A Text-Independent Speaker Recognition System

Catie Schwartz
Ph.D. Student, Applied Mathematics and Scientific Computing
Department of Mathematics
University of Maryland, College Park
schwa2cs AT math.umd.edu

Dr. Ramani Duraiswami
Associate Professor, Department of Computer Science and
Department of the University of Maryland Institute of Advanced Computer Studies (UMIACS)
University of Maryland, College Park
ramani AT umiacs.umd.edu

Abstract

Speaker recognition is the computational task of validating a person's identity based on their voice. The two phases of a speaker recognition system are the enrollment phase where speech samples from the different speakers are turned into models and the verification phase where a sample of speech is tested to determine if it matches a proposed speaker. In a text-independent system, there are no constraints in the words or phrases used during verification. Numerous approaches have been studied to make text-independent speaker recognition systems accurate with very short speech samples and robust against both channel variability (differences due to the medium used to record the speech) and speaker dependent variability (such as health or mood of the speaker). A text-independent speaker recognition system using Gaussian mixture models and factor analysis techniques will be implemented in Matlab and tested against the NIST SRE databases for validation.

1 Project Background/Introduction

Humans have the innate ability to recognize familiar voices within seconds of hearing a person speak. How do we teach a machine to do the same? Research in speaker recognition/verification, the computational task of validating a person's identity based on their voice, began in 1960 with a model based on the analysis of x-rays of individuals making specific phonemic sounds [1]. With the advancements in technology over the past 50 years, robust and highly accurate systems have been developed with applications in automatic password reset capabilities, forensics and home healthcare verification.

There are two phases in a speaker recognition system: an enrollment phase where speech samples from different speakers are turned into models and the verification phase where a sample of speech is tested to determine if it matches a proposed speaker, as displayed in Figure 1. It is assumed that each speech sample pertains to one speaker. A robust system would need to account for differences in the speech signals between the enrollment phase and the verification phase that are due to the channels used to record the speech (landline, mobile phone, handset recorder) and inconsistencies within a speaker (health, mood, effects of aging) which are referred to as channel variability and speaker dependent variability respectively. In text-dependent systems, the words or phrases used for verification are known beforehand and are fixed. In a text-independent system, there are no constraints on the words or phrases used during verification. This project will focus on text-independent speaker verification systems.

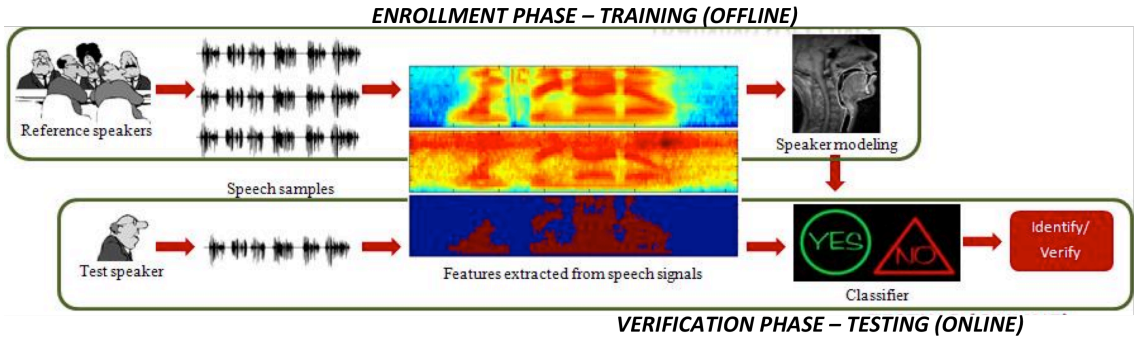

Figure 1: Speaker Recognition System (Courtesy of Balaji Srinivasan)

A variety of different features can be extracted from the speech samples, or utterances. Low level features relate to physiological aspects of a speaker such as the size of the vocal folds or length of vocal tract, prosodic and spectro-temporal features correspond to pitch, energy or rhythm of the speech, and high level features are behavioral characteristics such as accents or pronunciation. When extracting features, voice activity detectors (VADs) can be used to remove segments in an utterance where there is no speech. VADs can be energy based or based on periodicity. Many advanced systems account for multiple features and fusion is used to find the best overall match [2].

For text-independent speaker verification, the most popular modeling approaches are vector quantization (VQ), Gaussian mixture models (GMMs) and support vector machines (SVM). VQ is a technique that divides the features into clusters using a method such as K-means. GMM is an expansion of the VQ model, allowing each feature to have a nonzero probability of originating for each cluster [2]. A universal background model (UBM) representing an average speaker is often used in a GMM-based model. Adaptations of the UBM are used to characterize each of the individual speakers making the models robust even when the full phonemic space is not covered by the training data. SVM take labeled training data and seeks to find an optimized decision boundary between two classes which can be used to discriminate the different speakers.

Various techniques have been researched to assist in compensating for channel variability and speaker dependent variability, including speaker model synthesis (SMS) and feature mapping (FM). Most approaches require the speaker models to be organized into a high- and fixed-dimensional single vector called a supervector so that utterances with varying numbers of features can be represented in a general and compatible form. Popular methods that focus on compensating SVM supervectors include generalized-linear discriminant sequence (GLDS) kernel and maximum likelihood linear regression (MLLR) [2]. Factor analysis (FA) is a common generative modeling technique that is used on supervectors from GMMs to account for variability by learning low-dimensional subspaces. FA methods used in speaker verification include joint factor analysis (JFA) which model channel variability and speaker dependent variability separately, and total variability which model channel variability and speaker dependent variability in the same space. Normalization methods such as nuisance attribute projection (NAP), within-class covariance normalization (WCCN) and linear discriminant analysis (LDA) are also used for intersession variability compensation [2].

2 Approach

In this project, three separate speaker recognition systems will be developed, each of which will build upon the previous. All cases will be text-independent and will be implemented using mel-frequency cepstral coefficients (MFCCs) as the features with a voice activity detector (VAD) used to remove silent frames. The first speaker models will be UBM-adapted GMMs as shown in Figure 2. Once speaker models are created, a log likelihood ratio will be used to classify whether a test speaker is the same as a hypothesized speaker.

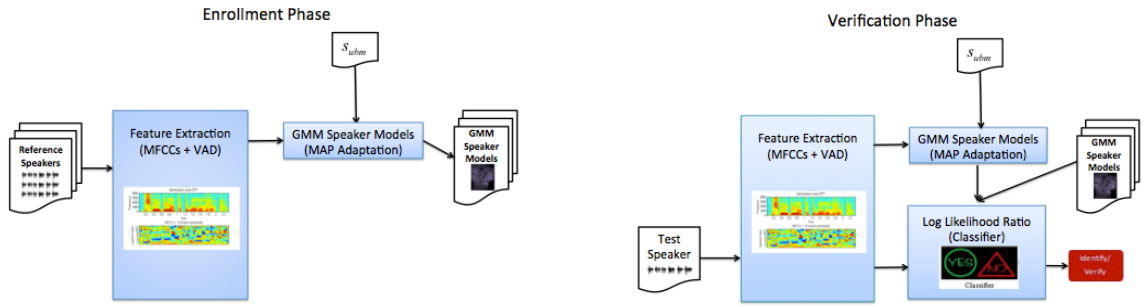

Figure 2: The first implementation for speaker models will be Gaussian mixure models based on the adaptation of the UBM. In the enrollment phase, a gallery of speaker models is generated from reference speakers. In the verification phase, a probe or test speaker is compared to a hypothesized speaker from the gallery using log likelihood ratio

Displayed as an input to the MAP Adaptation algorithm, the GMM Universal Background Model, s_{ubm}, is trained beforehand and only needs to be completed once. Other variables, such as the total variability space, T, and the reduced subspace, A, needed for the other speaker models are also trained prior to the enrollment phases of the different speaker recognition systems. The data used to create these variables are a large set of background speakers that do not overlap with the reference speakers in the speaker verification system. That said the background speakers should be a good representation of the

diversity found within the reference speakers. For example, if the speakers used as reference speaker are all male citizens of the United States, the background-training speakers should also be representative of all male citizens of the United States. Figure 3 illustrates all the training variables that need to be computed prior to speaker models being created.

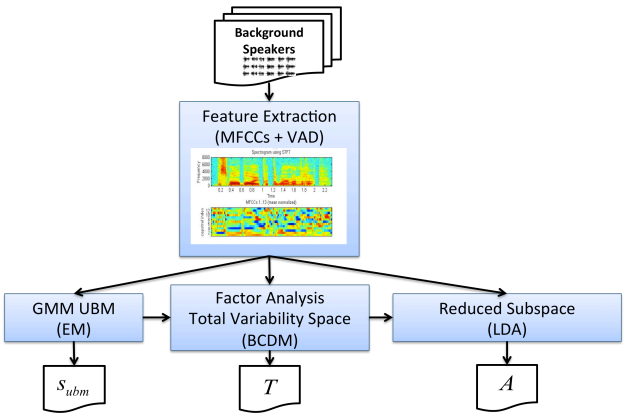

Figure 3: Algorithm flow chart for Background training

The second speaker models will be based on the idea that mean components in the GMMs concatenated into supervectors can create a new space. Factor Analysis (FA) techniques will also be used on the GMM supervectors to learn the low-dimensional total variability space. i-vectors will be extracted from the low-dimensional total variability space which uniquely represent the same information contained in the GMM supervectors. The i-vector speaker model flow chart is displayed in Figure 4. The i-vector speaker models build off of the GMM speaker models and use the total variability space trained using background speaker models. The classifier for i-vectors will be the discrete cosine score (DSC).

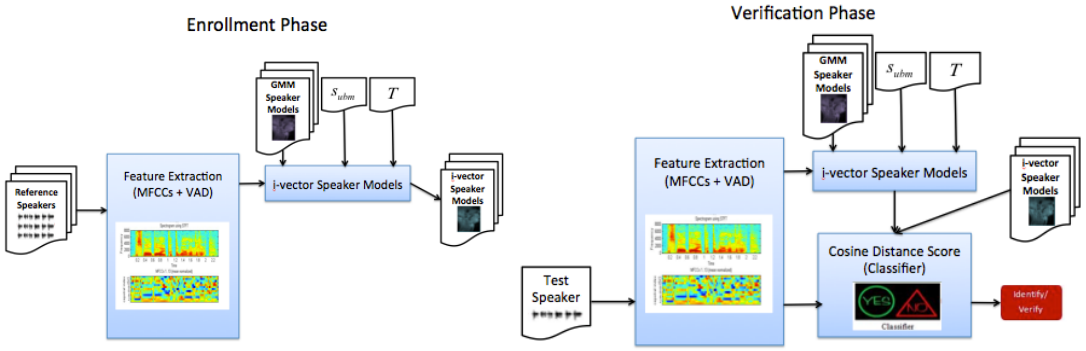

Figure 4: i-vector speaker verification

Linear discriminant analysis (LDA) techniques will then be applied to the i-vectors of the reference speakers to create the third and final speaker recognition system. Using LDA, a lower dimensional subspace will be found that will on maximize the inter-speaker variability and minimize speaker-

dependent variability. The algorithm flow chart for the LDA reduced i-vectors will be very similar to Figure 4, except i-vector speaker models will be used as an input, along with the total variability matrix and the reduced subspace matrix found in the background training section. Discrete cosine scoring (DCS) will also be used for verifying if a test utterance matches a proposed speaker.

2.1 Feature Extraction

The features used for this project will be mel-frequency cepstral coefficients (MFCCs). In order to make the MFCCs more robust, a voice activity detector (VAD) algorithm is implemented to find segments in the utterance in which no speech is detected.

2.1.1 MFCCs

Low-level features called mel-frequency cepstral coefficients (MFCCs) will be extracted from the speech samples and used for creating the speaker models. The mel-frequency scale maps lower frequencies linearly and higher frequencies on a logarithmic scale, as displayed in Figure 5, in order to account for the widely supported result that humans' can differentiate sound best at lower frequencies. Cepstral coefficients are created by taking a discrete cosine transform on the logarithm of the magnitude of the original spectrum. This step removes any relative timing, or phase, information between different frequencies and significantly alters the balance between intense and weak components [3]. MFCCs relate to the physiological aspects of a person such as the size of their vocal folds or length of their vocal tract and were first used starting in the 1980s [2]. They have been found to be fairly successful in speaker discrimination.

