The Sequential GMM:

A Gaussian Mixture Model Based Speaker Verification System that Captures Sequential Information

Stephen James Stafford

May 16, 2005

Table of Contents:

1 Introduction

This report presents a novel speaker verification system that generates a new feature set that captures long duration speaker identifying characteristics while taking advantage of the well-established and well-studied Gaussian Mixture Model system (GMM). Much of the innovation in the system is contained in the intelligent exploitation of traditional cepstral features such that temporal aspects of speech, which are otherwise disregarded in traditional GMM frameworks, can be explicitly modeled. The system consists of a collection of independent GMMs, one for each phoneme, built on these long duration feature vectors. The outputs of these GMMs are then combined at the score level using a neural network.

Despite using traditional tools with respect to the GMM and the front-end feature extraction, combining this system with a run-of-the-mill GMM system dramatically reduced both the equal error rate (EER) and the minimum value of the decision cost function (DCF) on a standard speaker verification test set, in comparison to the GMM system alone. This improvement indicates that the long duration features are capturing speaker characterizing information that the regular GMM ignores. The min DCF fell by nearly 65% and the EER fell by approximately 36%. Moreover, the new system's performance, when operating in isolation, approached that of the state-of-the-art GMM.

1.1 Motivation

Speaker recognition is a task that is familiar to everyone. When answering the telephone, people often know immediately who is on the other end of the line. Unfortunately, speaker recognition is not such a simple task for computers. Part of the problem is that it is difficult for humans to determine what characteristics they use in identifying speakers. Perhaps they recognize a phrase the person commonly uses or maybe just the way the person laughs. Human based speaker recognition can be studied, and has been to some extent [1]. However, perhaps humans are not the optimal system; perhaps machines can do much better.

There are a number of distinguishing speech characteristics that can be utilized, such as acoustic qualities, prosodic patterns, pronunciation preferences, and word usage, to name a few. The sources of these different pieces of information depend on factors ranging from the shape of the nasal passage to where the person was raised [2]. The aspiration for speaker recognition systems is to use all of the above-mentioned sources of information; simply stated, the goal is to capture every piece of information that reveals the identity of the speaker. The difficulty, however, is in modeling these complex speaker idiosyncrasies.

The current state-of-the-art system, the GMM, generates its speaker hypothesis based on information derived from frames, which are obtained by dividing the speech sample into approximately ten-millisecond segments. These frames are then used without ordering to model a speaker's voice. This so-called "bag of frames" technique generally works very well. However, there is good reason to believe that by treating each frame independently, and hence forfeiting sequential information, the GMM loses potentially speaker-identifying characteristics of the speech. The system proposed in this paper, which is

referred to as the *Sequential GMM*, harnesses the power and simplicity of the GMM while capturing sequential information by using frames that represent entire phonemes. A phoneme is typically between 50 and 300 milliseconds long; therefore, the new frames will represent a time span of approximately an order of magnitude longer than traditional frames. Part of the reason this long-term modeling is possible is due to increased quantities of training data available through NIST's Extended Data task [3]. As the availability of training data grows, the feasibility of modeling higher-level and potentially rarer temporal speaker idiosyncrasies grows.

1.2 Overview

The primary innovation presented in this paper is the method by which a new feature vector is created and where each new frame represents the information from an entire phoneme. This task is accomplished by stacking the frames that constitute a phoneme and treating this sequence of frames as a single new feature vector. Thus there is one new frame for each instance of a phoneme in the speech stream.

This approach introduces a few challenges. First, the system now requires a phoneme-level transcription of the acoustic data. Second, because the GMM must model a probability space of fixed dimension, each of the new feature vectors must be the same dimension. This is a significant hurdle because rates of speech can be highly variable, so different instances of the same phoneme will generally be of different duration. Since the new frames are created by stacking the old frames, phonemes of different length will create stacked frames of different dimension. As was mentioned, this system builds phoneme-specific GMMs; therefore, it is necessary to warp each sequence of frames, for a specific phoneme, to the same length. The process for warping will be explained in Section 4.5. The final issue introduced by this system is training the GMMs. Under the new system, there are fewer frames because there is only one feature vector per phoneme. Additionally, there are more model parameters to train due to two reasons: the dimension of the training vectors has grown, and a separate GMM is being trained for each phoneme.

The nature of the speaker recognition task is briefly described below; for further explanation see [4]. Below, the basic GMM system will be briefly described (Section 2), the creation of the new feature set will be explained (Section 4), a number of experiments will be investigated (Section 7), and possible future directions of the research will be proposed (Section 8).

2 The GMM

Before the GMM system is discussed, it is important to understand the speaker verification task. A single trial in the task consists of a test speech segment and a putative target speaker. The test segment contains speech from only one speaker, and the goal is to determine whether the test segment was created by the target speaker. Hence, the task can be formulated as a hypothesis test, where the hypotheses are denoted by H_0 and H_1. H_1 is the hypothesis that the test data was created by the target speaker, a

"match," and H_0 is the hypothesis that the test data was not created by the target speaker, but instead by an "impostor."

The decision criterion is given in the usual form for a hypothesis test,

$$P(\text{match} \mid \text{test data}) \underset{H_0}{\overset{H_1}{\gtrless}} P(\text{impostor} \mid \text{test data}). \quad (1)$$

This equation is further modified using Bayes rule, resulting in

$$P(\text{test data} \mid \text{match})\, P(\text{match}) \underset{H_0}{\overset{H_1}{\gtrless}} P(\text{test data} \mid \text{impostor})\, P(\text{impostor}). \quad (2)$$

This formulation gives the provably optimal solution, assuming the probabilities are accurately modeled; however, in speaker verification, it is assumed that priors are unknown to the researcher and are therefore disregarded. Hence, the metric of interest is measured in terms of a likelihood ratio:

$$\frac{P(\text{test data} \mid \text{match})}{P(\text{test data} \mid \text{impostor})} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma \quad (3)$$

If the likelihood ratio is above a threshold, γ, H_1 is accepted, otherwise H_0 is accepted.

The GMM system [5, 6] is an effort to model those probabilities. It is the de facto standard for text-independent speaker verification, and it performs very well despite its simplicity. The system attempts to probabilistically model the frames of speech using Gaussian mixtures, where a frame of speech is a vector resulting from some sort of signal processing on a slice of speech. Frames are typically computed over a 30ms window, at 10ms steps through the speech utterance.

The first step in constructing the system, assuming the frames have been created, is to create a universal background model (UBM) [5]. The UBM uses a GMM to model the frames of speech from a generic speaker and is, therefore, trained on frames from a large held-out set of speakers. The UBM can be used to calculate the probability of a frame being created by a generic, non-target speaker; thus it produces the denominator of the likelihood ratio. The GMM's parameters are trained using Expectation Maximization (EM). The trainable parameters are the Gaussian means, covariance matrices, and weights. The covariance matrices are assumed to be diagonal, to limit the number of parameters.

The second step is to create speaker-specific models. A speaker-specific GMM is created through MAP adaptation [7] of the means from the UBM. The adaptation is computed using training data, which consists of frames from speech utterances from the target speaker. The amount of training depends on the application. Naturally, there is a separate adapted GMM for each speaker of interest. The speaker-specific GMM is used to generate the numerator of the likelihood ratio.

The log likelihood ratio is calculated as

$$\sum_i \log \frac{P(x_i \mid \text{match})}{P(x_i \mid \text{impostor}),} \tag{4}$$

where x_i is a frame from the test data and the sum is over all of the frames. Factoring the joint probability of the test frames, from equation (3), into probability functions on the individual frames implicitly assumes that the frames are independent. This assumption is necessary because the GMM system contains no models of correlation among frames. With the UBM, the speaker-specific adapted model, and the frames from the test data, the likelihood ratio can be computed. The final step is to determine a threshold so a decision can be made.

3 System Overview

The Sequential GMM system can be broken into three stages, which will each be explained in Sections 4, 5, and 6, respectively:

1) Creating the new feature vectors
2) Developing the background and target models
3) Scoring and fusion

3.1 The Task

This system was run on the Extended Data task of the 2001 NIST Speaker Recognition Evaluation [3]. The data in the task are drawn from recorded telephone conversations of the switchboard one corpus which contains 2400 two-sided telephone conversations from around 540 speakers [8, 9], and the task is strictly defined as speaker verification, as described in Section 2. The test data for one trial consists of a single "conversation-side" of speech from a single speaker, and the training condition is one in which eight conversation-sides are available from the target speaker. The eight conversation-side condition is on the large side of the spectrum of training conditions, with many systems using much less. The Sequential GMM is well suited for a setting with more training data because it models larger, and therefore sparser, speech events. A conversation-side is the speech from one of the two speakers in a five-minute conversation.

The task is divided into six "splits," where each split consists of a disjoint set of speakers; therefore, testing on one split can use other splits in a non-cheating way. Splits 1, 2, and 3 have 4797 trials and splits 4, 5, and 6 have 5008 trials. When testing on the first three splits the last three splits were used to train the background models, and vice versa.

4 Developing the New Feature Set

The heart of the sequential GMM is the creation of the new feature vectors, in which the new features are assembled from the typical Mel-Frequency Cepstral Coefficients (MFCC) features [10]. Since there are two feature sets, one built from the other, particular care must be taken to note when the new feature vectors are being referenced. The process of creating these new feature vectors can be broken down into these six stages, which will each be examined in detail in the following subsections:

1) Feature extraction
2) Automatic Speech Recognition (ASR)
3) Frame sifting
4) Calculating average state lengths
5) Frame warping
6) Stacking the frames

4.1 Feature Extraction

This system uses the traditional MFCCs, C0 through C19 extracted using HTK [11]. The cepstra are produced every 10 ms over 25ms of speech. See Figure 1. The speech is preemphasised, with a preemphasis coefficient of 0.97, and a Hamming window is used to window the speech. Cepstral mean subtraction (CMS) [12] is applied on a per conversation-side basis, with the means estimated from the speech portion of the segment.

Delta parameters were not used because of the large dimension of the stacked feature vectors and because stacking neighboring frames should inherently encode this information. The primary difficulty with large feature vectors arises in training the large number of resulting GMM parameters. The new frames, resulting from multiple stacked frames, are as large as 300 dimensions without deltas. Using deltas would increase this number to 600. Training this increased number of parameters is limited by two factors: training data, and computation time. Although the mushrooming computation time could be dealt with, the amount of training data cannot. Data for training the background models can be pulled from other data corpora; however, the amount of data available for adapting target models is limited to eight conversations by the task.

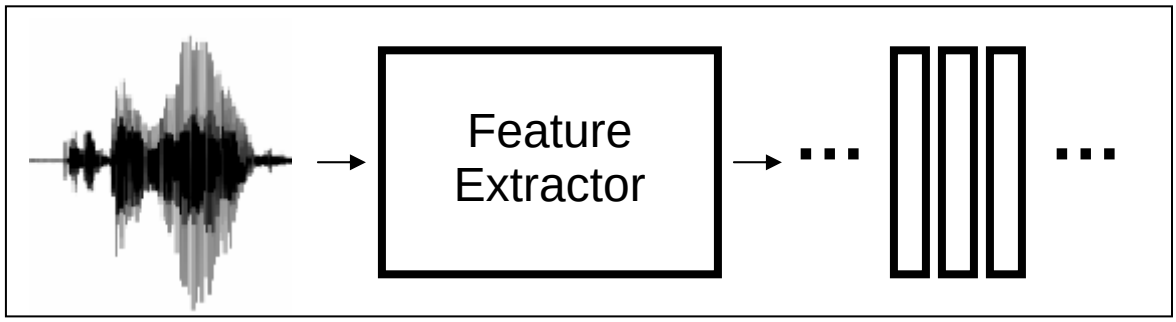

Figure 1: The speech waveform passes into the feature extractor, which does signal processing and produces a frame with 20 elements every 10 ms.

4.2 Automatic Speech Recognition (ASR)

The waveforms are separately run through a traditional word-based speech recognizer [13], courtesy of SRI [14]. Because the ASR system was trained using the same switchboard one data it was run on, somewhat stripped-down models were used, in order to make the ASR quality practical. As a by-product of the speech recognition, the system also produces phoneme and phoneme-state time alignments. Each phoneme, as modeled by a Hidden Markov Model (HMM), contains three states. The frame sequence created in the feature extraction is now labeled at the phoneme-state level (see Figure 2). The SRI phoneme set consists of the 47 phonemes listed in Figure 11. Some of the nonstandard phonemes include:

fpv: filled pause vowel (as in the vowel in “uh” or “um”),
fpn: filled pause nasal (as in the “m” in “um”),
bgn: background noise,
pum2: similar to fpv (sound of words like “hm”),
mtn: mouth noise, and
lau: laughter.

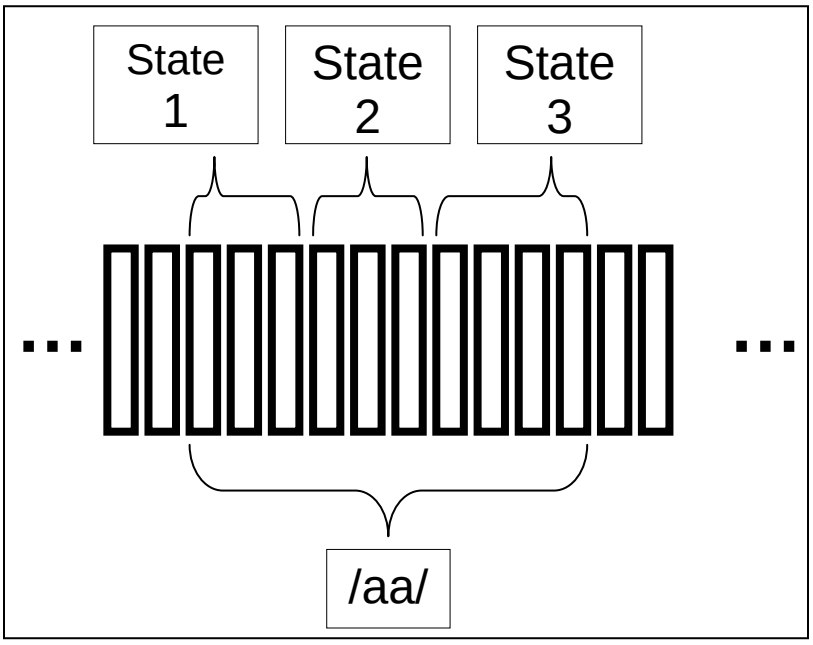

Figure 2: The sequence of frames has been tagged, with the center 10 frames belonging to the phoneme /aa/; states 1, 2, and 3 have three, three, and four frames, respectively.

4.3 Frame Sifting

Once the frames have been labeled by means of ASR, the frame sequences associated with the different phonemes are sifted apart. This can be pictured as 47 bins, one for each phoneme, and in each bin is a set of frame sequences. Naturally, each of the frame sequences in a bin would correspond to the phoneme for which the bin was labeled; moreover, it is crucial that the ordering of the frames within these sequences remain intact. After all, this system is attempting to model the time-evolution of phonemes.

