Channel-dependent GMM and Multi-class Logistic Regression models for language recognition

David A. van Leeuwen

TNO Human Factors
Soesterberg, the Netherlands
david.vanleeuwen@tno.nl

Niko Brümmer

Spescom Datavoice
Stellenbosch, South Africa.
nbrummer@za.spescom.com

Abstract

This paper describes two new approaches to spoken language recognition. These were both successfully applied in the NIST 2005 Language Recognition Evaluation. The first approach extends the Gaussian Mixture Model technique with channel dependency, which results in actual detection costs (C_{DET}) of 0.095 in NIST LRE-2005, and which should be compared to a traditional 2-gender dependency of GMM language models achieving 0.120. The second approach is a Multi-class Logistic Regression system, which operates similarly to a Support Vector Machine (SVM), but can be trained for all languages simultaneously. This new approach resulted in a C_{DET} of 0.198. The joint TNO-Spescom Datavoice (TNO-SDV) submission to NIST LRE-2005 contained two more systems and obtained a result of 0.0958.

1. Introduction

In spoken language recognition, the objective is to identify the language spoken in a given segment of speech. The nature of language identification makes it essentially a speaker- and text independent task. This means that a yet wider variability of speech signals needs to be modeled than in related technologies such as text-independent speaker recognition (where the language is generally dependent on the target speaker) and speaker-independent speech recognition (where the language is assumed known). The result is that very large training corpora are used (typically 60 hours per language, as in CallFriend [1]). Although this may seem an enormous requirement for data collection effort, the positive side is that, unlike in speech recognition, the training material does not need any orthographic annotation.

Traditional approaches to the problem may be summarized as (1) phonotactic techniques, (2) generative spectral modeling techniques and (3) discriminative spectral modeling techniques. Although these are certainly not the only imaginable techniques [2] they are used widely because of the relatively limited training effort required. In ref. [3] a system based on all these approaches is described. In this paper, we will describe efforts to extend Gaussian Mixture Models (GMMs, an example of (2)) by adding transmission channel dependence, and a novel discriminative approach (an example of (3)), the Multi-class Logistic Regression (MLR) approach.

An important stimulus to the development of language recognition systems has been the series of National Institute of Standards and Technology Language Recognition Evaluations (NIST LREs) held in 1996, 2003 [4] and recently in 2005 [5]. In these evaluations, a common task is defined, and both training and development test material is provided to participants. The task is cast as a *detection* problem, where the question is whether or not a speech segment is spoken in a specified target language. With the task, an evaluation metric is given which measures the performance of the detection *decisions* in terms of a cost function C_{DET}, with specific cost parameters. The evaluation protocol also requires a *score* for each trial, but at the November 2005 LRE workshop some questions arose that have cast doubts on the applicability of the measures such as equal error rate (EER) and minimum C_{DET}, which are traditionally used for speaker recognition score analysis. Therefore, we will report results exclusively in terms of C_{DET}. We have restricted ourselves to reporting only on the primary task, and not on other parts of the evaluation.

The paper is organized as follows. First, a resumé of the NIST 2005 LRE task is given. Then, the development of our TNO-SDV system is described with a focus on the channel-dependent GMM approach and the multi-class logistic regression technique. Next, the results of the various sub-systems in the NIST LRE-2005 are presented and discussed, as well as some post-evaluation experiments.

2. Task, data, and system design

2.1. The NIST LRE-2005 task

After two earlier evaluations in 1996 and 2003, the speech group of NIST in the US organized a language recognition evaluation (LRE) in the fall of 2005. Where in earlier evaluations a larger part of the test material was drawn from the CallFriend data collection, the 2005 evaluation promised to be exciting, because for the first time

in 10 years a new LRE collection effort had been carried out, mainly at the Oregon Health State University (OHSU). The set of target languages was reduced from 12 in 2003 to 7 in 2005, where the latter was a subset of the former, namely English, Hindi, Japanese, Korean, Mandarin, Tamil and Spanish. The *primary task* was defined to contain test segments of approximately 30 seconds, drawn exclusively from data collected at OHSU in the 7 target languages. It was made explicit that part of the English test trials would be in *Indian accented English*, and some sample segments were distributed.