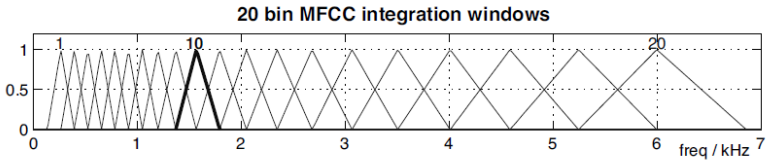

Figure 5: Illustration of 20 bins used for creating a 20-channel mel-frequency filterbank [3]. The highlighted bin 10 shows that most of the emphasis in this channel is on the center frequency with some weight placed on frequencies down to the center of the previous channel and up to the center of the next channel.

Given an utterance, it is first segmented using a 20 ms windowing process at a 10 ms frame rate. If the waveform is sampled at 16kHz, the 20 ms segment will contain 320 samples. The Fast Fourier Transform (FFT) algorithm with 512 points is applied to the speech sample. Then a mel-frequency filter bank is used to obtain an M-channel filterbank denoted as $Y(m), m = 1, \dots, M$. For this project, a 40-channel filterbank will be used. To convert the FFT power spectrum into a 40-channel filterbank, first the mel-frequency centers for the 40 different channels are determined. For each channel, weights for different frequencies will be computed with the most emphasis placed on the center frequency for the current channel and some emphasis placed on frequencies down to the center of the previous channel and up to the center of the next channel as displayed in Figure 5. The channels are then scaled so that each

channel has approximately a constant energy per channel [3]. Once the mel-frequency filterbank is computed, the MFCCs are found using the following formula:

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

where n is the index of the cepstral coefficient. The 19 lowest DCT coefficients will be used for purposes of this project along plus 1 energy value (c_0). For initial verification, the 12 lowest DCT coefficients will be used along with the energy term. The algorithm to obtain the MFCCs is outlined in Figure 6.

Input: *utterance*; *sample rate*
Output: *matrix of MFCCs by frame*
Parameters: *window size* = 20 ms; *step size* = 10 ms
nBins = 40; *d* = 13 (*nCeps*)

Step 1: Compute FFT power spectrum
Step II : Compute mel-frequency m-channel filterbank
Step III: Convert to ceptra via DCT

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

(0th Cepstral Coefficient represents "Energy")

Figure 6: MFCC algorithm

In addition to MFCCs, differences between the MFCCs at a current frame compared to a frame 10 away can be used as features. This is called the derivative of the MFCCs. The second derivative of the MFCCs (found by taking the derivative of the first derivative of the MFCCs) is also commonly used as features.

2.1.2 VAD

Since it is natural for people to pause while speaking, many of the 20 ms frames will contain no useful information. A simple energy based voice activity detector (VAD) will be applied to the speech signals in order to locate the specific frames that include speech segments and specific frames that are silent [2]. The algorithm for the VAD implemented is displayed in Figure 7. Note that the input and parameters to the VAD must match the input and parameters to the MFCC algorithm. For each 20 ms frame found to be silent, the entire frame is removed.

Input: *utterance*, *sample rate*
Output: *Indicator of silent frames*
Parameters: *window size* = 20 ms; *step size* = 10 ms

Step 1 : Segment utterance into frames
Step II : Find energies of each frame
Step III : Determine maximum energy
Step IV: Remove any frame with either:
a) less than 30dB of maximum energy
b) less than -55 dB overall

Figure 7: VAD algorithm

2.2 Gaussian Mixture Models using a Universal Background Model

Gaussian mixture models (GMMs) were first introduced as a method for speaker recognition in the early 1990s and have since become the *de facto* reference method [2, 4]. GMMs represent each speaker, s_i, by a finite mixture of multivariate Gaussians based on the *d*-dimensional feature vector $\boldsymbol{x}$:

$$p(\boldsymbol{x}|s_i) = \sum_{k=1}^{K} \pi_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \sum_k) \tag{2}$$

where K is the number of components, $\pi_k \geq 0$ represent the mixture weights that are constrained by $\sum_{k=1}^{K} \pi_k = 1$ and

$$N(\boldsymbol{x}|\mu_k, \sum_k) = (2\pi)^{-\frac{d}{2}} |\sum_k|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \sum_k{}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)\} \tag{3}$$

where μ_k of dimension $1 \times d$ represents the mean value of mixture component *k* and $\sum_k$ of dimension $d \times d$ represents the covariance of mixture component *k* [4]. Given the sequence of T training vectors $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ the GMM likelihood can be rewritten as

$$p(X|s) = \prod_{t=1}^{T} p(\boldsymbol{x}_t|\, s). \tag{4}$$

The values of $\pi, \mu, \sum$ representing each speaker, s, will be learned using maximum likelihood (ML) estimation techniques, which seek to find model parameters which maximize the likelihood of the GMM given the input training data, $\boldsymbol{x}$. Using full-covariance GMM normally requires a significant amount of training data and is very computationally intensive, therefore diagonal covariance matrices will be used.

A universal background model (UBM) or speaker-independent model is first created using speech samples from a large number of background speakers. Based on the newly found GMM-UBM components, $\pi^{UBM}, \mu^{UBM}, \sum^{UBM}$, a Bayesian adaptation technique is used to determine the components of the GMM for each individual speaker using an algorithm called Maximum a posteriori (MAP) adaptation.

2.2.1 GMM UBM

The parameters of the UBM are found using an expectation-maximization (EM) algorithm which iteratively refines a random initialization of GMM parameters to monotonically increase the likelihood of the estimated model based on the given feature vectors. This algorithm is part of the background training shown in Figure 3. For this algorithm, the MFCCs created from feature extraction will all be concatenated into a large matrix. It is important to have a large amount of data that equally represents the reference speakers that will be part of the speaker verification system. In a case where there is too much data to run on a given system, the data can be scaled down to 2*8*nCeps*K [reference], where K corresponds to the number of Gaussian centers, and still yield good results. In the case where the number of centers and the number of features are both low, the algorithm will choose a maximum between this number and 10000. It important for the data to be scaled down in a uniform and random manner to ensure that there is a good representation of the entire domain. For the initial

implementation, 512 Gaussian centers will be used. In most state of the art speaker recognition systems, 2048 Gaussian centers are currently used.

In the estimation step, the Bayesian statistics are used to determine the probability of mixture component c:

$$\gamma_t(c) = p(c|\boldsymbol{x}_t, s) = \frac{\pi_c N(\boldsymbol{x}|\boldsymbol{\mu}_c, \sum_c)}{\sum_{k=1}^{K} \pi_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \sum_k)} \tag{5}$$

In the maximization step, the following are formulas are used which guarantee a monotonic increase in the model's likelihood value [5]:

Mixture weights:

$$\pi_c = \frac{1}{T} \sum_{t=1}^{T} \gamma_t(c) \tag{6}$$

Means:

$$\boldsymbol{\mu}_c = \frac{\sum_{t=1}^{T} \gamma_t(c) \boldsymbol{x}_t}{\sum_{t=1}^{T} \gamma_t(c)} \tag{7}$$

Variances:

$$\sigma_c = \frac{\sum_{t=1}^{T} \gamma_t(c) \boldsymbol{x}_t^2}{\sum_{t=1}^{T} \gamma_t(c)} - \boldsymbol{\mu}_c^2 \tag{8}$$

The expectation step and the maximization step are iterative. The algorithm will cease to iterate when $\frac{abs(maxlikelihood_{old} - maxlikelihood_{new})}{abs(maxlikelihood_{old})} < 0.01$. There is also a maximum number of iteration that is set such that the algorithm should converge before the designated value. The algorithm is detailed in Figure 8. Remember that diagonal covariances will be used and a tolerance will be set so that the covariances will not grow too small.

Input: *Concatenation of the MFCCs of all background utterances* ($\mathbf{x} = \mathbf{x}_{background}$)

Output: $s_{ubm} = \{\pi^{ubm}, \mu^{ubm}, \Sigma^{ubm}\} = \{\pi, \mu, \Sigma\}$

Parameters: K = 512 (*nComponents*); *nReps* = 10

Step I : Initialize $\{\pi, \mu, \Sigma\}$ randomly

Step II: (Expectation Step)

Obtain conditional distribution of component c

$$\gamma_t(c) = p(c \mid \mathbf{x}_t, s) = \frac{\pi_c N(\mathbf{x}_t \mid \boldsymbol{\mu}_c, \Sigma_c)}{\sum_{k=1}^{K} \pi_k N(\mathbf{x}_t \mid \boldsymbol{\mu}_k, \Sigma_k)}$$

Step III: (Maximization Step)

Mixture Weight: $\pi_c = \frac{1}{T}\sum_{t=1}^{T} \gamma_t(c)$

Mean: $\boldsymbol{\mu}_c = \frac{\sum_{t=1}^{T} \gamma_t(c)\mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(c)}$

Covariance: $\sigma_c = \frac{\sum_{t=1}^{T} \gamma_t(c)\mathbf{x}_t^2}{\sum_{t=1}^{T} \gamma_t(c)} - \boldsymbol{\mu}_c^2$

Step IV: Repeat Steps II and III until

$$\frac{abs(maxlikelihood_{old} - maxlikelihood_{new})}{abs(maxlikelihood_{old})} < 0.01$$

Figure 8: EM Algorithm for GMM

2.2.2 MAP Adaptation for Speaker Models

Given the GMM UBM, the GMM Speaker models can be determined as shown the algorithm flow chart in Figure 2. The speaker models will be made up of the parameters $s = \{\pi^{ubm}, \mu, \Sigma^{ubm}\}$, where π^{ubm}, Σ^{ubm} are values from the UBM and μ will be computed using the MAP algorithm and will be unique for each speaker. The algorithm could be modified to include adaptations of the weights and the covariances, but it has been found to degrade performance compared to only adapting the means.

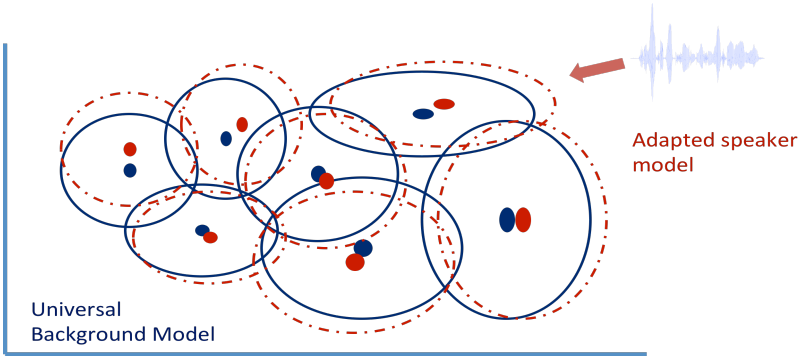

Figure 9: Maximum a posteriori (MAP) algorithm used to adapt the UBM (Courtesy of Balaji Srinivasan)

The first step of the MAP algorithm is the same as steps II and III in EM algorithm, that is, the values of μ_c are found using Bayesian statistics and ML estimations as in (6) and (7) on the reference database corresponding to a specific speaker using the UBM, s_{ubm}. Once these parameters are found, they are

used to update the old UBM parameters $\boldsymbol{\mu}_c^{UBM}$ for mixture component c to create the adapted parameters for mixture component c with the equation:

$$\widehat{\boldsymbol{\mu}}_c = \alpha_c^m \boldsymbol{\mu}_c + (1 - \alpha_c^m)\boldsymbol{\mu}_c^{UBM} \tag{9}$$

where α_c^m represent the adaptation coefficients controlling the balance between the old and the new estimates for the means [4]. The values of α_i^m are defined as

$$\alpha_c^m = \frac{\sum_{t=1}^{T} \gamma_t(c)}{\sum_{t=1}^{T} \gamma_t(c) + r^m} \tag{10}$$

where r^m is a fixed relevance factor. For this project, $r^m = 16$ will be used since experimental results have found performance to be rather insensitive to values in the range of 8-20 [4]. Using data-dependent adaptation coefficients enables de-emphasis on new parameters when a mixture component has a low probabilistic count with more emphasis on the old parameters. If a mixture component has high probabilistic counts, more emphasis can be placed on the new parameters. Having the ability to adjust the adaptation coefficients based on the data leads to robustness against limited training data. The MAP adaptation concept is illustrated in Figure 9 and the algorithm is outlined in Figure 10. Note that unlike the EM algorithm to determine the UBM, the MAP adaptation is not iterative.