Since different phonemes tend to be different lengths, it was decided not to try to warp all the frame sequences to the same target length, but rather warp each instance of a phoneme to the average length of that phoneme type. The decision to treat each of the phonemes separately is not without negative consequences. As was mentioned earlier, data sparsity is a serious issue with this system, and by treating each phoneme separately, the training data is spread out across 47 different GMMs. Moreover, it is actually worse than simply dividing the training data by the number of phonemes, because some phonemes are very infrequent. For example, the phoneme /zh/ often does not even appear in conversations in the switchboard one corpus.

4.4 Calculating Average State Lengths

Before the frame sequences can be warped, it is necessary to know to what length they will be warped. That is, how many frames will be in the sequence of frames for a phoneme, after warping? Choosing a rather long target length has the benefit of preserving the information in the sequence of frames, whereas choosing a short target length results in a smaller probability space. Each choice has its advantage. Reducing

the dimension of the probability space decreases the number of trainable parameters. On the other hand, condensing the frame sequences likely throws out some sequential information. For this system, a happy medium was chosen.

The number of desired frames is determined by finding the average number of frames in each phoneme-state over all instances of the phoneme-state in the data. That number is then rounded to the nearest integer. For example, after rounding, the phoneme /aa/ is found to have an average of three, five, and three frames, respectively, for its three states. Therefore, after warping, each /aa/ frame sequence will be eleven frames. This method is slightly objectionable because the target lengths are being determined from the same speakers (and conversations) the system is being tested on. Nevertheless, this is a minor infraction because these average lengths are reasonably stable across the corpus.

To summarize the preceding discussion: For each phoneme there is a set of frame sequences that are each labeled with three phoneme-states. Moreover, the desired number of frames for each phoneme-state is known. All that remains is to warp each of the frame-sequences to the appropriate size and then stack the frames.

4.5 Frame Warping

The key step in the new feature creation is a time warping which is implemented in order to make all frame sequences corresponding to the same phoneme have the same number of frames. This time warping is computed component-wise for each cepstrum. Figure 3 shows a sequence of frames for the phoneme /aa/ being warped to the appropriate length.

Two different methods of interpolation were used: linear, and sinc kernel, also known as the “sampling function,” convolution. A third method of warping simply collapsed each state of a phoneme into a single frame. This method is referred to as the “average.” Each of these methods are explained below.

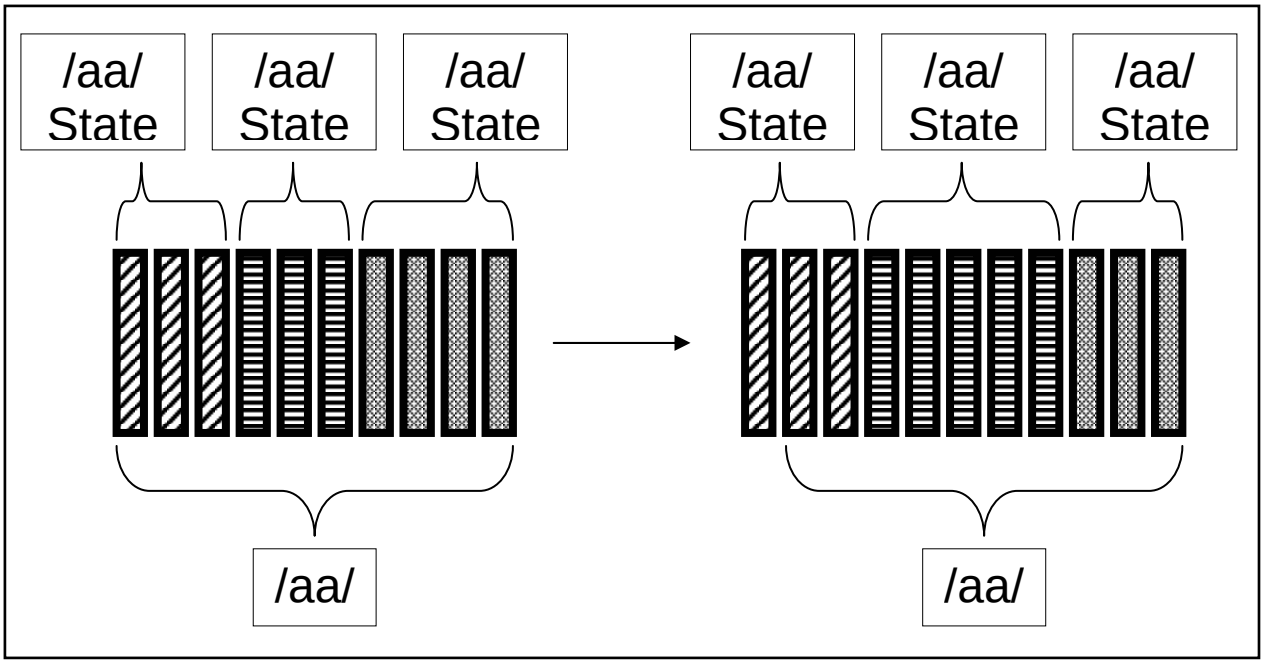

Figure 3: An instance of the phoneme /aa/ with three, three, and four frames, respectively, is warped into a frame sequences with three, five, and three frames, respectively.

4.5.1 Warping Method: Linear Interpolation

In this method, adjacent points are simply connected with a line, then sampled at evenly distributed points, with the first and last values remaining unchanged. The figures below are an example of the process for a single component in which a five-frame phoneme-state is warped to a seven-frame sequence. In Figure 4, there are five values of the cepstrum corresponding to a five-frame phoneme-state, and in Figure 5, these five points are linearly interpolated. Finally, Figure 6 shows the resulting seven cepstra values. In situations where a single frame must be warped into multiple frames, it is repeated.

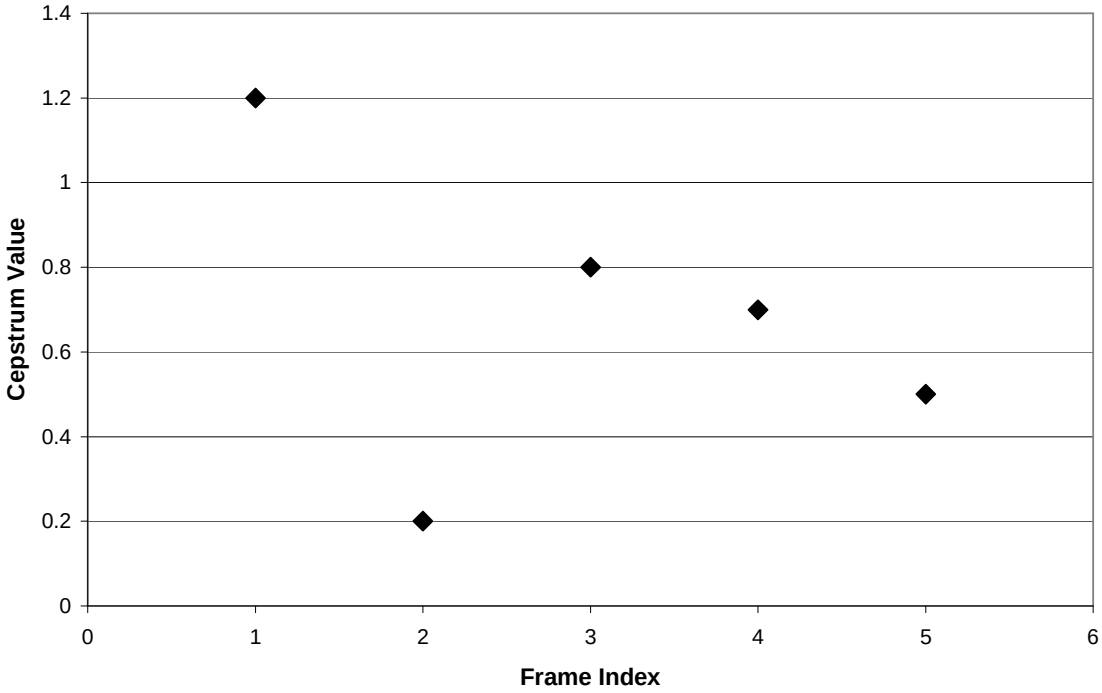

Figure 4: The value of a cepstrum over five frames.

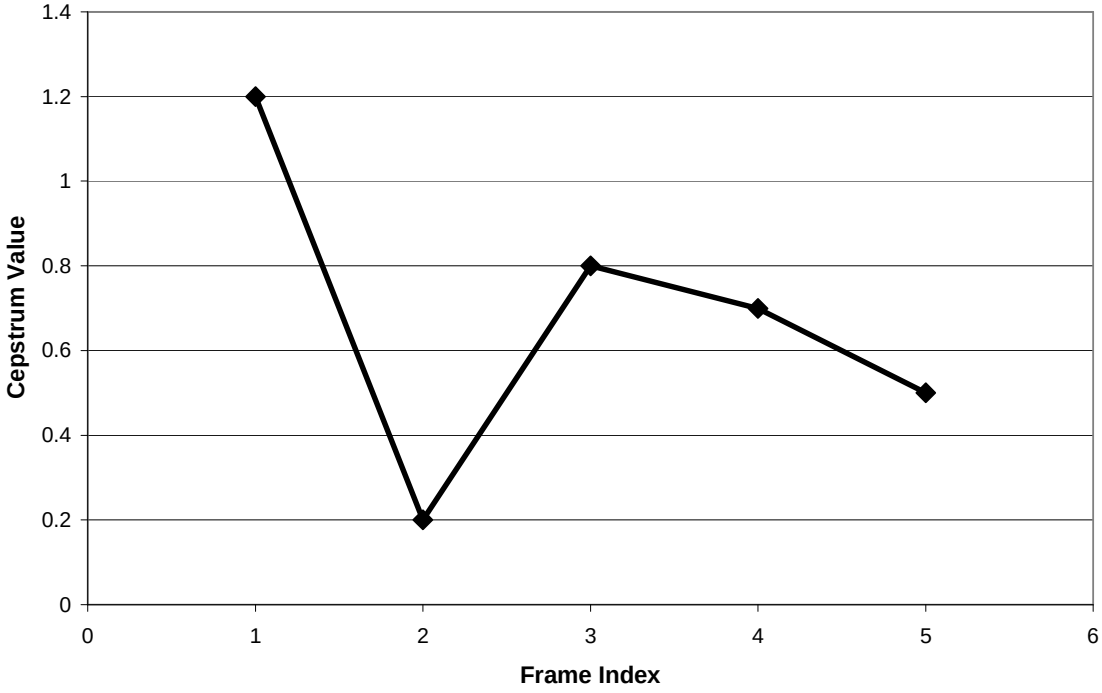

Figure 5: The five cepstra values from Figure 4 are linearly interpolated.

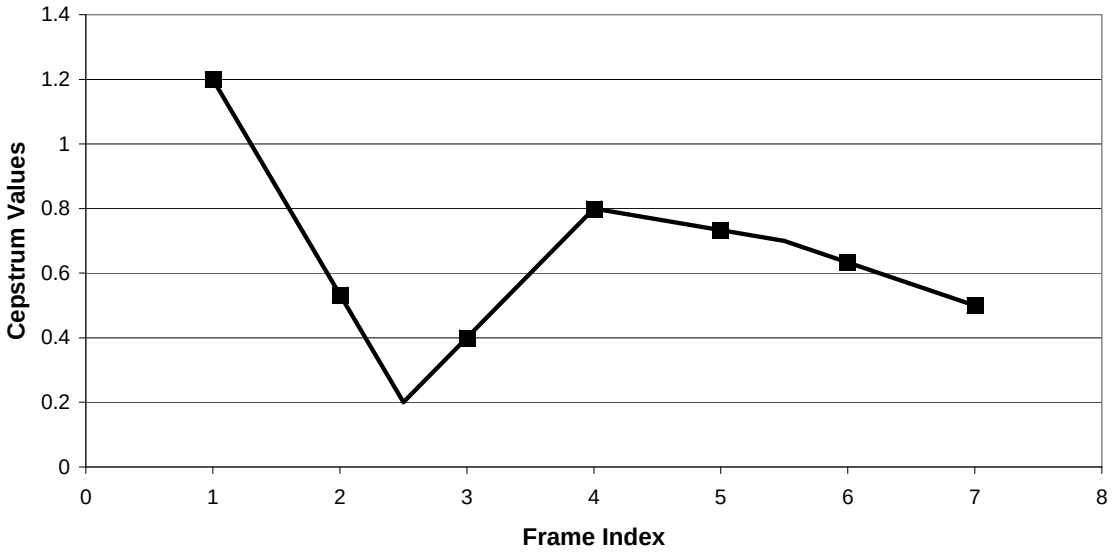

Figure 6: The linear interpolation of the five cepstra from Figure 5 is sampled at seven equally spaced intervals.

4.5.2 Warping Method: Sinc Kernel

This method of warping is based on a digital-signal-processing perspective. The discrete data points are treated as a sequence of appropriately weighted dirac deltas that are convolved with a sinc kernel and then sampled in the same fashion as in the linear interpolation case. The sinc kernel is defined as

$$\operatorname{sinc}(x) \equiv \begin{cases} 1 & \text{for } x = 0, \\ \dfrac{\sin(\pi x)}{\pi x} & \text{otherwise.} \end{cases}$$

See Figures 7, 8, and 9 for an example of the sinc warping method.

Figure 10 shows the difference between the two warping methods. The linear interpolation method is a weighted average of neighboring points. On the other hand, since the sinc interpolating function consists of numerous low frequency sines, the interpolation overshoots the original values. See Figure 10. From a heuristic standpoint, it is not clear which method is more appropriate.

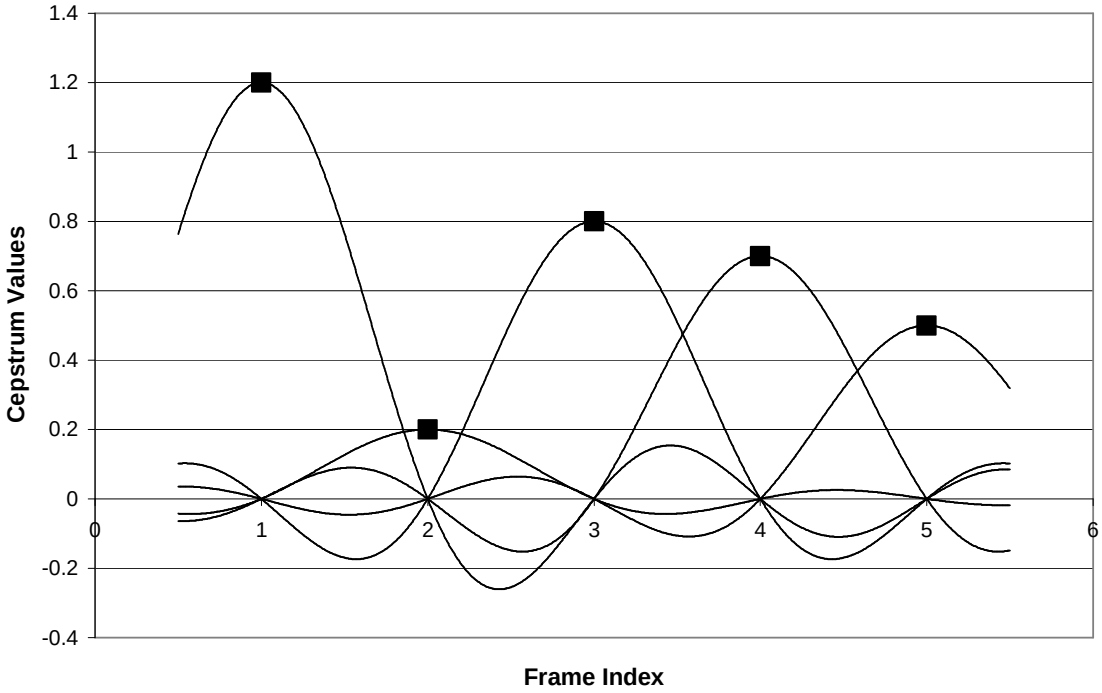

Figure 7: The five cepstra from Figure 4 are superimposed with a sinc of appropriate scale.

