Modelling of a Speech-to-Text Recognition System for Air Traffic Control and NATO Air Command

Grant Zietsman[1], Reza Malekian[1,2,3*]

[1] Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa
[2] Department of Computer Science and Media Technology, Malmö University, Sweden
[3] Internet of Things and People Research Centre, Malmö University, Sweden
grant.zietsman@tuks.co.za, reza.malekian@ieee.org

Abstract

Accent invariance in speech recognition is a challenging problem especially in the are of aviation. In this paper a speech recognition system is developed to transcribe accented speech between pilots and air traffic controllers. The system allows handling of accents in continuous speech by modelling phonemes using Hidden Markov Models (HMMs) with Gaussian mixture model (GMM) probability density functions for each state. These phonemes are used to build word models of the NATO phonetic alphabet as well as the numerals 0 to 9 with transcriptions obtained from the Carnegie Mellon University (CMU) pronouncing dictionary. Mel-Frequency Cepstral Coefficients (MFCC) with delta and delta-delta coefficients are used for the feature extraction process. Amplitude normalisation and covariance scaling is implemented to improve recognition accuracy. A word error rate (WER) of 2% for seen speakers and 22% for unseen speakers is obtained.

Keywords: Automatic Speech Recognition (ASR), Hidden Markov Model (HMM), Gaussian Mixture Model (GMM), Mel-Frequency Cepstral Coefficients (MFCC), Covariance scaling

1 Introduction

SPEECH recognition has been very well established over the past two decades in particular in intelligent transportation systems and has realised a significant growth in applications. Nonetheless, the critical challenge of accent independence remains to be solved [1]. The limited availability of speech data for certain accented groups contributes to this complexity. An area of considerable importance is in aviation. It is critical that communication between pilots and air traffic controllers be accurate and efficient hence the development of the NATO phonetic alphabet [2]. The issue is more apparent with international flights, involving people of different nationalities and mother languages which frequently results in miscommunication due to accents. It is proposed that the development of a speech-to-text recognition system for accented speech could resolve this issue by transcribing speech and providing a secondary channel of information transfer.

Most research efforts use Gaussian Mixture Models (GMMs) with Hidden Markov Models (HMMs) for speaker independent speech recognition [3]. In this paper a novel technique is proposed to improve recognition of accented speech [21] when a limited group of speakers are used in the training set. This entails the application of a covariance scaling factor which is inversely proportional to the number of speakers. In parallel, amplitude normalisation is used to improve MFCC feature matching accuracy. The novelty of this work is continuous speech to text recognition for air navigation systems with better accuracy than current systems using HMMs and GMM probability density functions and frequency cepstral coefficients for feature extraction.

Novelty: a continuous speech recognition system using GMM-HMMs is developed to model phonemes and a unigram based language model to be used in aviation communication with improved recognition accuracy by implementation of amplitude normalisation and covariance scaling.

Contribution: The contributions of this paper can be summarized as follows. We model phonemes using HMMs with GMM probability density functions for each state. These phonemes are used to build word models of the NATO phonetic alphabet as well as the numerals 0 to 9 with transcriptions obtained from the CMU pronouncing dictionary. MFCC with delta and double-delta coefficients are used for the feature extraction process. We also implement the hypothesis search for continuous speech recognition. Moreover, we propose and teste a method for speech recognition that is not affected by accents as much as current approaches by using the speech recognition problem between pilots and air traffic controllers as a motivating application.

The paper is organised as follows. Section II discusses the work done prior to the implementation of this project. A brief high level description of the proposed approach is given in Section 3. The details of the system design as well as the subsystem implementation is described in Section 4. The design choices are justified with consideration to available resources and scope of the project. This section also tabulates the results obtained by the speech recognition system and describes the relevance of these results. Finally, Section 5 concludes with an overview as to how the project outcomes were obtained.

Some additional resources are provided in the appendix. This includes Section A which defines the speech recognition lexicon; Section B which defines the mathematical notation

*Corresponding Author: Reza Malekian; E-mail: reza.malekian@ieee.org
DOI: 10.53106/160792642022122307008

used in this paper and Section C, a proof of the forward algorithm scaling equations.

2 Related Work

An initial effort to perform speaker independence recognition using Hidden Markov Models (HMMs) is presented in [4].

An area of research that is relevant to the effort presented in this paper involves the use of Hidden Markov Models with Gaussian Mixture Model (GMM-HMMs) probability density functions. Hidden Markov Models have been applied to speech recognition since the 1970s [4]. Since then, the application of the expectation maximization (EM) and GMMs have drastically improved speech recognition accuracy [22]. Such models proved to be very effective as probability density functions for HMMs in speech recognition [3]. GMM-HMMs formed the foundation of speech recognition systems until the recent application of deep learning models.

In more recent years, specifically since 2009, Deep learning Neural Network HMMs (DNN)-HMMs have surpassed the performance of GMM-HMMs [5]. This motivated significant research in the area of artificial neural networks (ANNs) which was previously limited by computational performance. Much of the development of deep learning in speech recognition was achieved by IBM, Microsoft, Google and Baidu [6]. To over- come the challenges of training with big data both Microsoft and Baidu employed parallelism using graphics processing units (GPUs) while Google employed central processing units (CPUs) with the asynchronous stochastic gradient descent (ASGD) algorithm [7].

A significant improvement was achieved by IBM with the application of sequence discriminative training (SDT) [8]. However, the use of Hessian-free training was required to reduce the larger training time [9]. The development of the context dependent (CD)-DNN-HMM has proved very successful in speech recognition. DNNs have also shown a reduction in the requirement for speech preprocessing [10]. DNNs using only log Mel-scale filter bank preprocessing have been shown to outperform DNNs using Mel-frequency cepstral coefficients (MFFC) [11].

As of 2013, long short-term memories (LSTMs) which are a form of recurrent neural networks (RNNs) have been reported to have the lowest phone error rate (PER) when benchmarked on the TIMIT[1] corpus [13]. LSTMs can realise a greater recognition accuracy than CD-DNN-HMMs even with a reduced network size. The implementation of these techniques makes use of both the ASGD and the truncated back propagation through time (BPTT) algorithms.

3 System Overview

The development of a speech-to-text solution, in the context of the above-mentioned aviation problem, first entails the definition of system requirements in terms of typical speech recognition parameters. Since communication between pilots and air traffic controllers is continuous and involves multiple speakers, a continuous accent independent model is developed. Although aviation communication necessitates a comprehensive vocabulary, only the NATO phonetic alphabet and numerals are modelled. The use of a reduced vocabulary is justified by the implementation of phoneme models which are adaptable to larger vocabularies.

A summarised overview of the system is presented in Figure 1 which shows all subsystems that are developed in this paper.

The system comprises of six core components which are executed in sequence to accurately transcribe spoken speech. Although each component operates independently, the system is configurable for two modes of operation, training and classification, which are denoted in Figure 1 by the dotted and solid lines respectively. The training mode is used to develop the acoustic models which are stored in the phoneme database while the classification mode is used to perform transcription of speech. The first component of the system is the feature extraction subsystem which receives raw speech from either the microphone or speech database (depending on the operating mode) and generates a temporal sequence of feature vectors. The following component is the acoustic model which generates phoneme models and stores them in the phoneme database during the training mode. During classification, this component computes the likelihoods of temporal sequences of feature vectors with respect to the existing phoneme models stored in the phoneme database. The next subsystem is the hypothesis search which determines the transcription with the greatest likelihood. To allow for classification of continuous speech, the hypothesis search interfaces with a language model used for inter-word modelling. The system architecture can be further simplified to a linear model following the sequence of speech input, feature extraction, acoustic model, hypothesis search and transcription output with the language model and database providing auxiliary services.