During preparations for the evaluation, one of the present authors noted an undesirable property of the original evaluation metric [4], namely that a better estimation of the number of test segments per language in the evaluation would be beneficial to a site for obtaining lower error scores. Since mere good guesswork cannot be the objective of a technology evaluation, NIST changed the primary evaluation metric such that the number of target trials per language is no longer of influence to the measure. Hence the primary evaluation measure C_{DET} is defined as an average

$$C_{\mathrm{DET}} = \frac{1}{N}\sum_{i=1}^{N} C^i_{\mathrm{DET}}, \qquad (1)$$

where C^i_{DET} is the detection cost for the subset of trials for which the target language is i

$$C^i_{\mathrm{DET}} = C_{\mathrm{miss}} P^i_{\mathrm{miss}} P_{\mathrm{target}} + C_{\mathrm{FA}} \sum_{j\neq i} P^j_{\mathrm{non}} P^{ij}_{\mathrm{FA}}. \qquad (2)$$

Here N is the number of target languages (seven), C_{miss} and C_{FA} normalized cost parameters (set to unity in the evaluation), and P_{target} the prior probability for target language i that must be considered in the decision (set to $\frac{1}{2}$ in the evaluation). Finally P^j_{non} is the prior probability that the test segment is in non-target language j (set by NIST to $(1-P_{\mathrm{target}})/(M-1)$, where M is the number of test languages, in the primary task $P^j_{\mathrm{non}} = 1/12$).

The error probabilities P^i_{miss} and P^{ij}_{FA} are determined by the evaluation results, where P^i_{miss} is the proportion of true trials in language i where the system's decision was 'false,' and P^{ij}_{FA} is the proportion of trials with target language i and test segment language j where the decision was 'true.' [1]

2.2. System design

The overall TNO-SDV system is designed in a similar way to that of earlier work [3]. In a first stage, we extract spectral or phonetic features, which are then modeled by so-called *score producers*, which consist of a battery of language-dependent engines, each producing a score that tends to be higher when a test speech segment comes from a particular class of speech, *e.g.*, a certain spoken language. In a second stage, all these scores, in total numbering N_p, are combined using a Gaussian back-end. Since this back-end used a language-independent full covariance model, it effectively also forms a Linear Discriminant Analysis (LDA) reduction from the N_p scores to $M-1$ log likelihood ratios. In a final stage, these log likelihood ratios are further processed to make minimum-expected-cost Bayes' decisions.

Both the first and second stages need training data. For the first stage the data consists of large amounts of speech conditioned to the class that the score producer models. For the second stage the training data consists of a large number of speech segments of appropriate duration in the required seven target languages. In our NIST LRE-2005 system, we used four different score-producing technologies, giving a total of $N_p = 149$ scores.

2.3. Training and development test data

In this paper we refer to the set of test segments of a particular NIST LRE as 'lid96d1,' 'lid96e1,' etc., indicating the year and purpose (development/evaluation). For development testing we used 1280 segments 'lid03e1' (a held out portion of the CallFriend collection) and 40 Indian English development trials 'lid05d1' distributed with LRE-2005. Development test figures reported here are only considering the seven target languages from LRE-2005.

For the training of the language recognition systems we used two largely overlapping data resources. For the training of score producing engines we used the entire CallFriend database, *i.e.*, the three parts designated for 'training,' 'development testing,' and 'evaluation.' For the training of the LDA Gaussian back-end we also used trials from previous NIST LREs. In Table 1 we have summarized the usage of these trials for training the back-end, for both the development test (LRE-2003) and the evaluation (LRE-2005). Note that for training the back-end parameters we included trials that are drawn from the same data collections used for training the score producing engines. We found that this compromise of independence was less detrimental to the performance than the benefit obtained from more example trials.

3. TNO-SDV LRE-2005 system

3.1. Channel-dependent GMM system

3.1.1. Current approaches

In their Eurospeech 2003 paper, Singer *et al.* [3] described a state-of-the-art generative modeling system based on GMMs. The Universal Background Model (UBM) / GMM approach adopted from speaker recognition was successfully applied to language identification problems

[1] Note, that NIST sometimes reports 'normalized C_{DET}' which is twice C_{DET} reported here, and reflects the benefit of the system w.r.t. a trivial system deciding 'true' (or 'false') for every trial. The normalization factor 2 assigns a normalized $C_{\mathrm{DET}} = 1$ to such a system.

Table 1: Summary of the trials used for training of the LDA back-end for the development test and evaluation test. The symbol ∘ means that this trial set was used only in post-evaluation experiments.

Trial set	Development lid03e1+lid05d1	Evaluation lid05e1
lid96d1	•	•
lid96e1	•	•
lid03e1		∘
lid05d1		•

by introducing special spectral features, the *shifted delta cepstra* (SDC) [6]. These features are formed from normal cepstral features by repeatedly augmenting features for the current analysis frame with features from future frames. SDC features proved to have good discriminative powers when modeled by GMM or SVM [7].

One of the problems encountered in speaker recognition has been the drop in performance when different transmission channels between train and test segments are used [8], due to, *e.g.*, differences in microphone or speech coding. Since the training data for this evaluation stemmed from a decade ago, we anticipated that the introduction of new channels in the new data collection, such as cellular telephones and hands-free terminals, would be a major challenge.