Input: *MFCCs of utterance for speaker* ($\mathbf{x} = \mathbf{x}_{utterence}$);
$s_{ubm} = \{\pi^{ubm}, \boldsymbol{\mu}^{ubm}, \Sigma^{ubm}\}$

Output: $s_i = \{\pi^{ubm}, \boldsymbol{\mu}^{i}, \Sigma^{ubm}\}$

Parameters: K = 512 (*nComponents*); *r*=16

Step I : Obtain $\boldsymbol{\mu}_c$ via Steps II and III in the EM for GMM algorithm (using s_{ubm})

Step II: Calculate $\boldsymbol{\mu}_c^i = \alpha_c^m \boldsymbol{\mu}_c + (1 - \alpha_c^m)\boldsymbol{\mu}_c^{ubm}$

where $\alpha_c^m = \frac{\sum_{t=1}^{T} \gamma_t(c)}{\sum_{t=1}^{T} \gamma_t(c) + r}$

Figure 10: MAP Adaptation Algorithm

2.3 Factor Analysis

To create a better model for the speakers, it is now assumed that the MFCCs from each utterance comes from a 2-stage generative model. The first stage is the K-component GMM where the weights and covariance matrices come from the UBM as in the model above. The second stage relies on the concatenation of the mean components of the GMMs for each into a high- and fixed-dimensional single vector, called the supervector, of dimension $Kd \times 1$ where K is the number of Gaussian centers and d is the number of features. The second stage generative model is called a factor analysis model is described in the following equation:

$$\mu = m + Tw \tag{11}$$

where $\mu \in \mathbb{R}^{KD}$ represents the supervector of a given speaker or utterance, $m \in \mathbb{R}^{KD}$ represents the supervector of the UBM, $T \in \mathbb{R}^{KD}$ represents the total variability space and $w \in \mathbb{R}^{pT}$ represents the i-vector for a given speaker or utterance. The dimension pT can be chosen by the user. It has been found imperially that a good dimension to use for the speaker recognition problem is 400.

First, consider training of an i-vector. Given the MFCCs for a given utterance, $\boldsymbol{x} = \{x_t\}_{t=1}^{T}$ and fixed total variability space T, we want to find the maximum probability of a given speaker denoted by its supervector μ given the utterance, which is the same as minimizing the negative log-likelihood displayed in the following equations:

$$\max_{\mu} p(\mu|\boldsymbol{x}) = \max_{\mu} p(x|\mu)p(\mu) \tag{12}$$

$$= \min_{\mu}\{-\log\left(p(x|\mu = m + Tw) - \log\big(p(w)\big)\right\}$$

where $p(w) = N(0, I)$ is an assumption. This turning into minimizing:

$$\psi(w) = \frac{1}{2}\left\|W^{\frac{1}{2}}(\eta - Tw)\right\|_2^2 + \frac{1}{2}\|w\|_2^2 \tag{13}$$

where $W = \Gamma\Sigma^{-1}$, $\Gamma = diag(\Gamma_k)$, $\Gamma_k = \gamma_k I$ represents the 0[th] order statistic and $\eta = \mu - m$ is the centered 1[st] order statistics. These two represent the sufficient statistics for a given utterance or speaker. The algorithm to obtain the i-vector is described in Figure 10;

Input: $\mathbf{x}$ - the MFCCs for a given speaker/utterance
T - the total variability space
Output: w - the i-vector for a given speaker/utterance
Step 1: Obtain sufficient statistics $(\eta, \mathbf{W})$

Step 2: Given $\mathbf{W}$ is positive semi-definite,

$$\Psi(w) = \frac{1}{2}\left\|\mathbf{W}^{\frac{1}{2}}(\eta - \mathbf{T}w)\right\|_2^2 + \frac{1}{2}\|w\|_2^2$$

is strongly convex and therefore the minimization problem has a closed form solution:

$$w = (\mathbf{I} + \mathbf{T}^T\mathbf{W}\mathbf{T})^{-1}\mathbf{T}^T\mathbf{W}\eta$$

Figure 11: i-vector training algorithm

In order to train the total variability space *T*, all utterances $D = \{X_r\}_{r=1}^{R}$ from the background speakers must be considered, each being represented by their sufficient statistics (η_r, W_r). To train *T,* it is desired to minimize a the following equation with respect to *T* and all i-vectors w_r:

$$\min_{T,\{w_r\}} \sum_{r=1}^{R}\left(\frac{1}{2}\left\|W_r^{\frac{1}{2}}(\eta_r - Tw_r)\right\|_2^2 + \frac{1}{2}\|w_r\|_2^2\right) \tag{14}$$

When *T* and all w_r are variable, the above equation is not convex and therefore the solution cannot be determined directly. The equation will be solved using Block coordinate descent minimization, which is an alternating optimization algorithm, described in Figure XX. In steps 3 and 4, it is assumed certain variable are fixed. In each case, the remaining function is convex related to the other variable(s) and therefore a solution can be found directly by taking the derivative and setting it equal to zero. Due to the complexities of the problem, there are no convergence guarantees but the method has been found to work well by experiment. It has been found that 10-20 iterations yield good results. By default, 10 iterations will be used for the algorithm.

Input: $D = \{\mathbf{X}_r\}_{r=1}^{R}$
Output: $\mathbf{T}$ – total variability space
Step 1: Obtain sufficient statistics for each utterance $(\eta_r, \mathbf{W}_r)$
Step 2: Use random initialization for $\mathbf{T}$
Step 3: Assume $\mathbf{T}$ is fixed, minimize w_r

$$w_r = (\mathbf{I} + \mathbf{T}^T \mathbf{W}_r \mathbf{T})^{-1} \mathbf{T}^T \mathbf{W}_r \eta_r$$

Step 4: Assume w_r are fixed, minimize $\mathbf{T}$

$$\min_{\mathbf{T}} \sum_{r=1}^{R} \left\| \mathbf{W}_r^{\frac{1}{2}} (\eta_r - \mathbf{T} w_r) \right\|_2^2$$

with the solution $\mathbf{T}_{new}$ where

$$\mathbf{T}_{new}^{(k)} \left(\sum_{r=1}^{R} \gamma_{rk} w_r w_r^T \right) = \sum_{r=1}^{R} \gamma_{rk} \eta_r^{(k)} w_r^T$$

Step 5: Do 10 iterations of Steps 3 and 4

Figure 11: Algorithm to train the total variability space

2.4 Linear Discriminant Analysis

Another dimensionality reduction technique called linear discriminant analysis (LDA) will be used. Once the total variability space, T, and the i-vectors, w are learned using the algorithm in Figure 11, LDA can be used to project the i-vectors into a lower-dimensional space, ω using the following equation:

$$\omega = Aw \tag{15}$$

The matrix A is chosen such that within-speaker, or speaker-dependent, variability (via the within-speaker covariance) is minimized and inter-speaker variability (via the inter-speaker covariance) is maximized within the space. The within-speaker covariance can be found for K speakers as:

$$S_W = \sum_{k=1}^{K} S_k \tag{16}$$

where

$$S_k = \sum_{n \in C_k} (x_n - m_k)\,(x_n - m_k)^T \tag{17}$$

$$m_k = \frac{1}{N_k}\sum_{n\in C_k} x_n \tag{18}$$

and N_k is the number of utterances for Speaker C_k. In order to determine the inter-speaker covariance matrix, we first consider the total covariance matrix

$$S_T = \sum_{n=1}^{N}(x_n - m)(x_n - m)^T \tag{19}$$

where m is the mean of the total data set

$$m = \frac{1}{N}\sum_{n=1}^{N} x_n = \frac{1}{N}\sum_{k=1}^{K} N_k m_k \tag{20}$$

and $N = \sum_k N_k$is the total number of utterances. The total covariance matrix can be decomposed as the sum of the within-speaker covariance matrix plus the inter-speaker or between speaker covariance matrix S_B.

$$S_T = S_W + S_B \tag{21}$$

In order for (21) to be true, S_B must be defined as

$$S_B = \sum_{k=1}^{K} N_k (m_k - m)\,(m_k - m)^T \tag{22}$$

Similar matrices can be construction that represent the covariances, s_W, s_B of the projected subspace. In order to construct a matrix in which the inter-speaker covariances is large and the within-speaker covariance is small, one could maximize the matrix

$$J(A) = Tr\{s_w^{-1} s_B\} \tag{23}$$

Solving this problem is the same as finding the eigenvectors of $S_W^{-1}S_B$ which means solving the eigenvalue problem $S_B a_i = \lambda_i S_w a_i$. The dimension of the new subspace can be no greater than the number of speakers used for training. For initial implementation, the dimension of the new subspace will be 200.

2.4 Classifiers

Two classifiers will be used for the accept/reject decision. A log-likelihood ratio test will be used based on the GMMs models and cosine distance scoring will be used on both the i-vectors and the intersession-compensated LDA reduced i-vectors.

2.4.1 Log-likelihood ratio test

Given a GMM speaker model s_{hyp} and the GMM-UBM s_{UBM}, a log-likelihood ratio test can be applied on the extracted features of a test utterance, X using the following formula [4]:

$$\Lambda(X) = \log p\big(X|s_{hyp}\big) - \log p(X|s_{UBM}) \tag{24}$$

where $\Lambda(X) \geq \theta$ will lead to verification of the hypothesized speaker, s_{hyp} , and $\Lambda(X) < \theta$ will lead to rejection.

An algorithm was implemented in attempt to create faster scoring based on [4]. In this paper, it was stated that with the large dimensional GMM, the top C scoring mixture components could be used to determine the log-likelihood when comparing the utterance to the UBM. The paper stated that C=5 can be used. Implementing the algorithm would save computation time but is only beneficial if similar results are obtained using the faster method. After numerous experiments, it was found that the faster model significantly degraded performance, even if a C value of 256 is used where there were originally 512 GMM components.

2.4.2 Discrete cosine score

The discrete cosine score (DCS) can be applied to both the i-vectors, w, and the intersession-compensated i-vectors using LDA, ω using the following equation [9]:

$$score(\omega_1, \omega_2) = \frac{\omega_1^* \omega_2}{\|\omega_1\| \|\omega_2\|} = \cos(\theta_{\omega_1, \omega_2}) \tag{25}$$

where $score(\omega_1, \omega_2) \geq \varphi$ will lead to verification of the hypothesized speaker and $score(\omega_1, \omega_2) < \varphi$ will lead to rejection.

3 Implementation

In Phases II-V (described in Section 7), a simple yet complete speaker recognition system was implemented in Matlab on a modern Dell desktop computer. A software package that extracts the MFCCs from the utterances was used [12] along with the tool set to plot the DET curves [18]. All of the other main algorithms were developed along with the two classifier tests to validate code at different the phases. Testing the TIMIT database and the SRE databases required some special scripts to read certain files and those scripts were used when necessary.

The code finishes running in a reasonable amount of time for lower dimensional features and modest sized training sets, for example in the case where 12 MFCCs were used with 512 Gaussian components for testing the TIMIT database or the SRE databases. In the case where 19 MFCCs were used along with the derivative and the second derivative of the MFCCs to create a 57 dimensional space with 1024 Gaussian components for testing the SRE databases, the code runs but takes a long time to complete. The code was transferred to a computing cluster (the UMD cluster named Chimera) and many sections were run in parallel in order to complete in an acceptable amount of time. Most parallelization of the code was done ad hoc as not much time was left in the semester. It was noted in the proposal that the numerical complexities and high memory requirements were expected in with higher dimensional features and that the code was not expected to be able to process large amounts of data.

4 Databases

4.1 TIMIT Database

The TIMIT dataset designed by SRI International, Texas Instruments, Inc (TI) and Massachusetts Institute of Technology (MIT) in 1990 was used for initial testing in order to obtain results on a smaller set of speakers. This database contains recording of 630 speakers from eight different American dialect regions, each reading ten sentences that have been chosen to encapsulate the phonetic space. The waveforms are recorded at 16kHz. Two-thirds of the speakers were used as background speakers to train the UBM, the total variability space and the LDA reduced subspace while the remaining one-third was used for testing. Performance on the TIMIT dataset is expected to be good as the TIMIT files are relative clean and have little variations due to channel variability and speaker-dependent variability. The Figure 12 below displays the demographics of the speakers.

