

Efficient Language Identification using Anchor Models and Support Vector Machines

Elad Noor¹, Hagai Aronowitz^{2,3}

¹The Weizmann Institute of Science, Rehovot, Israel

²Department of Computer Science, Bar-Ilan University, Israel

³IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

elad.noor@gmail.com, haronow@us.ibm.com

Abstract

Anchor models have been recently shown to be useful for speaker identification and speaker indexing. The advantage of the anchor model representation of a speech utterance is its compactness (relative to the original size of the utterance) which is achieved with only a small loss of speaker-relevant information. This paper shows that speaker-specific anchor model representation can be used for language identification as well, when combined with support vector machines for doing the classification, and achieve state-of-the-art identification performance. On the NIST-2003 Language Identification task, it has reached an equal error rate of 4.8% for 30 second test utterances.

1. Introduction

Language identification (LID) systems typically try to extract high-end phonetic information from spoken utterances, and use it to discriminate among a closed set of languages. The best known method for this is PPRLM (Parallel Phone Recognition and Language Modeling) [1] which has been quite successful. In PPRLM, a set of phone recognizers are used to produce multiple phone sequences (one for each recognizer), which are later scored using n-gram language models. Lately, there has been a large improvement with Gaussian Mixture Model (GMM) based techniques, due to the introduction of Shifted Delta Cepstra (SDC) features [2] [3]. SDC are derived from the cepstrum, over a long span of time-frames, and this enables the frame-independent GMM to model long time-scale phenomena, which are likely to be significant for identifying languages. The advantage of the second method is that it requires much less computational resources.

It is desirable for a language identification system to be speaker independent. A common way to do this is to mix together a large set of utterances from various speakers for training the language model, and thus minimize the speaker dependency. However, this approach is suboptimal because there is only a single speaker in each test scenario, so it is unlikely that the distribution of features will match well to the multi-speaker GMM, even for the same language.

The present work proposes a more robust way for achieving speaker independence, without changing this basic scheme of SDC and GMM. Instead of using a single GMM for each language (or 2 in the case of gender-dependent models), a GMM is trained for *every* speaker in the language database. The number of these models can be hundreds of times more than the number of languages. Each test utterance is compared to each one of these models, and the results are stored in a vector called the

Speaker Characterization Vector (SCV, see section 3). The set of GMMs is usually referred to as anchor models, when used for speaker recognition, speaker indexing, and speaker clustering [4] [5]. In this paper, the entire CallFriend database has been used for training the anchor models. Note that although language and gender labels are given for all conversations in CallFriend, they were discarded and not actually used for anchor modeling. The only requirement is that there should be enough anchors for each language that is to be identified.

The SCVs are a compact and fixed-length representation of the original utterance, and therefore it is much easier to apply standard normalization methods on them. Two papers [6] and [7], which are aimed at speaker verification, use similar representations and propose methods of modeling the intra-speaker inter-session variability. In this paper, discriminative methods are used to compensate for the intra-language inter-speaker variability, and to identify the language. A linear-kernel Support Vector Machine (SVM) is trained for discriminating between the target language's SCVs, and the SCVs from the non-target languages. For this purpose, the NIST-2003 language development data was used, where labels were provided for 12 languages. These were the only language labels used for training.

As in the case of using anchor models for speaker recognition, any speaker distance measure can be used for producing the SCV. In this work, first GMM-UBM likelihood ratio scores were used [8], with SDC as the frontend feature. Later, the GMM scoring method was substituted with an extremely efficient approximation called Test Utterance Parameterizations [9].

The organization of the remainder of this paper is as follows: Section 2 describes the corpora and evaluation methods used in the LID experiments. Section 3 presents a detailed description of the combined anchor model and support vector machine (AM-SVM) system and introduces a few alternative implementations for it. Section 4 analyzes the time complexity of some of the methods, and suggests a way to speed-up the algorithms. Section 5 presents the performance of the system on NIST language recognition evaluations and compares it to a GMM-SVM baseline and Section 6 follows with a brief summary and proposal for future work.

