Speaker Recognition: An SVM Approach Using Time-Averaged Features

Yimu Tu[1], *Minya Yao*[2], *Fangyao Liu*[3], *Ben Quachtran*[4]

Department of Electrical and Computer Engineering, University of California, Los Angeles, USA

fatandy567@ucla.edu[1], minyayao@ucla.edu[2], fangyaoliu@ucla.edu[3], benquachtran@ucla.edu[4]

Abstract

The task of speaker recognition remains a difficult problem that has yet to be solved, especially in noisy environments. Recent advancements in machine learning, however, have proved useful in developing recognition models through feature-based training on large data sets. In this paper, we propose a support vector machine approach to this task of speaker recognition, incorporating elementary voice characteristics, namely pitch and formant frequencies, as well as harmonic amplitude measures and Mel Frequency Cepstrum Coefficients (MFCC). Dimensionality reduction was performed using a time-average of these features. The magnitude difference between feature vectors and the corresponding binary labels (0 if the same speaker and 1 otherwise) were used for training of a support vector machine. Training was performed on both data sets with and without the presence of background babble noise. The model was evaluated on both clean and noisy data sets as well as two distance calculations: Euclidean and logarithmic. When trained on the clean data and Euclidean distance calculation, the model achieved a (FPR, FNR) of (23.8%, 14.8%) on clean testing data, and (30.8%, 20.7%) on noisy testing data. Training the machine learning model on both clean and noisy data achieved (FPR, FNR) of (29.3%, 9.6%) on clean testing data, and (36.6%, 16.3%) on noisy testing data.

1. Introduction

Speaker recognition has become an important topic in various fields, such as forensics, phonetics, and engineering. It is the process of identifying the speaker of the utterance based on the acoustic features embedded in the speech segment, and has been widely employed on occasions of identity verification and control access verification to facilate service to customers [1]. For humans, identifying a person by his or her voice is not a difficult task that most of us often take for granted. Several researches have shown that humans are able to outperform machines and identify a speaker given a very short utterance produced by a familiar person, even in a noisy environment [2].

In automatic speaker recognition (ASR), we need to provide “features” of utterances for a machine to make decisions. To this day, how exactly a human process the audio signal and what features that his or her brains use to make decisions are still unclear. Thus several features have been proposed, such as the Mel Frequency Cepstral Coefficients (MFCC) [3] and the Linear Prediction Cepstral Coefficients (LPCC) [4]. In this project, we first extracted several formant-related features using VoiceSauce toolbox [5]. With the help of machine learning, we developed a text-dependent system that is able to distinguish whether two utterances were spoken by the same person or not, and achieved error rates of 20% and 25% in a quiet and noisy environment, respectively.

2. Background

2.1. Voice Quality Features

In previous studies, researchers found that listeners of various languages are sensitive to H1-H2, which is the amplitude difference between the first and the second harmonics [6]. Also, H2*-H4*, which is the amplitude difference between the second and the forth corrected harmonics, is suggested to play an important role for a listener to distinguish the gender of the speaker [7]. Based on the previous work in [8], we decided to extract the following feature from the recorded audio files: F0, F1, F2, F3, H1*-H2*, H2*-H4*, H2k* (the amplitude of the harmonic near 2kHz), H4*, the difference between H2k* and H5k, and the cepstral peak prominence (CPP) [9]. The asterisks (*) indicate that the effect of formants and harmonics have been compensated [5].

2.2. Mel Frequency Cepstral Coefficients

MFCCs were first proposed by Davis and Mermelstein in 1980, and was indicated to have superior performance compared to other parametric representations, such as linear frequency cepstrum coefficients (LFCC), linear prediction coefficients (LPC), reflection coefficients (RC), and the cepstrum coefficients derived from the linear prediction coefficients (LPCC) [10]. The MFCCs implicitly represent the information about the vocal track, and remain one of the most popular short-term acoustic features utilized in ASR systems. The basic idea of the computation of MFCCs is to compute a frequency analysis based upon a filterbank with critical bands and weighted by a weighting function, typically triangular- or rectangular-shaped filters [11]. More detail will be provided in Section 3.