In speaker recognition several methods have been developed for dealing with channel variability. Most notably there is *Feature Mapping* [9], where after channel classification features are mapped 'back' to a channel-independent feature space, and eigen-analysis [10], where channel and speaker information are disentangled in an unsupervised way.

3.1.2. Features and baseline GMM

As cepstral features we used 5 Perceptual Linear Prediction (PLP) coefficients calculated every 16 ms using a 32 ms window, augmented with log energy. From these $n = 6$ cepstral features, we calculated differentials (deltas) over $d = 1$ frame. These delta-cepstra were repeated over a time-shift interval of $p = 2$ frames, $k = 4$ times. Features were normalized to zero mean, unit variance on a per-utterance basis. The SDC parameters were chosen to mimic approximately the same time-span for the shift and delta parameters as the famous $(7, 1, 3, 7)$, which were found to be optimal in MIT's system [3]. Our SDC-parameters led to far fewer features per frame (24 versus MIT's 49). We observed no degradation in performance by increasing the frame step size from 10 ms to 16 ms, but a significant decrease in computational effort for the GMM system.

Using all of CallFriend speech data (60 conversations per language/dialect, in 15 languages/dialects), we trained two 2048-component UBMs, one per speaker sex, using one in every 20 SDC frames. Then these UBMs were further MAP adapted (means only) to 12 language- and sex-specific GMMs [11, 12] using language specific training data, with a relevance factor of 16.

Using full CallFriend training for the 24 acoustic language models and development test trials for the LDA back-end, we obtained a baseline C_{DET} of 0.050 for lid03e1 evaluated on the 7 languages of interest in LRE-2005.

3.1.3. Feature mapping with SDC's

In a first attempt, we tried to apply feature mapping to language recognition. In order to be able to perform feature mapping, there must be *channels* defined with example speech material for each channel or class. We used a training set used in the TNO 2005 speaker recognition system [13]. This set consists of 591 speakers found in the databases Switchboard II phase 2 [14] and the NIST SRE data 2001–2003. For the former, MIT Lincoln Labs had distributed microphone class labels (carbon button or electret) determined by a classifier. For the latter, NIST had provided labels for the coding of the cellular network used in the conversation (GSM or CDMA). Together with the speaker's sex this resulted in 8 'channel' classes, 2 (sex) $\times$ 4 microphone/coding (carbon button, electret, GSM, CDMA). Using this data, we built a 1024-component 'root' UBM . These were further MAP adapted (means-only) to 8 channel-dependent GMMs using training speech conditioned to each of the channels.

There are two ways in which we tried to apply feature mapping to SDC's. In the first method, we built a root UBM and channel-GMMs based on the 6 'raw' cepstral features. Then we classified each speech file (train or test) using these channel GMMs, and applied feature mapping to each raw frame, after which SDC features were calculated from these mapped features. In the second method, we first calculated SDC features from all feature mapping training data, and then built a root UBM from that. Feature mapping was then applied to the full SDC features for each train and test speech segment.

Neither of these methods improved the language recognition results compared to our baseline.

3.1.4. New approach

A fact that we did not mention in the description of our 'baseline' system above is that the sex is not known for all speakers in CallFriend. In fact, only the sex of the person initiating the call is known, and not that of the person called. As was observed in earlier work [3] we had noticed that our GMM-based system benefited from doubling the number of score-producers by separating models for speaker sex. This was different from, *e.g.*, SVMs. Since the GMM approach also benefits from more training speakers, it had been desirable for our baseline system to automatically determine the gender[2] of the callee.

[2]We refer to the sex of the speaker, if determined by analysis of the speech, as *gender* to reflect the socio-linguistic character of classification.

We could use the intermediate classification result in the feature mapping process for this purpose.

Since we had observed that the GMM score producers somehow benefited from separation into classes that were determined automatically (gender), we extended this notion to *channel* classes. So rather than doing feature mapping, we used the classification made by the (raw) feature mapper to divide the training data in channel-dependent classes. Since we had relatively few observations in the GSM and CDMA classes, we decided to collapse these to a single class 'cellular', so that we ended up with 6 classes (2 'gender' $\times$ 3 'channel'). Thus, we trained 6 channel-dependent UBMs, each of which was MAP adapted to 12 language-dependent GMMs using training data conditioned on channel and language. For one condition (Japanese female cellular) there were no training examples so we ended up with 71 GMM scores.

This new approach resulted in a C_{DET} of 0.0352 on the development test data, which was a significant improvement with respect to the 2-gender baseline result 0.050.

3.2. Multi-class Logistic Regression

Logistic regression is a discriminative learning technique, which is very similar to support vector machines (SVM) and which, in our experiments, gave language recognition accuracy very similar to SVM. Just like SVMs can be extended to multi-class recognition [15][3], we extended the logistic regression approach to a multi-class recognition problem. We refer to this type of logistic regression as *multi-class logistic regression* (MLR). Although MLR gave no clear advantage over the more well-known 2-class SVM in these experiments, we nevertheless dedicate this section to recording the details of our implementation, because there may be other contexts in which MLR can indeed have important advantages.