2. Corpora and evaluation methods

There are 12 target languages in all the corpora used in this study: Arabic, English, Farsi, French, German, Hindi, Japanese, Korean, Mandarin, Spanish, Tamil and Vietnamese. Additional utterances in Russian were introduced only in the test data.

2.1. LDC CallFriend

Anchor models training data was obtained from the Linguistic Data Consortium CallFriend corpus (development, train and test sets). Each set consists of 20 two-sided conversations from each language, approximately 30 minutes long. There are 13 languages (Mandarin was divided to Mainland and Taiwan dialects). This sums up to a total of 1560 utterances (in 780 hours of speech).

2.2. NIST LRE-03

In this paper, the experiments are done using the NIST 2003 language recognition evaluations (LRE-03) [10]. The development data was taken from NIST LRE-96 [11], and consists of the lid96d1 and lid96e1 sets, in the 12 basic languages. The task was to recognize the language that was used in each test utterance, out of a 13-language set (Russian added for out-of-set evaluations). The durations of the test utterances are 3, 10 and 30 seconds. There are 1280 test utterances for each duration (80 from each language except 160 in Japanese and 240 in English).

2.3. Universal background model

The feature vectors used for the anchor models (and in any other method mentioned in this paper) were SDC with 7-1-3-7 parameter configuration (a 49-dimensional feature vector). This configuration was chosen based on the results in [3]. The UBM was trained from 700 utterances of male and female speakers from Cellular phones, which were randomly obtained from NIST SRE-01. The covariance matrix for each Gaussian in the GMM was diagonal. Multiple models of several sizes were produced (256/512/1024/2048 Gaussians).

2.4. Evaluation methods

Reported scores are given in the form of Detection Error Trade-off (DET) [12] curves and equal error rates (EER). In the case of multiple language recognition task, these scores are computed by first pooling the entire set of scores from all languages together and then creating the DET curve.

3. Algorithms

3.1. Anchor models

This is a technique usually used for speaker ID. The anchors are a predetermined set of speakers, that is non-intersecting with the set of target speakers in the test utterances. This study uses the GMM-UBM framework [8] for modeling these speakers. The anchor models are trained in advance, and denoted λ_e , $e = 1, \dots, E$. Each new utterance, X , is projected into the anchor speaker space, using the average log likelihood ratio of X for each anchor model relative to the UBM (λ_{UBM}). The result, $A(X) \in \mathbb{R}^E$, is called the Speaker Characterization Vector (SCV).

$$\hat{s}(X|\lambda_e) = \frac{1}{F} \log \frac{p(X|\lambda_e)}{p(X|\lambda_{UBM})} \quad (1)$$

$$A(X) = \begin{bmatrix} \hat{s}(X|\lambda_1) \\ \hat{s}(X|\lambda_2) \\ \vdots \\ \hat{s}(X|\lambda_E) \end{bmatrix} \quad (2)$$

where F is the number of acoustic feature vectors in X . This projection is used for both the speech utterances of the known target speakers and the unknown test segments. The identifica-

tion is done by calculating the distance between target and test characterization vectors and comparing to a threshold. Possible distance measures are Euclidean, absolute value, Kullback-Leibler and angle. A probabilistic approach has been proposed in [5] where each speaker is represented by a Gaussian distribution in the anchor space.

In this study, we use a similar scheme for language identification. For training the anchor models, the UBM of size 512 Gaussians was chosen, since larger GMMs require more computational resources. 1560 anchor GMM models ($E = 1560$) were trained with MAP adaptation, using the CallFriend corpus. The weights, means and variances were all adapted with a relevance factor of ($r = 16$) [8]. Each utterance in the development and test sets of NIST LRE-03, was projected to the anchor speaker space using SCV, exactly like described in [5]. However, the identification phase was different and was done using an SVM.

3.2. Support vector machine

A support vector machine is a two-class classifier. For a given *kernel* function $K(\cdot, \cdot)$, it is described by the function:

$$f(x) = \sum_{i=1}^T \alpha_i y_i K(x, x_i) + b \quad (3)$$

where $x \in \mathbb{R}^D$ is the D -dimensional input vector of a test example. $\{x_i\}_{i=1}^T$ are the support vectors that are obtained from the training set by an optimization process [13], $y_i \in \{-1, 1\}$ are their associated class labels. $\alpha_i > 0$ and b are the parameters of the model, and $\sum_{i=1}^T \alpha_i y_i = 0$. Decision is made according to a threshold for $f(x)$, specifically $y = \text{sign}(f(x))$.