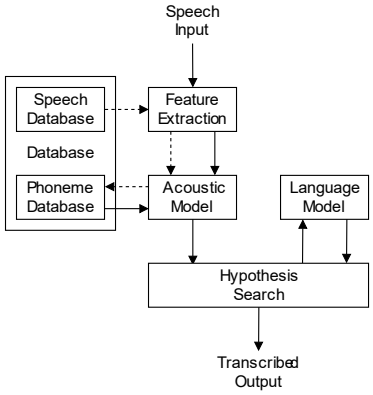


Figure 1. Overview of the speech-to-text system architecture

Since the design of the system that follows involves significant mathematical derivation, the notation that is used is defined in appendix B.

4. System Design

A. Feature extraction

Many feature extraction techniques for speech recognition already exist [14-15]. However, none of the techniques consistently outperform their alternatives, independent of

[1] A common benchmark for speech recognition systems is PER on the TIMIT corpus [12].

operational conditions [14]. On the other hand, certain techniques such as Mel-Frequency Cepstral Coefficients (MFCC) and Perceptual Linear Prediction (PLP) are very well established and have been shown to typically perform within a narrow margin of the best techniques for the given conditions [11]. In [16], it was established that MFCC with delta (Δ-MFCC) and delta-delta (ΔΔ-MFCC) coefficients outperformed MFCC without the delta coefficients. Therefore, ΔΔ-MFCC is selected as a baseline in this paper. This baseline enables a reference point for comparison to previous work and creates a new baseline for future comparisons.

The ΔΔ-MFCC feature extraction technique consists of multiple independent steps that are shown in Figure 2.

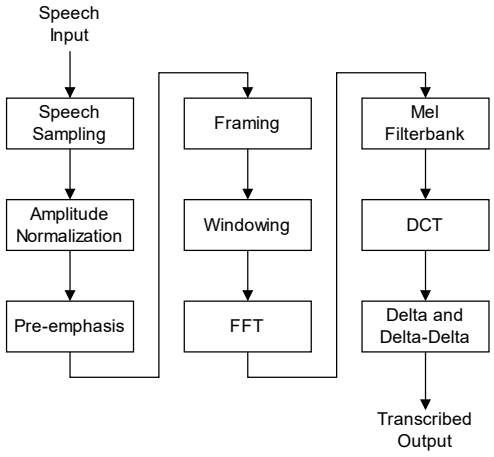


Figure 2. Overview of the speech-to-text system architecture

1) Speech Sampling: Since digitising and capturing of speech is a lossy process, the sampling parameters need to maintain sufficient information to classify speech. For consistency, the Samson C01U Pro microphone with a pop filter and Samson Sound Deck software version 1.1.2.0 was used for all recordings. The recordings were stored in WAVE format with 16-bit quantisation and a sampling frequency of 16 *kHz*. According to the Nyquist-Shannon sampling criterion, this sampling rate enables the capturing of the speech waveforms with frequencies within the 8 *kHz* baseband (audio wideband). This is sufficient information for speech decoding [17].

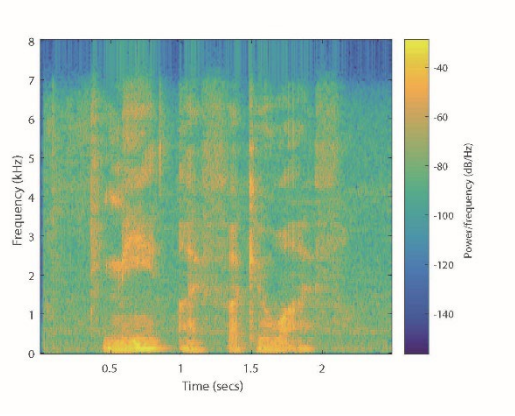
2) Amplitude Normalisation: After sampling speech, the amplitude is scaled using the absolute maximum as the normalisation factor. This ensures that the features extracted from the signal are independent of the magnitude of the original sampled speech.

3) Pre-emphasis: Since speech energy tends to be more densely distributed at lower frequencies, a first order high-pass filter is used to marginally suppress the lower frequency amplitudes, thus emphasising the higher frequency amplitudes [17]. Therefore, the pre-emphasis filter is given by Equation (1), with α denoting the pre-emphasis coefficient.

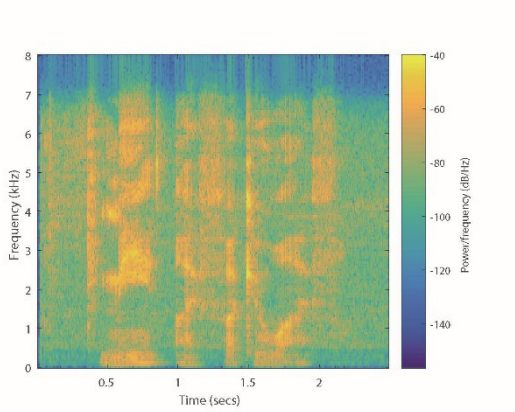
$$y[n] = x[n] - \alpha x[n-1] \quad where \quad 0.9 \leq \alpha \leq 1. \tag{1}$$

where, x(n) is speech signal and y(n) is the output signal.

Figure 3 illustrates the effect of the pre-emphasis filter where Figure 3(a) and Figure 3(b) are the speech spectrograms without and with the pre-emphasis filter respectively. Close inspection reveals the amplitude differences in both the low and high frequency bands. Therefore, a pre-emphasis filter coefficient of $\alpha = 0.95$ was selected.



(a) Without pre-emphasis filter



(b) With pre-emphasis filter

Figure 3. Speech spectrogram without (a) and (b) with the pre-emphasis filter

4) Framing: Since speech is a temporally variant signal it can only be considered stationary for very short time frames. Based on previous work and literature a time frame of 25 *ms* is selected with an overlap of 10 *ms* [17]. The overlap is necessary to ensure that the information is not lost at the frame boundaries due to windowing. Before windowing, the frame energy is calculated using Equation (2) where T is the frame length and $x[t]$ is the frame amplitude at time t.

$$E = \log\left(\sum_{t=0}^{T} x^2[t]\right). \tag{2}$$

5) Windowing: To convert each stationary frame to the frequency domain, a window function is first applied. The window function is required to remove discontinuities at the frame boundaries. This can be achieved using Discrete Fourier Transform (DFT) for example. The window function achieves this by tapering the frame boundary values to zero. Both the Hamming and Hann window functions are considered and both are applicable but the Hamming window is selected due to the more uniform distribution of spectral leakage. The Hamming window is given by the Equation (3), where the signal in a frame is denoted by $w(n)$, $n = 0, \ldots L\text{-}1$ and L is number of samples in each frame.

$$w[n] = \begin{cases} 0.54\text{-}0.46\cos\left(\frac{2\pi n}{L}\right) & 0 \leq n \leq L\text{-}1 \\ 0 & otherwise \end{cases}. \tag{3}$$

Equation (4) shows how the Hamming window function $w[n]$ is applied to the pre-emphasised frame, $x[n]$, to produce $y[n]$, the windowed frame. The result is depicted in Figure 4 where the frame is enveloped by the Hamming window function.

$$y[n] = w[n]x[n]. \tag{4}$$

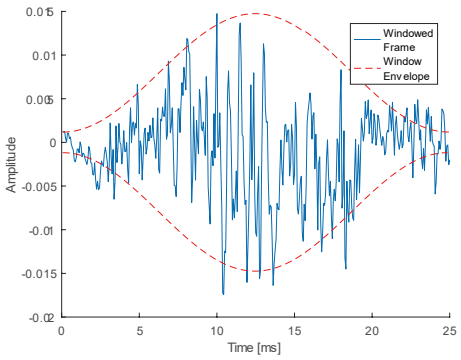


Figure 4. 25 ms speech frame after windowing

6) FFT: The Fast Fourier Transform (FFT) is a more efficient implementation of the DFT, Equation (5), which is used to convert the windowed frame from the time domain to the frequency domain.

