strategy

2.1. Model Background

Speaker verification models can vary drastically based on the application and techniques used. We will simplify it down into a couple stages. First, features relevant to the application are extracted. The literature is extensive, so we will not review the many features that one can extract for speaker verification[1][2], but we will briefly cover the features we tried in the next section. For the baseline, we were provided with a script that utilizes the multi-band summary correlogram-based pitch detection method [3]. Pitch, or the fundamental frequency, is an example of a commonly used and effective method of distinguishing speakers as it varies significantly between speakers [1]. In the baseline method, the mean of the pitch across the audio file (from voiced sections), was returned into a feature dictionary.

After feature extraction, the model is trained using any number of scoring techniques. The literature indicated this aspect of speaker verification is an extensive and complex problem space[6]. For the baseline, the mean pitch from the file pairs were subtracted, and the absolute value difference was saved as the score. Using this single value score, it was possible to compute the EER and set the threshold at that point.

Finally, the model is tested on new data. In the baseline case, the threshold was used to classify the computed scores in the test data. The resulting prediction vector is compared to the provided labels to determine the FPR and FNR results.

2.2. Strategy

Since we had limited time to work on this project, we wanted to utilize our time efficiently. Our initial strategy was to attempt pre-processing signals, to then explore and select features, and develop a scoring system for our classifier.

The literature indicated the pre-processing methods and features are well-known for this application[2]. A majority of our time should be dedicated to develop a novel scoring system.

After simple low-pass and band-pass filtering, we could not effectively de-noise our signal. In fact, the results were noticeably worse than the baseline results. We quickly moved on to selecting features for our classifier, as well as the scoring system that our classifier uses. In the following section, we will explain our philosophy behind testing and choosing the features and scoring system that we used.

3. Method

3.1 Feature Selection

The key to selecting features for this project is to make sure that the feature inherently gives good distinction of the two audio signals when they are spoken by different speakers. Additionally, the feature should also recognize when the audio signals are spoken by the same person with as little error as possible. To understand what feature can bring these properties, we had to first understand the physical properties of speech signals in general.

There were some ideas that inspired us to evaluate specific features. Theoretically speaking, autocorrelation of the two time signals and pitch of the two signals should be high if the two signals are correlated. That means that they should be overlapped if they are voiced (periodic). We would expect higher values when the speakers were the same, and vice versa.

Another pool of methods that might be helpful is to exploit image processing techniques. Although audio signals are often perceived as 1D, spectroam, mel-spectrogram, and other energy related plots are helpful when it comes to generating a 2D image. We believed that utilizing 2D data gave us more information so that there would be potentially more information. We employed centroid matching and dither correlation to align the spectrograms or mel-spectrograms of the two audio files by weight for a valid comparison.

Then, we used one of the most popular features in the realm of audio processing, MFCC (Mel-Frequency Cepstral Coefficients). We were very confident in the performance of this feature since it's been proven in many applications that it could work. In addition, delta (1st derivative of MFCC) and deltaDelta (2nd derivative of MFCC) have also been widely used along with MFCC[2][5]. However, we were aware of the drawback that MFCC does not perform well with noisy signals. After experimentation, we decided to use a 4.2ms window with 80% overlap.

Lastly, it made sense for us to use pitch as a feature. Intuitively, two different speakers should have distinct pitches. Nonetheless, pacing, mood, and many other factors can suppress the effectiveness of this feature.

Our goal to select the 'best' features would be to find a matrix of features that should naturally generate two non-overlapping distributions when the speakers are different people, and vice versa. That way, a unambiguous threshold can be drawn in between the two distributions so that the classifier can reference this threshold and label the signal pair as either 'same speaker' or 'different speakers'.

3.1.1 Feature Selection Method

A systematic approach to evaluate these features is to utilize the EER (Equal Error Rate) function. Ideally, we want to minimize EER for our choice of features.

3.1.2 Feature Experimentation

We computed a series of results with the EER function using the clean and multi training data. The scoring system in Table 2 was preliminary.

Table 2: Feature Evaluations - EER Results

	EER (Clean)	EER (Multi)
Baseline (F0 + MBSC)	35%	39%
Autocorrelation (Time + Squared Sum)	45%	50%
Autocorrelation (F0 + MBSC)	50%	48%
Mel-Spectrogram (SSIM)	43%	49%
Mel-Spectrogram (IMMSE)	45%	50%
MFCC + F0 (SSIM[7])	35%	-

Using MFCC and pitch with some simple scoring systems gave us the lowest EER. We then decided to further improve on this feature by implementing Dynamic Time Warping (DTW).