3.2.1. MLR versus SVM

When recognizing a closed set of N language classes $\{L_1, \ldots, L_N\}$, via SVM, it is customary to use N different, two-class, one-against-the-rest, language recognition SVMs. Each SVM i outputs a *score*, s_{it}, where a more positive score favours the hypothesis that the test segment t is in language class L_i and a more negative score favours the hypothesis that the test segment is in any of the other $N-1$ language classes. These N SVMs are separately trained. An SVM with a linear kernel effectively implements an affine transform from a D-dimensional input feature vector, $\mathbf{x}_t$, to the one-dimensional score: $s_{it} = \mathbf{w}_i \cdot \mathbf{x}_t + k_i$. The SVM 'model' parameters are the vector $\mathbf{w}_i$ and the scalar offset k_i.

In contrast, a single MLR model can be trained to discriminate between all N classes. Moreover, its scores can be interpreted as *log-likelihoods* or *log-likelihood-ratios*.

[3] We are indebted to a kind but anonymous reviewer for pointing this out to us.

Note that in a closed N-class recognition problem, one does not need N independent scores, because this is a redundant representation of the information extracted from the input data by the recognizer. In a non-redundant representation, the scores or log-likelihood-ratios need only have $N-1$ components. However, non-redundant representations are asymmetrical and inconvenient to work with, so we chose to work with a redundant symmetrical logistic regression that gives N scores, s_{it}, having log-likelihood interpretation, followed by a redundancy removal which yields $N-1$ scores, λ_j having log-likelihood-*ratio* interpretation. For each test segment t we calculated:

$$s_{it} = \mathbf{w}_i \cdot \mathbf{x}_t + k_i, \qquad i = 1, \ldots, N \tag{3}$$

$$\lambda_{jt} = s_{jt} - s_{Nt}, \qquad j = 1, \ldots, N-1 \tag{4}$$

where all of the model parameters $\mathbf{w}_i$ and k_i together form a single logistic regression model, M:

$$\mathsf{M} = \left\{(\mathbf{w}_1, k_1), (\mathbf{w}_2, k_2), \ldots, (\mathbf{w}_N, k_N)\right\} \tag{5}$$

which is trained with a single procedure rather than with N separate procedures as for SVM. This model, M, defines a single affine transform that takes input vector $\mathbf{x}_t$ to the $(N-1)$-dimensional log-likelihood-ratio space. The choice of likelihood-ratio denominator in equation 4 is arbitrary and can be changed via a linear transformation in log-likelihood-ratio space.

3.2.2. MLR Training Objective Function

The logistic regression scores function as log-likelihood-ratios because of the way they are trained. The scores are formed in exactly the same way as for SVM, namely by affine transform, but the model parameters end up being different. These log-likelihood-ratios are defined as:

$$\lambda_{jt} = \log \frac{p(\mathbf{x}_t | L_j, \mathsf{M})}{p(\mathbf{x}_t | L_N, \mathsf{M})}, \qquad j = 1, \ldots, N-1 \tag{6}$$

The basic logistic regression objective function[4] is simply the total log posterior probability of the labels of the training data, given the feature vectors for this data. Assuming trial-independence, this can be written as

$$\log \prod_{t \in \mathcal{T}} P(L_t | \mathbf{x}_t, \mathsf{M}), \tag{7}$$

where $\mathcal{T}$ is the set of training speech segments, L_t is the true language class for segment t and $\mathbf{x}_t$ is the feature supervector for that segment. Using equations 3 to 6 and Bayes' rule, we can write the posterior, $P(L_i | \mathbf{x}_t, \mathsf{M})$, in terms of the logistic regression scores, s_{it}:

$$P(L_i | \mathbf{x}_t, \mathsf{M}) = \sigma_{it} = \frac{e^{s_{it} + \gamma_i}}{\sum_{j=1}^{N} e^{s_{jt} + \gamma_j}} \tag{8}$$

[4] Note the similarity between this objective function and the MMI objective function as employed for GMM-based language recognition in [16].

where $\gamma_i = \log P(L_i)$ represents the prior probability distribution of the language classes. We used $P(L_i) = \|\mathcal{T}_i\| / \|\mathcal{T}\|$ where $\mathcal{T}_i$ denotes the subset of speech segments having language class L_i and where $\|\cdots\|$ is the cardinality operator.