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-2\pi jkn/N} = \sum_{n=0}^{N-1} x(n)w_n^k. \tag{5}$$

The Cooley-Tukey algorithm was used to perform the FFT by computing only the essential twiddle factors, w_N^k. As shown in Equation (6), the twiddle factors are symmetric and this redundancy significantly reduces the number of computations.

$$w_N^k = e^{-2\pi jk/N}. \tag{6}$$

This technique, however, constrains the FFT to sizes that are exponents of base 2, therefore, a 512-point FFT with zero padding was used for each frame.

7) Mel-Frequency Triangular Filters: Since human hearing is not linear with respect to frequency, the frequency spectrum is warped to match the sensitivity of the human ear. To achieve this, the Mel-frequency scale is used since it models frequencies below 1000 *Hz* linearly and frequencies above 1000 *Hz* logarithmically which is similar to that of human hearing [17]. The Mel-frequency scale is given by equation (7) below.

$$mel(f) = 1127\ln\left(1 + \frac{f}{700}\right). \tag{7}$$

To apply Mel-frequency scaling to the FFT of the frame, a Mel-frequency filterbank is used. The filterbank is represented by overlapping triangular filters warped according to the Mel-frequency scale. Although the speech sampling bandwidth was selected as $8\,kHz$, very little speech information resides outside the $100\,Hz$ to $6.8\,kHz$ frequency band [18]. Therefore the minimum and maximum frequencies for the Mel-frequency filterbank are set as $f_{low} = 100\,Hz$ and $f_{high} = 6.8\,kHz$ respectively. Additionally, the number of filters are selected to be $M = 26$ since that reduces the number of features significantly but still retains sufficient features to perform the Discrete Cosine Transform (DCT).

To calculate the positions of the filters, the frequency bandwidth $(f_{low} - f_{high})$ is divided by $M + 2$. The additional two positions are required since the filters have 50% overlap implying that both the boundary filters require an extra position each. Once the positions are obtained they are Mel-frequency scaled using Equation (7) and rounded to the nearest FFT bin. The result is stored in the vector $f(m)$ where $0 \leq m \leq M + 1$.

Finally, equation (8) is used to generate the Mel-frequency triangular filters. F_m represents the m^{th} Mel-frequency filter, with $1 \leq m \leq M$. The frequency input is given by k.

$$Fm(k)\begin{cases} 0, & \mathrm{k} < \mathrm{f(m-1)} \\ \frac{\mathrm{k-f(m-1)}}{f(m)-f(m-1)}, & \mathrm{f(m-1)} \leq \mathrm{k} \leq \mathrm{f(m)} \\ \frac{\mathrm{f(m+1)-k}}{f(m+1)-f(m)}, & \mathrm{f(m)} \leq \mathrm{k} \leq \mathrm{f(m+1)} \\ 0, & \mathrm{f(m+1)} \leq k \end{cases}. \tag{8}$$

Figure 5 shows the Mel-frequency filterbank with all 26 filters, Mel-frequency scaled across the $100\,Hz$ to $6.8\,kHz$ bandwidth. Included in this figure is the cumulative summation of the filter amplitudes across the bandwidth. This shows that the filters have a net effect of multiplying the frequencies within the bandwidth by one. The frequencies outside the bandwidth are multiplied by zero and therefore disregarded.

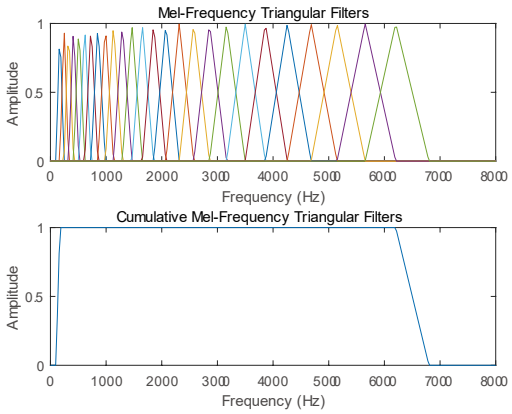


Figure 5. Mel-frequency filterbank with 26 filters and bandwidth between $100\,Hz$ and $6.8\,kHz$

Each of the Mel-frequency filters in the filterbank are multiplied with the frame and summed to get the respective filter coefficient. This results in 26 Mel-frequency filter coefficients that are logged to account for the logarithmic perception of human hearing.

8) DCT: The next step in calculating the MFCC features is to generate the cepstral coefficients using the Discrete Cosine Transform type 3 (DCT-3). The DCT-3 is the inverse of the DCT-2 which is identical to the DFT under certain

conditions. Therefore, the DCT-3 parallels the behaviour of the inverse DFT (IDFT) whilst decorrelating the cepstral coefficients. Although principal component analysis (PCA) is typically used for decorellating features, the DCT performs sufficiently as an approximation. The decorellation of features is a useful property especially when the features are used as input to GMMs. The DCT-3 is given by Equation (9) [19]. The i^{th} cepstral coefficient is given by c_i, where m_j is the j^{th} Mel-frequency filter coefficient and M is the number of filters in the Mel-frequency filterbank.

$$c_i = \sqrt{\frac{2}{M}} \sum_{j=1}^{M} m_j \cos\left(\frac{i\pi}{N}(j - 0.5)\right). \quad \textbf{(9)}$$

9) Delta and Delta-Delta Coefficients: The choice of 12 cepstral coefficients from the DCT is considered. This is combined with the frame log energy coefficient to form 13 features. Although MFCC is a very effective feature extraction technique, it can be enhanced by adding delta/velocity (Δ) and delta-delta/acceleration (ΔΔ) coefficients. These are derived from the original 13 coefficients as well as past and future coefficients. Equation (10) calculates the delta coefficients at time t, given by the vector $\Delta(t)$, using the MFFC coefficients, given by vector $c(t)$. The delta factor is set to $N = 2$. The same equation can also be used to calculate the delta-delta coefficients by substituting $\Delta(t)$ and $c(t)$ with $\Delta\Delta(t)$ and $\Delta(t)$ respectively.

$$\Delta(t) = \frac{\sum_{n=1}^{N} n(c(t+n)+c(t-n))}{2\sum_{n=1}^{N} n^2}. \quad \textbf{(10)}$$

Figure 6, depicts a sample of speech represented in the time domain in the first graph and represented as a spectrogram in the second graph. The final graph shows the MFCC, delta MFCC and delta-delta MFCC features for the same sample of speech. Table 1 indicates the order of the features in Figure 6.

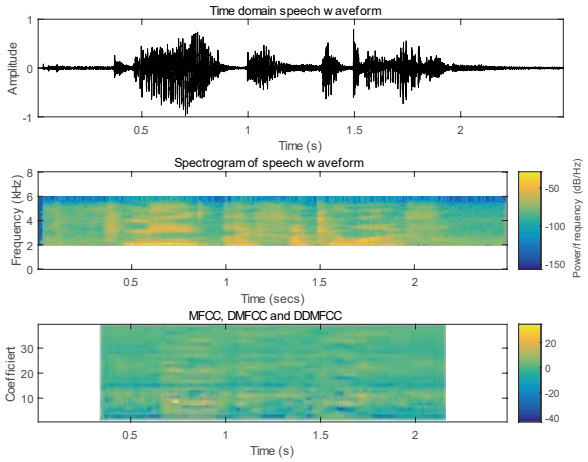


Figure 6. Time domain, spectrogram and MFCC features of a speech sample

Table 1. Description of the different features shown in Figure 6

Feature	Description
1-12	Cepstral Coefficients
13	Log Energy
14-25	Delta Cepstral Coefficients
26	Delta Log Energy
27-38	Double-Delta Cepstral Coefficients
39	Double-Delta Log Energy

B. Acoustic model

The acoustic model is developed using HMMs with GMM probability density functions (GMM-HMMs). Each English phoneme, as defined by the CMU pronouncing dictionary in Table 2, is modelled by a GMM-HMM. Words are formed by concatenating phonemes according to their CMU transcriptions. Finally, continuous speech is modelled by a sequence of words and optional silences.