3.1.3 Dynamic Time Warping

After settling on the MFCC and pitch combination, DTW was utilized to match the feature set[5] individually across files and produce a distance vector for each data comparison pair. We utilized DTW specifically on the feature vectors themselves (MFCCs and F0), and not the time signals. Although we briefly tried that method, the computational time was on the order of hours from minutes for this method. In principle, DTW should improve our MFCC and pitch features' EER.

3.1.3 Final Feature Selection Results

We could now compare the distributions of the baseline features and the numerous features listed above.

Figure 2: Clean Train Score Distribution (Baseline)

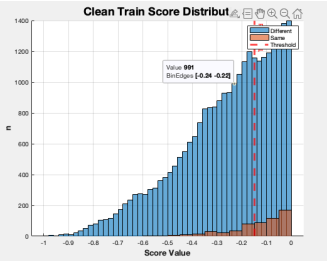
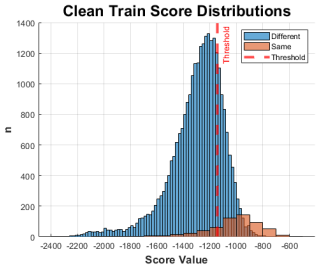


Figure 3: Clean Train Score Distribution (DTW MFCC + F0)



From Figures 2 and 3, MFCC related features and pitch with DTW have significantly more distinct distributions in which the thresholds result in lower EER than that of the baseline. We can also look at distributions generated with multi train-list.

Figure 4: Multi Train Score Distribution (Baseline)

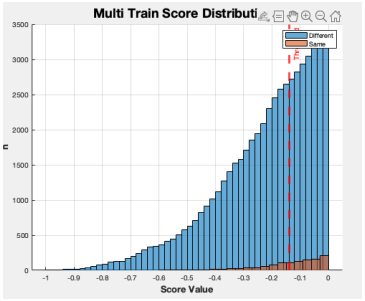


Figure 5: Multi Train Score Distribution (DTW MFCC + F0)

From Figures 4 and 5, MFCC related features, pitch with DTW have worse performance in EER than that of the baseline as expected.

Table 3: Feature Evaluations with DTW - EER Results

	EER (Clean)	EER (Multi)
Baseline (F0 + MBSC)	35%	39%
MFCC + F0 (DTW distance + Sum)	22%	49%
MFCC + F0 (DTW distance + log)	22%	49%
Combos of MFCC, delta, deltaDelta, F0 (DTW distance + sum)	22-35%	50%
PSTC + F0 (DTW distance + Sum)	37%	51%
Varying DTW number of features with constant weights (DTW distance + Sum)	22±0.7%	49±0.7%

Table 3 summarizes the final feature experimentation phase. We noticed that MFCC played a critical role in this `matrix of features'. Any other features yielded a higher ERR than MFCC + pitch in the clean data set. At this point, we decided to move forward with developing a more robust scoring system. Our final selected feature is a composition of 14 MFCCs and pitch (15 features).

3.2 Scoring for Training and Testing

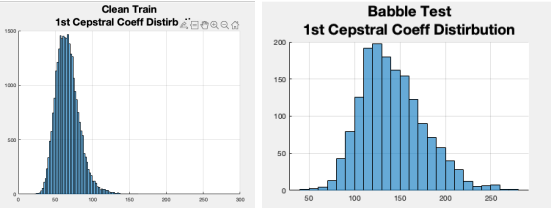
As discussed in our strategy, we wanted to focus on scoring due to the opportunity to experiment. After settling on our feature set and verifying the effectiveness using the EER, the next step was to develop a more robust scoring method using a variety of techniques. In order to utilize the provided ROC curves and EER as a matric, we needed to bring our features down to a single value. This was done by taking the sum of the Euclidean distances from the DTW algorithm. While this did bring the EER down significantly from the baseline, test verification was needed on another data set. Using the clean/clean clean/babble as a starting point, we initially got 15%/23% and 0%/100% error rates (which is discussed in the next section).

3.2.1 Scoring Intuition and Method

The previously mentioned clean/babble results of 0%/100% clearly indicated a normalization issue in mismatched data. While the clean/clean results showed promising signs, the threshold in terms of absolute value was meaningless for babble data or any scenario in which the test and train data

might vary significantly. The reason behind this can be illustrated in Figure 6 below.

