The Speaker and Language Recognition Workshop (Odyssey 2018)
26-29 June 2018, Les Sables d'Olonne, France

The LEAP Language Recognition System for LRE 2017 Challenge - Improvements and Error Analysis

Bharat Padi, Shreyas Ramoji, Vaishnavi Yeruva, Satish Kumar, Sriram Ganapathy

Learning and Extraction of Acoustic Patterns (LEAP) Laboratory
Electrical Engineering, Indian Institute of Science, Bengaluru, India.
{bharatp, shreyasr, sriramg}@iisc.ac.in, {vaishnaviy2, satish.pas}@gmail.com

Abstract

The language recognition evaluation (LRE) 2017 challenge comprises an open evaluation of the language identification (LID) task on a set of 14 languages/dialects. In this paper, we describe our submission to the LRE 2017 challenge fixed condition which consisted of developing various LID systems using i-vector based modeling. The front end processing is performed using deep neural network (DNN) based bottleneck features for i-vector modeling with a Gaussian mixture model (GMM) universal background model (UBM) approach. Several back-end systems consisting of support vector machines (SVMs) and deep neural network (DNN) models were used for the language/dialect classification. The submission system achieved significant improvements over the evaluation baseline system provided by NIST (relative improvements of more than 50% over the baseline). In the later part of the paper, we detail our post evaluation efforts to improve the language recognition system for short duration speech data using novel approaches of sequence modeling of segment i-vectors. The post evaluation efforts resulted in further improvements over the submitted system (relative improvements of about 22 %). An error analysis is also presented which highlights the confusions and errors in the final system.

1. Introduction

The NIST Language Recognition Evaluation (LRE) 2017 Challenge [1] targets the development of language recognition systems on a closed set of 14 language/dialects. The 14 language classes used in LRE consisted of 5 major language clusters and various dialects of these languages. For the language identification (LID) task, the performance is degraded due to presence of dialectal variations of the same language and the short segment duration of the speech recordings. In this paper, we describe various approaches used for our submission to the LRE 2017 challenge with a particular focus on improving the LID system for short-duration signals.

Traditionally, phoneme recognition followed by language modeling (PRLM) was one of the popular methods for automatic LID task [2, 3]. This approach uses a multilingual phoneme recognizer to generate phoneme sequences which are converted to language model (n-gram) features for the LID classifier. The success of this approach is dependent on the performance of the phoneme decoder. For relatively clean data with linguistic resources, the PRLM method provides good performance comparable to acoustic systems [4]. In the recent past, the use of deep neural network (DNN) based posterior features were attempted for LID [5]. The Tandem features have shown promising results for noisy language recognition as well [6]. The use of bottleneck features derived from a speech recognition acoustic model has also shown good improvements for language recognition [7, 8].

The development of i-vectors as one of the primary features for LID was first introduced in [9]. The i-vectors are features of fixed dimensions derived from variable length speech utterances using a background model [10]. The background model can be a Gaussian mixture model (GMM) [11] or a DNN model [12]. In this work, we use GMM based UBM for the i-vector modeling. The i-vectors capture long term information of the speech signal such as speaker and language. The i-vectors extracted from the training data are used to train classifiers such as support vector machines (SVMs) and deep neural networks (DNNs).

For the LRE evaluation, we submitted a combination of four systems where three of the sub-systems used DNN-bottleneck features and the fourth system used a two-dimensional auto-regressive (2D-AR) model features [13]. Each of the systems have different i-vector extraction approaches and back-end models (SVMs/DNNs). All the four systems were combined with score fusion using the foCal toolkit [14].

The i-vector feature extraction performs global modeling of the utterance and may ignore local speech information which may be crucial to discern the language/dialect. In order to overcome this limitation, we propose the extraction of short-segment i-vectors (i-vectors from 1sec segments similar to the i-vector extraction in speaker diarization [15]) which are processed using sequence models for LID. The early version of this model using simple averaging was attempted for the LRE evaluation and this work has been advanced in the post evaluation efforts using long short term memory (LSTM) models. The post evaluation efforts also attempted the use of linear discriminant analysis (LDA) after the i-vector extraction. This LDA based dimensionality reduction provided significant improvements in the LRE results. The post-eval system improves the LRE submission system by about 22 % relatively.