As with SVM training, it is usually necessary to *regularize* the objective function to avoid over-training. To this end, we used a quadratic penalty on the model weights, which tends to limit the magnitudes of the log-likelihood-ratios. Assembling all of this, we get the final objective function:

$$\mathcal{O}(\mathsf{M}) = \sum_{i=1}^{N} \sum_{t \in \mathcal{T}_i} \log \sigma_{it} - \frac{\alpha}{2} |\mathbf{w}_i|^2 \qquad (9)$$

where $|\mathbf{w}_i|^2 = \mathbf{w}_i \cdot \mathbf{w}_i$ and α is a positive regularization constant that can be chosen with a cross-validation procedure on some held-out data. For a given value of α, the logistic regression training is the unconstrained convex optimization problem of maximizing $\mathcal{O}(\mathsf{M})$ w.r.t. the $N(D+1)$ parameters of M. A few comments are in order:

- Logistic regression is traditionally used to give posterior probabilities, rather than log-likelihoods. However, by explicitly inserting the prior into the objective function, we train the logistic regression to give the latter.
- Unlike most SVM implementations, this form of logistic regression does not make use of a kernel (or Gram) matrix. It can be advantageous to do this (see *e.g.*, ref. [17]), but in our case the training problem was dimensioned such that $\|\mathcal{T}\| > D$. This meant that operating directly on the $\|\mathcal{T}\| \times D$ input matrix was more economical than working with a $\|\mathcal{T}\| \times \|\mathcal{T}\|$ kernel matrix.
- It is possible to re-weight the relative contributions of different language classes in the objective function in order to optimize for operating conditions different to that given by the proportions of classes in the training data, $P(L_i) = \|\mathcal{T}_i\| / \|\mathcal{T}\|$. However, in our experience, we found that this did not make a significant difference when the operating conditions between training and actual use are not too different.

3.2.3. MLR training implementation

Implementing two-class logistic regression can be very simple,[5] but multi-class implementations are considerably more difficult. Although several free implementations are available for download,[6] we nevertheless found it more convenient to write our own implementation of a non-linear conjugate-gradient optimizer, following the general methodology of [18].

Ideally a conjugate gradient optimizer should make use of both the *gradient* (vector of first derivatives of the objective function, w.r.t. each parameter) and the *Hessian*, (matrix of second derivatives). Roughly speaking, the gradient helps the optimizer to decide in which direction in parameter space to move next and the Hessian helps it to decide how far to move in that direction. With some patience and concentration, the gradient of $\mathcal{O}(\mathsf{M})$ is easy enough to derive analytically and to implement in code. However, analytically deriving and coding the Hessian is a much bigger and error-prone task. Also keep in mind that the Hessian matrix is *huge*, having $(D+1)^2 N^2$ elements. Fortunately, the need to calculate the Hessian can be replaced by *line-search* techniques [18]. Once the optimizer has decided in which direction (line) to move for the next iteration, a line search finds an approximate optimum along this line at the cost of a few objective function evaluations.

In our implementation, we noted that the most expensive part of the objective function evaluation was the linear operations of equation 3, having computational weight proportional to $DN\|\mathcal{T}\|$. Fortunately, however, because of the linearity, we were able to factor the computational work such that most of it could be done just once per line search, irrespective of the number of function evaluations along the line. (A similar economy was implemented for the quadratic penalty term.) This resulted in robust[7] and very fast line-searches.

3.2.4. SDC-MLR

The MLR supervector for SDC is found by a third order multinomial expansion of the SDC features, followed by an averaging and normalization operation. Component-specific averaging is performed over a whole test segment, and expanded features are normalized by the variance of the component determined from a large background data set, in this case the 'training' third of Call-Friend.

The MLR system used different SDC feature extraction to the GMM and SVM systems, namely MFCC features at a frame-rate of 10 ms shift, in a $(7, 1, 2, 4)$ SDC configuration. Three variants of SDC-MLR models were used, which, when combined, gave a slight performance benefit over the individual variants. One with plain MFCC-SDC features, and two with an additional feature indicating the speech activity detection coded as $\pm\frac{1}{2}$. These latter two systems only discarded frames where all shifted cepstra in the SDC had no speech activity detected. The two systems differed in training—one

[5] See, *e.g.*, `http://www.dsp.sun.ac.za/~nbrummer/focal/`, where there is a MATLAB implementation of logistic regression fusion for use in speaker detection.

[6] See, *e.g.*, `http://www.kyb.tuebingen.mpg.de/bs/people/seeger/software.html` or `http://www.isi.edu/~hdaume/megam/index.html`

[7] There are many subtleties involved in constructing numerically robust line-search algorithms for non-linear optimands. By making function evaluations along the line cheap, we could afford to use as many as needed to implement all of the mechanisms recommended in [18].

used the 'training' CallFriend third, the other the 'development' one third. The SDC-MLR gives a C_{DET} on development data (lid03e1) of 0.072.