Figure 6: Histograms of distances for the first cepstral coefficient between clean and babble data.



In different datasets, the distances between feature vectors can vary significantly. If the entire distribution is shifted, then we need a way to normalize our threshold according to the distribution shift. It is important to note our baseline data prior to DTW was normalized, but the distance measures themselves were not. Scaling them proved to be difficult, as we experimented with max-min window scaling, mean/standard deviation scaling, and a few less intuitive techniques to little success. Table 4 below shows the results from the best attempt which was a subtraction of the min and factor of the max scaling techniques (CC is Clean/Clean, MB is Multi/Babble, etc).

Table 4: DTW Sum Norm Results

Clean/ Clean	Multi/ Clean	Clean/ Babble	Multi/ Babble
9%/ 30%	19%/ 67%	17%/ 41%	6%/ 80%

Even after normalization, the results on mismatched data were often no better than random guessing. We believe the primary issue is the low-dimensional nature of this comparison. We have 15 features, all with their unique distances, and we are projecting them into a one-dimensional space prior to comparison. This method is attempting to discriminate two distributions from a summation of 15-D data. But high and low distance measures across features might cancel each other out in the process. The remainder of our efforts focus on preserving the higher-dimensional nature of our distance vectors during the scoring phase. Rather than modifying the EER computation to reflect this, we abandoned the matric for direct testing and FPR/FNR results only.

3.2.2 *Unsuccessful Scoring Methods*

We proceeded to attempt a wide variety of scoring methods guided by the literature and our own intuition [5][6][7]. In brief, we will recap a few methods that were unsuccessful either in scoring or in the implementation themselves. All the techniques we tried at this stage were still on the distance vectors themselves, and not on the underlying MFCC coefficients or F0 vector. This is in sharp contrast to most of the literature[6][7], which utilizes many of these methods on the features directly and then performs a comparison. In our method, our computational time was likely significantly less but not formally quantified. Our goal was to differentiate between a 'same' and 'different' distribution of distance vectors. We believe we were sacrificing potentially useful data by not going a more traditional route, but the strategy outlined above was to primarily focus on robust scoring within the framework of an effective feature set.

As an example of one failed scoring system, we did attempt to use a Gaussian Mixture Model (GMM) in a different way from the traditional methods [9]. The traditional method would train on a Universal Background Model (UBM), enroll speakers, and then comprare between a local speaker model and the UBM. We attempted to train a GMM for two distributions on our distance matrix with the intention of distinguishing our same and different speaker sets. This proved to be moderately effective, but less so than our summation baseline (with results ~35-42% clean/clean). Our intuition behind this was not great, but we attempted this method later in the process and did not spend significant time investigating.

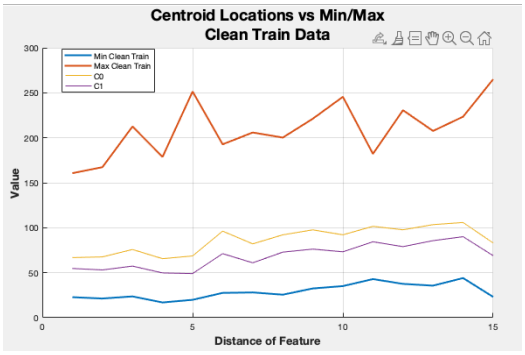
We also tried to develop a Hidden-Markov Model (HMM) [10]. Our intention was to step back one stage in our process and model state changes on the differences between the two feature vectors for a given feature. This step back would have been to account for the lack of transition data that a single distance score between two feature vectors held. Our plan was to instead model the absolute or euclidean distance of the features in time after DTW was applied to align the individual coefficient vectors. This proved to be very difficult to implement due to the time and the complexity associated with adapting our code base. We were unsuccessful in getting a working model, although we believe this topic merits revisiting considering the tremendous success of HMMs in the field.

Prior to any of the above methods, we started by applying various weight vectors to the distance vector and then summing to see if this approach yielded better results. The answer was definitely no, with EERs from 22-45%. We were often downweighting the very feature that was discriminating or over-weighting a distance that was spurious.