Table 2. List of all 40 phonemes in English defined by the CMU pronouncing dictionary [20]

AA	AE	AH	AO	AW	AY	B	CH
D	DH	EH	ER	EY	F	G	HH
IH	IY	JH	L	K	M	N	NG
OW	OY	P	R	S	SH	T	TH
UH	UW	V	W	Y	Z	ZH	SIL

1) Hidden Markov Models: Speech can be represented by a discrete sequence of phonemes (states) which vary over a discrete sequence of frames (time intervals). However, these phonemes can only be inferred (observed) by MFCC features which are extracted from the frames of speech. To make accurate statistically inferences the relationship between the MFCC features and phonemes are modelled by GMMs (probability density functions). Therefore, assuming that the Markov Property holds, HMMs are well suited for this application.

By modelling the initial phoneme likelihoods, phoneme transition likelihoods and GMMs, the likelihood of a phoneme sequence (word/sentence) given the phoneme models can be determined. The phoneme models (HMMs) were selected to have a 3-state feedforward structure as depicted in Figure 7. The feedforward structure permits only transitions to the same state or next (forward) state. Although a phoneme can be modelled by a single state, the first and last states are used to model the glide-on and glide-off inter-phoneme transitions. The silence phonemes were, however, modelled by a combination of 1, 2 and 3 states for varied lengths of pauses between words.

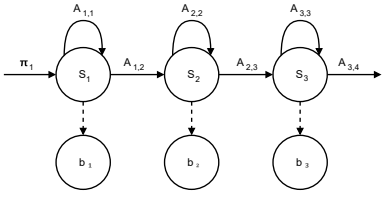


Figure 7. 3-State feed forward Hidden Markov Model

The following parameters were used to define the phoneme models:

- $\boldsymbol{\pi}$: Initial state likelihood vector, see Equation (11).
- $\boldsymbol{A}$: State transition likelihood matrix, see Equation (12).
- $\boldsymbol{b}$: Observation likelihoods, given by GMMs.

Note that the exit state transition likelihood, $A_{3,4}$, was calculated as $A_{3,4} = 1 - A_{3,3}$ since each of the rows in the state transition likelihood matrix are row stochastic. The transition likelihood, $A_{3,4}$, is used to chain the GMM-HMMs as part of embedded training.

$$\boldsymbol{\pi} = [1, 0, 0]. \tag{11}$$

$$\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & 0 \\ 0 & A_{2,2} & A_{2,3} \\ 0 & 0 & A_{3,3} \end{bmatrix}. \tag{12}$$

Furthermore, the phonemes are modelled as monophones instead of triphones. The distinction is that triphones require models for all combinations of adjacent phonemes. Therefore, assuming 40 distinct phonemes, $(40)^3 = 64\,000$ unique triphones need to be modelled. However, of the $64\,000$ only about $20\,000$ triphones are typically modelled since the remaining triphones do not occur in English speech [17]. State tying may be used to reduce the data and time required to triphone modelling which is not feasible for this paper.

2) Gaussian Mixture Models: As was mentioned earlier, Gaussian Mixture Models (GMMs) were used as the probability density functions for each state of the phoneme models. A Gaussian mixture model is effectively the mixture (summation) of M Gaussians in D dimensions with mixing proportions $\boldsymbol{c}$. To illustrate this, Figure 8 shows an example of a GMM with $M = 3$ mixtures and $D = 2$ dimensions. In this case, the mixing proportions are set to $\boldsymbol{c} = [0.4249, 0.2570, 0.3181]$.

Also, note that the GMM proportions always sum to unity as shown in Equation (13).

$$\sum_{m=1}^{M} c_m = 1. \tag{13}$$

Since the features extracted using MFCC are multidimensional, multivariate GMMs were used. Multivariate GMMs comprise of a mixture of multivariate Gaussians, which are individually expressed by Equation (14) below.

$$N(o, \Sigma, \mu) = \left(\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} \times \exp\left[-\frac{1}{2}(o-\mu)^{\tau}\Sigma^{-1}(o-\mu)\right]\right). \tag{14}$$

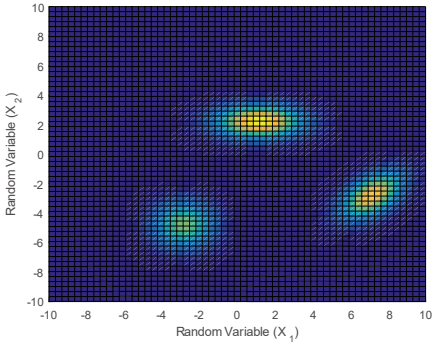
The symbols, $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$, represent the $(D \times D)$ Gaussian covariance matrix and the Gaussian mean vector of length D respectively. The Gaussian mean parametrised by i as $\boldsymbol{\mu}_i$ defines the mean of the i^{th} dimension vector component. The Gaussian covariance parametrised by i and j as $\boldsymbol{\Sigma}_{i,j}$ defines the covariance of the i^{th} dimension vector component with respect to the j^{th}dimension vector component. o represents the multivariate observation vector.

Equation (15) gives the Gaussian Mixture Model, $b(o)$, that is generated by summing each of the multivariate Gaussians multiplied by the associated proportions. The m^{th} proportion, covariance matrix and mean vector are defined by c_m, $\bar{\boldsymbol{\Sigma}}^{(m)}$ and $\bar{\mu}^{(m)}$ respectively.

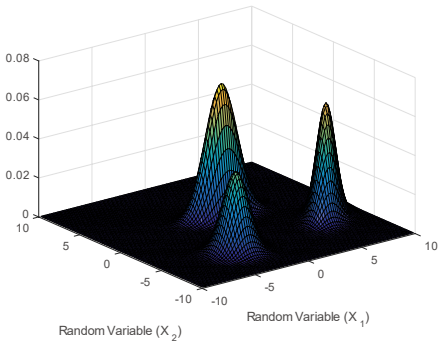
$$b(o) = \sum_{m=1}^{M} c_m N(o, \bar{\Sigma}^{(m)}, \bar{\mu}^{(m)} = \sum_{m=1}^{M}\left(\frac{c_m}{(2\pi)^{\frac{D}{2}}|\bar{\Sigma}^{(m)}|^{\frac{1}{2}}} \times \exp\left[-\frac{1}{2}\left(o-\bar{\mu}^{(m)}\right)^{\tau}\left[\bar{\Sigma}^{(m)}\right]^{-1}\left(o-\bar{\mu}^{(m)}\right)\right]\right). \tag{15}$$

An important design consideration relating to the covariance matrix, $\boldsymbol{\Sigma}$, arises when using GMMs since the covariance can either be represented as a full or diagonal matrix. The selection relates to decorrelation of features by means of the DCT. Assuming perfectly decorrelated features implies that diagonal covariance matrices may be used since decorrelation renders non-diagonal coefficients null. However, MFCC features are not completely decorrelated and using diagonal covariance matrices results in a trade off between performance and accuracy. It is found that the performance gain relative to the accuracy loss is not sufficient enough to justify the use of diagonal covariance matrices for this application.

Figure 8(a) illustrates the difference between diagonal and full covariance matrices. The two left most Gaussians are aligned with the axes since their covariance matrices are diagonal but the right most Gaussian does not have that restriction since it has a full covariance matrix. Therefore, fewer full covariance matrices are required to model correlated data points.



(a) 2D plot of a typical GMM with 3 mixtures in 2 dimensions



(b) 3D plot of a typical GMM with 3 mixtures in 2 dimensions

Figure 8. GMM with 3 mixtures in 2 dimensions represented by a 2D (a) and 3D (b) plot respectively.

3) Embedded Training: Embedded training, using the data in the speech database, is implemented to train the phoneme models. This enabled accurate modelling of phonemes using their phonetic transcriptions and eliminated the need to manually segment the speech data, which is tedious and produces inaccuracies. The transcriptions for the speech data were imported from the CMU pronouncing dictionary [20]. For example, the speech data transcribed by the word "alpha" is converted to the phoneme sequence [AE, L, F, AH]. By concatenating the GMM-HMM phonemes in that sequence, a composite GMM-HMM is constructed for that word. The composite GMM-HMM is initialised and then trained using the speech data for that word.