3.2.5. Phonotactic MLR

A phonotactic MLR was built on the basis of a phone-recognition system obtained from the University of Stellenbosch. This system produces 44 phone-probabilities every 15 ms. The probability vector stream is first segmented based on where the maximum phone probability label changes, and then averaged over these phone-like segments. MLR supervector components are formed by the 44 phone probabilities averaged over the segment and 44×44 bi-phone probabilities.

Our final phonotactic system is the combination of three variants, one plain system and two 'staggered' systems where the bi-phone probabilities are obtained by skipping one or more phones in the middle. The first staggered system augmented the standard unigram and bigram by adding 2-staggered bigrams and 3-staggered bigrams to the supervector features. Our second staggered system did the same but adding only 4-staggered bigrams. The final phonotactic MLR thus had 33 scores and operated at $C_{\mathrm{DET}} = 0.108$ on development test data.

We also attempted the classical phonotactic approach, where we used the maximum phone probability label per segment as phone token, followed by n-gram modeling. We used the SRILM toolkit [19] in order to estimate bigram probabilities for language modeling and calculation of the probability of the phone sequence given the model. This led to a higher C_{DET} of 0.169.

3.3. SVM system

For completeness, we will first describe the final score producer, an SVM implementation, briefly (see also section 3.2.1). This system is very similar to systems described earlier [7]. We used the same SDC features as for the GMM system, and expanded them using 3rd order monomials in order to obtain 2924 dimensional features. These were averaged per utterance, and normalized using the per-dimension root-mean square of all training data. After silence removal, each CallFriend training conversation side was divided in 5 equal length utterances. Twelve SVM language models were trained by taking the target language as positive examples amongst a background of all other CallFriend languages. We used IDIAP's `SVMTorch` for this [20]. Scores were produced for each test segment by subjecting the test segment features to the same expansion, averaging and scaling operation as the training files. Thus, 12 scores per test segment were produced.

3.4. Gaussian back-end and decisions

In the final stage, the 149 scores, produced with 4 technologies described above, are fused: 71 scores from the channel-dependent GMM, 12 SVM scores, 33 SDC-MLR scores and 33 phonotactic MLR scores. The back-end is an affine transformation from N_p scores to $N-1$ log-likelihood-ratios. N likelihood scores for each target language are obtained from multivariate Gaussians with target language specific means and a common full covariance matrix. then we form $N-1$ log-likelihood-ratios by normalization w.r.t. the Nth likelihood. The *linear* part of the affine transform is just the same as an LDA transform, which tries to maximize the ratio of between-class to within-class variance. The *translation* part of the affine transform is equivalent to the calibration task of setting 'language dependent thresholds.' In order to estimate the within-class covariance we found it was beneficial to work with all 12 CallFriend language classes even for the application of the 7 LRE 2005 target languages. Because the within-class covariance matrix needs inversion, and we found that this matrix can become ill-conditioned, we regularized it by adding $\lambda\mathbf{I}$ before the inversion, where we used $\lambda = 5$.

Decisions were made on the basis of minimum expected cost [21]. Using Bayes' rule we converted the LDA log-likelihood ratios into posteriors using prior 0.5 for the target language and 0.5/6 for the other languages. Then finally, we compared the posterior for the target language to the Bayes decision threshold of 0.5.

Table 2: Summary of the language detection performance for the development test and evaluation test.

System	Devel	Eval	Post-eval
GMM	0.0409	0.113	0.0951
SVM	0.073	0134	
SDC-MLR	0.072	0.198	
Phon-MLR	0.108	0.187	
All	0.0255	**0.0958**	0.0924

4. LRE 2005 results and analysis

4.1. LRE 2005 submission

The results of the primary evaluation metric C_{DET} is shown in Table 2. For comparison, the values of the development test set lid03e1+lid05d1 are also included, as well as the results of the individual subsystems.

4.2. Constellation plot

The effects of fusion of several systems can be summarized nicely in what is known as a *constellation plot*. This idea was presented first by Dr. Schriberg at the NIST SRE 2005 workshop [22]. The graph is a scatter-plot of various system fusions, where the axes denote two important parameters for the evaluated technology. For our purposes we chose to plot C_{DET} obtained in development testing against C_{DET} obtained in the LRE'05 evaluation tests. In figure 1 the data points of all our system combinations is shown. In the figure, the disks indicate C_{DET} values of the four single systems, pluses indicate the six binary system fusions, triangles indicate 4 triple fusions and the square indicates the four-system fusion. Arrows