2.3. Support Vector Machine

Support Vector Machine (SVM) is one of the most popular Machine Learning methods for classification problems. To illustrate the mechanisim behind SVM, we first assume two sets of data which are linear separable and denoted by:

$$\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \dots, (\mathbf{x_N}, y_N)\} \tag{1}$$

where $\mathbf{x_i} \in \mathcal{R}^d$ and $y_i = \{\pm 1\}$ for simplicity. To separate these two of sets, we would like to find a hyperplane classifier whose normal vector is denoted by $\mathbf{w}$ and have different signs for different sets of data. The classifier could be formulated as:

$$f(x) = sign(\mathbf{w^T x} - \mathbf{b}) \tag{2}$$

$$\mathbf{w^T x} - \mathbf{b} \geq +1,\ y_i = +1 \tag{3}$$

$$\mathbf{w^T x} - \mathbf{b} \geq -1,\ y_i = -1 \tag{4}$$

If equation (3) and (4) are the only constraints, it is possible that there are several $\mathbf{w}$s satisfying our needs. For example, hyperplane H_1 and H_2 in Figure 1 for a 2-D case both satisfy the constraints. However, in SVM, we want this hyperplane to have the maximum separating margin with respect to two data sets. From this point of view, H_1 is more preferred than H_2.

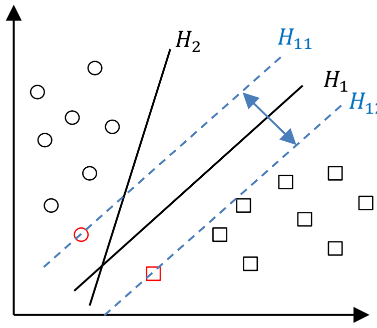

Figure 1: An illustration of the computation of SVM.

Assuming H_1 is the optimal solution to our 2-D simplified SVM problem, and two hyperplanes H_{11} and H_{12} are parallel to H_1 with equal distance, the red circle (x_c, y_c) and red square (x_s, y_s) in Figure 1 on H_{11} and H_{12}, respectively, will give us:

$$y_c = \mathbf{w}^{\mathbf{T}}\mathbf{x_c} - \mathbf{b} = +1 \tag{5}$$

$$y_s = \mathbf{w}^{\mathbf{T}}\mathbf{x_s} - \mathbf{b} = -1 \tag{6}$$

Recall that in 2-D dimension, the distance between a point (x_0, y_0) and a line whose equation is $Ax + By + C = 0$ could be formulated as:

$$\text{Distance } d = \frac{|\boldsymbol{Ax_0+by_0+C}|}{\sqrt{A^2+B^2}} \tag{7}$$

The formula in higher dimension is similar to the 2-D case. Generalizing the case in Figure 1 to higher dimension, the red square (x_s, y_s) to hyperplane $H_1$is

$$\text{Distance } d = \frac{\mathbf{w}^{\mathbf{T}}\mathbf{x_s} - \mathbf{b}}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|} \tag{8}$$

Thus, the distance between the two hyperplanes H_{11} and H_{12} is

$$\text{Distance } d = \frac{1}{\|\boldsymbol{w}\|} \times 2 = \frac{2}{\|\boldsymbol{w}\|} \tag{9}$$

To maximize the margin, we can formulate the SVM to an optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} \quad \text{subject to } y_i(\mathbf{w}^{\mathbf{T}}x_i - b) \geq 1 \tag{10}$$

3. Implementation

3.1. Database

The database of utterances where designed and collected at UCLA, with over two hundred female and male speakers participated. Each piece of utterance in the dataset is the same sentence “Help the woman get back to her feet”. The collection of data was performed in three separate recording sessions on different days. Two different environments are considered: with and without babble noise. All of the audio recordings were made in a sound-attenuated booth with a sampling rate of 22KHz [8].