Dialect Region	#Male	#Female	Total
1	31 (63%)	18 (27%)	49 (8%)
2	71 (70%)	31 (30%)	102 (16%)
3	79 (67%)	23 (23%)	102 (16%)
4	69 (69%)	31 (31%)	100 (16%)
5	62 (63%)	36 (37%)	98 (16%)
6	30 (65%)	16 (35%)	46 (7%)
7	74 (74%)	26 (26%)	100 (16%)
8	22 (67%)	11 (33%)	33 (5%)
8	438 (70%)	192 (30%)	630 (100%)

Figure 12: Demographics of TIMIT database

4.2 SRE Databases

The National Institute of Standards and Technology (NIST) has coordinated Speaker Recognition Evaluations (SRE) approximately every two years since 1996. In support of the SRE, databases consisting of sph formatted files with a sampling rate of 8kHz are provided for use of the participants. Approximately 1,400 different speakers were used from the SRE 2004, SRE 2005 and SRE 2006 databases as background speakers with over 16,000 wav files (derived from the sph files) in order to train the UBM, the total variability space and the LDA reduced subspace. The background training results from these three SRE databases were used in generating the speaker models for the reference and test speakers within the SRE 2010 database. The SRE 2010 database consisted of over 20,000 files that were tested in 9 different conditions. All of the NIST SRE databases contain speech data in several different conditions including data from interviews, microphones, telephone conversations with some differences in speaker dependent variability like vocal effort as seen in the table below.

Condition Number	Number of Trials	Description
Cond 1	46,923	All trials involving interview speech from the same microphone in training and test
Cond 2	219,842	All trials involving interview speech from different microphones in training and test
Cond 3	58,043	All trials involving interview training speech and normal vocal effort conversational telephone test speech
Cond 4	85,902	All trials involving interview training speech and normal vocal effort conversational telephone test speech recorded over a room microphone channel
Cond 5	30,373	All different number trials involving normal vocal effort conversational telephone speech in training and test
Cond 6	28,672	All telephone channel trials involving normal vocal effort conversational telephone speech in training and high vocal effort conversational telephone speech in test
Cond 7	28,604	All room microphone channel trials involving normal vocal effort conversational telephone speech in training and high vocal effort conversational telephone speech in test
Cond 8	28,604	All telephone channel trials involving normal vocal effort conversational telephone speech in training and low vocal effort conversational telephone speech in test
Cond 9	27,520	All room microphone channel trials involving normal vocal effort conversational telephone speech in training and low vocal effort conversational telephone speech in test

Figure 13: Description of the 9 different conditions tested for the SRE 2010 database

Unlike the TIMIT database that consists of short utterances, the audio samples for the NIST database are much longer and more varied. The files usually contain two speakers and are accompanied with an Automatic Speech Recognition (ASR) files that labels when a specific speaker is talking. This information should be combined with the VAD in order to determine the non-silent frames in which the speaker of interest is talking.

5 Validation

In order to verify that the algorithms have been implemented correctly, various techniques were used for the different algorithms and are summarized in Figure 14.

Algorithm	Validation Technique
MFCCs	Compared modified code to original code by Dan Ellis
VAD	Audio evaluation and visual inspection
EM for GMM	Visual inspection for 2D feature space with 3 Gaussian components to check for convergence • Analysis on various k values used in algorithm ($k < 3, k = 3, k > 3$) Compare results with vetted system (Lincoln Labs)
MAP Adaptation	Visual inspection for 2D features space with 3 Gaussian components • Analysis on speakers with varying levels of representation for different components Compare results with vetted system (Lincoln Labs)
BCDM for FA	Attempted to create dataset and compare orthonormal project onto the range of the total variability space • Determined too non-linear for validation method to be valid Compare results with vetted system (BEST Project Code)
LDA	Visual inspection in 2D and 3D space

Figure 14: Table of the Validation methods for the main algorithms

For the GMM speaker models and the i-vector speaker models, the validation included of comparing the Equal Error Rate (EER) and the Detection error trade-off (DET) curves to a vetted system. The EER is a measure that gives the accuracy at the decision threshold for which the probabilities of false rejection (miss) and false acceptance (false alarm) are equal. This measure is good at obtaining a first quick understanding of whether there are any bugs. DET curves used for a visual inspection of the performance for various thresholds.

5.1 Feature Extraction

5.1.1 MFCCs

The code to create the MFCCs was modified from a tool set created by Dr. Dan Ellis from Columbia University. In order to validate this code, the results of the modified code were compared to the results of the original code given the same inputs.

5.1.2 VAD

As shown in Figure 15, a tool was created to display an input waveform to the VAD algorithm along with the VAD results marking the detected speech segments. By visual inspection, it seems as though the detected speech segments looks reasonable. The tool also enables the user to listen to the original waveform, the concatenation of the silent frames along with the concatenation of the frames in which speech was detected. There is never any loss of speech data in the new speech segment that can be heard and silent frames sound silent. When the silent frames are normalized, the result sounds like noise.

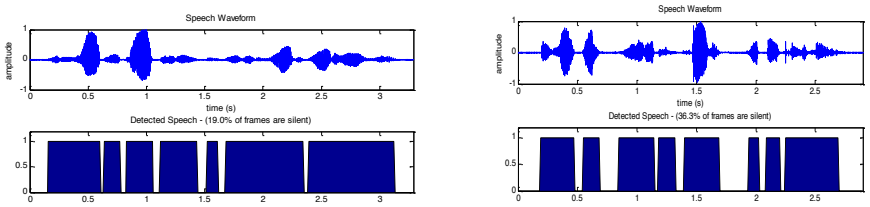

Figure 15: Speech Detection from VAD on two different waveforms

5.2 GMM

In order to test the GMM speaker models, a tool was created in order to make toy set of examples in two dimensional features space. The tool enables the user to choose the number of Gaussian centers for the data. Besides the number of Gaussian centers, the number of test speakers and the number of training speakers must be designated. Figure 16 shows examples in which the number of Gaussian centers were chosen to be 3 for (a) and 128 for (b). Each time the function is run, a new set of centers is created, even if the number of Gaussian centers is chosen to be the same. Numerous examples were used for validation but only one per number of components will be used in this presentation. The number 3 was chosen in order to let the reader visually understand the algorithm's performance. The number 128 was chosen to enable to reader to see the difficulties in understanding the results in 2-dimensional space when only 128 components are used. The smallest set of true data used in the speaker recognition project will be 512 Gaussian centers in 13-dimensional feature space.

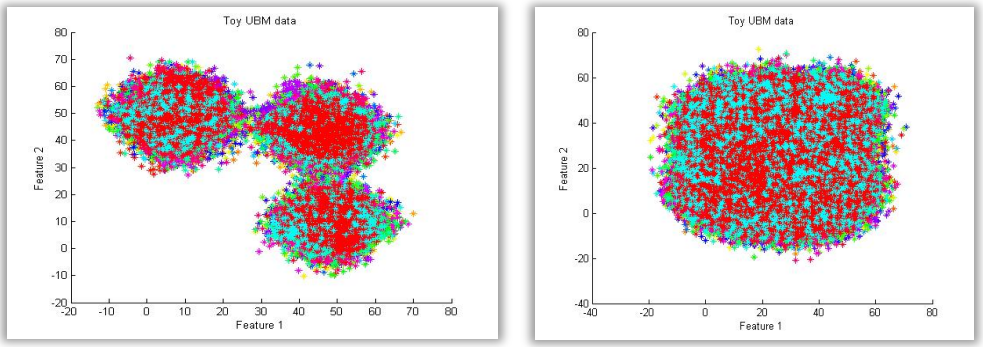

Figure 16: Result of tool to create toy datasets using different numbers of Gaussian Centers.
a) User designated 3 Gaussian centers b) User designated 128 Gaussian centers.

First, the true Gaussian centers are determined randomly on a uniform space. Then, for each speaker, a mean offset and covariance is chosen for the speaker for each true Gaussian component. The speaker can have a random number of data points for a given component, and different components have a different probability that speakers will have data related to the component. Once the mean offsets, the covariance and the number of data points is determined for a given speaker and a given component, random features are generated and added to the speaker's feature matrix that represents the features in an utterance. Two separate utterances are generated for each speaker that use the same mean

offsets and covariance matrices, but will have a different number of data points per component and different random data points. In Figure 16, each speaker is represented in a different color.

The same method of creating features for the speaker's utterances is used for both the training speakers and the test speakers, but all have different mean offsets and covariance matrices.

5.2.1 EM For GMM UBM

The results of both utterances for all of the speakers in the training set are used to create the GMM UBM. Figure 17 displays all the iterations of a random initialization of the EM algorithm for GMM. Each iteration comes closer to the true answer until the values converge. Figure 18 displays examples of other random initializations and the number of iterations it took for the EM algorithm to converge. The black data points represent all of the data points used to train the GMM UBM. The green x denotes the mean of a given Gaussian component and the green ellipses in which the x's are the centers represent the covariance matrices of the given Gaussian components. Note that not every random initialization will yield a desirable result. Therefore it is advised to complete multiple repetitions and keep only the parameters that yielded the largest maximum likelihood.

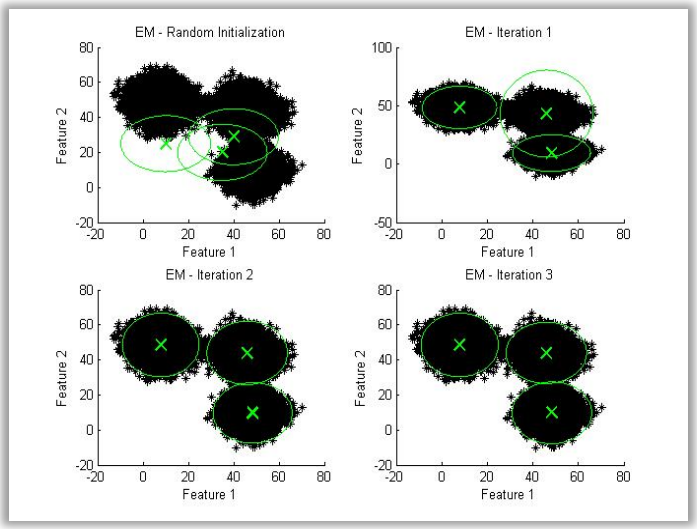

Figure 17: Iteration in the EM algorithm showing convergence

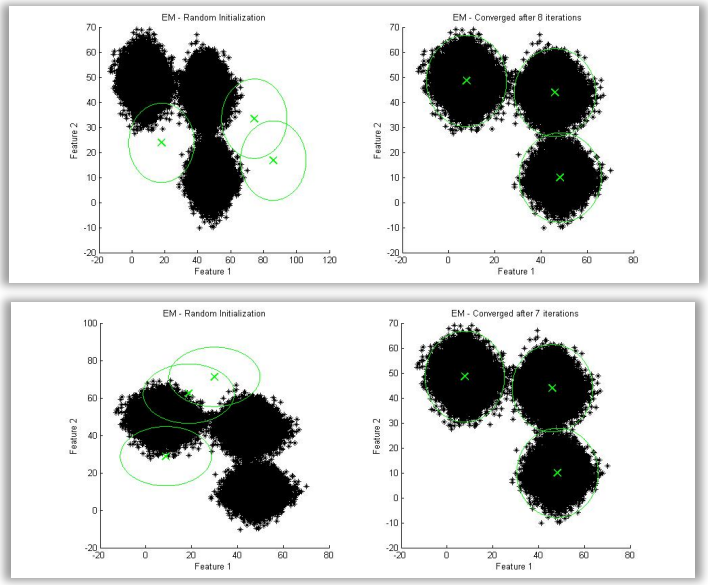

Figure 18: Convergence of the EM algorithm with different initializations.
a) converged after 8 iterations. b) converged after 7 iterations.

Both Figures 17 and 18 displayed results in which the number of Gaussian centers entered in as an input to the EM algorithm matched the number of true Gaussian centers. It is an interesting study to understand what happens when these numbers do not agree. As shown in Figure 19, if the value used in the EM algorithm is smaller than the number of true Gaussian components, the result is poor because the algorithm is forced to merge at least two true Gaussian components together. This will result in parameters that do not match any true Gaussian.