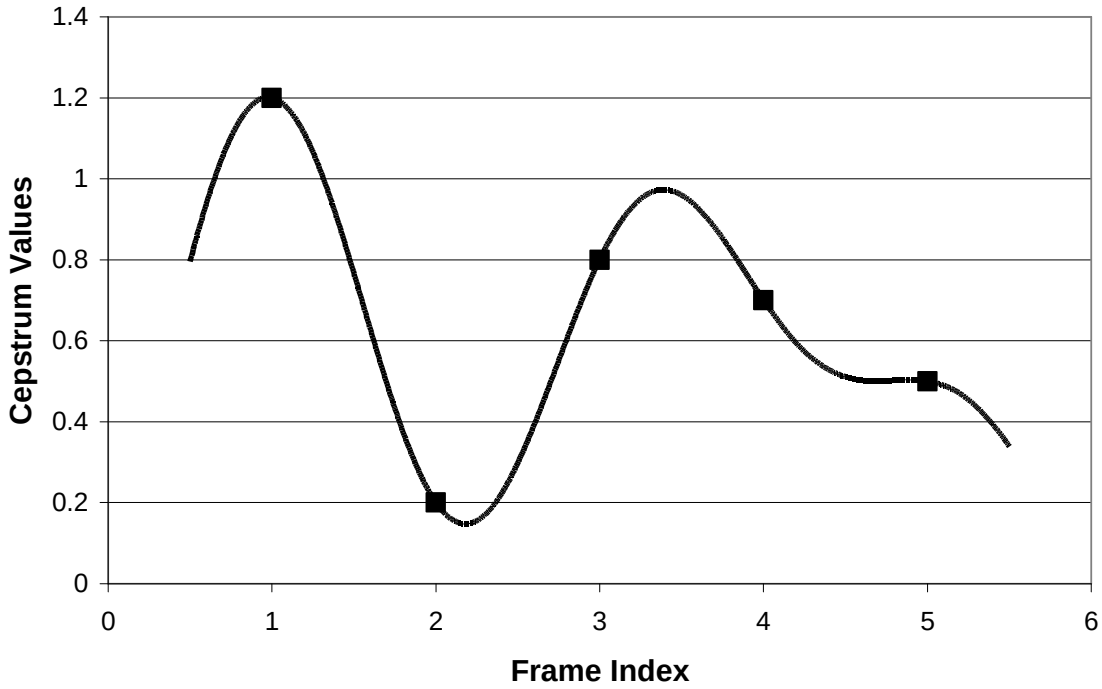

Figure 8: The five shifted and scaled sincs from Figure 7 are summed, to complete the convolution.

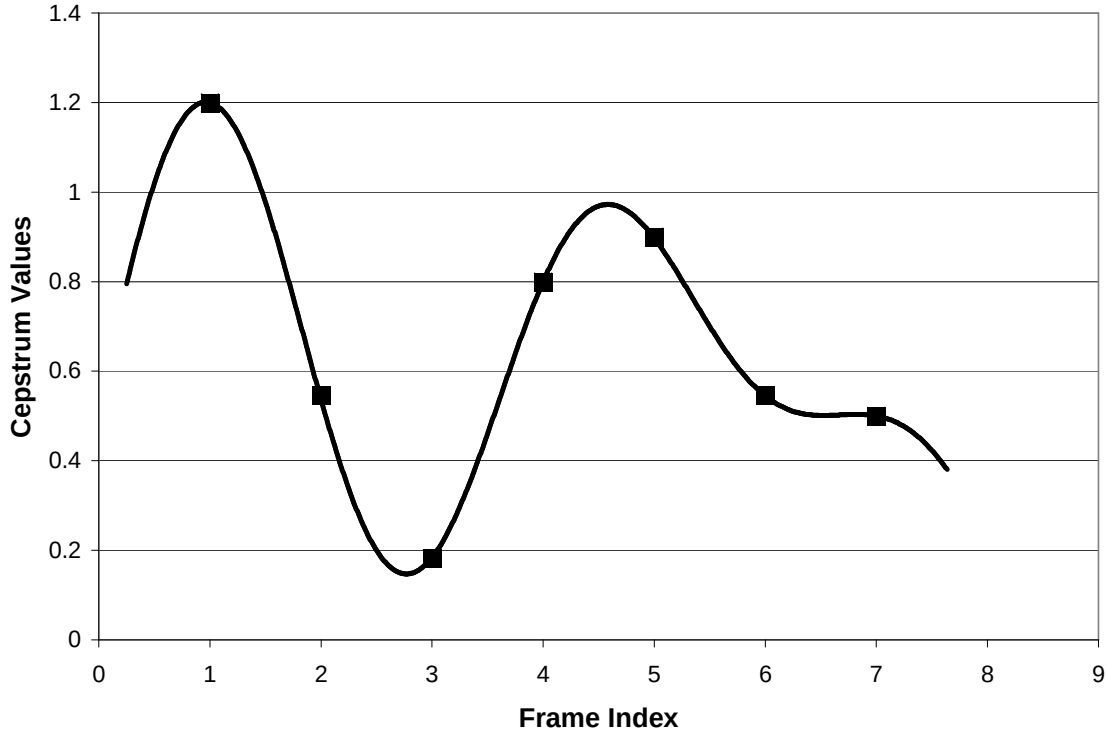

Figure 9: The sinc interpolation of the five cepstra from Figure 8 is sampled at seven equally spaced intervals.

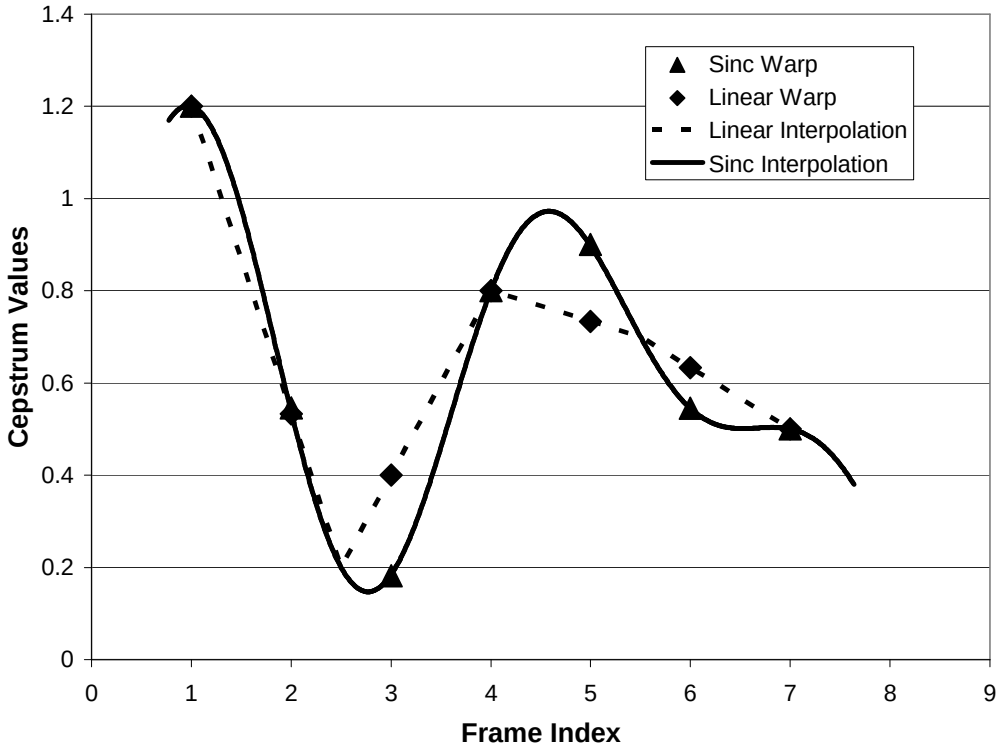

Figure 10: A comparison of the two warping methods.

4.5.3 Warping Method: Average

This warping method collapses each phoneme-state frame sequence into a single frame. This is done by component-wise averaging. This method is simple and results in relatively small stacked frames. Figure 11 shows a sequences of frames for the phoneme /aa/ being warped using this method.

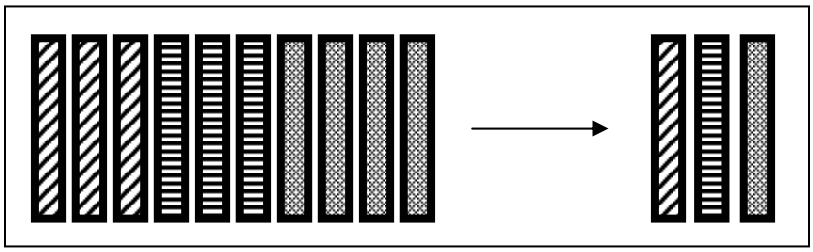

Figure 11: A sequence of frames corresponding to a phoneme, with three states, is transformed according to the average warping method.

4.6 Stacking the frames

The final step in the feature creation process is stacking the warped frame sequence into a single new feature vector. Refer to Figure 12. Each phoneme token in the data results in a single stacked feature vector.

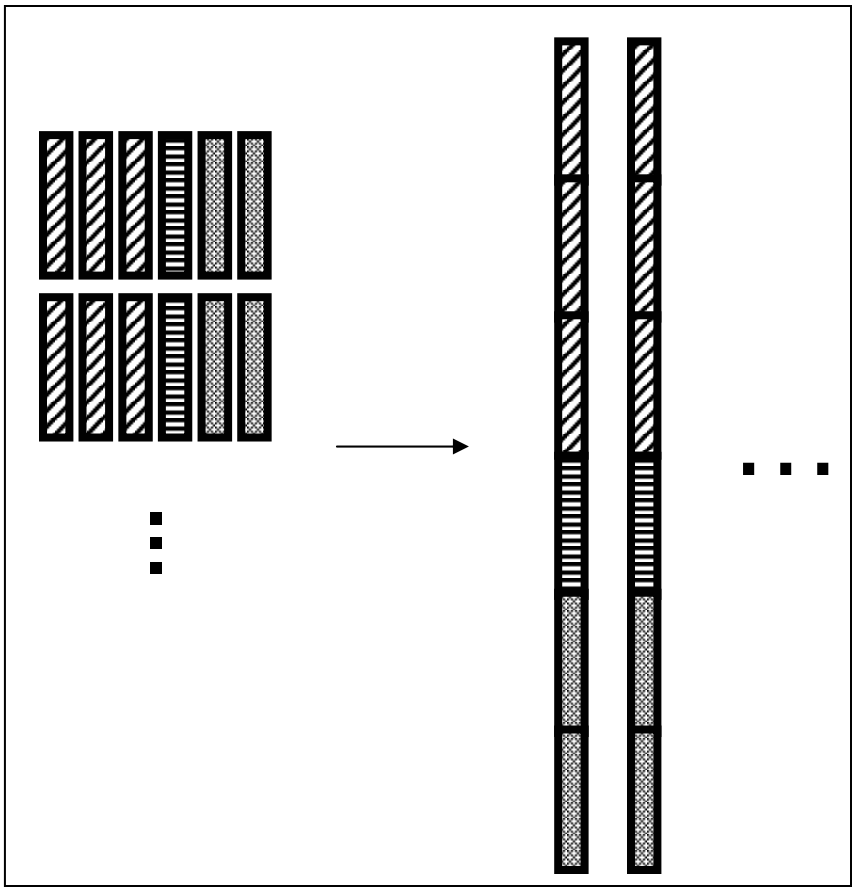

Figure 12: The warped frame sequences on the left are stacked and become the long frames on the right.

5 Model Training

Once this new set of feature vectors has been created, a traditional GMM system is used to train a background and adapt target models. A separate GMM is created for each phoneme and hence there will be separate background models and adapted target models for each phoneme.

5.1 Background Training

In the following experiments, testing on splits 1, 2, and 3 used splits 4, 5, and 6 for background training and vice versa for testing on splits 4, 5, and 6. The Gaussians were trained with diagonal covariance matrices, and the number of Gaussians in each background model varied across experiments. This will be explored later in the report. One of the major concerns with this system is whether the data would be sufficient to train the background models.

Table 1 shows the dimension of the feature vectors for the various phonemes, the amount of available training data for each phoneme, and a comparison to the amount of training

data for a typical GMM. For a typical GMM UBM, 3 hours of training data is usually sufficient, but commonly much more may be used. At 10ms per frame, this works out to 1,080,000 training points. The columns, "% Compared to a Typical GMM," refer to the percentage of the 1,080,000 points that is available to train the background model for a particular phoneme in the Sequential GMM. The percentages vary, with many of them being under 10%.

It is important to note confounding factors for this comparison. A typical UBM has around 2000 Gaussians, while the Sequential GMM system typically used 128 or 256 Gaussians for each phoneme model. On the other hand, typical GMM systems use feature vectors of dimension 40, while this system's feature vectors are typically four or five times this size. For diagonal covariance Gaussians, the number of training parameters is proportional to the dimension of the feature space and the number of Gaussians. Therefore, the GMM for a single phoneme in the Sequential GMM has about half as many parameters as a typical GMM. Many of the GMMs in this system are using significantly less training data than typical; however, after taking into account the number of parameters to train, the situation is not quite as bad as the table implies.