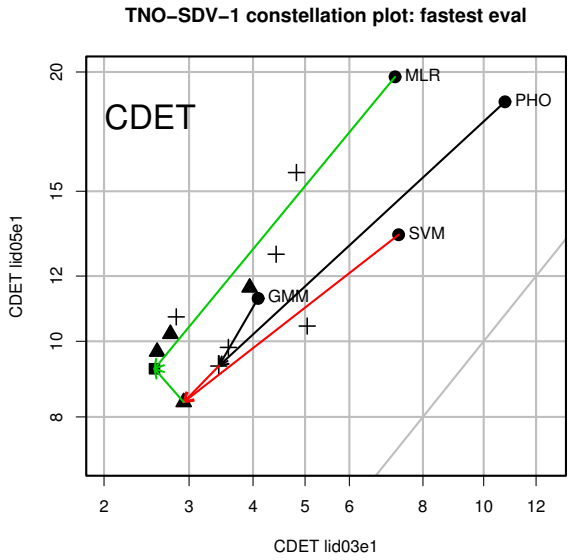

Figure 1: Effect of increasing the systems in the fusion. Progressively the number of systems is indicated by the symbol in the order of 1: disc, 2: plus, 3: triangle, and 4: square. Note that the axes (in %) are warped using the probit function, as is customary in DET plots.

are used to indicate how we can start from the best performing system (GMM), progressively adding 'most effective' alternative systems. From the figure, several observations can be made quickly:

- The GMM system performs best and is most effectively combined with first Phonotactic MLR (lowest +) and then with the SVM system.
- For the development test, it helped to add the MLR system to the other three (the square is more to the left than all other triangles), but in the evaluation it actually hurt.

Not all the system combinations are identified in the plot, in order not to clutter the graph too much, but one can imagine that the plotting method gives detailed diagnostic information about which system combinations are beneficial. Another interesting way of drawing arrows is to start at the full fusion and see which system made the biggest difference when being added last. In the case of SRI's constellation plots, where the axes are formed by EER and $C_{\mathrm{DET}}^{\mathrm{min}}$, the data points for $N-1$ subsystems nicely formed a group between the fully fused system and the group with combinations of, say, 2 subsystems. Hence the name 'constellation plot' was given.

4.3. Post-evaluation analysis

We have run a couple of experiments after the evaluation. The analysis of some of these are the following:

Indian English Although unmentioned in section 3.1, we had taken measures to deal with Indian English test segments, since the inclusion of these was announced in the evaluation plan. Using development test set 'lid05d1,' consisting of 40 conversation excerpts of Indian English speech, we MAP-adapted the American English UBM in order to obtain American/Indian English models, anticipating that the 120 hours of American English would not be completely 'spoilt' by the 20 minutes of Indian English, and that some of the Indian English character would be modeled by the new UBM. However, this turned out to be a bad decision, since the original American English models would have performed with a $C_{\mathrm{DET}} = 0.0951$ rather than 0.113.

SDC-MLR In an attempt to understand why the SDC-MLR system gave a negative contribution to the fused system in the evaluation, we ran the same system with the same PLP-SDC features as calculated for the GMM and SVM subsystems. This gave improved SDC-MLR performance on the evaluation test set, very similar to the SVM performance. However, in this case the SVM and SDC-MLR did no longer fuse to give a better combined result.

Regularization parameter During development testing we had found that a regularization parameter $\lambda = 5$ was necessary in order to prevent the between-class covariance matrix from becoming ill-conditioned. However, we found that with more LDA training trials (by including lid03e1 and lid05d1, see Table 1) we could have used $\lambda = 0$, leading to a better fused result of 0.0924.

4.4. Discussion and Conclusions

In the development of a language recognition system suitable for the NIST LRE 2005, we managed to obtain a set of technologies, each capable of performing language recognition to varying degrees. As a single technology, the channel-dependent GMM approach performs with lowest C_{DET}, but benefits from additional information from other classifiers. It is interesting that by merely separating the training database in different partitions, according to a channel classifier so that more score producers can be built, improves the perfomance of this GMM system significantly. This channel classifier is trained on English data only, and it is hard to speculate if more performance gain would have been obtained if labeled channel information would have been available in more languages. An observation we would like to make here is that the *generative* GMM seems to benefit more from multiple score producers and a Gaussian back-end than *discriminative* techniques such as SVM and MLR seem to do.

We further developed a new classifier for language recognition problems, the multi-class logistic regression technique. Both this implementation and our fusion use

an *objective function* to minimize the error, and therefore includes *calibration*. Simultaneously, there have been attempts to make the scores interpretable as *log-likelihood-ratios*, so that it is also possible to make optimal decisions based on *different* cost parameters than the ones defined in the NIST evaluation plan. However, this matter is more complicated than in the case of speaker recognition and deserves additional research [21].