In order to make the two-class SVM into a multi-class language identifier, one SVM was trained for each language, using SVMtorch [13]. SCVs of the target language were used as positive examples and the other SCVs were used as negative examples. The kernel used in this research was the standard inner-product kernel function $K(x, x_i) = x^t x_i$.

When evaluating a new test SCV, the raw scores were taken one-by-one for each SVM without applying the *sign* function, in order to have a confidence measure. Later, the scores were converted to log-likelihood ratios:

$$s'_i = s_i - \log \left(\frac{1}{L} \sum_{j=1}^L e^{s_j} \right); i = 1 \dots L \quad (4)$$

where s_i is the SVM score for language i . L is the number of target languages. This type of score normalization was found to be useful, although the SVM scores are not log-likelihood values.

3.3. Test utterance parameterization

In [9] [14] a new speaker recognition technique was presented. The idea is to train GMMs not only for target speakers but also for the test sessions. The likelihood of a test utterance is approximated using only the GMM of the target speaker and the GMM of the test utterance. This technique of representing a test session by a GMM was named test utterance parameterization (TUP) and the technique of approximating the GMM algorithm using TUP was named GMM-simulation. This can be used for greatly reducing the time complexity of evaluating the likelihood score of an utterance according to a large set of GMMs, which is required for computing the SCVs.

3.3.1. Simplified GMM-simulation

In this paper, a simplified form of TUP was used. The MAP adaptation process for GMM training was done only on the Gaussians' means, while the weights and covariance matrices were copied from the UBM. The GMM-simulation algorithm in this case is:

1. Estimate GMM Q for target speaker.
2. Estimate GMM P for test session.
3. Compute score $S = S(P, Q)$ using top- N pruning.
4. Normalize score using TZ-norm.

When using top- N pruning, each Gaussian in P is compared to a set of the top- N closest Gaussian in Q , which is selected in advance. The most simple case of $N = 1$ was used. Since both GMMs were adapted from the same UBM, it was assumed that $\text{top-1}(i) = \{i\}$. This was based on the general assumption that even after MAP adaptation, each Gaussian does not change drastically, and stays close to the original Gaussian in the UBM. The score function that approximates the classic GMM algorithm is therefore:

$$S(P, Q) = \sum_{g=1}^G w_g \left(\log w_g - \sum_{d=1}^D \frac{(\mu_{g,d}^P - \mu_{g,d}^Q)^2}{2\sigma_{g,d}^2} \right) + C \quad (5)$$

- D - dimension of the acoustic features
- G - order of the GMMs
- w_g - weight of the Gaussian g of the UBM distribution
- $\sigma_{g,d}^2$ - element (d, d) of the covariance matrix of Gaussian g of the UBM distribution
- $\mu_{g,d}^P$ - element d of the mean vector of Gaussian g of distribution P
- C - a constant.

Since w_g does not depend on P or Q , we can define $C' = C + \sum_{g=1}^G w_g (\log w_g)$ which is also constant, and $S(P, Q)$ can be rewritten as:

$$S(P, Q) = C' - \sum_{g=1}^G \sum_{d=1}^D \frac{w_g}{2\sigma_{g,d}^2} \cdot (\mu_{g,d}^P - \mu_{g,d}^Q)^2 \quad (6)$$

Defining the supervector $\tilde{P} \in \mathbb{R}^{G \cdot D}$ as the normalized concatenation of all Gaussian means of distribution P

$$\tilde{P}_{g \cdot D + d} = \mu_{g,d}^P \cdot \sqrt{\frac{w_g}{2\sigma_{g,d}^2}} \quad (7)$$

and the same for \tilde{Q} , one can see that

$$S(P, Q) = C' - \|\tilde{P} - \tilde{Q}\|^2 = 2\tilde{P}^t \tilde{Q} - \|\tilde{P}\|^2 - \|\tilde{Q}\|^2 + C' \quad (8)$$

TZ-norm is applied by using the means only (without normalizing the standard deviations). It is justified since it doesn't hurt performance and simplifies the equations. T-norm eliminates the values $\|\tilde{P}\|^2$ and C' , since they do not depend on the target speaker. $\|\tilde{Q}\|^2$ disappears after Z-norm since it does not depend on the test session. The 2 factor can obviously be ignored, therefore GMM-simulation is reduced to a simple inner-product between supervectors.