The rest of the paper is organized as follows. In Sec. 2, we describe the LRE 2017 challenge task and the data resources used in our model training. The details of various system components used in our LRE submission are given in Sec. 3. The post-eval efforts in improving the LID system are detailed in Sec. 4. The results and analysis are discussed in Sec. 5 followed by a summary of the paper in Sec. 6.

The work done was also partly supported by funds from Defense Research Development Organization (DRDO) under the DST0689 project and from the Prathiksha Grant. The views expressed in this paper are from the authors and do not express the views of the funding agencies.

10.21437/Odyssey.2018-5

2. LRE 2017 Data Description and Evaluation Metrics

The LID system training is performed on the LRE17 training LDC2017E22 dataset and the development and evaluation are performed using the LRE17 dev (LDC2017E23) and eval set. For the DNN Bottleneck (BN) feature extraction, we trained the DNN model using labeled speech data from Switchboard SWB1 and Fisher corpora ($\sim$2000 hours). The LRE17 training data has five major language clusters with 14 target dialects as shown in Table 1 with a total duration of $2,069$ hours in 16205 files. The development dataset consists of 3661 files which contain 253 hours of audio and the evaluation dataset consists of 25451 files with 1065 hours of audio. The development and evaluation datasets are further partitioned into utterances of durations 3sec, 10sec, 30sec or 1000sec.

The performance metrics are reported on the development data and evaluation data. For preliminary system development, the developed language recognition systems are trained on LRE17 train set and tested on LRE17 development set. For the final submission, these systems are trained on LRE17 train set and development set and tested on the LRE17 evaluation set.

The long speech files in train dataset (LDC2017E22) were split into roughly 30sec splits, and approximately $50k$ files were obtained by randomly sub-sampling the split files. This modified dataset will be referred to as "DATASET-B" hence-forth. Some of the systems developed use this dataset. We have also explored the use of all the 30sec splits ($160k$ approx) without any sub-sampling and we refer to this dataset as "DATASET-C".

The performance is measured using the primary metric described in the evaluation plan of NIST LRE 2017 [1]. In terms of conditional probabilities for the observed data (O) given a target language model (L_i), the log-likelihood score (l_i) is defined as

$$l_i = \log\left(P(O|L_i)\right). \tag{1}$$

The likelihood function in Eq. 1 is related to the posterior probability $P(L_i|O)$ via Bayes' rule as follows,

$$P(L_i|O) = \frac{P(L_i)\exp(l_i)}{\sum_{j=1}^{N_L} P(L_j)\exp(l_j)}, \tag{2}$$

where $P(L_i)$ is the a priori probability of the language class i, and N_L is the number of target languages. The pair-wise language recognition performance will be computed for all the target-language/non-target-language pairs (L_T, L_N). An average performance cost for each system is computed as

$$C_{avg}(\beta) = \frac{1}{N_L}\left\{\sum_{L_T} P_{miss}(L_T) + \frac{\beta}{N_L - 1}\sum_{L_T}\sum_{L_N} P_{FA}(L_T, L_N)\right\}, \tag{3}$$

where P_{miss} is the probability of miss detection, P_{FA} is the probability of false alarm, which are computed by applying detection threshold of $\log\beta$ to log-likelihood ratios derived from the log-likelihood values output by the system. The primary metric used for LRE17 is the average cost performance defined in Eq. 4,

$$C_{avg} = \frac{C_{avg}(\beta_1) + C_{avg}(\beta_2)}{2}, \tag{4}$$

where $\beta_1 = 1$, $\beta_2 = 9$. The lower the C_{avg} score, the better the language recognition performance.

Table 1: LRE17 training set : target languages, language clusters and total number of hours.