		Background: Splits 1, 2, 3			Background: Splits 4, 5, 6		
Phoneme	Dimension of Feature	Number of Stacked Feature Vectors	% of Total	% Compared to a Typical GMM	Number of Stacked Feature Vectors	% of Total	% Compared to a Typical GMM
aa	220	64,377	1.4%	6.0%	65,003	1.4%	6.0%
ae	260	149,295	3.1%	13.8%	146,301	3.1%	13.5%
ah	160	114,966	2.4%	10.6%	110,182	2.4%	10.2%
ao	240	55,557	1.2%	5.1%	55,502	1.2%	5.1%
aw	300	30,223	0.6%	2.8%	29,763	0.6%	2.8%
ax	80	483,115	10.2%	44.7%	474,827	10.1%	44.0%
ay	260	134,222	2.8%	12.4%	132,695	2.8%	12.3%
b	140	82,869	1.7%	7.7%	81,252	1.7%	7.5%
bgn	400	14,992	0.3%	1.4%	15,792	0.3%	1.5%
ch	240	17,470	0.4%	1.6%	17,750	0.4%	1.6%
d	120	159,847	3.4%	14.8%	156,301	3.3%	14.5%
dh	100	153,843	3.2%	14.2%	149,824	3.2%	13.9%
dx	120	51,453	1.1%	4.8%	51,090	1.1%	4.7%
eh	160	120,246	2.5%	11.1%	119,898	2.6%	11.1%
er	200	92,945	2.0%	8.6%	91,165	1.9%	8.4%
ey	240	73,710	1.6%	6.8%	73,604	1.6%	6.8%
f	200	63,411	1.3%	5.9%	62,814	1.3%	5.8%
fpn	340	13,551	0.3%	1.3%	10,792	0.2%	1.0%
fpv	560	37,334	0.8%	3.5%	33,361	0.7%	3.1%
g	160	51,192	1.1%	4.7%	51,055	1.1%	4.7%
hh	240	87,615	1.8%	8.1%	88,213	1.9%	8.2%
ih	160	150,649	3.2%	13.9%	145,652	3.1%	13.5%
iy	200	164,547	3.5%	15.2%	163,507	3.5%	15.1%
jh	180	25,171	0.5%	2.3%	24,233	0.5%	2.2%

k	180	140,105	3.0%	13.0%	138,953	3.0%	12.9%
l	180	166,644	3.5%	15.4%	165,330	3.5%	15.3%
lau	400	68,078	1.4%	6.3%	67,239	1.4%	6.2%
m	140	121,127	2.6%	11.2%	120,409	2.6%	11.1%
mtn	400	33,211	0.7%	3.1%	31,567	0.7%	2.9%
n	140	318,296	6.7%	29.5%	314,812	6.7%	29.1%
ng	180	53,701	1.1%	5.0%	52,383	1.1%	4.9%
ow	280	99,080	2.1%	9.2%	98,122	2.1%	9.1%
oy	320	3,446	0.1%	0.3%	3,665	0.1%	0.3%
p	160	72,219	1.5%	6.7%	71,703	1.5%	6.6%
pum2	660	20,461	0.4%	1.9%	23,191	0.5%	2.1%
r	180	172,788	3.6%	16.0%	173,001	3.7%	16.0%
s	200	201,085	4.2%	18.6%	196,800	4.2%	18.2%
sh	240	22,927	0.5%	2.1%	23,295	0.5%	2.2%
t	120	326,383	6.9%	30.2%	319,022	6.8%	29.5%
th	160	37,769	0.8%	3.5%	36,325	0.8%	3.4%
uh	120	26,872	0.6%	2.5%	26,746	0.6%	2.5%
uw	180	78,978	1.7%	7.3%	79,127	1.7%	7.3%
v	120	74,914	1.6%	6.9%	73,235	1.6%	6.8%
w	200	117,304	2.5%	10.9%	114,903	2.5%	10.6%
y	180	88,142	1.9%	8.2%	88,246	1.9%	8.2%
z	180	109,893	2.3%	10.2%	108,625	2.3%	10.1%
zh	180	2,060	0.0%	0.2%	1,922	0.0%	0.2%
Total:		4,748,084	100.0%		4,679,197	100.0%	

Table 1: A table representing the distribution of the data across the phonemes. For each phoneme, the dimension of the stacked feature vector and the number of training instances available for each background model are given. The "% total" column represents the percentage of the new feature vectors that are in a specific phoneme class. Finally, the "% Compared to a Typical GMM" column shows the percentage of frames available to train the phoneme-specific model compared to the number of frames typically used to train a traditional UBM, where a frame for the phoneme-specific background model is a stacked feature vector.

5.2 Target Model Adaptation

Data sparsity is a more serious problem for the adaptation of the phoneme-specific background models to the targets. The amount of data used in training the UBM is only limited by the size of the available corpora. The amount of data available for adapting the target models, however, is defined by the task and is constrained to eight conversation-sides. Thus, when comparing the amount of training data available for this system to that of a typical GMM, model adaptation is much more starved than the background model. For example, at one frame per 10ms, there will be about 120,000 (8 [conversations] x 2.5 [minutes / conversation] x 6000 [frames / minute]) training points available for typical model adaptation; whereas in the Sequential GMM system there are approximately only 300 training points on average over a feature space four or five times the dimension. The number of stacked feature vectors available in the training data was calculated by taking into account that the average phoneme in this corpus is 90ms, and there are 47 phonemes. Moreover, this assumes that the speakers use all phonemes

equally, which we know is false from Table 1. Therefore, some models are being adapted from even fewer data.

6 Scoring & Fusion

The final step is to compute the score as in equation (4) for each phoneme model, using the new feature vectors, and then combine the scores resulting from each of the phonemes. The combination was done with LNKnet [15]. LNKnet was used to train a neural net with seven hidden nodes. The training is done in a round-robin process using the first three splits of the task and the last three splits. When training the neural network on splits one, two, and three the testing is done on splits four, five, and six, then vice versa. Note that the neural network parameters are optimized using the same data used in constructing the background models, which is not ideal.

7 Experiments

The experiments on the system include exploring the different frame-sequence warping methods and the number of Gaussians components used in the GMMs, and combining the system with the state-of-the-art GMM system, among others. Results presented were run on the entire 2001 Extended Data task. The performance is measured in terms of the EER and min DCF [3], and the Detection Error Tradeoff (DET) curve [16] is provided in some key experiments. The EER is defined as the error rate where the probability of false alarm is equal to the probability of missed detection. The min DCF is the minimum value of the decision cost function, as defined by NIST:

$$\mathrm{DCF} = C_{\mathrm{MISS}} \times P_{\mathrm{MISS|MATCH}} \times P_{\mathrm{MATCH}} + C_{\mathrm{FALSE\ ALARM}} \times P_{\mathrm{FALSE\ ALARM|IMPOSTOR}} \times P_{\mathrm{IMPOSTOR}},$$

where the costs are defined by $C_{\mathrm{MISS}} = 10$ and $C_{\mathrm{FALSE\ ALARM}} = 1$, and the probability of an impostor is 0.99.

7.1 Baseline System

As a matter of comparison, a generic GMM without z-norm [6], t-norm [17], h-norm [6], or feature mapping [18] is run on the task, since these features have not yet been introduced into the Sequential GMM system. Its models consist of 2048 Gaussian components, and its performance is shown in Table 2. As a matter of notation, this system will be referred to as "GMM – 1."

In addition, a state-of-the-art GMM system will be used in experiments to demonstrate that the Sequential GMM can improve performance through combination. The state-of-the-art GMM system, which will be referred to as "GMM – 2," benefited from t-norm followed by h-norm. The importance of these normalizations is underlined by a comparison of Tables 2 and 3.

GMM – 1	
min DCF	0.0220
EER	4.88%

Table 2: Performance of the baseline GMM, without normalizations.

GMM – 2	
min DCF	0.00509
EER	0.90%

Table 3: Performance of the state-of-the-art GMM.

7.2 Sequential GMM

The experiments detailed in this section test the performance of the Sequential GMM system described above for the different frame-warping methods and various numbers of Gaussian components.

Figure 13 compares the three different warping methods for the case where 128 Gaussian components are used. The most interesting result is that all three warping methods score so similarly, especially in the EER region. However, the linear interpolation-based warping method prevails in the low false-alarm region. Finally, it is clear that the system is vastly superior to the GMM system without any normalization. The metrics of interest for this system are reported in Table 4. The poor performance in the high false-alarm region is believed to be a result of LNKnet optimizing for DCF, and this phenomena is discussed in Section 8.

The next experiment is similar to the one above, except 256 Gaussians components are used. The DET curves are shown in Figure 14. Again, the different warping methods' performances are comparable. However, the sinc interpolation-based warping faired slightly better than the rest except in the high false-alarm region. In addition, examination of Table 4 reveals that using 256 Gaussian components tends to be superior to 128. The sinc interpolation-based warping with 256 Gaussian components resulted in the best EER for the system: 1.14%. The DCF for the same configuration was .0057. Both of these scores are close to the marks reached by the state-of-the-art GMM.

Figures 15 and 16 compare the DET curves for 128, 256, and 512 Gaussian components for the average and sinc interpolation warping methods, respectively. For the average warping method, which has smaller feature vectors, the system performs comparably for each number of Gaussians; however, for the sinc interpolation warping method the best performance is with 256 Gaussian components, particularly in the low false-alarm region.

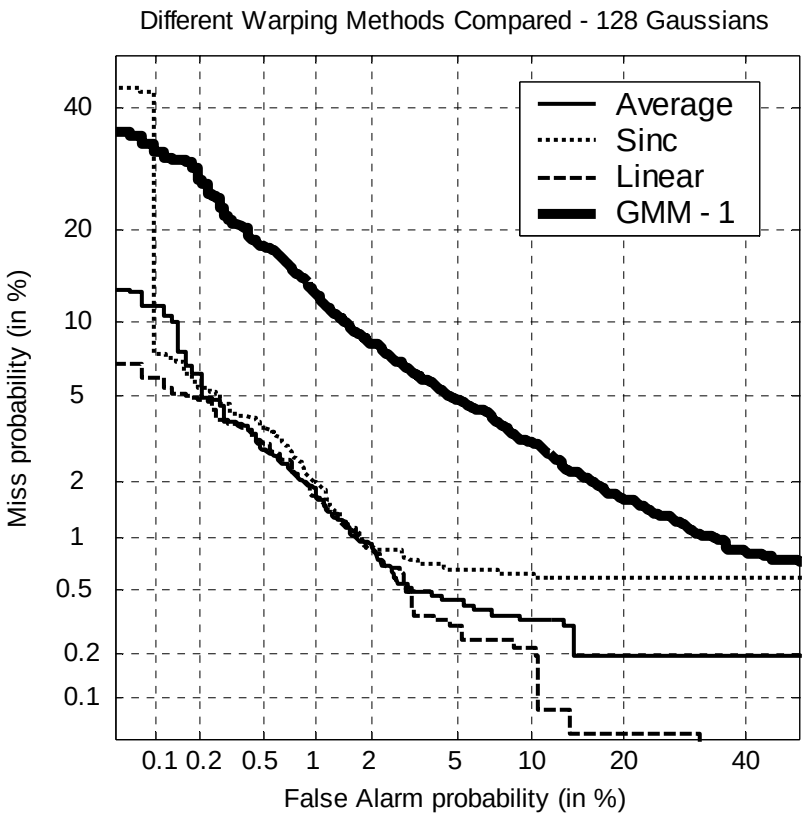

Figure 13: DET curves for different warping methods, using 128 Gaussian components, versus the baseline GMM.

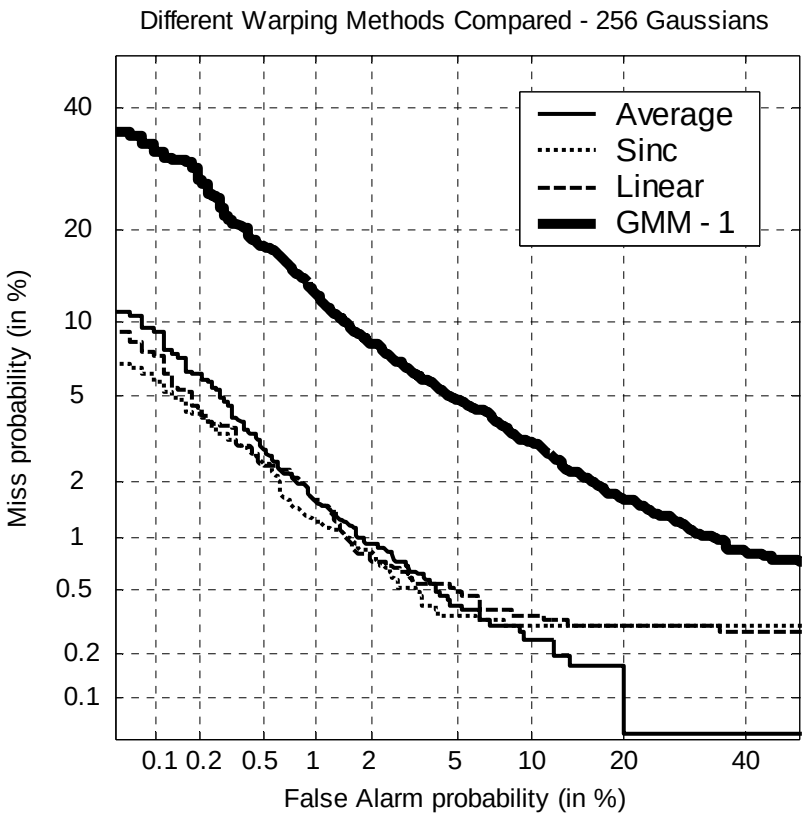

Figure 14: DET curves for different warping methods, using 256 Gaussian components, versus the baseline GMM.

		Sinc Interpolation	Linear Interpolation	Average
128 Gaussian Components	EER min DCF	1.34% 0.00728	1.33% 0.00644	1.28% 0.00681
256 Gaussian Components	EER min DCF	1.14% 0.00575	1.25% 0.00605	1.33% 0.00715
		GMM – 1	GMM – 2	
2048 Gaussian Components	EER min DCF	4.88% 0.0220	0.90% 0.00509	

Table 4: System performance for various configurations.

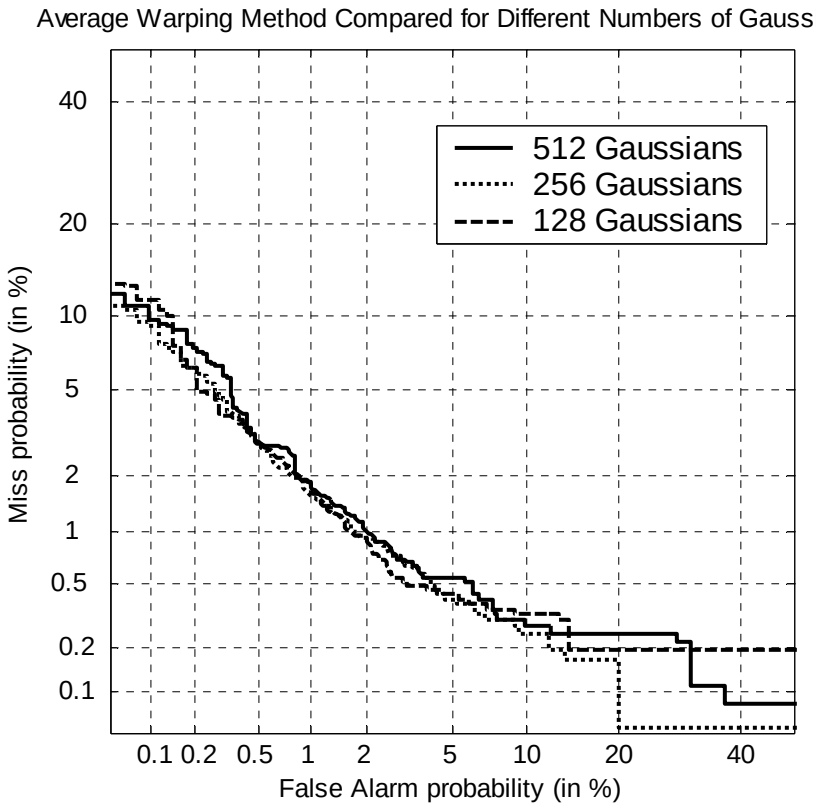

Figure 15: DET curves for various numbers of Gaussian components for the average warping method.