3.3.2. Anchor supermatrix

Let λ_e be one of the anchor models, X a test session and P_X the GMM trained from X . Let $\tilde{\lambda}_e$ and \tilde{P}_X be the supervectors corresponding to λ_e and P_X as defined in equation (7). Using GMM-simulation, the log-likelihood ratio, $\hat{s}(X|\lambda_e)$ from equation (1), is approximated by:

$$\hat{s}(X|\lambda_e) \approx \tilde{\lambda}_e^t \tilde{P}_X \quad (9)$$

Arranging $\tilde{\lambda}_e$ as columns in a matrix $\tilde{\Lambda}$, produces a $E \times (G \cdot D)$ matrix, called the anchor supermatrix. Defining the approximation $A'(X)$ as follows:

$$A'(X) = \tilde{\Lambda}^t \tilde{P}_X = \begin{bmatrix} \tilde{\lambda}_1^t \tilde{P}_X \\ \tilde{\lambda}_2^t \tilde{P}_X \\ \vdots \\ \tilde{\lambda}_E^t \tilde{P}_X \end{bmatrix} \approx \begin{bmatrix} \hat{s}(X|\lambda_1) \\ \hat{s}(X|\lambda_2) \\ \vdots \\ \hat{s}(X|\lambda_E) \end{bmatrix} = A(X) \quad (10)$$

where $A(X)$ is the SCV, as defined in equation 2. The time complexity of computing $A'(X)$ is low, since the most time-consuming operation is calculating \tilde{P}_X , which is actually training a GMM for X [14]. When using 1560 anchor models with 512 Gaussians each, there is a speedup of about 16 compared to classic UBM-GMM (with top-5 Gaussian pruning [15]). When using 2048 Gaussians, TUP is about 4.8 times faster. A further speed-up factor of about 36 (173 in total) can be achieved using a technique for speeding-up GMM adaptation using a vector quantization arranged in a tree structure for fast categorization of frames and selection of frame-dependent Gaussian short-lists (see subsection 4.1).

3.3.3. TUP-SVM-COV

Applying a kernel function on the approximated SCVs ($A'(X) = \tilde{\Lambda}^t \tilde{P}_X$), we get the following expression:

$$K(A'(X), A'(Y)) = (\tilde{\Lambda}^t \tilde{P}_X)^t (\tilde{\Lambda}^t \tilde{P}_Y) = \tilde{P}_X^t \tilde{\Lambda} \tilde{\Lambda}^t \tilde{P}_Y \quad (11)$$

It is possible to define a new kernel function (K'):

$$K'(\tilde{P}_X, \tilde{P}_Y) = \tilde{P}_X^t (\tilde{\Lambda} \tilde{\Lambda}^t) \tilde{P}_Y = K(A'(X), A'(Y)) \quad (12)$$

Therefore, using linear-kernel SVM classification on the approximated SCVs, is equivalent to using an SVM with the new kernel directly on the supervectors. Note that $(\tilde{\Lambda} \tilde{\Lambda}^t)$ is the covariance matrix of the CallFriend supervectors (assuming their mean is 0). However, it is not efficient to use this representation in practice, since this covariance matrix is of enormous size - $(G \cdot D) \times (G \cdot D)$. This classification method will be denoted TUP-SVM-COV.

3.4. Multiple discriminant analysis

Multiple Discriminant Analysis (MDA) [16] is a natural generalization of Fishers linear discrimination (LDA) in the case of multiple classes. It can be used as a different approach for dealing with the variability of speakers using the same language, by applying a dimension-reducing linear transformation. It assumes a Gaussian distribution for each class (language in this case).