The first step of the embedded training involves initialising the GMM parameters ($\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$) to the mean and covariance of the entire speech database with a random factor to distribute the mixture components. This technique is similar to that of a flat-start except that it ensures that the GMMs overlap with sufficient data points to perform Expectation Maximisation (EM). It is found that if there are insufficient data points overlapping with the Gaussians the covariance matrices be- come singular. To resolve this, more data points are added by collecting more speech recordings. If the Gaussians still become singular, they are expanded by scaling the diagonals of the covariance matrices. Additionally, the number of training iterations are capped at 10 epochs.

Figure 9 represents only 2 dimension of a GMM-HMM with 3 states each comprising of 5 full covariance Gaussians. The Guassians are the large elliptic regions initialised to the data points as described above. The data points are deliberately segmented into three regions to visually verify the training process.

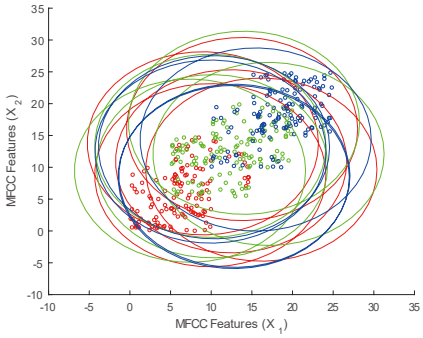


Figure 9. HMM-GMM initialisation with 3 states, 5 mixtures and 2 dimensions

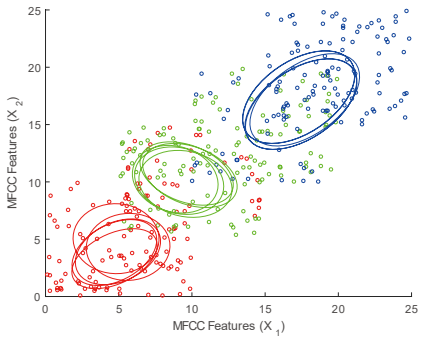


Figure 10. HMM-GMM of Figure 9 after 10 epochs of training

Figure 10 and Figure 11 illustrate the training of the GMMHMM in Figure 9. After 10 epochs the Gaussians are well generalised/aligned to the data points.

By 30 epochs the Gaussians started to over train. Apart from this leading to poor performance for speaker independence, it also leads to the issue described above where the Gaussian in the bottom left of Figure 11 may assume too little data and become singular.

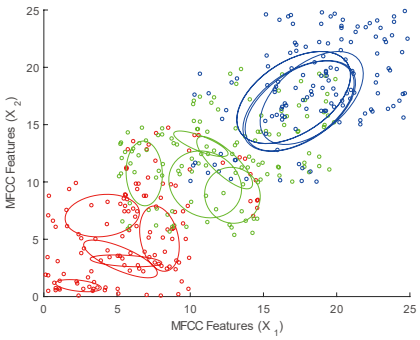


Figure 11. HMM-GMM of Figure 9 after 30 epochs of training

After initialising the GMMs, phoneme models are concatenated to form word models. The Baum-Welch algorithm, which iteratively re-estimates the model parameters, is implemented. The first step of the Baum-Welch algorithm is to compute the forward likelihoods using the forward algorithm.

To compute the forward likelihoods, the likelihood of an observation or multivariate data point being associated with astate in a GMM-HMM model needs to be defined. Equation 16 defines $b_j(o_t)$, the GMM probability density function for state j of the HMM at time t.

$$b_j(o_t) = \sum_{m=1}^{M}\left(\frac{c_m}{(2\pi)^{\frac{D}{2}}\left|\bar{\Sigma}_j{}^{(m)}\right|^{\frac{1}{2}}} \times \exp\left[-\frac{1}{2}\left(o_t - \bar{\mu}_j{}^{(m)}\right)^{\tau}\left[\bar{\Sigma}_j{}^{(m)}\right]^{-1}\left(o_t - \bar{\mu}_j{}^{(m)}\right)\right]\right). \quad \textbf{(16)}$$

Equation (17) is the parametrised form of Equation (16), using the parametrisation factor m.

$$b_j{}^{(m)}(o_t) = \left(\frac{c_m}{(2\pi)^{\frac{D}{2}}\left|\bar{\Sigma}_j{}^{(m)}\right|^{\frac{1}{2}}} \times \exp\left[-\frac{1}{2}\left(o_t - \bar{\mu}_j{}^{(m)}\right)^{\tau}\left[\bar{\Sigma}_j{}^{(m)}\right]^{-1}\left(o_t - \bar{\mu}_j{}^{(m)}\right)\right]\right). \quad \textbf{(17)}$$

Therefore, the relationship between Equation (16) and (17) is given by Equation (18).

$$b_j(o_t) = \sum_{m=1}^{M} b_j^{(m)}(o_t). \quad \textbf{(18)}$$

The forward algorithm can now be defined which is a recursive technique to calculate the likelihood of an observation resulting from state j at time t, given the observations up until and including time t. Since the algorithm is recursive, it is defined by Equations (19) and (20)

which are the initial step and recursive step respectively. The forward likelihoods are denoted by $\alpha_j(t)$. Note that $\boldsymbol{\pi}$ and $\boldsymbol{A}$ give the HMM initial state and state transition likelihoods respectively.

$$\alpha_j(1) = \pi_j b_j(o_1). \tag{19}$$

$$\alpha_j(t) = [\sum_{i=1}^{N} \alpha_i\,(t-1)A_{i,j}]b_j(o_t). \tag{20}$$

The most significant parameter of the forward algorithm is the full forward likelihood, $P(O|\lambda)$, where O represents the observation sequence and λ the GMM-HMM model. This is computed by Equation (21) using the forward likelihoods. T defines the length of the observation sequence.

$$P(O|\lambda)\ \sum_{i=1}^{N} \alpha_i\,(T). \tag{21}$$

Despite being mathematically sound, the forwar algorithm tends to produce very small likelihoods which are a result of successive multiplications of probabilities. To prevent underflow, the forward likelihoods are scaled at each time t to sum to one. This property is illustrated by Equation (22) where $\hat{\alpha}_j(t)$ is the scaled forward likelihood for state j at time t.

$$\sum_{j=1}^{N} \hat{\alpha}_j\,(t) = 1. \tag{22}$$

The scaled form of equations (19) and (20) are given by Equations (23) and (24) respectively. The forward scaling proof is given in the appendix.

$$\hat{\alpha}_j(1) = s_1\alpha_j(1). \tag{23}$$

$$\hat{\alpha}j(t) = \prod_{k=1}^{t} s_k\alpha_j\,(t). \tag{24}$$

Equations 23 and 24 can also be parametrised by the index *m* as given in Equations 25 and 26.

$$\alpha_j^{(m)}(1) = \pi_j b_j^{(m)}(o_1). \tag{25}$$

$$\alpha_j^{(m)}(t) = [\sum_{i=1}^{N} \alpha_i\,(t-1)A_{i,j}]b_j^{(m)}(o_t). \tag{26}$$

The second step of the Baum-Welch algorithm is to compute the backward likelihoods. Much like the forward algorithm, the backward algorithm is a recursive technique to calculate the likelihood of an observation resulting from state i at time t, given the observations from time T back until and including t. Since the algorithm is recursive, it is defined by Equations (27) and (28) which are the initial step and recursive step respectively. The backward likelihoods are denoted by
$\beta_i(t)$.

$$\beta_i(T) = 1. \tag{27}$$

$$\beta_j(t) = \sum_{j=1}^{N} A_{i,j}\,b_j(o_t + 1)\beta_j(t+1). \tag{28}$$

It was already shown by induction how the scaling factors are applied to the forward likelihoods. The exact same scale factors are applied to the backward likelihoods as shown in Equation (29).