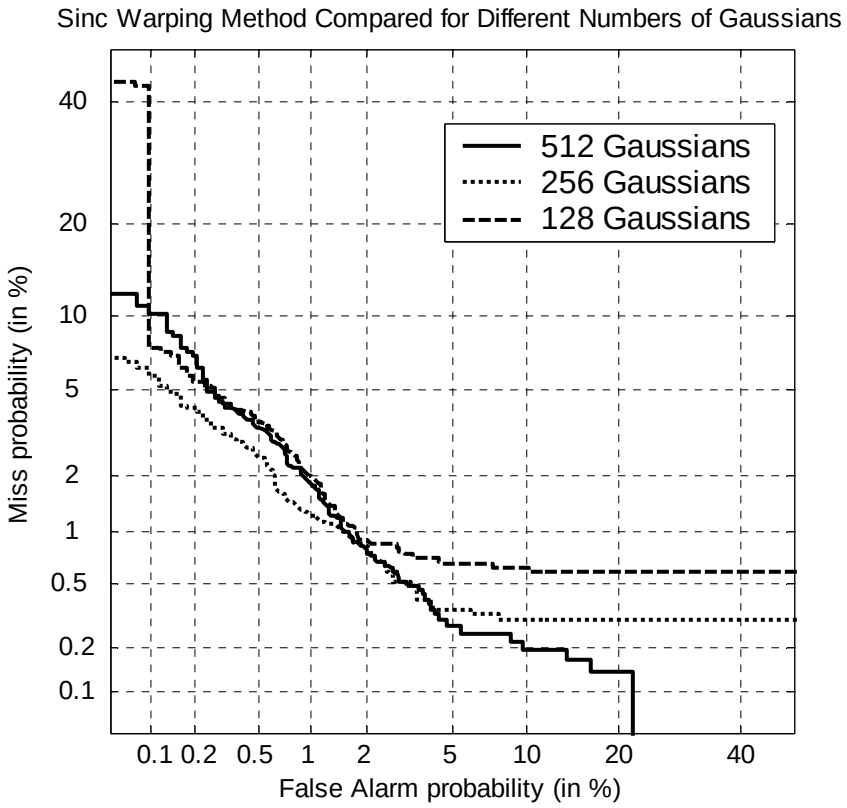

Figure 16: DET curves for various numbers of Gaussian components for the sinc interpolation warping method.

7.3 Combination with the State-of-the-Art GMM

In this experiment, the systems from the previous section are combined with the state-of-the-art GMM (GMM – 2). Combination is done at the score level using LNKnet. The GMM – 2 scores are treated as an additional phoneme, i.e. the input layer of the neural net now has 48 nodes instead of 47, one for each phoneme and one for GMM – 2.

The ability of a system to combine with the state-of-the-art system is crucial. For example, systems based on idiolect [19] traditionally have rather poor performance in isolation; however, they are very valuable in combination with the GMM. The objective of the Sequential GMM system is not to outclass the state-of-the-art system, but rather to capture speaker characterizing information that the regular GMM disregards, specifically sequential information. That is exactly what occurred. As was stated in the introduction, combing the two systems caused the EER to fall by 36% and the min DCF to fall by 65%, relative to the GMM alone.

Figure 17 and Table 5 show the results from combining the various warping methods, using 256 Gaussian components, with GMM – 2. The DET curves show the significant improvement over the state-of-the-art GMM, particularly in the important low false-alarm region. In addition, it is clear that the average warping method system does not combine quite as well as the others. Fortunately, this is what would have been expected;

since the average method captures the least amount of sequential information because it collapses many frames into one, it is the most similar to the regular GMM.

256 Gaussian Components	Sinc Interpolation + GMM – 2	Linear Interpolation + GMM – 2	Average + GMM – 2	GMM – 2
EER min DCF	0.57% 0.00180	0.65% 0.00224	0.79% 0.00289	0.90% 0.00509

Table 5: System performance for various configurations.

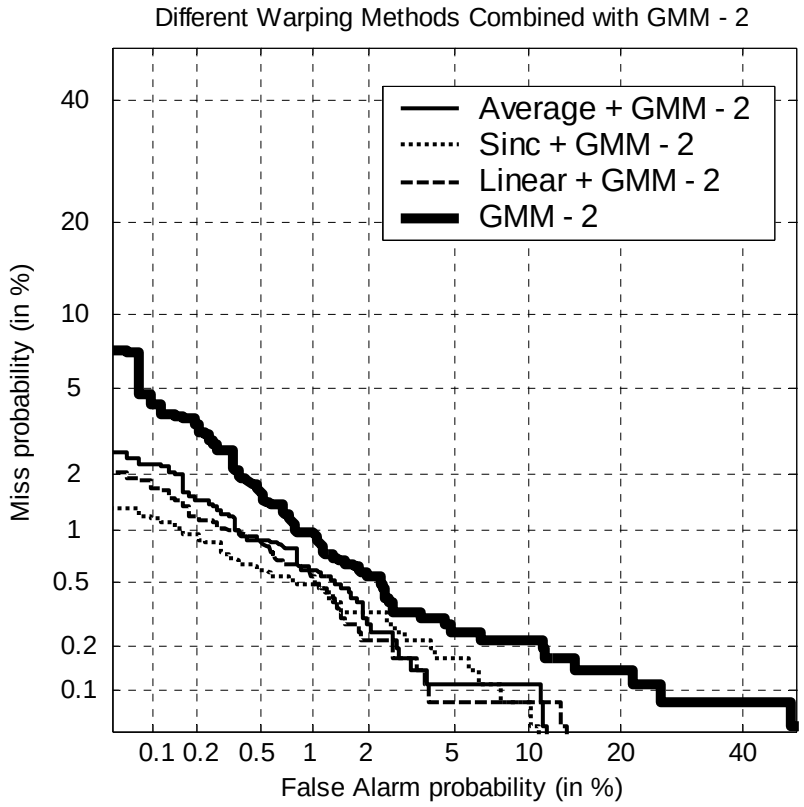

Figure 17: DET curves for the Sequential GMM systems combined with the state-of-the-art GMM system. The Sequential GMMs used 256 Gaussian components.

7.4 Limiting the Length of the Phoneme States

In this experiment, the method used to calculate state lengths, described in Section 4.4, is slightly modified. The maximum length for a phoneme-state is capped at three, i.e. the phoneme-state length is the minimum of three and the previously determined value. The motivation behind the experiment is to see if limiting the size of the feature vectors will improve system performance. The result, as seen in Table 6, is a slight degradation in performance. This implies that using the longer phoneme-state lengths did not particularly harm training the models, and that further compressing the frame sequences tends to remove valuable speaker identifying characteristics.

256 Gaussian Components	Linear Interpolation	Linear Interpolation – Capped Phoneme States
EER min DCF	1.25% 0.00605	1.38% 0.006730

Table 6: System performance with and without capped phoneme-state lengths.

7.5 Principal Component Analysis (PCA)

In order to alleviate potential data sparsity problems, PCA is used to reduce the dimension of the feature space, using the linear interpolation feature set as a typical system. The PCA is implemented on each phoneme individually, and the transformation is trained on all instances of a phoneme type in the corpus. As with calculating the phoneme-state average lengths, this transformation is technically cheating because the system is supposed to compute speaker hypotheses on-line, without additional knowledge of the other trials in the task. Although it is uncertain how this reflects itself in the system performance, cheating in this way should undoubtedly only help performance; thus this method gives us an upper bound on the potential of PCA. Table 7 shows how much information, in terms of energy, is preserved by transforming the feature spaces to retain only the 50 (or 100) largest eigendirections. On average across the phonemes, 88% and 96% of the energy is preserved for compression to 50 and 100 features, respectively. For the majority of the phonemes, the amount of retained energy is rather high, considering the dramatic decrease in the feature space. The least compressible phoneme is /ch/.

Unfortunately, PCA results in a degradation of performance. In Figure 18 the DET curves for before and after PCA can be compared, for 256 Gaussian component models. When using 128 Gaussian components, the PCA feature set's performance is closer to the baseline; however, it is still inferior. It is important to note that it is something of a mystery what information is discarded in the transformation. It is certainly possible that the few percent of information that has been abandoned has most of the discriminating power. An alternative approach would be to use Linear Discriminant Analysis (LDA), which explicitly preserves the directions with the most discriminating power.

PCA Compression			
Phoneme	Dimension of Feature Vector	50 Largest Eigenvalues	100 Largest Eigenvalues
aa	220	90.7%	97.0%
ae	260	90.2%	96.3%
ah	160	92.1%	98.1%
ao	240	89.5%	96.3%
aw	300	89.0%	95.7%

ax	80	96.4%	100.0%
ay	260	88.3%	95.6%
b	140	88.8%	97.9%
bgn	400	85.6%	92.6%
ch	240	76.7%	89.8%
d	120	91.5%	99.2%
dh	100	93.5%	100.0%
dx	120	94.7%	99.5%
eh	160	93.3%	98.5%
er	200	90.6%	97.1%
ey	240	89.4%	96.3%
f	200	80.6%	92.7%
fpn	340	85.0%	92.7%
fpv	560	84.3%	91.4%
g	160	86.8%	96.6%
hh	240	90.2%	96.1%
ih	160	92.6%	98.2%
iy	200	91.5%	97.4%
jh	180	83.1%	94.4%
k	180	83.1%	94.5%
l	180	90.8%	97.4%
lau	400	80.4%	90.1%
m	140	91.3%	98.5%
mtn	400	80.5%	89.4%
n	140	93.2%	98.8%
ng	180	91.8%	97.7%
ow	280	88.7%	95.5%
oy	320	85.3%	94.1%
p	160	82.4%	95.2%
pum2	660	83.5%	90.5%
r	180	92.2%	97.8%
s	200	81.8%	93.2%
sh	240	79.3%	90.9%
t	120	90.1%	99.1%
th	160	83.6%	95.6%
uh	120	93.7%	99.4%
uw	180	92.6%	97.9%
v	120	91.8%	99.3%
w	200	88.5%	96.3%
y	180	89.6%	96.9%
z	180	86.2%	95.6%
zh	180	86.4%	95.6%

Table 7: This table shows the amount of energy contained in the 50 and 100 largest eigenvectors.

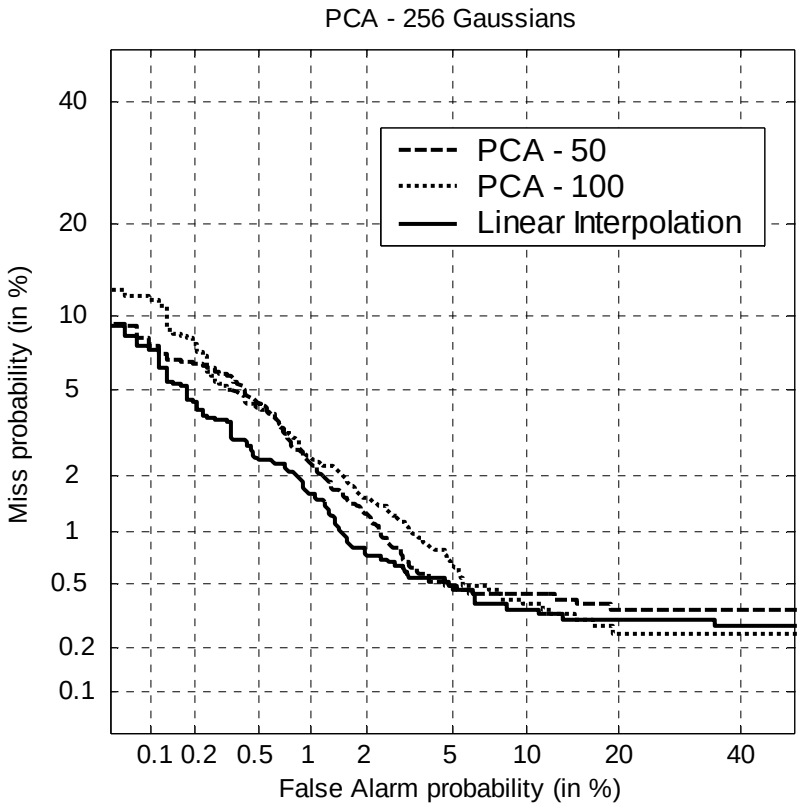

Figure 18: DET curves displaying the result of PCA. The baseline uses linear interpolation warping and 256 Gaussian components. The PCA is performed on the same feature set.

7.6 Adapting from a UBM

Rather than directly building phoneme-specific background models, a UBM is created. The UBM models all of the phonemes, and the phoneme-specific background models are then adapted from the UBM. See Figure 19. From that point onward, the system is the same as before. This staged approach was inspired by phoneme-model experiments of [20]. The adaptation of the UBM to phoneme-specific background models uses the same features that are used to create phoneme-specific background models directly in the single-stage approach. In all cases, the adaptation was only done on the means of the Gaussians.

The primary advantage of adapting the models in this method is that the initial UBM can take advantage of all the available data. Thus, the phoneme-specific UBMs should be able to capitalize on the fact they are adapting from a well-developed background model and not from scratch. One limitation of this method is that it requires that the feature vectors for all the phonemes be the same dimension. For this reason, the average warping method was used in this experiment. This is unfortunate because the other warping methods, which have larger dimension vectors, are more likely to benefit from this technique.

Figure 20 displays the DET curves comparing the two adaptation schemes and the baseline GMM. Both the single-stage and the two-stage adaptation methods are on the feature set created with the average warping style, and they both use 512 Gaussians components. The single-stage adaptation significantly outperforms the two-stage adaptation. However, the reason for this may be that the UBM should have been trained with more Gaussian components.