3.2.3 *Supervised K-means and Scaling*

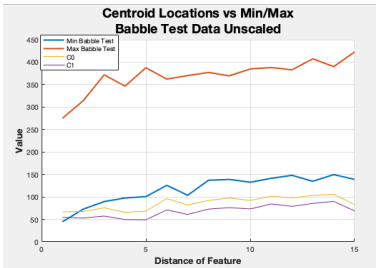
The first scoring method that proved reasonably robust was the widely used K-means method [11]. We won't discuss the general algorithm in detail here, but we utilized a supervised version in which we generate a 15-D centroid from our train data set for both the same and different distributions. Our initial clean/clean results were 10%/19% which was much better than the sum and a significant improvement over the baseline. Unfortunately, the method suffered from scaling issues as well with mismatched data. In Figure 7 below, we visualize the K-means algorithm by plotting the centroid values at each feature (15 of them) along with the minimum and maximum value in the train data set.

Figure 7: K-means Centroid Values vs Clean Train Min/Max



The issue can be seen in Figure 8 below in which the same centroid values are no longer meaningful in the babble test data set in which the scores are much higher across the board for same and different files.

Figure 8: K-means Centroid Values vs Babble Test Min/Max



In order to correct this, we took a statistical approach by saving the K-means centroid values as the number of standard deviations (sigma) away from the mean (mu) at each feature. When testing, the centroids are mapped to the new feature space by updating the values. Equation 1 shows the threshold calculation (sigmas away from mean), and equation 2 shows the new centroid remapping (which scales the centroid to the same place in the new distribution).

Equation 1: K-means centroid thresholding

$$\widehat{c0_1} = \frac{c0_1 - \mu_1}{\sigma_1}$$

$$\widehat{c1_1} = \frac{c1_1 - \mu_1}{\sigma_1}$$

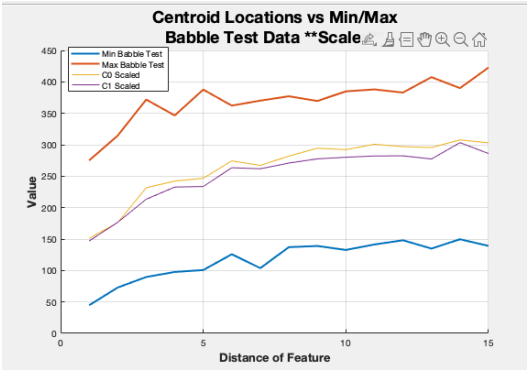
Equation 2: Centroid re-mapping

$$c_{0_{new}} = \sigma_{new} * \widehat{c0_1} + \mu_{new}$$

$$c_{1_{new}} = \sigma_{new} * \hat{1} + \mu_{new}$$

The results of this can be seen in the same visualization in Figure 9 with the clean train centroid mapped to the babble test data space.

Figure 9: K-means Centroid Values Mapped from Clean Train to Babble Test



We can see the results of these scaling efforts in Table 5 below. The results are certainly reasonable and a significant improvement over the sum of distances. But error rates are still not much better than random guessing when using mismatched data, nor is the multi/clean well balanced. Although the scoring implementation was a good exercise in statistics, slightly better performance was desired.

Table 5: DTW K-means Scaled Results

Clean/ Clean	Multi/ Clean	Clean/ Babble	Multi/ Babble
22%/ 21%	28%/ 57%	41%/ 14%	46%/ 44%

3.2.4 Mahalanobis Distance

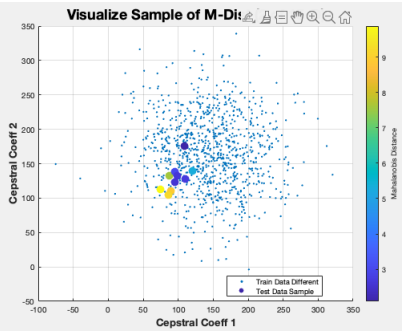
The Mahalanobis distance is an often used multivariable difference measure that seemed well suited for our task[12]. Again, full details will not be covered, but essentially this is a metric for the distance of a point from a distribution in any dimensional space. Equation 3 shows the calculation where x is our distance vector of observation, m is the mean vector of the distributions being compared to, and C is the inverse of the covariance matrix of the distributions.