3.2. System Implementation

For any speaker recognition system, the first step is to experiment and identify the components of a speech signal that are the most effective in terms of training a speaker model to distinguish between different speakers. However, not all features are equally important. Ideally, we want to utilize features that have large between-speaker variability and small within-speaker variability, and at the same time easy to be measured from the speech signal [12]. The features used in this project could be grouped into two categories: voice quality features and MFCCs. We extracted 10 voice quality features and 13 MFCC's and used them to form a supervector to train our SVM.

3.2.1. Voice quality features

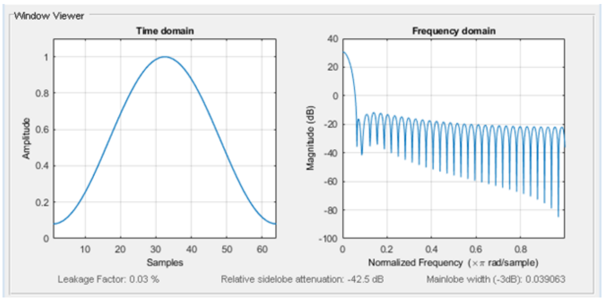

Figure 2: Hamming window of duration 25ms.

Voice quality feature extraction was performed using the VoiceSauce toolbox [5], a program for voice analysis, generating a time-series vector for each voice feature that was calculated. The features extracted include: pitch frequency F0, F1, F2, F3, energy, CPPs, and the harmonic-amplitude measurements (H1*-H2*, H2*-H4*, H4*-2K*, H2K*-H5K). The pitch frequencies and harmonic amplitudes were computed by VoiceSauce, whereas the CPPs were computed in the following sense. We first applied a hamming window of duration 25ms with a frame shift of 1ms and computed the cepstrum values per frame (Figure 2). Then we used a linear regression model to fit to the actual cepstrum, and computed the differences between the cepstral peaks and the corresponding regression values.

3.2.2. Mel frequency cepstrum coefficients (MFCC)

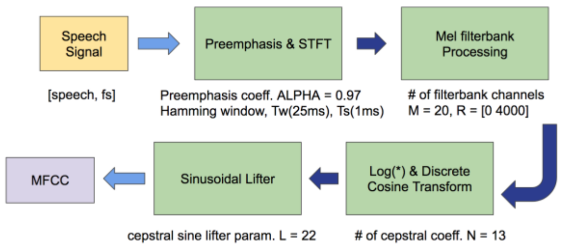
Speech Signal
[speech, fs]
Preemphasis & STFT
Preemphasis coeff. ALPHA = 0.97
Hamming window, Tw(25ms), Ts(1ms)
Mel filterbank Processing
# of filterbank channels
M = 20, R = [0 4000]
Log(*) & Discrete Cosine Transform
# of cepstral coeff. N = 13
Sinusoidal Lifter
cepstral sine lifter param. L = 22
MFCC

Figure 3: Block diagram for MFCC extraction.

Mel frequency cepstrum coefficients (MFCC) were computed using the an adjusted version of an algorithm written by Kamil Wojcicki [13]. As illustrated in Figure 3, the audio files were loaded into MATLAB and transformed into a speech signal with sampling frequency of 8KHz. The speech signal is pre-emphasized with a pre-emphasis coefficient of 0.97, and applied the Short Time Fourier Transform (STFT) with a Hamming window of frame duration of 25ms and frame shift of 1ms. The frames were then fed into the Mel filterbank that consisted of 20 triangular filters uniformly spaced on the mel-scale between 0Hz to 4000Hz. The filterbank was applied to the magnitude spectrum values to produce filter bank energies.

Next, we applied logarithmic operation to compress the resulting filterbank energies due to the fact that the human auditory system does not perceive loudness in a linear scale. Then we performed the Discrete Cosine Transform (DCT) to retain 13 leading coefficients. The DCT was employed for two reasons: first, it reduces feature dimension by compressing the energy of the speech signal to a few coefficients, and second, its decorrelation property ensures that the resulting feature vectors are uncorrelated, even if the filters in the mel filterbank are overlapping, as shown in Figure 6, and therefore produce correlated filterbank energies [3]. Denoting the mel filterbank energies as $Y(m)$, where $m = 1, 2, \dots, 20$, the MFCCs can be obtained using the following equation [12]:

$$c_n = \sum_{m=1}^{M} \log Y(m) \cos\left(\frac{\pi n}{M}\left(m - \frac{1}{2}\right)\right) \qquad (11)$$

where M is the number of filters used in the filterbank, and n is the index of the cepstral coefficient. The final step is to apply sinusoidal liftering to give 13 liftered MFCCs. Typically, the first and second derivatives of the MFCCs (deltas and double deltas) are also used as features in order to capture the dynamic aspects of the speech signals; however, we chose to include variances instead of deltas and double deltas to keep the training time as short as possible, while taking the dynamic features of the utterances into consideration.

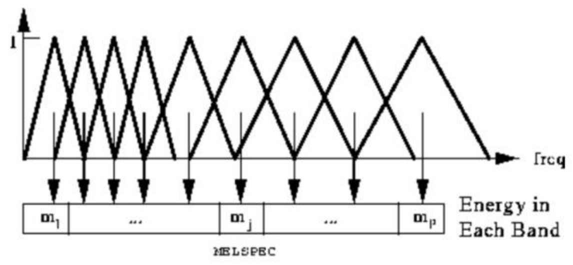

Figure 6: A mel-scale filterbank [16].

The next procedure is to perform dimension reduction. The 10 voice quality features and 13 MFCC's were concatenated into an *n-by-23* matrix for each utterance and the rows of each utterance matrix were pruned using an energy threshold to remove the trivial data for the silient period. In Figure 4 and Figure 5 illustrate this process. For the frames with energy levels below the threshold, we treat them as “silence” segments and remove them from the dataset. We then calculate the mean of these features across time, which will give us a vector of length 23 for each audio file.

At the end of these feature vectors, we concatenate 10 more variances, which are the varances across time of each voice quality features obtained from VoiceSauce after silence removal. A variance could be seen as a 1-D metric for how fast a set of data change. If those data vary slowly, the variance could be expected to be small, and vice versa. In this project, because people are saying the same sentence,”Help a women get back to her feet”, if two files come from the same person, intuitively, how those voice quality features vary across time will be similar, given that people have their own ways speaking and didn’t change it intentionally during the recording.

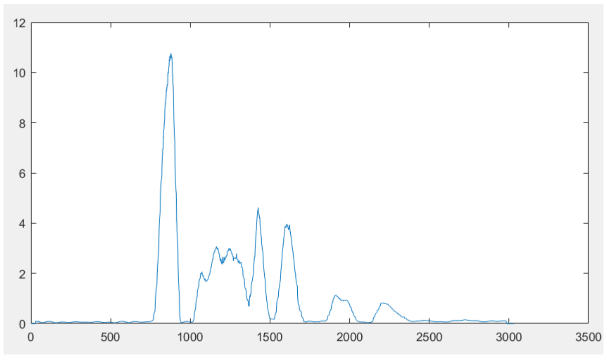

Figure 4: Energy of an audio file.

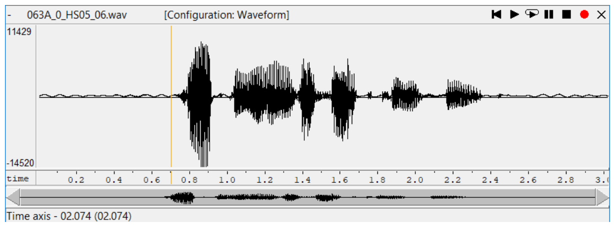

Figure 5: Audio time waveform by WaveSurfer.

Adding 10 more varainces results in a 1-dimensional feature vector of length 33 for each file (23 feature averages and 10 feature variances). These 1-dimensional feature vectors were generated for all utterances in the training and testing sets.

Finally, SVM training was performed using the absolute value difference between two speakers' feature vectors and a binary label of 0 or 1, 0 if the two utterances were spoken by different speakers and 1 if they were spoken by the same speaker. MATLAB's built-in functions *svmtrain* and *svmclassify* were used for this machine learning portion of the model. The SVM training was performed with parameters in Table 1. A visualization of our speaker recognition model is shown in Figure 7, depicting the feature detection and transformation methods that were used as input into the SVM classifier.