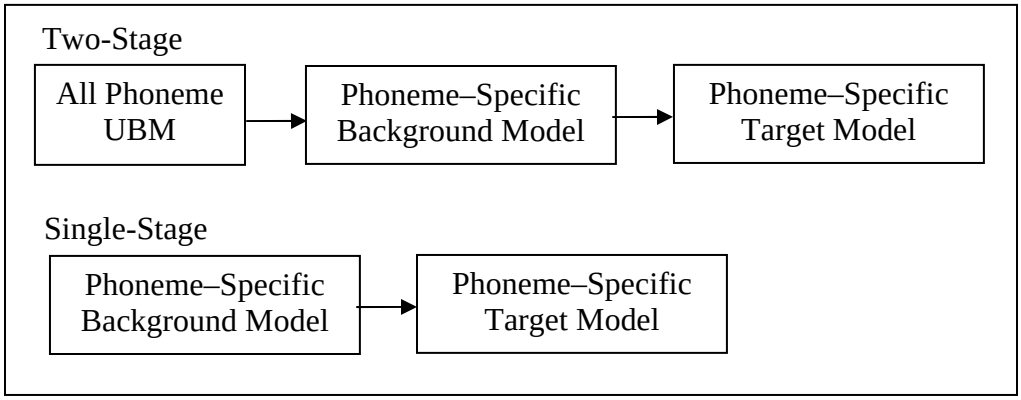

Figure 19: Two different GMM adaptation approaches [20].

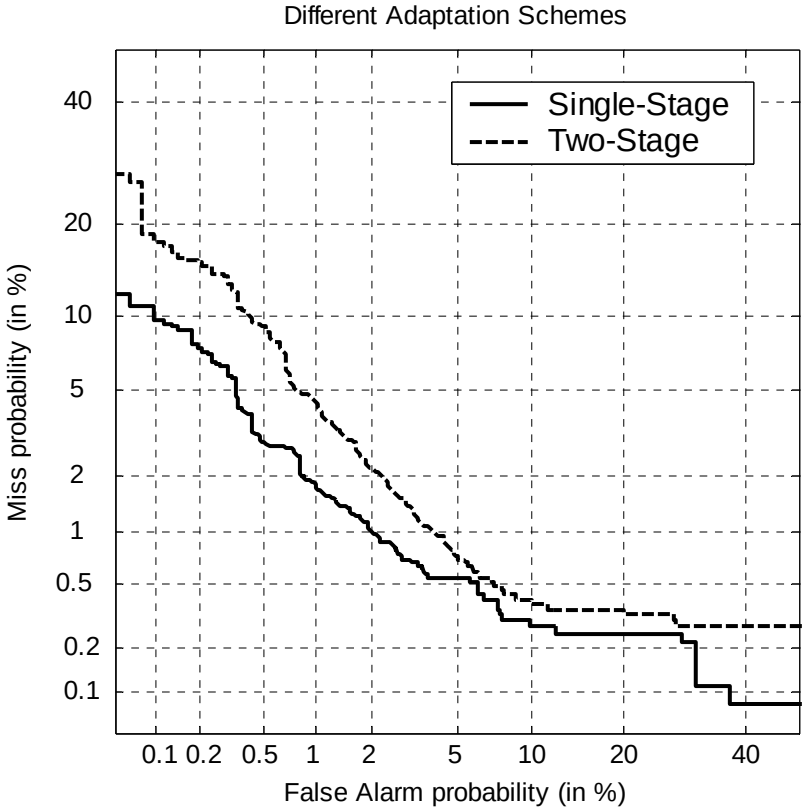

Figure 20: DET curves for the different adaptation methods.

7.7 Comparison to other Sequential Systems

In this section, the Sequential GMM system is compared to two other systems, both of which capture phoneme-length sequential information: one that is non-parametric, and another that models the sequences with HMMs.

7.7.1 Comparison to a Non-Parametric System

The first system, referred to as the Sequential Non-Parametric (SNP) system, compares phoneme-length frame sequences in the test and target data. As its name implies, the comparison is done non-parametrically [21]. In the SNP system, each trial of a test segment against a putative target model consists of scoring each frame sequence in the test data, corresponding to a phoneme, versus every instance of that phoneme in the training data. The frame sequences are scored by aligning the frames using a Dynamic Time-Warping (DTW) algorithm and then taking a Euclidian distance between aligned frames. This is a k^{th} nearest neighbor technique where only the distance to the closest training token is stored for each test token. The particular SNP system configuration used in this combination is exactly as described in [21] for phoneme-unigrams. Note that phoneme-unigrams are not the optimal token size for the SNP system: phoneme-trigrams performed better. This offers hope that the performance for the Sequential GMM system could be improved by switching to a longer token unit.

These systems make an interesting comparison because they are both based on the same phoneme-level frame sequences. There are two major differences between the systems. First, the SNP system uses no parametric models; and second, the frame sequences are warped differently. In the SNP system each pair of frame sequences are optimally warped for each other with DTW, whereas in the Sequential GMM system all the frame sequences are warped at the beginning to a common length.

Figure 21 shows the DET curves and Table 8 displays the EERs and DCFs for the SNP system, the Sequential GMM system, and a combination of the two using LNKnet. The Sequential GMM system performs better than the SNP, but the interesting fact is that combining the two systems results in very little improvement in system performance. This indicates that the speaker discriminating power of the SNP system is largely subsumed by the Sequential GMM system. Moreover, Figure 22 shows that the Sequential GMM system combines better with the state-of-the-art GMM system.

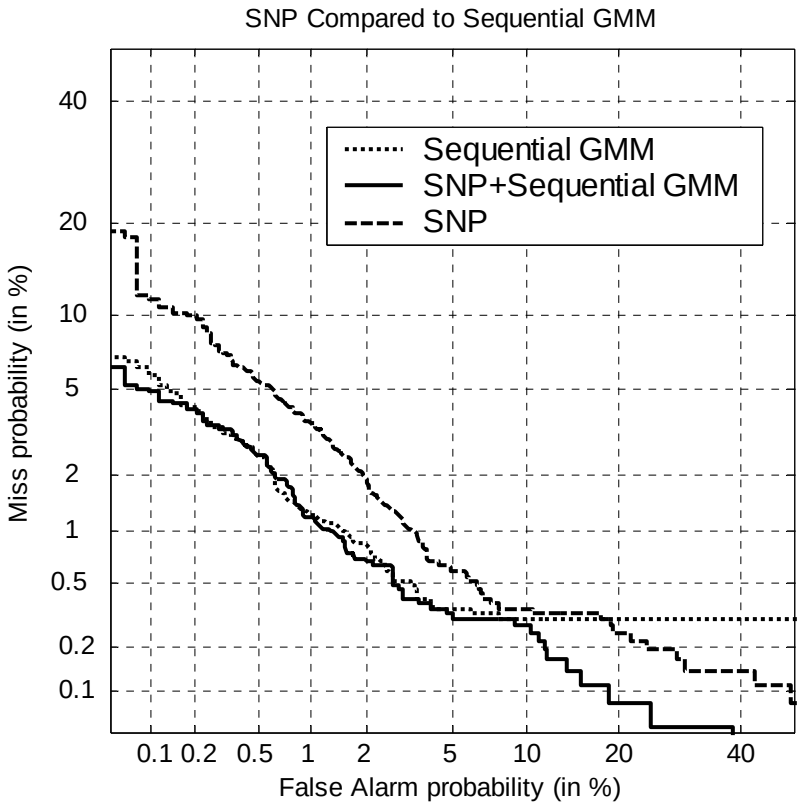

Figure 21: DET curves for the SNP, the Sequential GMM system, and a combination of the two. The Sequential GMM uses 256 Gaussian components and the sinc interpolation warping method.

	Sequential GMM	SNP	Sequential GMM + GMM – 2	SNP + GMM – 2	GMM – 2
EER min DCF	1.14% 0.00575	1.85% 0.00937	0.57% 0.00180	0.68% 0.00264	0.90% 0.00509

Table 8: EERs and DCFs for various systems. The Sequential GMM uses 256 Gaussian components and the sinc interpolation warping method.

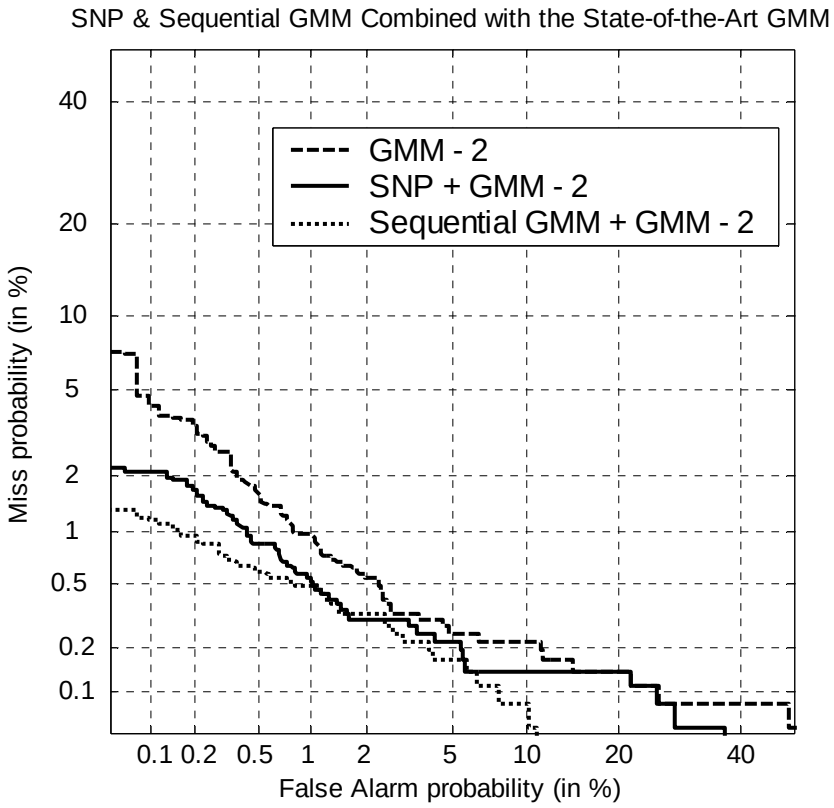

Figure 22: DET curves for state-of-the-art GMM and its combination with the SNP system and the Sequential GMM system. The Sequential GMM uses 256 Gaussian components and the sinc interpolation warping method.

7.7.2 Comparison to a Hidden Markov Model (HMM) based System

The HMM system takes the phoneme frame sequences and models them with HMMs. The phoneme-level HMM system is comparable to the system described in [22]. This system produced speaker hypotheses using HMM models on a handful of specially chosen words whereas the phoneme-level HMM system models phonemes. There are HMM models for 43 phonemes; each model has three states and 128 Gaussians components. This works out to 384 Gaussians per model, slightly more than, but comparable to, the Sequential GMM system. Training consisted of Baum-Welch re-estimation, and successive splitting of the Gaussians, starting from one Gaussian per state.

The Sequential GMM system and the phoneme-level HMM systems are analogous, save that different probabilistic models were used. The Sequential GMM system explicitly models all of the (warped) frames in the phoneme, i.e. the complete trajectory of the sequence of frames. The HMM model is "looser" and only models the frames in reference to their being in one of the three ordered states; i.e. the frames are split into three ordered groups. Modeling with HMMs has the advantage that it does not require any frame warping.

In terms of performance, the Sequential GMM has a vastly superior EER, while the HMM system has a modestly better DCF. See Table 9. Note that the performance of the system using the whole word models [22] is superior to that of the phoneme-level HMM system.

	Sequential GMM	HMM
EER	0.66%	1.16%
min DCF	0.00537	0.00460

Table 9: EERs and DCFs for the two systems. All scores are reported on a single split of switchboard one, consisting of 1624 trials because the HMM system's results are only available for this split.

8 Discussion & Future Work

Since this system is still in a preliminary stage, there are still many avenues that could potentially lead to performance improvements. Some of these ideas and other observations are discussed below.

8.1 Phoneme Performance

Tables 11, 12, 13, and 14, presented in Appendix A, show the EERs and min DCFs for each phoneme in most of the systems discussed above. One easily identifiable fact from these tables is that each phoneme's performance is relatively similar for each system. One thing not so readily identifiable, but intuitive, is that the EER is strongly correlated with how frequent the phoneme is. See Figure 23. In general, the more frequent a phoneme, the lower the EER is. This graph helps motivate interest in reducing the dimension of the models and moving to a larger corpus. Also, note that phonemes such as /zh/ and /oy/ have such poor EERs because they don't even appear in many test utterances.

One other aspect investigated is using subsets of the phonemes [20, 23]. These experiments illustrate that every phoneme is contributing to the score since removing any one phoneme results in a degradation in performance. Moreover, contrary to the results in [23], no performance gain is realized from using any subset of phonemes tried. Figure 24 shows the result for two different possible subsets: using the six best phonemes and using the forty best phonemes. The "best phonemes" are measured in terms of EER in isolation, according to Table 11. In both cases, the EER and min DCF of the systems based on the subsets of phonemes scored worse than the system that used all of the phonemes.

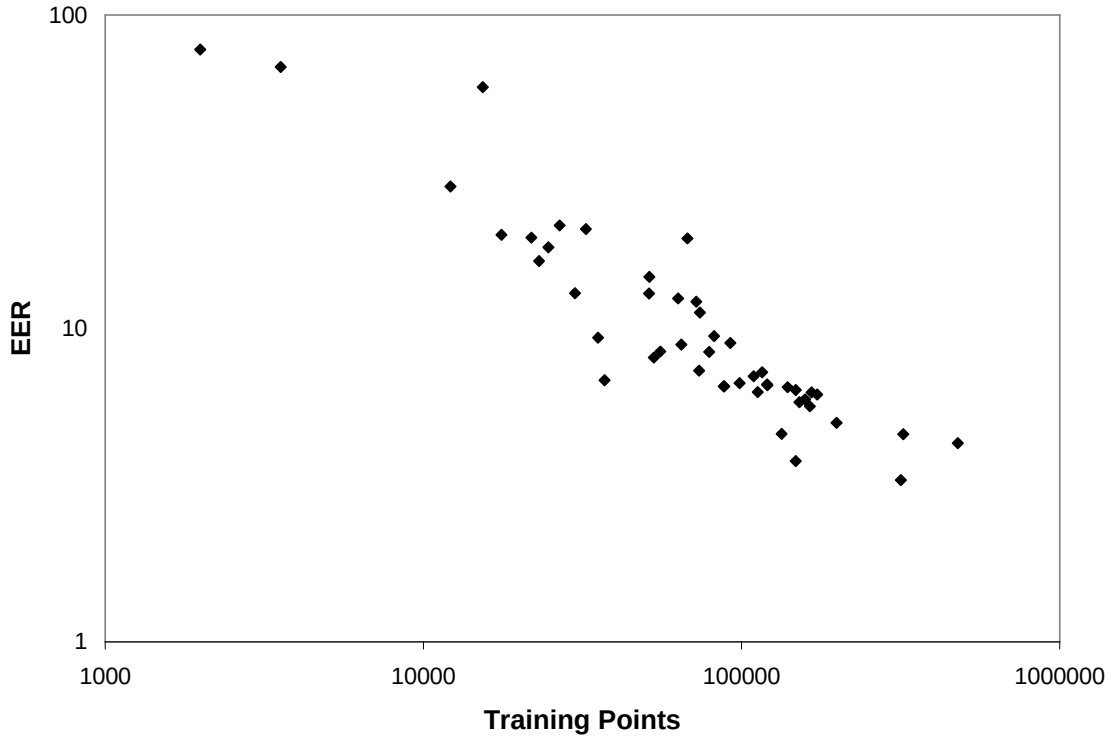

Figure 23: A log-log plot of EER versus the average number of training vectors for the phoneme-specific UBM. The EERs were taken from the sinc interpolation warping method with 256 Gaussian components given in Table 11.