Cluster	Target Languages	Hours
Arabic	Egyptian Arabic (ara-arz)	190.9
	Iraqi Arabic (ara-acm)	130.8
	Levantine Arabic (ara-apc)	440.7
	Maghrebi Arabic (ara-ary)	81.8
Chinese	Mandarin (zho-cmn)	379.4
	Min Nan (zho-nan)	13.3
English	British English (eng-gbr)	4.8
	General American English (eng-usg)	327.7
Slavic	Polish (qsl-pol)	59.3
	Russian (qsl-rus)	69.5
Iberian	Caribbean Spanish (spa-car)	166.3
	European Spanish (spa-eur)	24.7
	Latin American Continental Spanish (spa-lac)	175.9
	Brazilian Portuguese (por-brz)	4.1

3. LRE Submission

The block schematic of the LRE training and testing setup used in our systems is shown in Fig. 1.

3.1. Front-end Feature Extraction

Several feature extraction techniques were implemented, followed by the UBM training and the rest of the i-vector pipeline as shown in Fig. 1. The i-vector systems require the front-end features to contain language rich information and suppress the speaker and channel information. Language information, however is embedded in the long term characteristics of the speech signal. Hence, the front-end features should capture the long term phonotactic characteristics.

3.1.1. Bottleneck features (BNF)

We extracted bottleneck features (BNF) from a DNN trained for automatic speech recognition using Kaldi [16] framework. The DNN was trained on speech from combined Switchboard (SWB1) and Fisher corpora (about 2000 hours of labeled audio). The model uses hidden layers with ReLU activation with layer-wise batch normalization.

Once the BNF features are extracted for the LID data, a speech activity detection (SAD) algorithm was applied to remove the unvoiced frames [17]. We use the implementation of SAD from the Voicebox toolkit [4]. This was followed by cepstral mean variance normalization (CMVN) done over each utterance, followed by a sliding window CMVN where the mean and variance are normalized over a sliding window of 3sec.

3.1.2. 2D-AR model coefficients

One of our submission systems use the two-dimensional autoregressive (2D-AR) model based acoustic features [18]. These features characterize the temporal trajectory of the speech signal in long-temporal segments (1sec segments) in mel spaced subbands. The sub-band windowing and the configuration of the 2D-AR model are similar to those mentioned in [18].

3.2. Feature Extraction for Language Classification

The front-end features discussed above is a time sequence of features, where the length of the sequences is proportional to the corresponding duration of the speech files. The task of lan-