$$\beta_j(t) = \prod_{k=1}^{t} s_k\beta_i(t). \tag{29}$$

Both the forward and backward algorithms are used as part of the Baum-Welch algorithm. Therefore, the likelihood of transitioning from state i to state j at time t is given by $\zeta_{i,j}(t)$ in Equation (30). $\zeta_{i,j}(t)$ is not defined for $t = T$ since it relies on the observation at time $t = T + 1$ which does not exist.

$$\zeta_{i,j}(t) = \frac{\alpha_t(t)A_{i,j}\,b_j(o_{t+1})\beta_j(t+1)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i(t)A_{i,j}b_j(o_{t+1})\beta_j(t+1)}$$

$$= \frac{\alpha_i(t)A_{i,j}b_j(o_{t+1})\beta_j(t+1)}{P(O|\lambda)}. \tag{30}$$

The scaled forward and backward likelihoods are substituted into equation (30) and the scaling factors cancel out with $P(O|\lambda)$ to produce equation (31).

$$\zeta_{i,j}(t) = \frac{\alpha_t(t)A_{i,j}\,b_j(o_{t+1})\beta_j(t+1)}{P(O|\lambda)}$$

$$= \frac{[\prod_{k=1}^{t} s_k]^{-1}\alpha_i(t)A_{i,j}\,b_j(o_{t+1})[\prod_{k=t+1}^{T} s_k]^{-1}\hat{\beta}_j(t+1)}{[\prod_{k=1}^{T} s_k]^{-1}}$$

$$= \hat{\alpha}_i(t)A_{i,j}\,b_j(o_{t+1})\hat{\beta}_j(t+1). \tag{31}$$

The likelihood of occupying the state i at time t is given by $\gamma_i(t)$ in equation (32).

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N}\alpha_j(t)\beta_j(t)}$$

$$= \frac{\alpha_i(t)\beta_i(t)}{P(O|\lambda)}\ . \tag{32}$$

The scaled forward and backward likelihoods are substituted into equation (32) and the scaling factors cancel out with $P(O|\lambda)$ to produce equation (33).

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{P(O|\lambda)}$$

$$= \frac{[\prod_{k=1}^{t} s_k]^{-1}\hat{\alpha}_i(t)\,[\prod_{k=t}^{T} s_k]^{-1}\hat{\beta}_i(t)}{[\prod_{k=1}^{T} s_k]^{-1}}$$

$$= \frac{\hat{\alpha}_i(t)\hat{\beta}_i(t)}{s_t}. \tag{33}$$

An important relationship is given by equation (34) which relates the state transition likelihoods $\zeta_{i,j}(t)$ to the state occupation likelihoods $\gamma_i(t)$.

$$\gamma_i(t) = \sum_{j=1}^{N}\zeta_{i,j}(t). \tag{34}$$

As with the forward algorithm, the state occupation likelihoods are used as part of the expectation maximisation of the GMMs. It is therefore imperative that the state occupation likelihoods also be calculated with respect to the Gaussian mixture components as shown in equation (35).

$$\gamma_i^{(m)}(t) = \frac{\alpha_i^{(m)}(t)\beta_i(t)}{P(O|\lambda)}. \quad (35)$$

The final step is expectation maximisation (EM) which updates the HMM transition likelihoods and the GMM proportion, covariance and mean parameters. The equations below assume that R observation sequences are used to train the GMM-HMM.

$$\hat{A}_{i,j} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r-1}\zeta_{i,j}^{r}(t)}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\gamma_i{}^{r}(t)}. \quad (36)$$

$$\hat{c}_i^{(m)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\gamma_i{}^{r,(m)}(t)}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\gamma_i{}^{r}(t)}. \quad (37)$$

$$\hat{\mu}_i^{(m)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\gamma_i{}^{r,(m)}(t)o_t{}^{r}}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\gamma_i{}^{r,(m)}(t)}. \quad (38)$$

$$\hat{\Sigma}_i^{(m)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\gamma_i{}^{r,(m)}(t)(o_t{}^{r}-\bar{\mu}_t^{(m)})(o_t{}^{r}-\bar{\mu}_t^{(m)})^{\tau}}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\gamma_i{}^{r,(m)}(t)}. \quad (39)$$

C. Language model

Once the likelihood of states given an observation can be determined, the language model is used to determine the interphoneme likelihood. Although it is planned that an n-gram language model be developed for speech recognition [21], its use is not imperative since the vocabulary size is too small. A language model only becomes effective provided the structure of the language is known. For a small vocabulary containing only NATO phonetic words and digits there is insufficient structural language to define a robust language model. A simpler but still very effective model is developed as shown in Figure 12.

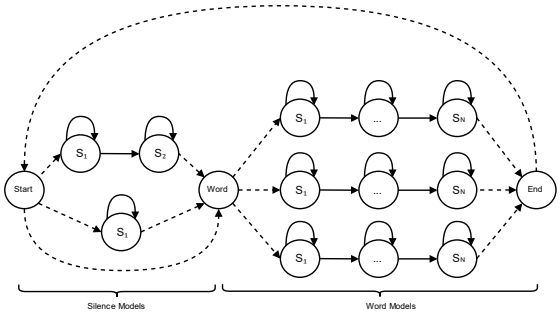


Figure 12. Structure of the language model for continuous speech recognition

This model provides an equal likelihood for each of the words making their likelihood completely dependent on the acoustic model. Additionally, optional silence models are considered between words to account for possible pauses.

D. Hypothesis search

The language model is converted into a trellis for Viterbi decoding as shown in Figure 13. Each word and silence model is expanded using the various pronunciations, and phonemes. The vertical axis defines the states of the trellis while the horizontal axis defines the observation sequence. The use of a trellis allows for dynamic programming and reduced computation.

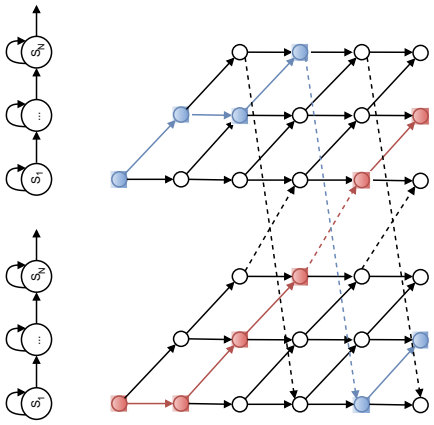


Figure 13. The structure of the trellis used for the Viterbi algorithm

Despite the reduced time taken to search the entire trellis, beam search could be used to prune the least likely search paths using a given tolerance. This will result in a tradeoff between accuracy and efficiency since beam search is greedy, thus not optimal. Beam search, however, is not implemented. For each observation, the likelihood of each GMMHMM state is computed using the GMM probability density function defined in equation (16). When the state likelihoods are computed for the next observation, only the maximum state likelihood prior to the current state is multiplied by the current state likelihood. To improve efficiency and accuracy, the probability density function uses logarithmic probabilities such that these probabilities are summed instead of multiplied. This also removes the need for scaling to prevent underflow when computing likelihoods of long observation sequences.

E. Database

The database is constructed using a hierarchy of JavaScript Object Notation (JSON) files to establish data relationships. Figure 14 defines the data relationships used for the database and Figure 15 defines the data relationships used for the dictionary. The use of a JSON database over a SQL database is to reduce the overhead of handling large quantities of speech data. This structure allowed for incremental loading of data whilst still being robust enough to maintain coherency. The hierarchical structure of the database ensured that loading of data required a reduced search space.