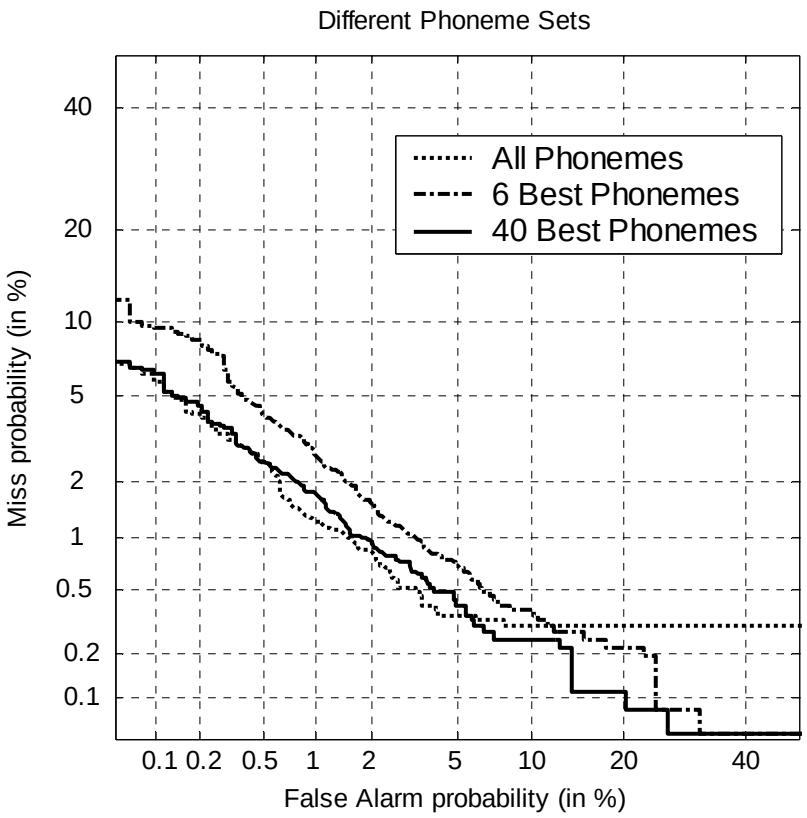

Figure 24: DET curves for various sets of phonemes, using the sinc interpolation warping method and 256 Gaussian components.

8.2 ASR

The ASR step in the system is suboptimal for a number of reasons. First, because switchboard one is an older task, the ASR system used is several years old and therefore not as good as current ASR systems. Moreover, due to differences in frame indexing conventions between SRI, who provided the ASR output, and HTK, which was used to do the feature extraction, there are likely slight frame alignment errors. Even small frame alignment differences can be significant, since phoneme states can be as short as a single frame. In newer systems, this issue was resolved by not chunking the feature files. If these modifications were made it is reasonable to expect an improvement in performance. For example, compare the improvement for a word-based HMM speaker recognition system in Table 10.

In order to assess the effect of ASR errors on system performance, it would be valuable to run the system using forced alignments based on truth transcripts. Although the word-level transcriptions do exist for switchboard one, the author did not have access to a phoneme state-level forced alignment based on these truth transcriptions. Nevertheless, the ASR based system provides the most important result because it represents the score that the fully-automatic speaker-recognition system produces. It is important that the system be automatic from beginning to end, and unfortunately truth transcription is a labor-intensive task.

Word-Based HMM System Performance		
	Before Improvements	After Improvements
EER	2.0%	1.7%
DCF	.012	.0092

Table 10: This table shows the change in system performance for the system in [22] that results from using newer ASR recognition and not segmenting the feature files.

8.3 Grouping Phonemes

It would be an interesting experiment to attempt to warp all the phonemes to the same length without simply reducing all of the phoneme-states to a single frame, as presented in section 7.6. Another method to fight data sparsity is to group phonemes together, based on acoustic similarity. Thus the training data would not be divided into as many "bins." The trade-off is between the amount of training data and the tightness of the probability models. That is, the more phonemes that are grouped together, the larger the acoustic space they will cover and hence the more spread out the GMMs will be. For example, the phonemes /ah/ and /aa/ may be close enough that it would be better to group them together for training the GMMs.

8.4 Alternative Approaches to the Average Phoneme-State Lengths

Section 4.4 includes a discussion of the trade-offs between different phoneme-state lengths. Alternative approaches may result in superior performance. It is possible to choose the target phoneme-state length based on how rich the acoustic information within the phoneme-state is. For instance, some of the factors of interest may be the time-evolution of the formants or whether the sound is voiced or unvoiced. Alternatively, one could just choose the longest instance of a phoneme-state as the desired length. As was discussed before, this method prevents discarding any temporal information.

8.5 Other Research Possibilities

Since the system is still in an early stage of development, there are many respects in which it could be improved. Some of these are listed below.

- Tweak the interpolation method. There are a number of methods to interpolate points, ranging from splines to different kernels. It is possible that some other, yet uninvestigated, method could be superior.
- Apply LDA to the feature vectors. The feature vectors are large and have been shown by way of PCA to be highly reducible. Perhaps LDA can better reduce the dimension while retaining the speaker discriminating information.
- Optimize the score combination for the phonemes. LNKnet has numerous parameters, including the type of neural network, the topology of the neural network, and the optimization parameters for training the neural network. Numerous times LNKnet has resulted in patently non-optimal combinations; therefore, it is reasonable to suspect that the combinations for this system are also

suboptimal. See Figure 25 for an example of how changing the neural network configuration affects performance.

- Adapt the system for use in an SVM. This research avenue is more speculative than the others; however, by creating vectors in a method similar to [24], but based on the stacked feature vectors in this system, an SVM can be explored.
- Change the basic unit from phonemes to tri-phonemes, words, etc. Recall that using longer tokens improved performance in the SNP and HMM systems [21, 22]. One major issue is that by increasing the basic unit size, data sparisty becomes an even larger issue.
- Use open loop phone decoding for the ASR. The main interest in the system is parsing the frames into acoustically similar chunks. The phoneme sequence as dictated by the lexicon is of little interest.
- Replace GMMs with other probabilistic models, such as Conditional Random Fields (CRF).
- Investigate the front-end. Perhaps there is an alternative signal processing method that is better suited for this system.
- Add normalization. Z-norm, t-norm, h-norm, or feature mapping could easily be applied to this system, and they have the potential for large performance gains.
- Move to a larger and more challenging corpus, such as switchboard two. A larger corpus would supply ample data for training background models and applying the various normalizations mentioned above.

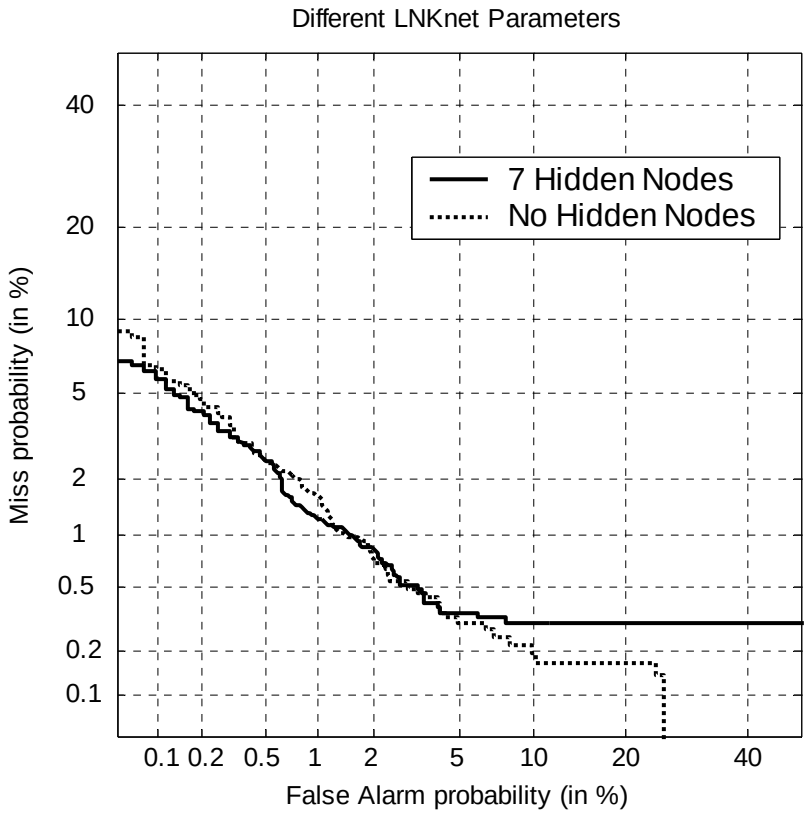

Figure 25: DET curves for the system, with sinc interpolation warping and 256 Gaussians components, using different LNKnet options. One configuration had a hidden layer with seven hidden nodes, and the other configuration had no hidden layers.

9 Conclusion

Since speaker recognition is as of yet an unsolved problem, it is still an open question as to what form the optimal solution will take. However, the optimal systems should use all possible sources of speaker identifying information—something the GMM system does not do. This does not mean the GMM system should be scrapped; it is good at collecting considerable speaker relevant information. The goal should be to build systems that capture the other sources of information, such as prosodic patterns, pronunciation preferences, word usage, grammar, and other speaker idiosyncrasies.

The Sequential GMM is one such system because it is a powerful addition to the traditional GMM system. Despite the undeveloped state of the system, it is amazingly able to cut the min DCF of the traditional GMM by over 50%. The min DCF is the principal metric by which speaker verification systems are measured by NIST and an important operating point for security applications. A primary advantage of this system is that it is composed of off-the-shelf parts: ASR, cepstral-based signal processing, and GMMs. This makes the system simple to implement, especially considering how commonplace GMM systems have become. Moreover, system performance should

improve as ASR does. Finally, since this system is an adaptation of well-known technology, the GMM, much of the research on GMMs can be applied to this system, e.g. feature mapping. By bringing to bear some of these technologies, it is reasonable to expect even further improvements.

10 Acknowledgments

It was once astutely observed that blame needn't sum to 100%; likewise, neither does credit. This report would not exist if it were not for the help of many individuals. There are a number of people, particularly my colleagues at the International Computer Science Institute (ICSI), responsible for the work presented. This project, although the culmination of four months of research, builds upon knowledge gained during the past year at ICSI.

Much of my thanks go to the members of the speaker verification group. Everything I learned about speaker verification I learned from them (that is meant to be complimentary). First, I would like to thank Barbara Peskin. Always knowledgeable—not only was she willing to hire me, she helped guide my research. Moreover, she edited this paper (multiple times). Also, thank you to my advisor, Professor Nelson Morgan. He helped edit this paper as well. I would also like to thank: Dan Gillick & Barry Chen, who showed me the ropes when I was first getting started; Kofi Boakye, who extracted the features used for the system; Andy Hatch, who showed me the LNKnet combination software and lent me his thesis for 5 months (I'll give it back someday); and Nikki Mirghafori, who helped too much to be mentioned in list form. Nikki offered guidance in my research and in turn I bugged Nikki with all sorts of questions. She was always quick to help, even when she was busy. I also must thank George Doddington, who helped guide our entire group. He is extremely bright, and the advice he gave was invariably good. Thanks to SRI for the ASR and to Andreas Stolcke for getting it to me. Finally, I could never have done this project without the support of Sachin Kajarekar. He is chiefly responsible for the GMM code. I appreciate the time he spent adapting his GMM code for this new application. He is also responsible for the so-called "state-of-the-art" GMM scores used as a matter of comparison in the report.

I would also like to thank all of those other miscellaneous people that helped make me the person I am—you know, my family and friends. I sincerely thank my parents for everything. Mom and Dad, I only made it this far because of your patience. Your support has kept me on the path. Theresa, your presence made me feel like I belonged at the University of Maryland. (By the way, I am happy I helped). Finally, thanks to Joe: without the trichotomy this acknowledgment might not have existed.

No acknowledgments section is complete without thanking the sponsor. NSF, thank you for grant IIS-0329258.

11 Appendix A

EERs (%)						
Warping Method	Sinc Interpolation	Sinc Interpolation	Linear Interpolation	Linear Interpolation	Average	Average
Number of Gaussians	256	128	256	128	256	128
aa	8.87	8.49	8.90	8.39	8.06	7.60
ae	3.77	4.23	3.80	4.10	3.72	4.18
ah	6.27	6.49	6.40	6.78	6.43	6.51
ao	8.44	9.09	10.26	9.20	8.71	8.25
aw	12.94	11.94	13.03	11.86	11.23	10.69
ax	4.31	4.48	4.23	4.56	4.18	4.45
ay	4.61	6.27	4.29	4.37	4.31	4.23
b	9.44	8.44	9.72	8.52	8.52	7.73
bgn	58.89	56.23	58.72	57.42	51.13	11.53
ch	19.92	18.37	20.08	18.67	18.45	16.72
d	5.94	5.67	5.86	5.32	5.67	5.62
dh	5.81	5.07	5.94	5.51	5.78	5.37
dx	14.60	13.24	14.60	13.24	13.95	12.56
eh	6.62	6.68	7.03	6.78	6.54	6.68
er	8.98	8.41	8.79	8.36	7.90	5.97
ey	7.33	7.54	7.38	6.97	6.54	6.89
f	12.48	11.99	13.60	12.16	11.53	10.23
fpn	28.36	26.73	28.66	26.92	25.97	24.29
fpv	9.34	8.49	9.47	8.66	9.06	8.36
g	12.92	11.51	13.35	11.72	11.91	10.53
hh	6.54	6.35	6.95	6.21	6.08	5.13
ih	6.35	6.65	6.19	6.27	6.16	6.38
iy	5.64	9.34	5.94	6.13	5.97	5.92
jh	18.15	16.12	18.43	16.55	16.01	14.84
k	6.49	6.11	6.43	6.30	6.35	6.16
l	6.24	6.38	6.40	6.38	6.05	6.16
lau	19.38	5.64	19.62	18.37	18.07	16.99
m	6.59	5.32	5.37	5.45	10.53	5.13
mtn	20.73	19.97	20.95	19.57	17.83	17.10
n	3.28	3.66	3.23	3.66	3.42	3.53
ng	8.06	7.44	8.14	7.68	7.03	6.59
ow	6.68	6.59	6.70	6.40	6.13	6.05
oy	68.25	65.48	68.74	65.70	55.50	52.08
p	12.16	10.28	10.83	10.23	10.31	9.58
pum2	19.51	17.77	19.35	17.86	17.31	15.66
r	6.16	6.27	6.16	6.02	6.13	6.11
s	4.99	68.25	5.07	4.86	4.75	4.97
sh	16.42	12.16	16.53	15.71	14.36	13.46
t	4.59	4.59	4.45	4.72	4.15	4.37
th	6.84	16.26	18.15	16.09	15.6	13.79
uh	21.30	19.65	21.3	19.65	20.46	19.24

uw	8.41	7.79	8.25	7.73	7.60	7.25
v	11.23	10.61	11.34	10.53	10.69	9.61
w	7.25	6.84	7.38	7.22	7.08	6.59
y	6.54	6.68	6.38	6.27	5.97	5.73
z	7.03	6.62	7.38	6.49	6.62	6.49
zh	77.67	78.94	77.96	78.53	73.76	72.37

Table 11: EERs for each phoneme for various systems.