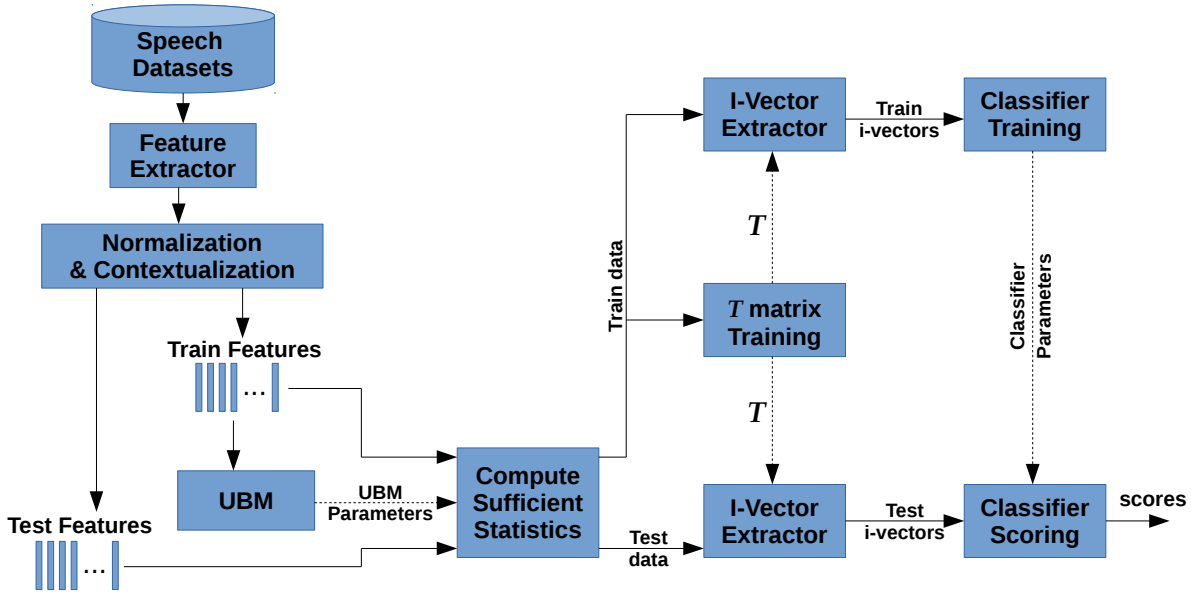

Figure 1: The i-vector based language recognition pipeline used in the LEAP LRE submission system.

guage recognition is typically accomplished by deriving statistical features from the variable length sequence of the front-end features. In this section, we discuss the approaches used to derive i-vector features for classification using the front-end features.

3.2.1. *I-vector Feature Extraction*

The i-vectors are one of the most widely used features for language recognition [9]. They are features of a fixed dimension derived from a variable length sequence of front-end features. A Gaussian Mixture Universal Background Model (GMM-UBM) is obtained by pooling the front end features from all the utterances in the train dataset. The means of the GMM are adapted to each utterance using the Baum-Welch (BW) statistics of the front-end features. Assuming that the front-end features are of dimension F with the GMM-UBM containing C mixture components, the zeroth and centered first order BW statistics of a recording s and UBM mixture component c are given by,

$$N_c(s) = \sum_{i=1}^{H(s)} \mathrm{p}(c \mid \boldsymbol{x}_i),$$

$$\boldsymbol{F}_{X,c}(s) = \sum_{i=1}^{H(s)} \mathrm{p}(c \mid \boldsymbol{x}_i)(\boldsymbol{x}_i - \boldsymbol{\mu}_c),$$

where $\boldsymbol{x}_i$ is the front end feature at time index i, $H(s)$ denotes the number of frames in recording s, and $\boldsymbol{\mu}_c$ is the mean of the UBM mixture component c.

A Total Variability Model (TVM) is assumed as a generative model for the adapted GMM mean supervector, which is given by,

$$\boldsymbol{M}(s) = \boldsymbol{M}_0 + T\boldsymbol{y}(s),$$

where $\boldsymbol{M}_0 = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_C \end{pmatrix}$ and $\boldsymbol{M}(s) = \begin{pmatrix} \boldsymbol{\mu}_1(s) \\ \vdots \\ \boldsymbol{\mu}_C(s) \end{pmatrix}$ are the UBM mean supervector and the adapted mean supervector of recording s respectively. Here, T is a matrix of dimension $CF \times R$, and $\boldsymbol{y}(s) \sim \mathcal{N}(\mathbf{0}, I)$ is a latent variable which is a vector of dimension R. The subspace spanned by $\boldsymbol{y}(s)$ is called the total variability space. The *maximum aposteriori* (MAP) estimate of $\boldsymbol{y}(s)$ given the front-end features of the recording s is called the i-vector of recording $\boldsymbol{y}^*(s)$. The i-vectors extracted for each of the speech files are processed with various normalization and dimensionality reduction techniques and are used as features for the language classification task.

In our experiments, the front-end features were of dimensions $F = 80$ in case of bottleneck features and $F = 39$ in case of 2D-AR features. The number of UBM mixture components was $C = 2048$ and the dimension of the total variability space was fixed to be $R = 500$.

3.2.2. *Short-term i-vectors*

In order to capture the local speech information which may get ignored in the traditional global i-vector models, we extracted short term i-vectors for every 1000msec window on the front-end features with a 200msec hop for each utterance. So, instead of a single i-vector representation for the entire utterance, we have a sequence of short term i-vectors $[\boldsymbol{y}_1^*, \ldots, \boldsymbol{y}_n^*]$ representing each utterance, where n is proportional to the duration of the utterance. While the evaluation submission used a simple averaging of these short-term i-vectors, the post-eval efforts used the short-term i-vectors in conjunction with a sequence classification model.