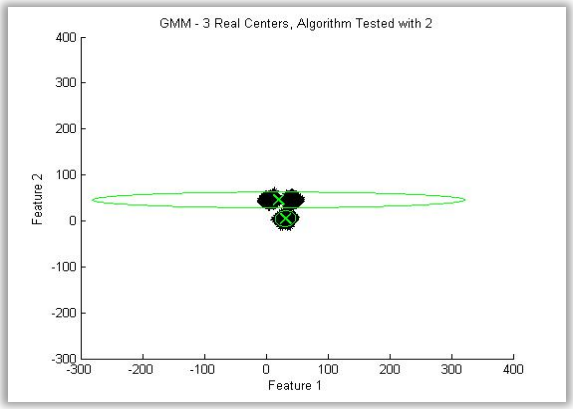

Figure 19: K=2 is less than the number of true Gaussian components (3)

If the value entered into the EM algorithm is larger than the number of true Gaussian components and there is a sufficient amount of data, the true Gaussian components are estimated well and any additional parameters represent made up Gaussian centers that are created by less than 5% of the data

per center and often much less. This is true even when the number of proposed Gaussian centers for the EM algorithm is over twice the number of true centers, as displayed in Figure 20 b.

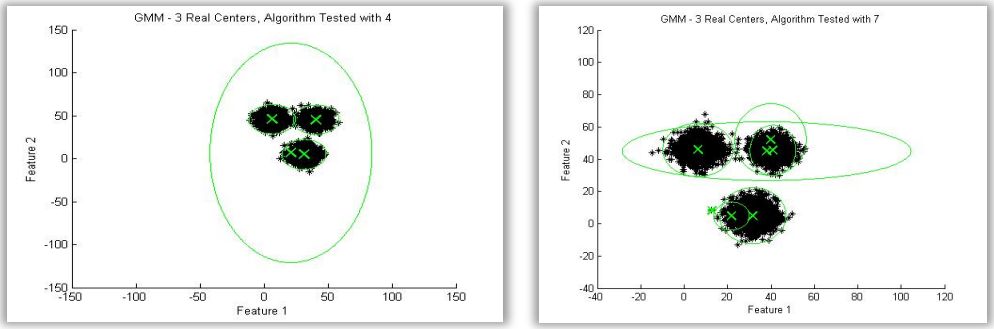

Figure 20: a) K=4 and b) K = 7 is more than the number of true Gaussian components (3)

All of the above examples were using a data set in which there were 3 true Gaussian components. The lowest number of Gaussian components analyzed in the speaker recognition project will be 512 and will be in 13 dimensional space. To illustrate the difficulties in visualizing the data set, Figure 21 shows an example in 2 dimensional space with 128 components. The covariance for only three components are drawn, otherwise the entire figure would be covered with green ellipses. In this case, the algorithm converged after 3 iterations.

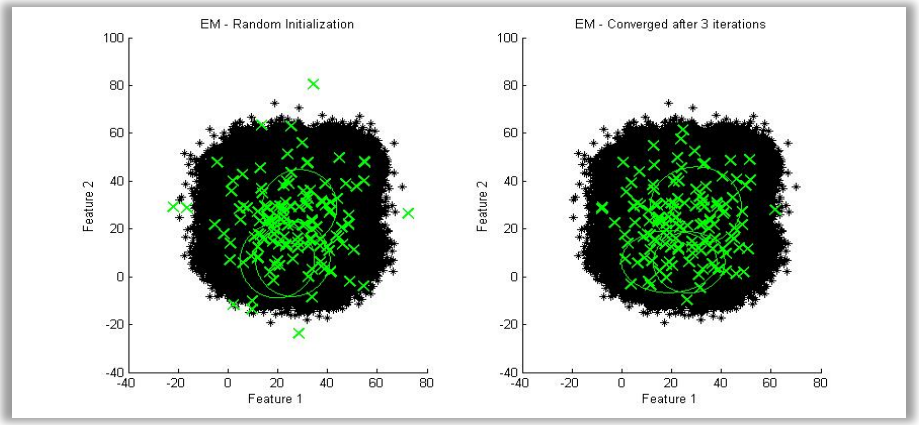

Figure 21: EM algorithm results with 128 true components and K=128

5.2.2 MAP Adaptation For GMM Speaker Models

The MAP adaptation algorithm should set the means for a speaker model to be the means of the UBM if there is very little or no data representing a given component for a given speaker. If there is sufficient data, the means for the speaker model should be representative of the mean for the speaker increasingly as the amount of data for the component per speaker is increased. Given these expectations, a visual inspection can be used to validate the algorithm when the number of components

can be observed. Figure 22 shows results for 4 different speakers when the number of true components is equal to 3. The black data points represent the data points in the Universal Background Model whereas the red data points represents the data points for the specified speaker. As expected, when the number of data points for a given component for the speaker is increased, the mean for the component lays on top of the mean for the speaker model. When there is no data for the speaker for a given component, as observed in one of the components for speaker 2, the mean is the same for the speaker as it is for the UBM. In cases where there are some data points corresponding to a component, but not a significant amount, it should be observed that the mean for the speaker is in between the mean of the UBM and the estimated mean of the speaker for the given component.

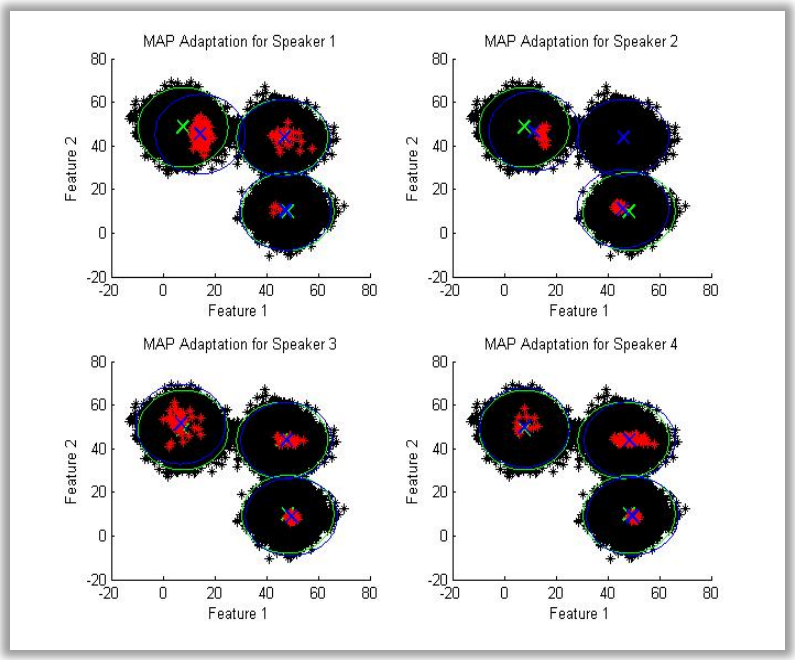

Figure 22: The results of the MAP algorithm of four separate speakers

Figure 23 illustrates the MAP adaptation results in the case in which there were 128 Gaussian components. There are many more cases in which a component for a given speaker is not represented by the data in a given utterance and therefore the means of the components are often represented by the mean of the UBM.

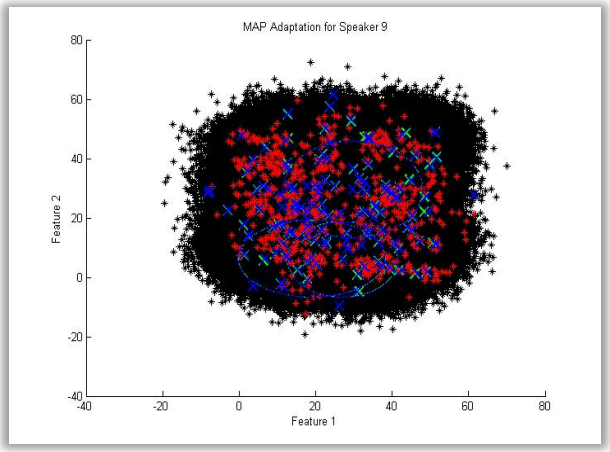

Figure 23: Example MAP Adaptation for speaker with the number of Gaussian Components being 128

5.2.3 GMM Speaker Models: Comparison to Vetted Code

In addition to validating the algorithms for the GMM speaker models by analyzing the results in two-dimensional space with three Gaussian components, the results of the GMM speaker models generated by the Matlab algorithms were compared with the results of the GMM speaker models generated by a vetted set of code written by MIT Lincoln Labs . The TIMIT dataset was used for comparison with the DET curves and the EER displayed in Figure 24.

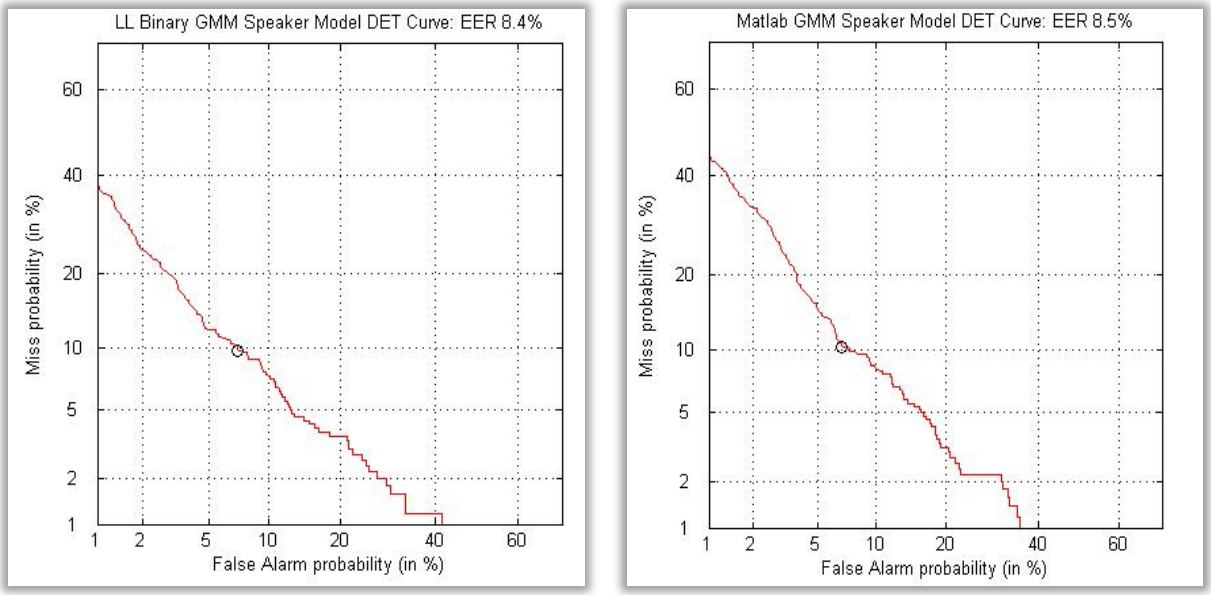

Figure 24: Comparison of results from vetted Lincoln Lab algorithm (left) to the results from the matlab implemented code (right)

Comparing the implemented code to the vetted code was an important step in the validation process. After the code validated by analyzing the behaviors of the algorithm in lower dimensional space, the implemented code initially obtained an EER of 16.2% compared to the EER of 8.4%. The Matlab code originally used random initializations whereas the vetted code used k-means to initialize the EM algorithm for GMM. Even after changing the initialization, I performed worse than the vetted code. It was determined after analysis that the sampling method implemented in Matlab was not ideal. Originally, all data points for the background speakers were concatenated into a large matrix and then down-sampled to the appropriate number whereas the a better method would be to determine the number of down-sampled data points needed and take an equal number of data points from each utterance. This led to a comparable performance to the vetted code.

5.3 Factor Analysis

To validate the block coordinate descent minimization (BCDM) algorithm, an attempt was made to work backwards from some given data, such as a total variability space and a set of i-vectors and generate a set of GMM means to represent speaker models and then from there generate a set of features that represent different utterances for different speakers. Given this set of data created, the algorithms developed for this project would determine a UBM and a total variability space. The idea was to compare the orthonormal projection onto the range of the total variability space as described in Figure 25. After a significant amount of work in trying to determine if the validation method was invalid or if there was a problem with the code, it was determined that with the amount of non-linearity of the algorithms, it is not expected that this validation method will lead to the original goal.