3.4.1. The MDA algorithm

The input to MDA is a set of D -dimensional vectors, $\{X_1, \dots, X_n\} \in \mathbb{R}^D$, and a mapping $l : \{1, \dots, n\} \rightarrow$

$\{1, \dots, L\}$ representing the labels. First, calculate the class means (μ_i):

$$n_i = |\{j : l(j) = i\}| \quad (13)$$

$$\mu_i = \frac{1}{n_i} \sum_{l(j)=i} X_j \quad (14)$$

The global mean is:

$$\mu = \frac{1}{n} \sum_{j=1}^n X_j \quad (15)$$

Define the within-class and between-class scatter matrices (S_W and S_B) as follows:

$$S_W = \sum_{i=1}^L \sum_{l(j)=i} (X_j - \mu_i)(X_j - \mu_i)^t \quad (16)$$

$$S_B = \sum_{i=1}^L n_i (\mu_i - \mu)(\mu_i - \mu)^t \quad (17)$$

Suppose a linear transformation W is applied on the input vectors ($X_j \mapsto W^t X_j$). Then, the new scatter matrices will be $W^t S_W W$ and $W^t S_B W$. The optimal transformation is defined as the one that maximizes the Rayleigh quotient $J(W)$.

$$J(W) = \frac{|W^t S_B W|}{|W^t S_W W|} \quad (18)$$

It can be proven that the columns of an optimal W are the generalized eigenvectors (w_i) that correspond to the largest eigenvalues (λ_i) in:

$$S_B w_i = \lambda_i S_W w_i \quad (19)$$

3.4.2. Block diagonal MDA

The anchor supervectors are used as input for MDA, where the class labels are the CallFriend language labels of these utterances. However, the dimension of the supervectors ($G \cdot D$) is usually very large, and much greater than the number of anchors, resulting in low rank scatter matrices which harms the MDA algorithm. To solve this problem, it is possible to use a block diagonal version of MDA like in [6]. It has been empirically verified that the covariance between GMM mean values corresponding to different features ($COV(\mu_{g_1, d_1}, \mu_{g_2, d_2})$ when $d_1 \neq d_2$) is relatively low. Therefore, it is justified to factor the supervector space into D disjoint subspaces (each one of dimension G) and to apply the MDA algorithm separately to each subspace (reducing the dimension to G'). The result of this algorithm is a block-diagonal transformation matrix W , of size $(G \cdot D) \times (G' \cdot D)$, where $G' < G$.

3.4.3. TUP-SVM-MDA

In order to create a model for language i , one can use the mean vector of the transformed supervectors of that language's utterances. Suppose \mathcal{L}_i is the set of examples for that language from the NIST development data, then the mean (\tilde{F}_i) is defined as:

$$\tilde{F}_i = \frac{1}{|\mathcal{L}_i|} \sum_{X \in \mathcal{L}_i} W^t \tilde{P}_X \quad (20)$$

Scoring a test utterance Y is done by computing the supervector, applying the MDA transformation, and taking the inner product with \tilde{F}_i .

$$s_i = (W^t \tilde{P}_Y)^t \tilde{F}_i = \frac{1}{|\mathcal{L}_i|} \sum_{X \in \mathcal{L}_i} \tilde{P}_Y^t W W^t \tilde{P}_X \quad (21)$$

where $\tilde{P}_X^t(WW^t)\tilde{P}_Y$ is a kernel-type function, and the simple MDA score can be viewed as calculating the score for all the examples from the language in this kernel-space and taking their average. However, one mean vector might not be a sufficient representation for an entire language distribution. In order to support other types of distributions an SVM is trained, this time with a polynomial kernel of degree 2. Comparing to TUP-SVM-COV (subsection 3.3.3), this procedure is equivalent for using the following kernel directly on the supervectors:

$$K''(\tilde{P}_X, \tilde{P}_Y) = (\tilde{P}_X^t(WW^t)\tilde{P}_Y + 1)^2 \quad (22)$$

This classification method will be denoted TUP-SVM-MDA.

3.5. TUP-SVM

For the sake of completeness, a final version of TUP-SVM is implemented, with the simple inner-product kernel. The great advantage of this method is that the data from CallFriend is not used at all. The SVM is trained only on the NIST development supervectors. However, since no reducing transformation is applied, the dimension of these vectors is very high, and therefore requires a large amount of memory. It was possible to use only small GMMs of size $G = 256$, since the SVM-Torch software cannot deal with vectors much larger than $G \cdot D = 12544$.