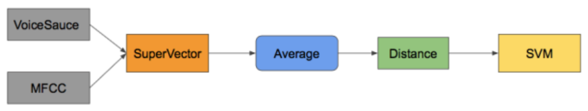

Figure 7: A summary of the procedures in the speaker recognition model

Table 1: SVM parameters.

Parameter Name	Value
kernel_function	‘linear’
kernel_function	‘linear’
MaxIter	1e7
kernelcachelimit	1e7
method	‘SMO’

3.3. Experiment

To evaluate our speaker recognition model, two sets of experiments were conducted: one using an SVM trained on clean audio data, labeled as *trainCleanList*, and another trained using both clean and noisy data, labeled as *trainMultiList*. For both experiments, accuracy was measured using classification of both clean and noisy audio files. The training and testing data was comprised of feature data evaluated between two speakers and their corresponding labels. Accuracy was measured using the classifier's false positive rate (FPR) and false negative rates (FNR) and compared against a baseline classification model, where lower rates indicate improved accuracy.

Two distance measures were tested for comparing two speakers' feature vectors, Euclidean and logarithmic distance. The results for the Euclidean distance metric are shown in Table 2 and Table 3, while the results of logarithmic distance are in Table 4 and Table 5. The math formula for two kinds of distance between two vectors, $\mathbf{v_1}$ and $\mathbf{v_2}$ are:

$$\text{Euclidean: } abs(\mathbf{v_1} - \mathbf{v_2}) \tag{12}$$

$$\text{Logarithmic: } log_{100}^{abs(\mathbf{v_1} - \mathbf{v_2})} \tag{13}$$

Logarithmatic functions, as shown in Figure 8, are able to suppress the larger terms while enlarge the smaller terms. By using logarithmic distance, we expect better ability to distinguish two files whose feature vectors are close.

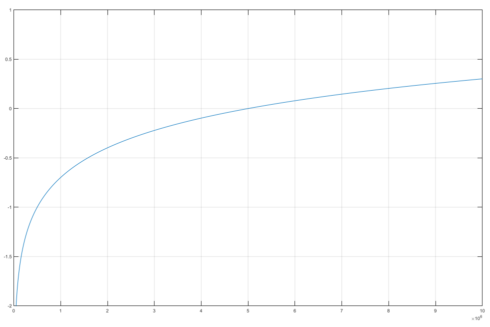

Figure 8: A logarithmic function.

3.4. Results

With the training with *trainCleanList* and Euclidean distance, we achieved FPR and FNR of 23.8% and 14.8% for the clean data, respectively. FPR and FNR of 30.8% and 20.7% were achieved for testing on noisy data, respectively. In comparison, the baseline model achieved 34.5% and 30.4% on the clean data, and 46.1% and 37.8% on the noisy data. Using our model, we see improvement rates of 10.7% and 15.6% for the clean data, and 15.3% and 17.1% for the noisy data. We also train our SVM without 10 variances and using logarithmic distance calculation. The results are tabulated in Table 2 and Table 4.

When trained with *trainMultiList* and Euclidean distance, our model achieved FPR and FNR of 29.3% and 9.6% when tested on the clean data, and 36.6% and 16.3% when tested on the noisy data. The baseline model, trained on the same data set, achieved rates of 32.3% and 33.3% on the clean data, and 42.8% and 42.2% on the noisy data. We therefore see FPR and FNR improvements of 3% and 23.7% for the clean testing data, and 6.2% and 25.9% for the noisy testing data. These speaker recognition improvements demonstrated by our SVM model demonstrate its effectiveness over the compared baseline model in all forms of evaluation. Without 10 variances and using logarithmic distance calculation were also trained. The results are tabulated in Table 3 and Table 5.