Input: $\mathbf{T}, m, w_r, \Sigma, \pi_r$

Output: $\mathbf{T}_{est}$

Step 1: Generate $\mu_r, \mathbf{X}_r$ using input variables

Step 2: Use code to determine $\mathbf{T}_{est}$ based on $m, w_r, \Sigma, \pi_r, \mu_r, \mathbf{X}_r$

Step 3: Compare the orthogonal projection onto $Ran(\mathbf{T})$ of the original vs the estimate

$$norm\left\| T \cdot (T^T T)^{-1} \cdot T^T - T_{est} \cdot (T_{est}^T T_{est})^{-1} \cdot T_{est}^T \right\|_{fro} \sim 10^{-10}$$

Figure 25: Method used in attempt to validate BCDM algorithm

Therefore, besides a through analysis of the code, comparing the results of the algorithm using the 9 different SRE 2010 conditions to a results of the algorithm of a vetted system using the same conditions completed the validation. Below is a Figure containing a table of these results:

	Cond1 (46,923)	Cond2 (219,842)	Cond3 (58,043)	Cond4 (85,902)	Cond5 (30,373)	Cond6 (28,672)	Cond7 (28,356)	Cond8 (28,604)	Cond9 (27,520)
i-vectors (12 MFCCs, 512 GC)	6.5%	14.5%	10.9%	11.5%	12.2%	16.1%	16.6%	5.3%	8.2%
i-vectors BEST (12MFCCs, 512 GC)	7.6%	14.8%	14.5%	12.2%	13.8%	17.7%	18.0%	6.0%	7.8%

Figure 26: Table of results comparing results of the matlab implemented BCDM algorithm (top) compared to the results of the vetted BCDM algorithm (bottom)

5.4 Linear Discriminant Analysis

The LDA algorithm was validated by looking at different classes in 2 dimensional space and 3 dimensional space, as displayed in Figure 27 and Figure 28.

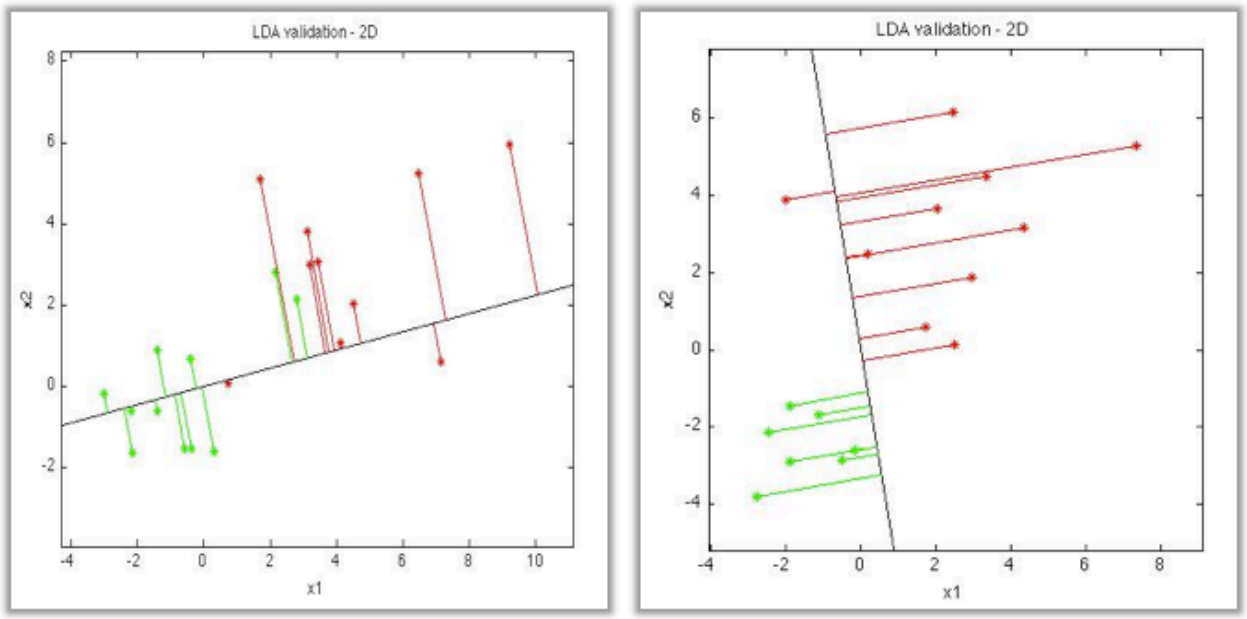

Figure 27: 2 dimensional visual validation of LDA algorithm

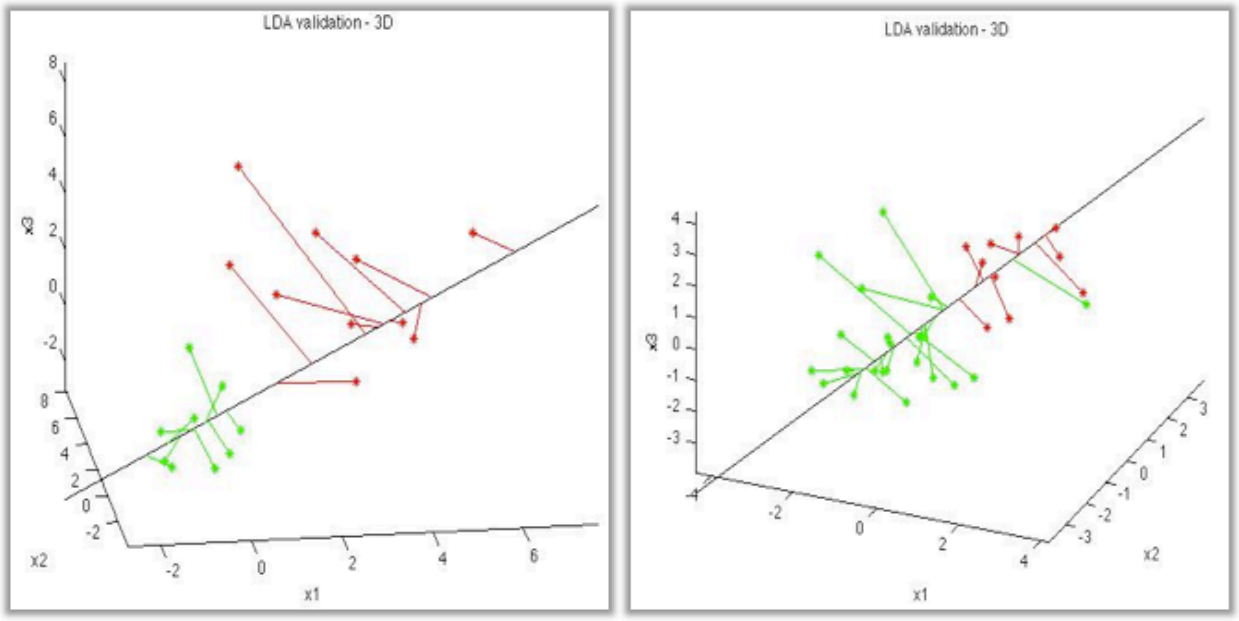

Figure 28: 3 dimensional visual validation of LDA algorithm

6 Testing

6.1 Testing using the TIMIT database

Initial testing was completed using the TIMIT database. Two-thirds of the speakers were used as background speakers to train the UBM, the total variability space and the LDA reduced subspace while the remaining one-third was used for testing. The first set of analysis was completed using the MFCC algorithm with 12 cepstral coefficients + 1 energy feature (c_0), the VAD, and the GMM model with 512 components, the Detection Error Tradeoff (DET) curve is displayed with the results of the log-likelihood classifier in Figure 29.

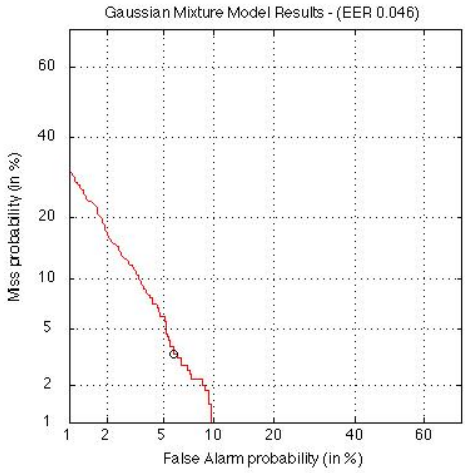

Figure 29: 12 unnormalized MFCCs + 1 energy feature (c_0), the VAD, and the GMM model with 512 components, the Detection Error Tradeoff (DET) curve is displayed with the results of the log-likelihood classifier

The features used to create the above DET curve were un-normalized which typically is not good practice but are reported here to remain consistent with the previous report. Below are the DET curves for the Matlab code (compared to the LL code in section 4). 12 normalized MFCCs were used, along with a VAD and the GMM model had 512 Gaussian components with results of the log-likelihood classifier.

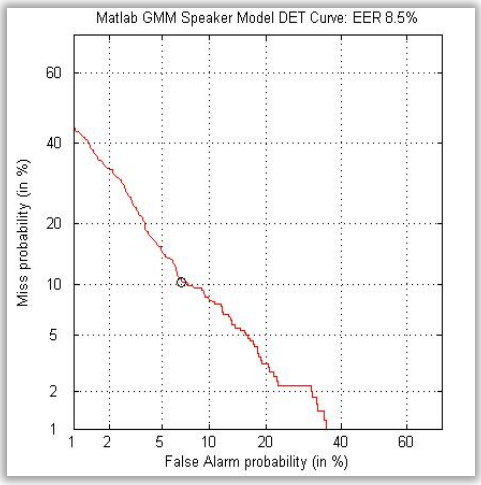

Figure 30: 12 normalized MFCCs, the VAD, and the GMM model with 512 components, the Detection Error Tradeoff (DET) curve is displayed with the results of the log-likelihood classifier

The i-vector speaker models and LDA reduced i-vector speaker models were created and tested using cosine distance scoring. The results were not always as expected. The EERs and DET curves were analyzed after various iterations of the BCDM algorithm for the Factor Analysis model and sometimes the earlier iterations obtained better scores than later iterations. The LDA reduced i-vectors also led to

inconsistent results. After analysis, it was determined that because the TIMIT dataset consisted of only a few utterances of only a few seconds long there was not enough information for each speaker to obtain statistically significant results for the higher level speaker models that rely heavily on statistical information.

6.2 Testing using the SRE databases

The main testing results for this project were completed on the SRE databases. Approximately 1,400 different speakers were used from the SRE 2004, SRE 2005 and SRE 2006 databases as background speakers with over 16,000 wav files (derived from the sph files) in order to train the UBM, the total variability space and the LDA reduced subspace. The background training results from these background speakers were used in generating the speaker models for the reference and test speakers within the SRE 2010 database. The SRE 2010 database consisted of over 20,000 files that were tested in 9 different conditions (see Section 4 for more details). Figure 31 displays the results for Condition 1 using the GMM speaker models with 12 normalized MFCCs and 512 Gaussian components. The histograms on the right display the distribution of the scores for the speakers that match the speaker model (true speakers) and the distribution of scores for the speakers that do not match the speaker model (imposter speakers). Given a specific threshold, the percentage of bins below the threshold in the true speaker histogram represents the miss probability and the percentage of bins above the threshold in the imposter speaker histogram represents the false alarm probability. In general, it is desired that enough data is obtained so that the distributions of the true speakers and the imposter speakers are normal. It is also desired that the means of the two distributions be as different as possible.