4. Time complexity analysis

Computing the SCV of anchor models under the UBM-GMM framework involves a very large amount of single Gaussian computations. For an utterance of F frames, the normal likelihood ratio evaluation requires FG Gaussian computations for the UBM likelihood, and FGE for all the anchor scores, i.e. $FG(1 + E)$ in total. A well known way to accelerate this is to first evaluate the top- N Gaussians for each test frame, according to the UBM, and calculate the score of that frame only according to these Gaussians. This requires FG computations for finding the top- N and FEN for the likelihood ratios. Altogether, there are $F(G + EN)$, which is $FE(G - N)$ less computations. This becomes quite substantial for very large E . The default value for such pruning is $N = 5$, and usually doesn't seem to have any significant effect on the scores. All GMM-UBM evaluations in this paper have been done using this pruning method.

By using GMM-simulation, as described in section 3.3, one can reduce further the number of Gaussian computations, to exactly FG . As mentioned earlier, the additional computations needed after performing the TUP are negligible.

4.1. VQ-tree for fast GMM-UBM decoding

This subsection describes a technique for accelerating the process of finding the top- N best scoring Gaussians for a given frame. This process is used by both classic GMM scoring and by MAP-adaptation which is used by the GMM simulation algorithm. The goal of the top- N best scoring process is given a UBM-GMM and a frame, to find the top- N scoring Gaussians. Note that a small percentage of errors may be tolerated, since all the scores along the utterance are averaged, and small effects become negligible. Finding the top- N best scoring Gaussians is usually done by scoring all Gaussians in the UBM-GMM and then finding the N best scores. Our technique introduces an indexing phase in which the Gaussians of the UBM-GMM are examined and associated to different clusters defined by a vector-quantizer. During recognition, every frame is first associated to a single cluster and then only the Gaussians mapped to that cluster

ter are scored. Note that a Gaussian is usually mapped to many clusters. In order to be able to locate the cluster quickly, we design the vector-quantizer to be structured as a tree (VQ-tree) with L leaves.

The VQ-tree is created by a top-bottom leaf-splitting technique, where in each step, the most distorted leaf is split into two leaves using k -means (with $k = 2$) and Mahalanobis distance. The distortion of a leaf is defined as the sum of squared Mahalanobis distances between every vector in the leaf and the center of the leaf. This step is repeated $L - 1$ times, i.e. until the tree has exactly L leaves. After building the tree, a Gaussian g is assigned to the short-list of cluster l (denoted G_l) if and only if the probability for a random feature vector associated to cluster l to have Gaussian g in the top- N Gaussians exceeds ϵ .

$$g \in G_l \Leftrightarrow \Pr_{x \in l} [g \in \text{top}N(x)] > \epsilon \quad (23)$$

It is difficult to compute this probability, therefore it is estimated from the training data by creating a G by L histogram. For each feature vector $x \in l$, compute the top- N scoring Gaussians, and in column l add 1 to each row in the histogram corresponding to these Gaussians. Then normalize each column by its sum. All cells in column l with values higher than ϵ will be in G_l .

For finding the top- N scoring Gaussians for a new frame, first find its cluster in the VQ-tree, using hierarchical search, then compute the score of only the Gaussians in the short-list G_l . Use the top- N Gaussians out of this list. The time complexity of the search is $D + S$, where the expected depth of the VQ-tree is $D = O(\log L)$ and the expected size of the Gaussian short-list is S . The original complexity is G , therefore the speedup is $G/(D + S)$.

A VQ-tree with 10,000 leaves was trained on the SPIDRE corpus. ϵ was set to 0.0001. For a UBM-GMM of 2048 Gaussians the average size of a Gaussian short-list was 40. The expected depth of a leaf in the VQ-tree was 17. The effective speedup factor is therefore 36. On the NIST-2004 SRE, no degradation in accuracy was observed.