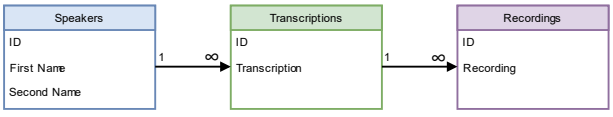


Figure 14. Database relationship for the speech database

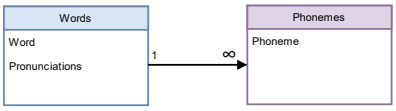


Figure 15. Database relationship for the speech dictionary

End-pointing using thresholding is performed on the speech recorded in the database to ensure that the training data accurately aligned with the transcriptions.

Table 3. Test results for both speakers trained speakers' data

World	Speaker	
	1	2
Alpha	100	100
Bravo	100	100
Charlie	100	100
Delta	90	100
Echo	100	100
Foxtrot	100	100
Golf	100	100
Hotel	100	100
India	100	100
Juliette	100	100
Kilo	100	100
Lima	100	100
Mike	100	100
November	100	100
Oscar	100	100
Papa	100	100
Quebec	100	100
Romeo	100	100
Sierra	100	100
Tango	100	100
Uniform	100	100
Victor	100	100
Whiskey	100	100
X-Ray	100	100
Yankee	100	100
Zulu	90	100
Zero	100	100
One	100	100
Two	80	100
Three	100	100
Four	90	100
Five	100	100
Six	100	100
Seven	100	100
Eight	100	100
Nine	80	100
Accuracy (%)	Speaker	
	1	2
	98.06	100

F. Seen speakers

When the speech recognition system was tested with the same speakers used for training, an average accuracy of 99.03% was obtained, Table 3. Two male speakers were used to conduct this test.

Table 4. Results for training using all the speakers except the selected speaker with reduced covariance scaling

World	Speaker		
	1	2	3
Alpha	100	90	0
Bravo	100	100	90
Charlie	100	100	90
Delta	100	100	100
Echo	100	80	80
Foxtrot	90	100	80
Golf	100	100	0
Hotel	100	100	100
India	90	0	80
Juliette	100	70	70
Kilo	100	100	40
Lima	100	70	100
Mike	90	100	100
November	100	100	0
Oscar	80	100	60
Papa	50	100	0
Quebec	100	80	70
Romeo	90	100	30
Sierra	100	100	100
Tango	100	60	80
Uniform	90	100	70
Victor	60	90	60
Whiskey	100	100	100
X-Ray	60	100	60
Yankee	100	100	50
Zulu	100	100	80
Zero	90	100	10
One	90	100	0
Two	0	80	60
Three	100	100	100
Four	70	100	0
Fife	70	100	90
Six	100	100	90
Seven	100	100	30
Eight	50	0	70
Niner	100	100	100
Accuracy (%)	Speaker		
	1	2	3
	88.06	89.44	62.22

G. Unseen speakers

When the speech recognition system was tested with unseen speakers, an average accuracy of above 80% was obtained, Table 4. The test was conducted using three male speakers. This was achieved after applying an improved form of covariance scaling. It was found that when the GMMs were trained using an individual speaker, the covariance matrix needed to be scaled to be more tolerant of the unseen speakers. The magnitude of this scaling reduces as the number of speakers in the training set increases. This is due to the variances between speakers implicitly scaling the covariance.

End-pointing and amplitude normalisation were two preprocessing techniques that resulted in significant gains in accuracy. It was also noted that an increase in the quantity of training data results in better speaker independence.

H. Response time

The average response time per word was measured as 3.06 seconds. It is important to note that full covariance matrices, a MATLAB simulation platform and a mid range ultrabook were used to obtain these results.

5. Conclusion

We proposed and tested a method for speech recognition that is not affected by accents as much as current approaches by using the speech recognition problem between pilots and air traffic controllers as a motivating application. The strongest aspect of the approach is the high level of accuracy for trained speakers. This is likely a result of both MFCC

decorrelating features and the manner in which GMMs accurately model the feature space with minimal iterations of training. Therefore, if the system was implemented with speaker enrolment, accents would be handled even more effectively. It was also discovered that the accuracy of the system with regard to speaker independence depends on the volume of training data. Despite limited training data, the use of covariance scaling, end-pointing and amplitude normalisation increased speaker independence drastically.

Appendix A

Vocabulary listing is shown in Table 5 to Table 7.

Table 5. Listing of the full NATO vocabulary used

NATO	
ALPHA	AE-L-F-AH
BRAVO	B-R-AA-V-OW
CHARLIE	CH-AA-R-L-IY
DELTA	D-EH-L-T-AH
ECHO	EH-K-OW
FOXTROT	F-AA-K-S-T-R-AA-T
GOLF	G-AA-L-F
	G-AO-L-F
HOTEL	HH-OW-T-EH-L
INDIA	IH-N-D-IY-AH
JULIET	JH-UW-L-IY-EH-T
KILO	K-IH-L-OW
LIMA	L-AY-M-AH
	L-IY-M-AH
MIKE	M-AY-K
NOVEMBER	N-OW-V-EH-M-B-ER
OSCAR	AO-S-K-ER
PAPA	P-AA-P-AH
QUEBEC	K-W-AH-B-EH-K
ROMEO	R-OW-M-IY-OW
SIERRA	S-IY-EH-R-AH
TANGO	T-AE-NG-G-OW
UNIFORM	Y-UW-N-AH-F-AO-R-M
VICTOR	V-IH-K-T-ER
WHISKEY	W-IH-S-K-IY
	HH-W-IH-S-K-IY
X-RAY	EH-K-S-R-EY
YANKEE	Y-AE-NG-K-IY
ZULU	Z-UW-L-UW

Table 6. Listing of the full numeric vocabulary used

Numeric	
ZERO	Z-IY-R-OW
ONE	W-AH-N
	HH-W-AH-N
TWO	T-UW
THREE	T-R-IY
FOUR	F-AO-R
FIFE	F-AY-F
SIX	S-IH-K-S
SEVEN	S-EH-V-AH-N
EIGHT	EY-T
NINER	N-AY-N-ER

Table 7. Listing of silence lengths used

Silence	
SILENCE	SIL-SIL-SIL
	SIL-SIL
	SIL

Appendix B

Mathematical notation

Symbol	Description
x	Variable
X	Constant
$\boldsymbol{x}$	Vector
$\boldsymbol{x_i}$	Vector indexed by variable i
$\boldsymbol{X}$	Matrix
$\boldsymbol{X}_{i,j}$	Matrix indexed by variables i, j
$\boldsymbol{X}^T$	Matrix transpose
$\bar{x}$	Array of vectors
$\bar{x}^{(i)}$	Array of vectors indexed by variable i
$\bar{\boldsymbol{X}}$	Array of matrices
$\bar{\boldsymbol{X}}^{(i)}$	Array of matrices indexed by variable i
$F(x)$	Function of x
$P(x)$	Probability of x
$[1, \ldots, N]$	Expanded vector
$\begin{bmatrix} X_{1,1} & \cdots & X_{1,N} \\ \vdots & \ddots & \vdots \\ X_{N,1} & \cdots & X_{N,N} \end{bmatrix}$	Expanded matrix

Appendix C

Proof of forward algorithm scaling

The intermediate forward likelihoods prior to scaling are defined by $\tilde{\alpha}_j(t)$.

$$\tilde{\alpha}_j(1) = \alpha_j(1). \tag{40}$$

$$\tilde{\alpha}_j(t) = [\textstyle\sum_{i=1}^{N} \tilde{\alpha}_i(t-1)\, A_{i,j}] b_j(o_t). \tag{41}$$

The scaling factor at time t is denoted by s_t and calculated using Equation 42.

$$s_t = \frac{1}{\sum_{j=1}^{N} \tilde{\alpha}_j(t)}\ . \tag{42}$$

As a result the scaled forward likelihood $\hat{\alpha}_j(t)$ can be calculated with respect to the intermediate scaled forward likelihood $\tilde{\alpha}_j(t)$. This is shown in Equation 43 using substitution of Equation 42.

$$\hat{\alpha}_j(t) = \frac{\tilde{\alpha}_j(t)}{\sum_{j=1}^{N} \tilde{\alpha}_j(t)} = s_t \tilde{\alpha}_j(t). \tag{43}$$