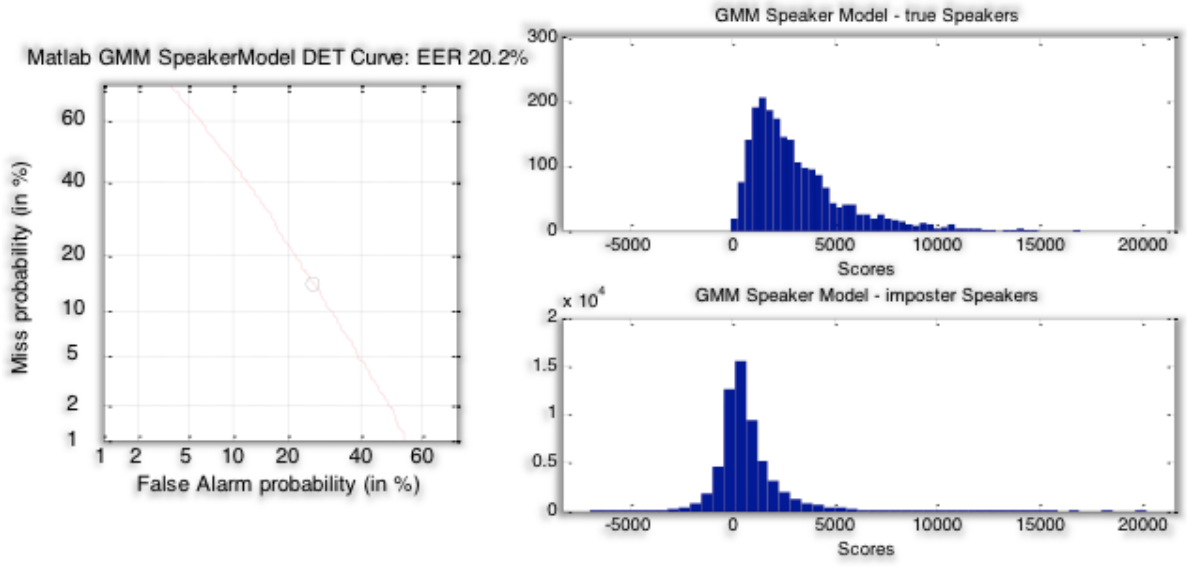

Figure 31: 12 normalized MFCCs, the VAD, and the GMM model with 512 components, the Detection Error Tradeoff (DET) curve is displayed with the results of the log-likelihood classifier along with the histograms of the true and imposter speakers.

There are different ways in which the DET curve and the EER can obtain less favorable results. It is possible that for any given speaker model, the true speaker has the best score compared to all the imposter speakers and that the scores between different speaker models do not align. It is also possible that there are a high number of "wrong" counts, where a "wrong" count is defined to be a case where imposter speaker score is higher than the true speaker score for a given speaker model. Therefore, plotting the scores per speaker model can be helpful in determining the cause of the given performance. This plot is shown for the GMM speaker model below:

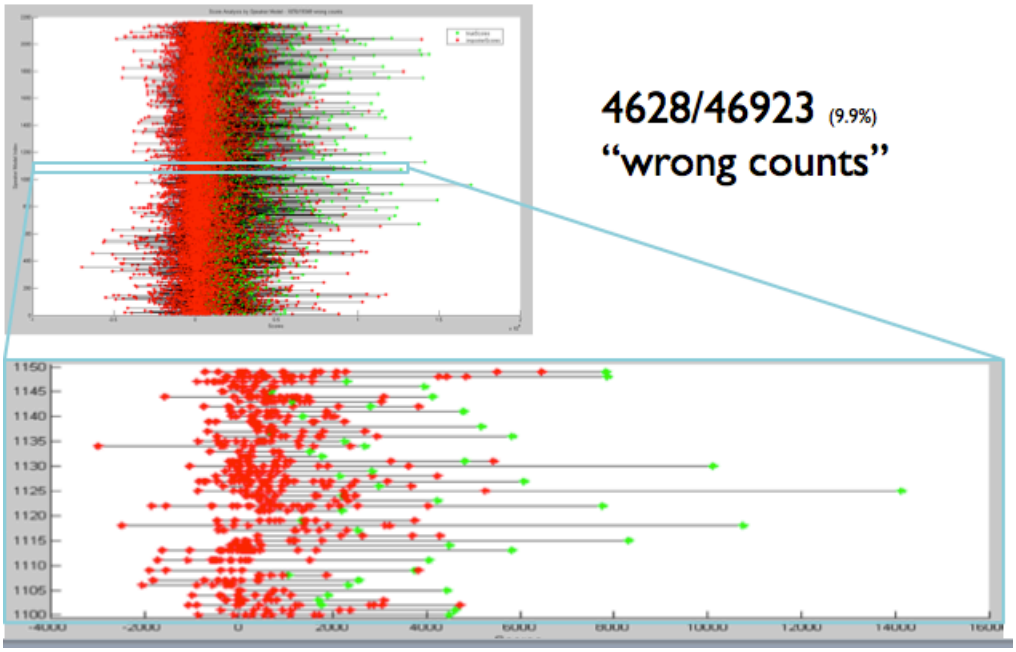

Figure 32: Score analysis by speaker model (same setup as Figure 31)

Using the same features as the GMM speaker model, the EER of the Factor Analysis i-vector model for SRE 2010 Condition 1 is 6.5% as displayed in Figure 33. The distributions of the true speakers and imposter speakers for the i-vector model look more normal and the means are definitely more separated than that of the GMM speaker model case as desired.

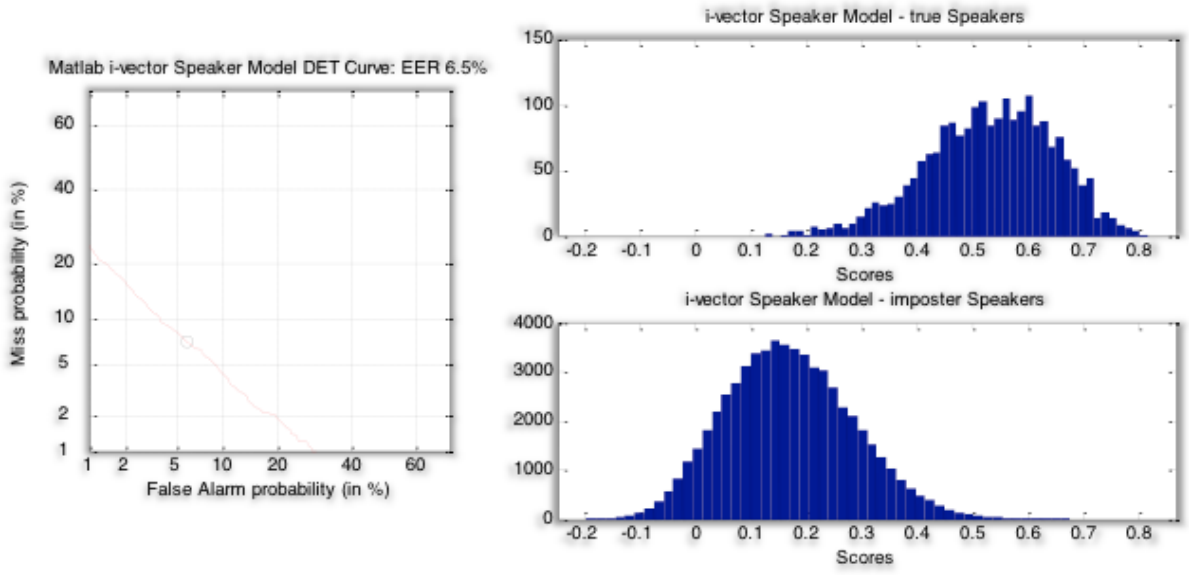

Figure 33: 12 normalized MFCCs, the VAD, and the i-vector model with 512 Gaussian components, the Detection Error Tradeoff (DET) curve is displayed with the results of the cosine distance scoring classifier along with the histograms of the true and imposter speakers.

As displayed in Figure 34, the number of wrong counts also significantly decreases with the factor analysis i-vector speaker models.

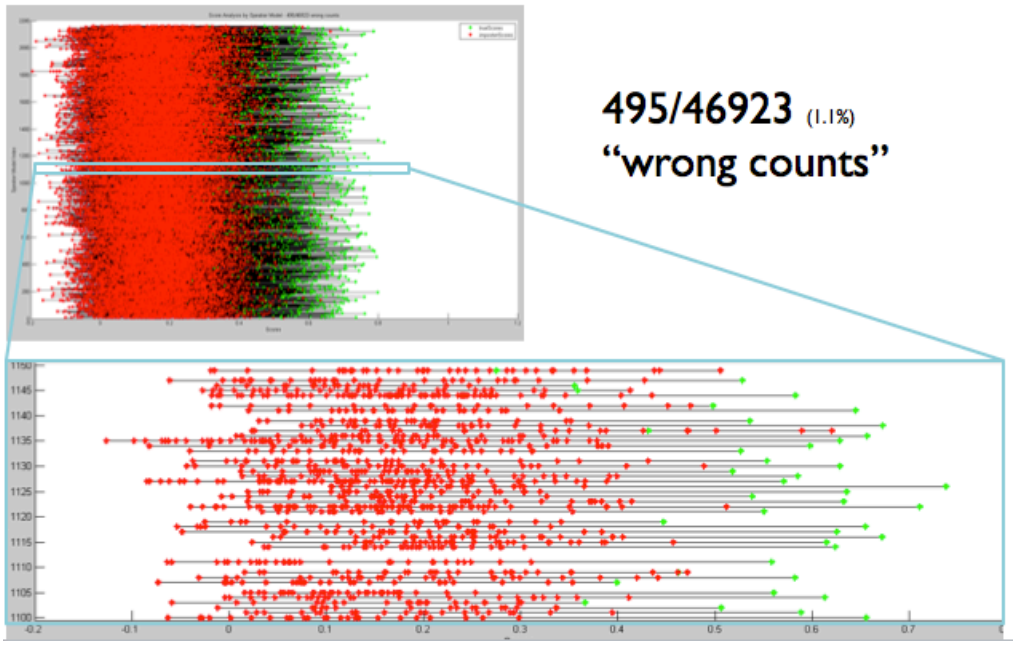

Figure 34: Score analysis by speaker model (same setup as Figure 33)

:

Using LDA to minimize the within speaker covariance and maximize the between speaker covariance of the i-vector models to create the LDA reduced i-vectors also improves results, as seen in Figures 35 and 36.

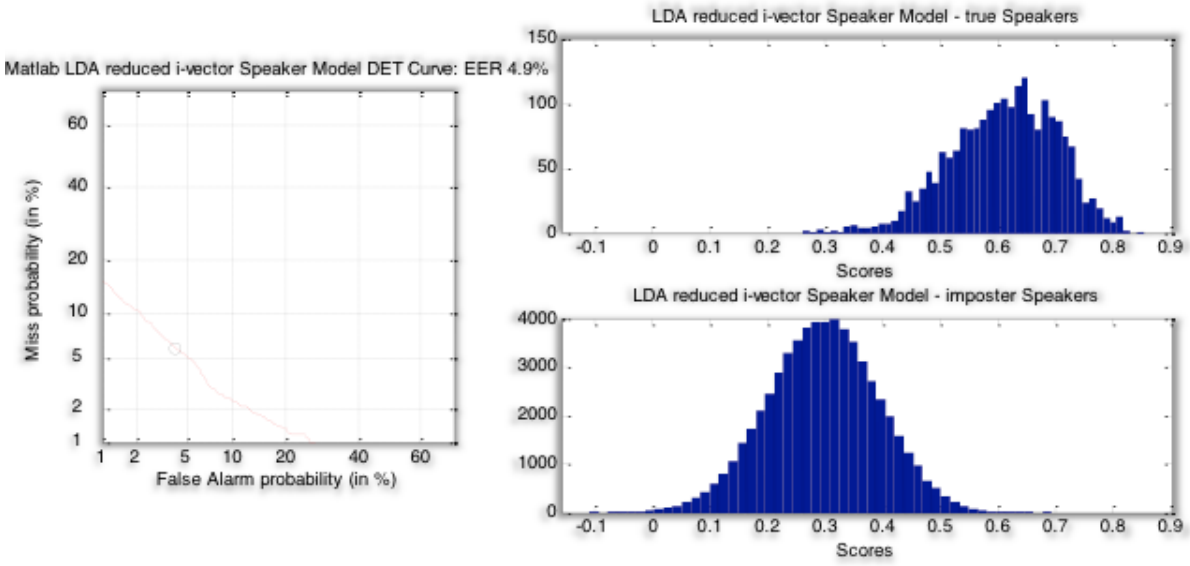

Figure 35: 12 normalized MFCCs, the VAD, and the LDA reduced i-vector model with 512 Gaussian components, the Detection Error Tradeoff (DET) curve is displayed with the results of the cosine distance scoring classifier along with the histograms of the true and imposter speakers.