Equation 3: Mahalanobis distance

$$D^2 = (x - m)^T \cdot C^{-1} \cdot (x - m)$$

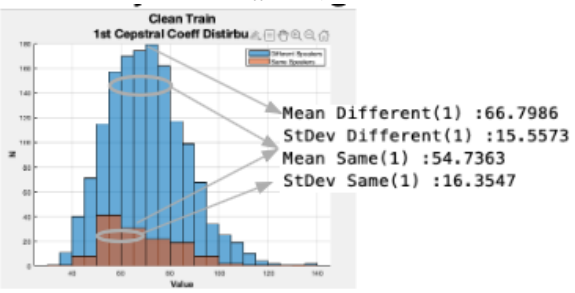
The benefit of this method is its consideration for multivariate variance. A widely distributed feature calls for a distance in terms of that variance. We can see a two dimensional visualization of this using the first and second cepstral coefficient in Figure 10.

Figure 10: 1st and 2nd feature Mahalanobis Score



Once more, our thresholds needed to be scaled for mismatched data. We decided to take a novel approach again and model our train distribution as three gaussians: same, different, and total. We save the mean and sigmas of the same and different distributions as adequate sufficient statistics. We also save the mean of the total distribution for offsetting with our test data. Figure 11 shows this statistics calculation on the clean training data for the 1st coefficient.

Figure 11: Saving the statistics on our train data (1st MFCC Coefficient)



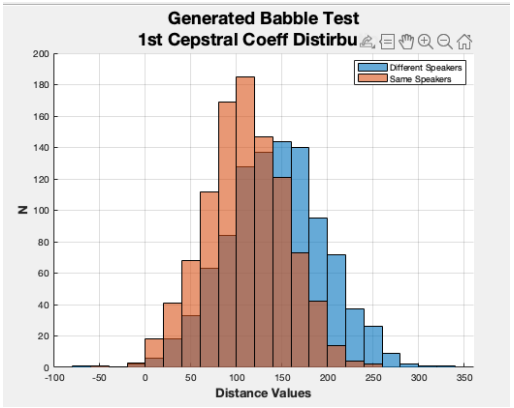
Note the mean of the total distribution is not pictured but was collected and sent as a threshold as well. When testing, we first calculate the mean of the new distribution. We can then compute the mean offset by subtracting the mean of the two total distributions and offset our threshold means to align them for our test data set as shown in Equation 4.

Equation 4: $\boldsymbol{\mu}$ offset calculation for Mahalanobis distribution scaling

$$\mu_{offset} = mean(testData) - \mu_{train}$$
$$\mu_1 = \mu_{1\_train} + \mu_{offset}$$
$$\mu_0 = \mu_{0\_train} + \mu_{offset}$$

Finally, we used a unique method for implementation by randomly generating a "same" and "different" distribution of 1000 samples centered at their respective "offset" means also using the train data sigma statistics. The test data is then compared to each of these two distributions across all features using the Mahalanobis distance calculation. The generated distributions can be seen in Figure 12 below.

Figure 12: Simulated distribution at offset means for 1st Cepstral Coefficient



Results will be shared in the next section, but this technique gave the best results across the train/test combos, and proved to be an effective classifier on clean data with significant improvements over the baseline.

3.3 Noise Robustness

As was briefly discussed in the feature selection methodology, we attempted basic filtering in preprocessing to de-noise the signals. This proved ineffective, but our intention was to revisit noise robustness in a more detailed way after improving the clean baseline. Due to time, we were unable to do so. But we did segment the multi dataset into a babble only training data set. This allowed us to generate error rates on babble/clean and babble/babble data to get an additional measure on the noise robustness of our final model. The next section includes these results.

4. Final Model and Results

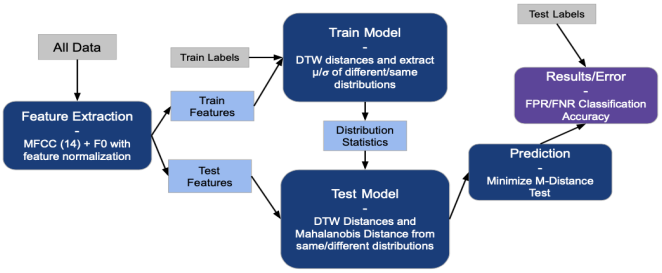
Our final results are listed in Table 6 below. Our model yielded significant improvements from the baseline on clean test data, but was comparable to the baseline when testing with babble noise data.

Table 6: Final Model Results

	Clean/ Clean	Multi / Clean	Clean/ Babble	Multi/ Babble	Babble/ Clean	Babble/ Babble
Base-Line	34%/ 30%	46%/ 37%	32%/ 33%	43%/ 42%	30%/ 36%	40%/ 47%
Final Model	18%/ 30%	38%/ 50%	12%/ 27%	35%/ 48%	10%/ 31%	34%/ 47%

Figure **13** below illustrates our final model encompassing the major steps and flow.