In order to find the relation between the scaled forward likelihood $\hat{\alpha}_j(t)$ and the original forward likelihood $\alpha_j(t)$, mathematical induction is required. Equation 45 continues the induction to the second time step $t = 2$.

$$\hat{\alpha}_j(1) = s_1\, \tilde{\alpha}_j(1) = s_1\, \alpha_j(1). \tag{44}$$

$$\hat{\alpha}_j(2) = s_2\ \tilde{\alpha}_j(2)$$

$$= s_2[\sum_{i=1}^{N} \hat{\alpha}_i(1)A_{i,j}\ ]\ b_j(o_t)$$

$$= s_2[\sum_{i=1}^{N} s_1\alpha_i(1)A_{i,j}\ ]\ b_j(o_t)$$

$$= s_1 s_2 \alpha_i(2). \quad \textbf{(45)}$$

It is intuitive that at time $t-1$, the following relation shown in Equation 46 will exist.

$$\hat{\alpha}_j(t-1) = \prod_{k=1}^{t-1} s_k\ \alpha_j(t-1). \quad \textbf{(46)}$$

Thus, the final step of the induction is shown in Equation 47 which defines the forward scaling relation.

$$\hat{\alpha}_j(t) = s_t[\sum_{i=1}^{N} \hat{\alpha}_i(t-1)A_{i,j}\ ]\ b_j(o_t)$$

$$= s_t[\sum_{i=1}^{N} \prod_{k=1}^{t-1} s_k\ \alpha_i(t-1)\ A_{i,j}]\ b_j(o_t)$$

$$= \prod_{k=1}^{t} s_k\ \alpha_j(t). \quad \textbf{(47)}$$

Using the scaled forward likelihoods, the full forward likelihood can be simplified to Equation 48.

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_i(T) = \frac{\sum_{i=1}^{N} \hat{\alpha}_i(T)}{\prod_{k=1}^{T} s_k} = [\prod_{k=1}^{T} s_k]^{-1}. \quad \textbf{(48)}$$

Equation 48 is used for the Viterbi algorithm, therefore, the full forward log-likelihood needs to be computed. This is more efficient since it only uses additions as shown in Equation 49.

$$\log\ P(O|\lambda) = \log[\prod_{k=1}^{t} s_k]^{-1} = -\sum_{k=1}^{T} s_k. \quad \textbf{(49)}$$

Finally, the forward likelihoods are parameterised by the Gaussian mixture components, m, in Equations 50 and 51 respectively.

$$\alpha_j^{(m)}(1) = \pi_j b_j^{(m)}(o_1). \quad \textbf{(50)}$$

$$\alpha_j^{(m)}(t) = [\sum_{i=1}^{N} \alpha_i(t-1)A_{i,j}]b_j^{(m)}(o_t). \quad \textbf{(51)}$$

Acknowledgment

The author would like to thank ATNS (Air Traffic Navigation Services) South Africa for their contribution to the project of speech resources, feedback and finances. Much thanks also goes to Mr. Andre Smith (ATNS) for providing guidance and feedback during the course of this research project.

References

[1] F. Biadsy, Automatic dialect and accent recognition and its application to speech recognition, PhD thesis, Columbia University, New York, USA, 2011.

[2] International Civil Aviation Organization, *Aeronautical telecommunications*, Annex 10, Vol. 2, pp. 5-14, October, 2001.

[3] L. R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257-286, February, 1989.

[4] L. Rabiner, B. Juang, An introduction to hidden markov models, *IEEE assp magazine*, Vol. 3, No. 1, pp. 4-16, January, 1986.

[5] A.-r. Mohamed, G. Dahl, G. Hinton, Deep Belief Networks for phone recognition, *Nips Workshop on Deep Learning for Speech Recognition and Related Applications*, Whister, BC, Canada, 2009, pp. 1-9.

[6] L. Deng, Deep learning: from speech recognition to language and multimodal processing, *APSIPA Transactions on Signal and Information Processing*, Vol. 5, Article No. e1, January, 2016.

[7] F. Niu, B. Recht, C. R´e, S. J. H. Wright, A lock-free approach to parallelizing stochastic gradient descent, November, 2011, https://arxiv.org/abs/1106.5730.

[8] B. Kingsbury, Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling, *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, 2009, pp. 3761-3764.

[9] T. N. Sainath, B. Kingsbury, B. Ramabhadran, Improving training time of deep belief networks through hybrid pre-training and parallel stochastic gradient descent, *Proc. NIPS Workshop on Log-linear Models*, Lake Tahoe, Nevada, USA, 2012, pp. 1-7.

[10] A.-r. Mohamed, G. Hinton, G. Penn, Understanding how deep belief networks perform acoustic modelling, *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, 2012, pp. 4273-4276.

[11] D. Yu, L. Deng, Automatic Speech Recognition: A Deep Learning Approach, Springer, 2015.

[12] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, N. Dahlgren, V. Zue, *TIMIT Acoustic-Phonetic Continuous Speech Corpus*, LDC93S1, Philadelphia: Linguistic Data Consortium, 1993, https://catalog.ldc.upenn.edu/ldc93s1.

[13] A. Graves, N. Jaitly, A.-r. Mohamed, Hybrid speech recognition with deep bidirectional lstm, *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, Olomouc, Czech Republic, 2013, pp. 273-278.

[14] N. Dave, Feature extraction methods lpc, plp and mfcc in speech recognition, *International journal for advance research in engineering and technology*, Vol. 1, No. 6, pp. 1-5, July, 2013.

[15] U. Sharma, S. Maheshkar, A. N. Mishra, Study of robust feature extraction techniques for speech recognition system, *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, Greater Noida, India, 2015, pp. 654-658.

[16] S. Nell, R. Malekian, *Speech-to-text conversion project*, Technical Report, University of Pretoria, November, 2015.

[17] D. Jurafsky, J. H. Martin, *Speech And Language Processing: An Introduction to Natural Language*

Processing, Computational Linguistics, and Speech Recognition, Pearson, 2009.
[18] S. K. Kopparapu, M. Laxminarayana, Choice of Mel filter bank in computing MFCC of a resampled speech, *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*, Kuala Lumpur, Malaysia, 2010, pp. 121-124.
[19] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, P. Woodland, *The HTK book*, Vol. 3, ResearchGate, 2006.
[20] K. Lenxo, *The CMU Pronouncing Dictionary*, Carneigie Mellon University, Pittsburgh, USA, 2016. http://www.speech.cs.cmu.edu/cgibin/cmudict.
[21] P.-S. Huang, P.-S. Chiu, J.-W. Chang, Y.-M. Huang, M.-C. Lee, A Study of Using Syntactic Cues in Short-text Similarity Measure, *Journal of Internet Technology*, Vol. 20, No. 3, pp. 839-850, May, 2019.
[22] K.-K. Tseng, X.-X. An, C. Chen, Online Handwritten Verification Algorithms Based on DTW and SVM, *Journal of Internet Technology*, Vol. 21, No. 6, pp. 1725-1732, November, 2020.

Biographies

Grant Zietsman received a B.Eng. degree in computer engineering from the University of Pretoria, Pretoria, South Africa, where he is currently pursuing postgraduate degree with the Department of Electrical Electronic and Computer Engineering. His research interests include intelligent systems and development of intelligent algorithms.

Reza Malekian (M'10), (SM'17) Reza Malekian is a professor of computer science at Malmö University, Sweden and an extraordinary professor in the Department of Electrical, Electronic and Computer Engineering at the University of Pretoria, South Africa, where he previously led the Advanced Sensor Networks research group and received the Vice-Chancellor and Principal's Exceptional Young Researcher's Award. He is also nominated for the Tage Erlander Prize, an award by the Royal Swedish Academy of Science for outstanding scientific research in natural sciences and technology. His research focuses on the Internet of Things, connectivity and sensor systems and networks.