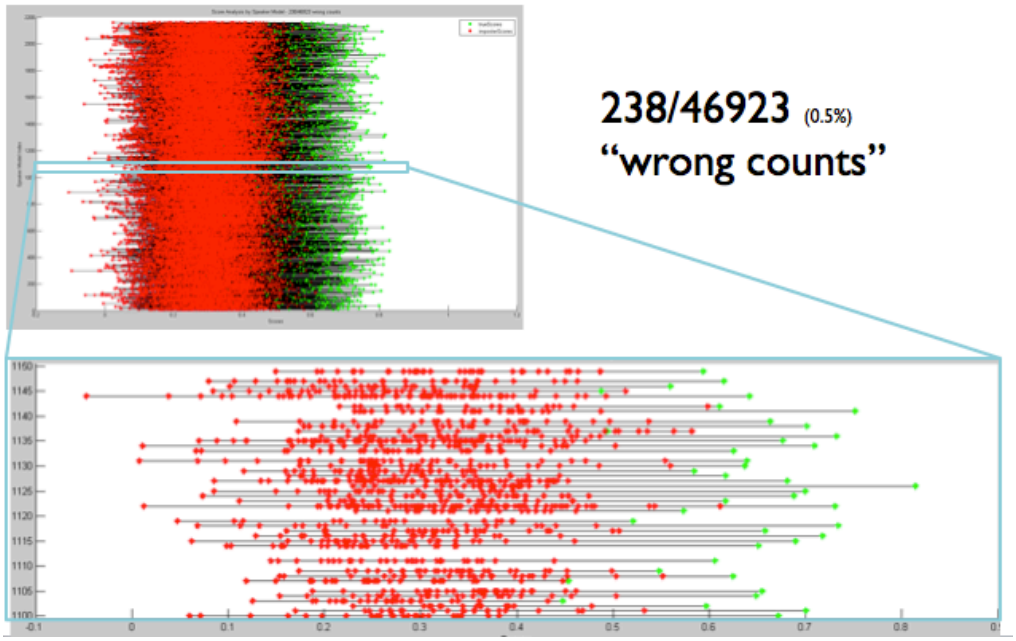

Figure 36: Score analysis by speaker model (same setup as Figure 35)

Overall, it is shown that the Factor analysis i-vectors obtain significant improvement over the GMM speaker models and the LDA reduced i-vectors are even better. In addition, the i-vectors and the LDA reduced i-vectors can be scored much faster. Table XX displays the results of the FA i-vectors and LDA reduced i-vectors in the 9 different conditions for SRE 2010. It should be noted that 10 iterations of the EM algorithm for training UBM is typically used, but due to timing constraints, only 2 iterations were completed for the 57 MFCC and 1024 Gaussian component (GC) case. Also, typically 10 iterations of the Factor Analysis model are used but only 7 were used for the 57 MFCC and 1024 GC case. This may be the reason for the degradation in performance when there was an increase in the number of features, although it may be because the higher dimensional feature space leads to worse performance in certain cases. More analysis will have to take place in order to determine the reasoning. More often than not, the increased number of features led to better performance, as expected.

	Cond1 (46,923)	Cond2 (219,842)	Cond3 (58,043)	Cond4 (85,902)	Cond5 (30,373)	Cond6 (28,672)	Cond7 (28,356)	Cond8 (28,604)	Cond9 (27,520)
i-vectors (12 MFCCs, 512 GC)	6.5%	14.5%	10.9%	11.5%	12.2%	16.1%	16.6%	5.3%	8.2%
LDA reduced i-vectors (12 MFCCs, 512 GC)	4.9%	9.9%	10.1%	7.7%	10.7%	16.2%	14.4%	5.4%	4.9%
i-vectors (57 MFCCs, 1024 GC)	8.1%	17.2%	10.7%	14.4%	10.5%	16.7%	17.0%	3.6%	8.5%
LDA reduced i-vectors (57 MFCCs, 1024 GC)	4.6%	8.5%	8.5%	7.8%	8.8%	14.5%	15.1%	3.4%	3.5%

Figure 37: Table of EERs for different models on the 9 different conditions in the SRE 2010 database

7 Project Schedule

Fall 2011

Phase I: ~(5 weeks)

Aug. 29 – Sept. 28 ~(4 weeks)
- ✓ Read a variety of Text-Independent Speaker Identification papers to obtain an understanding of the proposed project

Sept. 28 – Oct. 4 ~(1 week)
- ✓ Write proposal and prepare for class presentation

Phase II: ~(4 weeks)

Oct. 5 – Oct. 21 ~(2 weeks)
- ✓ Be able to extract MFCCs from speech data and apply simple VAD algorithm
- ✓ Understand SRE databases

Oct. 22 – Nov. 4
- ✓ Develop EM algorithm to trained UBM

~(2 weeks)	✓ Add MAP algorithm to create speaker models ✓ Add likelihood ratio test as a classifier ✓ Validate results using likelihood ratio test as classifier with EER and DET curves, bug fix when necessary
Phase III:	***~(5 weeks)***
Nov. 5 – Dec. 2 ~(3 weeks + Thanksgiving Break)	✓ Create supervectors from GMMs ✓ Write code to train total variability space ✓ Add ability to extract i-vectors from the total variability space ✓ Add cosine distance scoring (CDS) as a classifier ✓ Validate results using the CDS classifier with EER and DET curves, bug fix when necessary
Dec. 3 – Dec. 9 ~(1 week) *overlap*	✓ Prepare Project Progress Report
Dec. 3 – Dec. 19 ~(2 week) *overlap*	✓ Implement LDA on the i-vectors ✓ Validate results using the CDS classifier with EER and DET curves, bug fix when necessary

Spring 2012

Phase IV:	***~(4 weeks)***
Jan. 25 – Feb. 24 ~(4 weeks)	✓ Obtain familiarity with vetted a speaker recognition system ✓ Test algorithms of Phase II and Phase III on several different conditions and compare against results of vetted system ✓ Bug fix when necessary
Phase V	***~(7 weeks)***
Feb. 25 – Mar. 2 ~(1 week) *overlap*	✓ Create plan for rest of semester: (1) Make EM-GMM code usable when a large number of features and a large number of Gaussian Components are being used (2) Finish validating FA code (3) Attempt to make GMM Speaker Model scoring faster (4) Understand interesting behaviors in i-vector and LDA reduced i-vector speaker models for TIMIT database (5) Test code in all 9 SRE 2010 condition for i-vector and LDA reduced i-vector speaker models ✓ Read appropriate background material to make decision
Feb. 25 – Mar. 2 ~(1 week) *overlap*	✓ Work on Project Status Presentation
Mar. 3 – Apr. 20 ~(6 weeks + Spring Break)	✓ Update Schedule to reflect decision made in Phase IV ✓ (1) Make EM-GMM code usable when a large number of features and a large number of Gaussian Components are being used ✓ (2) Finish validating FA code ✓ (3) Attempt to make GMM Speaker Model scoring faster ✓ (4) Understand interesting behaviors in i-vector and LDA reduced i-vector speaker models for TIMIT database ✓ (5) Test code in all 9 SRE 2010 condition for i-vector and LDA reduced i-vector speaker models
Phase VI:	***~(3 weeks)***
Apr. 21 – May 10 ~(3 weeks)	✓ Create final report and prepare for final presentation

8 Milestones

Fall 2011

October 4	✓ Have a good general understanding on the full project and have proposal completed. Present proposal in class by this date. o *Marks completion of Phase I*
November 4	✓ Validation of system based on supervectors generated by the EM and MAP algorithms o *Marks completion of Phase II*
December 19	✓ Validation of system based on extracted i-vectors ✓ Validation of system based on nuisance-compensated i-vectors from LDA ✓ Mid-Year Project Progress Report completed. Present in class by this date. *Marks completion of Phase III*

Spring 2012

Feb. 25	✓ Testing algorithms from Phase II and Phase III will be completed and compared against results of vetted system. Will be familiar with vetted Speaker Recognition System by this time. o *Marks completion of Phase IV*
March 18	✓ Decision made on next step in project. Schedule updated and present status update in class by this date.
April 20	✓ Completion of all tasks for project. o *Marks completion of Phase V*
May 10	✓ Final Report completed. Present in class by this date. o *Marks completion of Phase VI*

9 Deliverables

- ✓ A fully validated and complete Matlab implementation of a speaker recognition system was delivered with at least two classification algorithms.
- ✓ Both a mid-year progress report and a final report were delivered which included validation and test results.

10 Bibliography

[1]*Biometrics.gov - Home*. Web. 02 Oct. 2011. <http://www.biometrics.gov/>.

[2] Kinnunen, Tomi, and Haizhou Li. "An Overview of Text-independent Speaker Recognition: From Features to Supervectors." *Speech Communication* 52.1 (2010): 12-40. Print.

[3] Ellis, Daniel. "An introduction to signal processing for speech." *The Handbook of Phonetic Science,* ed. Hardcastle and Laver, 2nd ed., 2009.

[4] Reynolds, D. "Speaker Verification Using Adapted Gaussian Mixture Models." *Digital Signal Processing* 10.1-3 (2000): 19-41. Print.

[5] Reynolds, Douglas A., and Richard C. Rose. "Robust Text-independent Speaker Identification Using Gaussian Mixture Speaker Models." *IEEE Transations on Speech and Audio Processing* IEEE 3.1 (1995): 72-83. Print.

[6] "Factor Analysis." *Wikipedia, the Free Encyclopedia*. Web. 03 Oct. 2011. <http://en.wikipedia.org/wiki/Factor_analysis>.

[7] Dehak, Najim, and Dehak, Reda. "Support Vector Machines versus Fast Scoring in the Low-Dimensional Total Variability Space for Speaker Verification." *Interspeech 2009 Brighton.* 1559-1562.

[8] Kenny, Patrick, Pierre Ouellet, Najim Dehak, Vishwa Gupta, and Pierre Dumouchel. "A Study of Interspeaker Variability in Speaker Verification." *IEEE Transactions on Audio, Speech, and Language Processing* 16.5 (2008): 980-88. Print.

[9] Lei, Howard. "Joint Factor Analysis (JFA) and i-vector Tutorial." *ICSI. Web. 02 Oct. 2011.* http://www.icsi.berkeley.edu/Speech/presentations/AFRL_ICSI_visit2_JFA_tutorial_icsitalk.pdf

[10] Kenny, P., G. Boulianne, and P. Dumouchel. "Eigenvoice Modeling with Sparse Training Data." *IEEE Transactions on Speech and Audio Processing* 13.3 (2005): 345-54. Print.

[11] Bishop, Christopher M. "4.1.6 Fisher's Discriminant for Multiple Classes." *Pattern Recognition and Machine Learning*. New York: Springer, 2006. Print.

[12] Ellis, Daniel P. W. *PLP and RASTA (and MFCC, and Inversion) in Matlab*. *PLP and RASTA (and MFCC, and Inversion) in Matlab*. Vers. Ellis05-rastamat. 2005. Web. 1 Oct. 2011. <http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/>.

[13] Tipping, Michael E., and Christopher M. Bishop. "Probabilistic Principal Component Analysis." *Journal of the Royal Statistical Society* B 3 (1999): 611-22. Print.

[14] Duda, Richard O., Peter E. Hart, and David G. Stork. *Pattern Classification*. New York: Wiley, 2001. Print.

[15] Shum, Stephen. *Unsupervised Methods for Speaker Diarization*. Thesis. Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2011. Print.

[16] Garcia-Romero, Daniel., Espy-Wilson, Carol Y. "Joint Factor Analysis for Speaker Recognition Reinterpreted as Signal Coding Using Overcomplete Dictionaries." *Odyssey 2010: The Speaker and Language Recognition Workshop* (2010). Print.

[17] D'Souza, Aaron. "Derivation of Maximum Likelihood Factor Analysis Using EM." *www-clmc.usc.edu/* Web. 21 Oct. 2011. <www-clmc.usc.edu/~cs599_an/factor_analysis.pdf>.

[18] " DET-Curve Plotting software for use with MATLAB." *NIST Multimodal Information Group Website*. Web. 15 May 2012. <http://www.itl.nist.gov/iad/mig/tools/DETware_v2.1.tar.gz >