Table 2: *Error rates by training with trainCleanList, Euclidean distance. (*) are without 10 varainces.*

	trainCleanList			
	Clean data		Noisy data	
	FPR	FNR	FPR	FNR
Ours	23.8%	14.8%	30.8%	20.7%
	(24.4%)	(10.4%)	(29.3%)	(18.5%)
Baseline	34.5%	30.4%	46.1%	37.8%
Improv.	-10.7%	-15.6%	-15.3%	-17.1%
	(-10.1%)	(-20.0%)	(-6.8%)	(-19.3%)

Table 3: *Error rates by training with trainMultiList, Euclidean distance. (*) are without 10 variances.*

	trainMultiList			
	Clean data		Noisy data	
	FPR	FNR	FPR	FNR
Ours	29.3%	9.6%	36.6%	16.3%
	(29.8%)	(8.1%)	(34.4%)	(17.0%)
Baseline	32.3%	33.3%	42.8%	42.2%
Improv.	-3.0%	-23.7%	-6.2%	-25.9%
	(-2.5%)	(-25.2%)	(-8.4%)	(-25.2%)

Table 4: *Error rates by training with trainCleanList, log distance. (*) are without 10 variances.*

	trainCleanList			
	Clean data		Noisy data	
	FPR	FNR	FPR	FNR
Ours	30.0%	15.6%	33.3%	30.0%
	(28.2%)	(14.1%)	(31.1%)	(23.7%)
Baseline	34.5%	30.4%	46.1%	37.8%
Improv.	-4.5%	-14.8%	-12.8%	-7.8%
	(-6.3%)	(-16.3%)	(-15.0%)	(-14.1%)

Table 5: *Error rates by training with trainMultiList, log distance. (*) are without 10 variances.*

	trainMultiList			
	Clean data		Noisy data	
	FPR	FNR	FPR	FNR
Ours	33.9%	14.1%	38.9%	21.5%
	(33.3%)	(14.1%)	(35.4%)	(23.7%)
Baseline	34.5%	30.4%	46.1%	37.8%
Improv.	-0.6%	-16.3%	-7.2%	-16.3%
	(-1.2%)	(-16.3%)	(-10.7%)	(-14.1%)

Table 6: *Improvement comparasion, trained with trainCleanList.*

	trainCleanList			
	Clean data		Noisy data	
	FPR	FNR	FPR	FNR
w.Var, Euc dist.	10.7%	15.6%	15.3%	17.1%
w/o.Var, Euc dist.	10.1%	20.0%	6.8%	19.3%
w.Var, log dist.	4.5%	14.8%	12.8%	7.8%
w/o.Var,log dist.	6.3%	16.3%	15.0%	14.1%

Table 7: *Improvement comparasion, trained with trainMultiList.*

	trainMultiList			
	Clean data		Noisy data	
	FPR	FNR	FPR	FNR
w.Var, Euc dist.	3.0%	23.7%	6.2%	25.9%
w/o.Var, Euc dist.	2.5%	25.2%	8.4%	25.2%
w.Var, log dist.	0.6%	16.3%	7.2%	16.3%
w/o.Var,log dist.	1.2%	16.3%	10.7%	14.1%

4. Discussion

We observe from the results that relatively substantial improvements were made to the speaker recognition model through use of an SVM classifier trained on time-averaged features. Most interestingly, we observe a 25% decrease of false negative prediction rates on noisy data when the SVM was trained with *trainMultiList*, with only a 3% decrease of false positive rate. The other differences in detection rates were more consistent, most falling within the 10-20% range, indicating a non-trivial improvement in classifier accuracy. Distance calculated using logarithmic function and without 10 varainces were also tested. With and without those 10 variances, based on our experiment, there is no significant difference. However, when using logarithmic distance calculation, the performance decays greatly.

While these results are promising, more comprehensive evaluation is a necessary step to demonstrate this increase in accuracy. Additionally, comparisons against a wider range of classification models to gauge the effectiveness of our feature and classifier choices are also required. The list of features were chosen based on prior works on speaker recognition that demonstrated the effectiveness of harmonic amplitude measures as well as MFCCs as a means of signal dimensionality reduction. Determining methods for feature dimension reduction proved to be a difficult problem, as reducing features dimension improved SVM training at the cost of lost information. Thus a trade-off had to be made for determining the level of feature information used in the SVM classifier in order to both capture key signal characteristics and not overload the classifier with unnecessary data.