As for the result of the combined technologies we are reasonably satisfied with our performance $C_{\mathrm{DET}} = 0.0958$ in the evaluation [5], given the fact that we had limited phonotactic information. Some choices we made turned out to be disadvantageous to the final C_{DET}. The effort to adapt English models to the Indian English development samples was perhaps naïve and the actual number of Indian English trials in the test was only 1/5 of all English trials, and therefore less important than we had anticipated. Despite all this, we believe the introduction of a new target language or accent, for which only a limited number of test-trials is available (and not amounts comparable to CallFriend training data) is an interesting problem, and certainly has applications.

The introduction of a regularization parameter λ would not have been necessary, but in situations with less back-end training trials or more score producers, this parameter may still be beneficial. One of the problems we encountered during development was the relatively low number of errors found in the development test. When error rates are around 2 %, the difference in C_{DET} between two conditions becomes less significant, and it is hard to make design decisions. This problem could have possibly been overcome by using the 10-sec. duration trials from the development set. Luckily, the new evaluation material gives rise to many more improvements to be made for further development.

The constellation plot is quite a powerful diagnostic tool for interpreting fusion results. Although identification of individual points (system combinations) quickly clutter the graph, we found that the indication of relationships between points, *e.g.*, adding a specific sub-system, can give further insight into the fusion process.

5. References

[1] Alexandra Canavan and George Zipperlen, "LDC callfriend," 1996, Catalog IDs LDC96S46 trough S60.

[2] M. A. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Trans. Speech and Audio Proc.*, vol. SAP-4, no. 1, pp. 31–44, 1996.

[3] E. Singer, P. A. Torres-Carrasquillo, T. P. Gleason, W. M. Campbell, and R. A. Reynolds, "Acoustic, phonetic, and discriminative approaches to automatic language identification," in *Proc. Eurospeech*, 2003, pp. 1345–1349.

[4] Alvin F. Martin and Mark A. Przybocki, "Nist 2003 language recognition evaluation," in *Proc. Eurospeech*, 2003, pp. 1341–1344.

[5] Audery Le and Alvin Martin, "Current language recognition performance as shown in the 2005 NIST language evaluation," in *Proc. Speaker Odyssey*, 2006, submitted.

[6] Pedro A. Torres-Carrasquillo, Elliot Singer, Mary A. Kohler, Richard J. Greene, Douglas A. Reynolds, and J. R. Deller Jr, "Approaches to language identification using gaussian mixture models and chifted delta cepstral features," in *ICSLP*, 2002.

[7] W. M. Campbell, E. Singer, P. A. Torres-Carrasquillo, and D. A. Reynolds, "Language recognition with support vector machines," in *Proc. Odyssey 2004 Speaker and Language recognition workshop*, 2004, pp. 285–288.

[8] George R. Doddington, Mark A. Przybocki, Alvin F. Martin, and Douglas A. Reynolds, "The NIST speaker recognition evaluation—Overview, methodology, systems, results, perspective," *Speech Communication*, vol. 31, pp. 225–254, 2000.

[9] Douglas A. Reynolds, "Channel robust speaker verification via feature mapping," in *Proc. ICASSP*, 2003, pp. 53–56.

[10] Patrick Kenny and Pierre Dumouchel, "Disentangling speaker and channel effects in speaker verification," in *Proc. ICASSP*, 2004, pp. 37–40.

[11] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori esitimation for multivariate gaussian mixture observations of markov chains," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 291–298, 1994.

[12] D.A. Reynolds, T.F. Quatieri, and R.B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, pp. 19–41, 2000.

[13] David van Leeuwen, "TNO speaker recognition system presentation," NIST SRE Workshop, Montreal, 2005.

[14] David Graff, Kevin Walker, and Alexandra Canavan, "LDC Switchboard II phase 2," 1999, Catalog ID LDC99S79.

[15] Koby Crammer and Yoram Singer, "On the algorithmic implementation of multiclass kernel-based machines," *Journal of Machine Learning Research*, pp. 265–292, 2001.

[16] L. Burget, P. Matejka, and J. Cernocky, "Discriminative training techniques for acoustic language identification," in *Proc. ICASSP*, Toulouse, France, 2006, submitted.

[17] J. Zhu and T. Hastie, "Kernel logistic regression and the import vector machine," *Journal of Computational and Graphical Statistics*, vol. 14, no. 1, pp. 185–205, 2005.

[18] W. H. Press et al., *Numerical Recipes in C*, Cambridge University Press, 1998.

[19] A. Stolcke, "Srilm—an extensible language modeling toolkit," in *International Conference on Spoken Language Processing*, Denver, 2002, vol. 2, pp. 901–904.

[20] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," Tech. Rep. IDIAP-RR 02-46, IDIAP, 2002.

[21] Niko Brümmer and David A. van Leeuwen, "On calibration of language recognition scores," in *Proc. Speaker Odyssey*, 2006, submitted.

[22] Elizabeth Schriberg et al., "SRI speaker recognition system presentation," NIST SRE Workshop, Montreal, 2005.