Figure 13: Final Model



5. Conclusion

5.1 Lessons Learned

The scope and nature of this project allowed for numerous learning opportunities along the way. The project goal of better classification , novelty, and knowledge demonstration left plenty of room for experimentation and exploration of unique ideas. One key lesson we took away from this is the importance of starting simple and early. We explored a lot of unique features through image processing such as mel spectrograms and even applied image processing to the MFCC/F0 matrix to begin with. The literature indicated this path was not utilized often, and for good reason we discovered (and enumerated in the sections above). We could have advanced much faster to our eventual feature selection had we begun with where most of the literature left off when it came to speech verification features.

Another major lesson was in the difficulty of both scoring normalization and noise robustness. Scoring normalization is a deep and nuanced exercise in domain knowledge, statistics, and experimentation, particularly when dealing with high dimensional and mismatched data in noisy conditions. Adjacently, noise proved to be the most difficult challenge that we could not address in the detail we hoped. The attempts we did make at noise robustness proved

negligible, and the existing literature also indicates the difficulty of training models with babble noise. Perhaps last and most importantly, given the time constraints on the course project, there was never enough time to attempt all the ideas we had. We would have benefited from better triaging earlier in the process to focus our efforts on the methods most likely to succeed with some room for novelty, and experimentation.

5.2 Future Research

In closing, we wanted to discuss areas we would have explored more. First and foremost, there is a vast library of methods to attempt basic and more advanced noise mitigation techniques that could have coupled well with our existing approach. We would have liked to explore more basic features with our final scoring models and attempt a fusion based approach to achieve the best results from a variety of features and scoring methods. Finally, we would have liked to explore different methods on a deeper level, such as HMMs, to get working models of comparable results for our own knowledge and perhaps to combine complementary features and methods for better results.

6. References

[1] Lawrence R. Rabiner and Ronald W. Schafer. 2007. *Introduction to Digital Speech Processing.* Now Publishers Inc., Hanover, MA, USA.

[2] S. K. Singh, Prof P. C. Pandey," Features And Techniques For Speaker Recognition",IIT Bombay.

[3] Tan, Lee & Alwan, Abeer. (2013). Multi-band summary correlogram-based pitch detection for noisy speech. Speech Communication. 55. 841–856. 10.1016/j.specom.2013.03.001.

[4] Bimbot, F., Bonastre, JF., Fredouille, C. *et al.* A Tutorial on Text-Independent Speaker Verification.*EURASIP J. Adv. Signal Process.* 2004, 101962 (2004).

[5] Muda, Lindasalwa & Begam, Mumtaj & Elamvazuthi, Irraivan. (2010). Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. J Comput. 2.

[6] Hourri, Soufiane & Kharroubi, Jamal. (2019). A Novel Scoring Method Based on Distance Calculation for Similarity Measurement in Text-Independent Speaker Verification. Procedia Computer Science. 148. 256-265. 10.1016/j.procs.2019.01.068.

[7] "Speaker Identification Using Pitch and MFCC." *Speaker Identification Using Pitch and MFCC - MATLAB & Simulink*, www.mathworks.com/help/audio/ug/speaker-identification-using-pitch-and-mfcc.html.

[8] Gonzalez, Ruben. "Better than MFCC audio classification features." *The Era of Interactive Media*. Springer, New York, NY, 2013. 291-301.

[9] Douglas A. Reynolds, Thomas F. Quatieri, Robert B. Dunn, *Speaker Verification Using Adapted Gaussian Mixture Models, Digital Signal Processing*, Volume 10, Issues 1–3, 2000,Pages 19-41,

[10] Sarkar, Achintya & Tan, Zheng-Hua. (2016). *Text Dependent Speaker Verification Using un-supervised HMM-UBM and Temporal GMM-UBM.* 10.21437/Interspeech.2016-362.

[11] Praveen, N., & Thomas, T. (2013). *Text Dependent Speaker Recognition using MFCC features and BPANN. International Journal of Computer Applications, 74*, 31-39.

[12] Prabhakaran, S. (2020, October 15). *Mahalanobis distance - understanding the math with examples (python)*. Retrieved March 12, 2021, from https://www.machinelearningplus.com/statistics/mahalanobis-distance/