4.2. Time complexity comparison

The different speedup methods are compared in Table 1, where the baseline is the common top- N pruning. The parameters of the systems are $G = 2048$, $E = 1560$, $N = 5$ and $D + S = 57$.

Table 1: Comparison of time complexity

method	time complexity	speedup
No pruning	$FG(1 + E)$	x 0.003
top- N pruning	$F(G + EN)$	x 1
VQ-tree	$F(D + S + EN)$	x 1.25
TUP	FG	x 4.8
TUP + VQ-tree	$F(D + S)$	x 173

It should be noted, that the speedup is calculated only for anchor model scoring, which is by far the most significant computational part of the baseline system. However, the other stages (feature extraction, SVM scoring) become significant when the speedup is large enough.

5. Results

Five types of experiments were conducted. The acoustic features for all of them were identical. The first, referred

to as *Language GMM*, is considered the baseline. Part of the CallFriend data was used to train 2 background models (for male and female speakers). Then 2 gender-dependent GMM models were trained for each language using MAP adaptation, to make a total of 26 models. An SVM back-end was trained with the log-likelihood ratio scores of these models for each utterance in the development set. The other systems are *Anchor GMM* which are the UBM-GMM anchor models followed by an SVM; *Anchor TUP* which is the same but with GMM-simulation (also denoted TUP-SVM-COV, see subsection 3.3.3); *TUP-SVM-MDA* and *TUP-SVM*. Table 2 gives the equal error rates of each system for 30sec, 10sec and 3sec utterances on the NIST LRE-03 test set, and the order of the UBM used in that system.

Table 2: EER (%) performance on the LRE-03 test set for NIST's primary condition

system	GMM size	30s	10s	3s
Language GMM	2048	7.4	13.8	27.8
Anchor GMM	512	4.8	12.3	27.0
Anchor TUP	2048	4.7	13.2	33.5
TUP-SVM-MDA	1024	6.7	15.6	30.8
TUP-SVM	256	8.9	-	-

The DET curve for the first 3 systems, for the 30sec test, is given in figure 1.

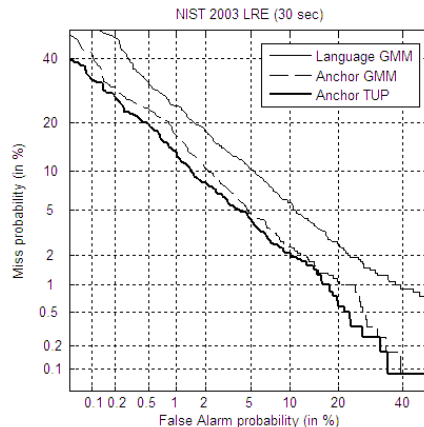


Figure 1: results on NIST LRE-03 (30s) test

Another way to measure accuracy is a matrix of identification confusion error-rates. Table 3 shows the confusion rates for the best performing system (*Anchor TUP*) on the 30sec task. The columns in the table correspond to the correct label of each utterance, and the rows correspond to the language identified by the system. Russian conversations were discarded, since they are irrelevant for the identification task (since there was no model for Russian). Out of the total 1200 utterances, there were 114 misclassifications (9.5%).

Table 3: Confusion matrix (columns indicate labels, rows indicate identification decisions)

	Ar	En	Fa	Fr	Ge	Hi	Ja	Ko	Ma	Sp	Ta	Vi
Ar	72	0	0	1	4	0	1	3	0	0	0	0
En	1	228	1	5	6	1	3	0	0	0	1	1
Fa	1	2	76	0	1	1	2	1	0	0	0	0
Fr	2	1	0	70	1	1	0	0	2	0	1	0
Ge	1	3	1	0	66	2	0	1	0	0	0	0
Hi	1	0	1	0	1	67	1	0	0	0	3	0
Ja	0	1	0	0	0	1	133	0	0	0	0	0
Ko	0	0	0	1	0	1	8	73	3	1	0	1
Ma	0	1	0	0	0	1	4	1	73	0	0	0
Sp	1	1	0	1	1	1	6	0	1	78	2	1
Ta	1	0	1	2	0	4	0	1	0	0	73	0
Vi	0	3	0	0	0	0	2	0	1	1	0	77
Total	80	240	80	80	80	80	160	80	80	80	80	80
Error-rate	10.0%	5.0%	5.0%	12.5%	17.5%	16.3%	16.9%	8.8%	8.8%	2.5%	8.8%	3.8%