EERs					
Warping Method	Linear Interpolation: PCA – 50 Components	Linear Interpolation: PCA – 100 Components	Linear Interpolation: PCA – 50 Components	Linear Interpolation: PCA – 100 Components	Linear Interpolation: Phoneme-States Capped at 3
Number of Gaussians	256	256	256	256	256
aa	8.82	8.09	11.10	9.63	9.01
ae	3.31	3.53	3.31	3.69	3.88
ah	6.27	6.38	7.84	7.52	6.40
ao	9.93	9.01	12.29	11.70	9.61
aw	11.8	10.72	14.19	12.97	13.00
ax	4.53	4.83	14.19	12.97	4.23
ay	4.26	4.34	4.37	4.53	4.42
b	9.12	8.79	11.78	10.61	9.72
bgn	42.93	40.65	11.78	10.61	55.14
ch	19.57	17.26	21.47	19.27	20.19
d	5.94	5.43	8.01	7.52	5.89
dh	5.75	5.78	6.95	7.00	5.94
dx	14.01	13.54	21.63	20.68	18.40
eh	6.38	6.35	8.39	7.90	7.03
er	8.63	7.84	10.75	9.69	8.93
ey	7.06	7.06	8.25	8.11	7.27
f	11.53	10.64	15.47	13.62	12.97
fpn	24.78	23.31	27.95	25.78	27.14
fpv	8.93	8.47	9.53	9.06	9.34
g	12.81	11.51	15.41	13.27	13.35
hh	6.49	5.70	9.61	9.31	6.81
ih	6.08	6.30	7.41	7.16	6.19
iy	5.81	5.97	6.46	6.35	6.08
jh	16.85	15.39	19.67	18.26	18.43
k	6.43	6.40	7.08	6.87	6.43
l	6.30	6.27	7.71	7.35	6.38
lau	42.66	42.33	20.95	19.84	19.84
m	5.64	5.48	6.32	6.13	5.37
mtn	19.54	17.99	24.04	21.22	20.57
n	3.20	3.34	3.88	3.74	3.23
ng	8.06	7.41	10.96	9.91	8.14
ow	5.75	5.70	7.52	6.46	6.68
oy	40.90	40.57	46.12	43.04	64.29

p	11.07	9.88	12.48	11.86	10.83
pum2	17.88	16.58	20.00	18.48	18.43
r	5.67	6.24	6.81	6.62	6.16
s	4.99	4.78	5.67	5.56	4.72
sh	15.90	14.55	19.35	16.66	15.98
t	4.83	4.86	6.02	5.81	4.45
th	16.20	14.74	20.38	18.05	18.13
Uh	21.98	19.78	25.73	24.40	21.30
Uw	8.36	7.54	10.15	9.93	8.25
V	11.78	10.45	15.39	13.27	11.37
W	7.63	7.33	8.96	8.66	7.30
Y	6.68	6.32	8.68	7.73	6.38
Z	6.92	6.46	8.85	7.95	7.38
Zh	54.55	49.61	59.21	50.66	78.40

Table 12: EERs for each phoneme for various systems.

DCFs						
Warping Method	Sinc Interpolation	Sinc Interpolation	Linear Interpolation	Linear Interpolation	Average	Average
Number of Gaussians	256	128	256	128	256	128
aa	0.0429	0.0435	0.0423	0.0434	0.040	0.0420
ae	0.0220	0.0225	0.0221	0.0228	0.0206	0.0225
ah	0.0339	0.0347	0.0325	0.0342	0.0328	0.0322
ao	0.0399	0.0437	0.0443	0.0446	0.0425	0.0419
aw	0.0534	0.0491	0.0566	0.0540	0.0508	0.0474
ax	0.0222	0.0228	0.0226	0.0231	0.0220	0.0224
ay	0.0242	0.0339	0.0252	0.0259	0.0259	0.0248
b	0.0460	0.0399	0.0432	0.0421	0.0426	0.0399
bgn	0.0976	0.0968	0.0975	0.0974	0.0945	0.0529
ch	0.0748	0.0689	0.0722	0.0689	0.0731	0.0679
d	0.0298	0.0301	0.0295	0.0286	0.0303	0.0290
dh	0.0279	0.0284	0.0275	0.0283	0.0302	0.0294
dx	0.0654	0.0613	0.0654	0.0613	0.0624	0.0589
eh	0.0371	0.0357	0.0337	0.0364	0.0327	0.0344
er	0.0413	0.0417	0.0416	0.0421	0.0427	0.0299
ey	0.0349	0.0352	0.0355	0.0356	0.0313	0.0337
f	0.0549	0.0515	0.0569	0.0546	0.0529	0.0492
fpn	0.0700	0.0675	0.0710	0.0696	0.0690	0.0672
fpv	0.0385	0.0361	0.0377	0.0365	0.0362	0.0377
g	0.0559	0.0544	0.0583	0.0546	0.0540	0.0488
hh	0.0311	0.0299	0.0307	0.0271	0.0288	0.0274
ih	0.0332	0.0337	0.0339	0.0348	0.0308	0.0325
iy	0.0285	0.0385	0.0303	0.0315	0.0299	0.0305
jh	0.0688	0.0673	0.0711	0.0672	0.0670	0.0593
k	0.0305	0.0306	0.0340	0.0328	0.0331	0.0299
l	0.0310	0.0309	0.0309	0.0317	0.0289	0.0308

lau	0.0668	0.0285	0.0704	0.0665	0.0654	0.0660
m	0.0330	0.0285	0.0286	0.0294	0.0488	0.0265
mtn	0.0645	0.0649	0.0638	0.0654	0.0598	0.0595
n	0.0176	0.0191	0.0183	0.0188	0.0185	0.0199
ng	0.0385	0.0342	0.0377	0.0355	0.0373	0.0344
ow	0.0337	0.0330	0.0326	0.0327	0.0318	0.0310
oy	0.0970	0.0970	0.0960	0.0957	0.0969	0.0956
p	0.0506	0.0486	0.0495	0.0471	0.0485	0.0446
pum2	0.0565	0.0538	0.0559	0.0534	0.0546	0.0531
r	0.0324	0.0322	0.0315	0.0336	0.0323	0.0335
s	0.0225	0.0970	0.0249	0.0239	0.0235	0.0243
sh	0.0608	0.0506	0.0571	0.0584	0.0588	0.0546
t	0.0225	0.0228	0.0229	0.0231	0.0222	0.0215
th	0.0384	0.0686	0.0705	0.0664	0.0630	0.0621
uh	0.0783	0.0834	0.0834	0.0783	0.0822	0.0779
uw	0.0404	0.0412	0.0424	0.0413	0.0402	0.0394
v	0.0541	0.0512	0.0526	0.0518	0.0509	0.0510
w	0.0395	0.0384	0.0367	0.0385	0.0375	0.0367
y	0.0316	0.0318	0.0324	0.0301	0.0301	0.0328
z	0.0340	0.0333	0.0336	0.0326	0.0347	0.0331
zh	0.0986	0.0986	0.0987	0.0984	0.0988	0.0980

Table 13: DCFs for each phoneme for various systems.

DCFs					
Warping Method	Linear Interpolation: PCA – 50 Components	Linear Interpolation: PCA – 100 Components	Linear Interpolation: PCA – 50 Components	Linear Interpolation: PCA – 100 Components	Linear Interpolation: Phoneme-States Capped at 3
Number of Gaussians	256	256	256	256	256
aa	0.0411	0.0411	0.0528	0.0452	0.0432
ae	0.0174	0.0188	0.0198	0.0216	0.0226
ah	0.0325	0.0334	0.0409	0.0373	0.0325
ao	0.0456	0.0428	0.0535	0.0515	0.0455
aw	0.0523	0.0495	0.0595	0.0586	0.0573
ax	0.0222	0.0247	0.0595	0.0586	0.0225
ay	0.0229	0.0233	0.0237	0.0230	0.0239
b	0.0458	0.0456	0.0550	0.0519	0.0432
bgn	0.1001	0.1000	0.1150	0.1019	0.0976
ch	0.0766	0.0723	0.0821	0.0769	0.0730
d	0.0314	0.0298	0.0409	0.0375	0.0295
dh	0.0307	0.0295	0.0364	0.0364	0.0275
dx	0.0617	0.0627	0.0887	0.0854	0.0713
eh	0.0346	0.0357	0.0403	0.0408	0.0337
er	0.0401	0.0407	0.0497	0.0474	0.0420
ey	0.0338	0.0336	0.0382	0.0364	0.0342
f	0.0561	0.0540	0.0678	0.0615	0.0573

fpn	0.0721	0.0685	0.0802	0.0757	0.0702
fpv	0.0371	0.0363	0.0393	0.0351	0.0390
g	0.0596	0.0556	0.0649	0.0600	0.0583
hh	0.0301	0.0285	0.0428	0.0427	0.0295
ih	0.0314	0.0323	0.0363	0.0358	0.0339
iy	0.0292	0.0307	0.0339	0.0332	0.0308
jh	0.0709	0.0685	0.0809	0.0748	0.0711
k	0.0339	0.0326	0.0364	0.0336	0.0340
l	0.0308	0.0337	0.0384	0.0385	0.0309
lau	0.0759	0.0749	0.0771	0.0764	0.0679
m	0.0294	0.0305	0.0338	0.0330	0.0286
mtn	0.0679	0.0663	0.0812	0.0760	0.0688
n	0.0187	0.0175	0.0211	0.0221	0.0183
ng	0.0390	0.0356	0.0492	0.0444	0.0377
ow	0.0313	0.0302	0.0350	0.0325	0.0333
oy	0.0988	0.0991	0.1003	0.0998	0.0971
p	0.0511	0.0492	0.0570	0.0538	0.0495
pum2	0.0569	0.0558	0.0614	0.0605	0.0580
r	0.0285	0.0307	0.0349	0.0345	0.0315
s	0.0274	0.0285	0.0331	0.0323	0.0232
sh	0.0654	0.0615	0.0763	0.0709	0.0593
t	0.0268	0.0263	0.0311	0.0315	0.0229
th	0.0661	0.0638	0.0769	0.0731	0.0705
uh	0.0822	0.0783	0.0905	0.0869	0.0833
uw	0.0437	0.0399	0.0497	0.0490	0.0424
v	0.0533	0.0495	0.0695	0.0638	0.0526
w	0.0400	0.0386	0.0442	0.0464	0.0376
y	0.0324	0.0305	0.0406	0.0383	0.0324
z	0.0355	0.0341	0.0436	0.0413	0.0336
zh	0.0999	0.1001	0.1001	0.1001	0.0987

Table 14: DCFs for each phoneme for various systems.

12 References

[1] A. Schmidt-Nielsen and T. H. Crystal. "Speaker Verification by Human Listeners: Experiments Comparing Human and Machine Performance Using the NIST 1998 Speaker Evaluation Data." Digital Signal Processing 10, 249-266, 2000.

[2] D. A. Reynolds, *et al*. "The SuperSID Project: Exploiting High-Level information for High-Accuracy Speaker Recognition." Proc ICASSP '03, Vol. 4, pp. 784-787, 2003.

[3] NIST 2001 Speaker Recognition website: http://www.nist.gov/speech/tests/spk/2001/index.htm.

[4] Joseph Campbell Jr. "Speaker Recognition: A Tutorial." IEEE, Vol. 85, No. 9, September 1997.

[5] D. A. Reynolds. "Speaker Identification and Verification using Gaussian Mixture Speaker Models." Speech Communication, Vol. 17, pp 91-108, August 1995.

[6] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. "Speaker verification using adapted Gaussian mixture models." Digital Signal Processing, 10(1-3), pp.19-41, 2000.

[7] J. L. Gauvain and C. H. Lee. "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains." IEEE Trans. Speech Audio Proc. 2 , 291–298, 1994.

[8] The Linguistic Data Consortium website for SWITCHBOARD-1 Release 2: http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC97S62.

[9] J. Godfrey, *et al*. "SWITCHBOARD: Telephone Speech Corpus for Research and Development." Proc ICASSP '92, San Francisco, March 1992.

[10] S. Furui. "Cepstral Analysis Technique for Automatic Speaker Verification." IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-29, pp. 254–272, 1981.

[11] Hidden Markov Model Toolkit (HTK) website: http://htk.eng.cam.ac.uk/.

[12] B. S. Atal. "Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification." J. Acoust. Soc. Am. 55 (6), 1304-1312, 1974.

[13] Ben Gold and Nelson Morgan. Speech and Audio Signal Processing: Processing and Perception of Speech and Music. New York: John Wiley & Sons, 2000.

[14] A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. R. Rao Gadde, M. Plauche, C. Richey, E. Shriberg, K. Sonmez, F. Weng, and J. Zheng. "The SRI March 2000 Hub-5 Conversational Speech Transcription System." Proc NIST Speech Transcription Workshop, College Park, MD, 2000. Website: http://www.nist.gov/speech/publications/tw00/html/cts80/cts80.htm

[15] LNKnet software is available from MIT Lincoln Laboratory. Website: http://www.ll.mit.edu/IST/lknet/index.html.

[16] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. "The DET Curve in Assessment of Detection Task Performance." Proc Eurospeech '97, Vol. 4, pp. 1895-1898, 1997.

[17] R. Auckenthaler, M. Carey, and H. Llyod-Thomas. "Score Normalization for Text-Independent Speaker Verification Systems." Digital Signal Processing, Vol. 10, pp. 42-54, January 2000.

[18] D. A. Reynolds. "Channel robust speaker verification via feature mapping." Proc ICASSP '03, pp. II53–II56, 2003.

[19] G. Doddington. "Speaker Recognition based on Idiolectal Difference between Speakers." Proc Eurospeech '01, Vol. 4, pp. 2521-2524, 2001.

[20] E. Hansen, R. Slyh, and T. Anderson. "Speaker Recognition using Phoneme-Specific GMMs." Proc Odyssey '04, Toledo, Spain, June 2004.

[21] D. Gillick, S. Stafford, and B. Peskin. "Speaker Detection Without Models." Proc ICASSP '05, Philadelphia, March 2005.

[22] K. Boakye and B. Peskin. "Text-Constrained Speaker Recognition on a Text-Independent Task." Proc Odyssey '04, Toledo, Spain, June 2004.

[23] R. Auckenthaler, E. Parris, and M. Carey. "Improving A GMM Speaker Verification System by Phonetic Weighting." Proc ICASSP '99, Phoenix, March 1999.

[24] V. Wan and W. M. Campbell. "Support Vector Machines for Speaker Verification and Identification." IEEE International Workshop on Neural Networks for Signal Processing, Sydney, Australia, December 2000.