In conclusion, this work has shown that SVM classification of time-averaged features is a promising approach to the problem of speaker recognition, necessitating further investigation into this model's efficacy. The improvements relative to a baseline model were shown to be significant based on the defined evaluation method, providing justification for the use of time-averaged features.

The task of speaker recognition remains a difficult task that has yet to be fully solved. However, advancements in machine learning have proven to be a possible solution that remains yet to be explored. Future works should investigate how modifications can be made to machine learning training to more effectively capture speaker characteristics and therefore improve recognition accuracy.

Future studies will include obtaining a larger dataset by recording longer audio files and utilizing GMM-UBM to better represent the feature supervectors to improve the robustness of the ASR system. In the early stage of this project, we did try GMM-UBM with several toolboxes available in the Internet. However, it seems that the dataset we have was not large enough to properly train the model, and the results were pretty disappointed, so we didn't adopt the design. Additionally, while humans are extremely good at distinguishing speech content from background noise, it requires more sophisticated algorithms for machines to adapt to that. We can further investigate on some mechanisms that effectively remove noise, such as Wiener filter, yet still be able to pertain the acoustic features of the original speech segments.

5. References

[1] J. Martinez, H. Perez, E. Escamilla and M. M. Suzuki, "Speaker recognition using Mel frequency Cepstral Coefficients (MFCC) and Vector quantization (VQ) techniques," *Electrical Communications and Computers (CONIELECOMP), 2012 22nd International Conference on,* February 2012.

[2] S. J. Wendt and R. L. Michell, "Machine recognition vs human recongition of voices," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference.,* 2012.

[3] John H.L. Hansen and Taufiq Hasan, "Speech recognition by machines and humans: A tutorial review," *IEEE Signal Processing Magazine,* pp. 74-99, 2015.

[4] H. Hermansky, "Perceptual Linear Predictive (PLP) Analysis of Speech," *The Journal of the Acoustical Society of America,* pp. 1738-52, 1990.

[5] Shue, Y.-L., P. Keating , C. Vicenik and K. Yu, "VoiceSauce: A program for voice analysis," in *Proceedings of the ICPhS,* 2011.

[6] C. M.Esposito, "The effects of linguistic experience on the perception of phonation," *Journal of Phonetics,* vol. 38, no. 2, pp. 306-316, 2010.

[7] Jason Bishop, and Patricia Keating, "Perception of pitch location within a speaker's range: Fundamental frequency, voice quality, and speaker sex," *J Acoust Soc Am,* pp. 1100-12, 2012.

[8] Soo Jin Park, Caroline Sigouin, Jody Kreiman, Patricia A. Keating, Jinxi Guo, Gary Yeung, Fang-Yu Kuo, and Abeer Alwan, "Speaker Identity and Voice Quality: Modeling Human Responses and Automatic Speaker Recognition," in *INTERSPEECH*, 2016.

[9] Rubén Fraile, Juan IgnacioGodino-Llorente, "Cepstral peak prominence: A comprehensive analysis," *Biomedical Signal Processing and Control,* vol. 14, pp. 42-54, 2014.

[10] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions On Acoustics, Speech, and Signal Processing,* August 1980.

[11] L. R. Rabiner and R. W. Schafer, Theory and Applications of Digital Speech Processing, Pearson.

[12] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *ScienceDirect,* 20 August 2009.

[13] K. Wojcicki, "HTK MFCC MATLAB," [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/32849-htk-mfcc-matlab?focused=5199998&tab=function.

[14] Markus Iseli, Yen-Liang Shue, and Abeer Alwand, "Age, sex, and vowel dependencies of acoustic measures related," *J Acoust Soc Am,* vol. 121, no. 4, pp. 2283-95, 2007.

[15] Steve Young, Gunnar Evermann, Dan Kershaw, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland, The HTK Book, 2002.

[16] "Filterbank Analysis," [Online]. Available: http://www.ee.columbia.edu/ln/rosa/doc/HTKBook21/node54.html. [Accessed 17 March 2018].