6. Discussion

In this paper, we have presented a novel language identification system that given utterances in a language, projects them onto a speaker space using anchor modeling, and then uses an SVM to generalize them. One advantage of this method is that very little labeled data is required. The only labels used for training (the SVM) were taken from NIST LRE-03 development data, which consists of about a hundred 30 second utterances per language. This is very helpful for automatic identification of languages that have little human-labeled available examples. A more efficient way to calculate the speaker characterization vectors was proposed, using test utterance parameterization instead of the classic GMM-UBM.

The future work includes further development of the TUP method to be more robust to the duration of the segments. Also, future experiments will be conducted for larger TUP supervectors using the TUP-SVM method, after overcoming the memory issues.

7. Acknowledgements

This research was supported by Muscle, a European network of excellence funded by the EC 6th framework IST programme.

8. References

- [1] M.A. Zissman, "Comparison of four approaches to automatic language identification of telephone speech", *IEEE Trans. Speech and Audio Processing*, Jan. 1996.
- [2] P.A. Torres-Carrasquillo, E. Singer, M.A. Kohler, R.J. Greene, D.A. Reynolds, and J.R. Deller, Jr., "Approaches to language identification using Gaussian mixture models and shifted delta cepstral features", in *Proc. ICSLP 2002*, Sept. 2002, pp. 89-92.
- [3] E. Singer, P. Torres-Carrasquillo, T. Gleason, W. Campbell, and D. Reynolds, "Acoustic, Phonetic, and Discriminative Approaches to Automatic Language Identification", in *Proc. Eurospeech 2003*, Sept. 2003, pp. 1345-1348.
- [4] D. Sturim, D. Reynolds, E. Singer, and J. Campbell, "Speaker Indexing in Large Audio Databases Using Anchor Models", in *Proc. ICASSP 2001*, May 2001, pp. 429-432.
- [5] M. Collet, Y. Mami, D. Charlet, and F. Bimbot, "Probabilistic Anchor Models Approach for Speaker Verification", in *Proc. INTERSPEECH 2005*, Sept. 2005.
- [6] H. Aronowitz, D. Irony, D. Burshtein, "Modeling Intra-Speaker Variability for Speaker Recognition", in *Proc. INTERSPEECH 2005*, Sept. 2005.
- [7] P. Kenny, G. Boulianne, P. Ouellet, P. Dumouchel, "Factor Analysis Simplified", in *Proc. ICASSP 2005*, Mar. 2005.
- [8] D.A. Reynolds, T.F. Quatieri, and R.B. Dunn, "Speaker verification using adapted Gaussian mixture models", *Digital Signal Processing*, Vol. 10, No.1-3, pp. 19-41, 2000.
- [9] H. Aronowitz, D. Burshtein, A. Amir, "Speaker Indexing In Audio Archives Using Test Utterance Gaussian Mixture Modeling", in *Proc. ICSLP, 2004*, Oct. 2004.
- [10] <http://www.nist.gov/speech/tests/lang/2003/index.htm>
- [11] <http://www.nist.gov/speech/tests/lang/1996/index.htm>
- [12] A. Martin et al., "The DET curve in assessment of detection task performance", in *Proc. Eurospeech 1997*, Sept. 1997, pp. 1895-1898.
- [13] R. Collobert, S. Bengio, and J. Mariétoz, "Torch: a modular machine learning software library", Technical Report IDIAP-RR 02-46, IDIAP, 2002.
- [14] H. Aronowitz, D. Burshtein, "Efficient Speaker Identification and Retrieval", in *Proc. INTERSPEECH 2005*, Sept. 2005.
- [15] J. McLaughlin, D. A. Reynolds, and T. Gleason, "A study of computation speed-ups of the GMM-UBM speaker recognition system", in *Proc. Eurospeech 1999*, pp. 1215-1218, Sept. 1999.
- [16] R. Duda and P. Hart, "Pattern Classification and Scene Analysis", New York: Wiley, 1973.