3.3. Back-end Classifiers

3.3.1. *Support Vector Machines (SVM)*

The i-vectors, processed with the within class covariance normalization (WCCN) technique [19], were used in a SVM kernel based classifier. We trained one-versus-rest SVM classifier which generated the log likelihood scores of the system. We used a radial basis (RBF) kernel and set the margin parameter C to 1.

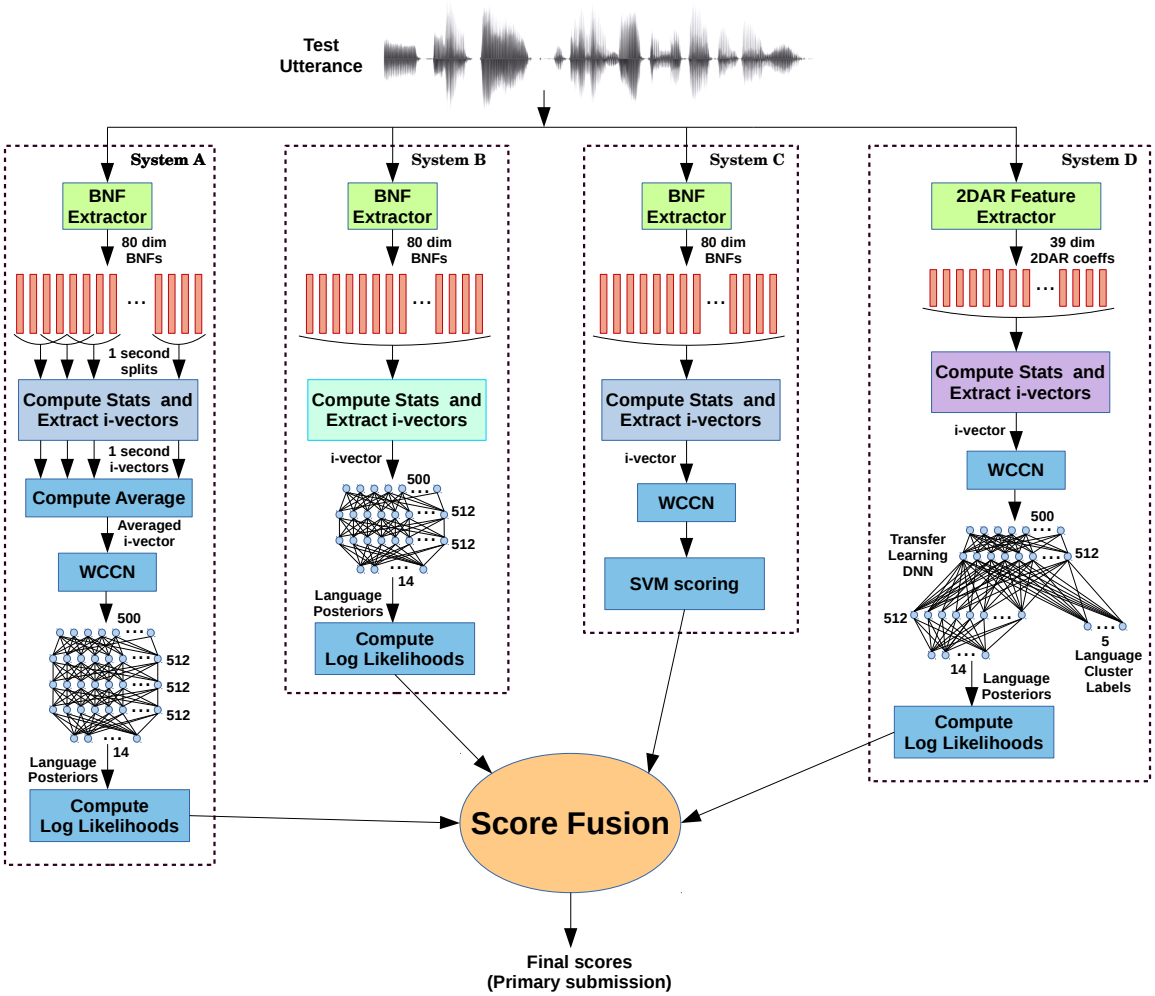

Figure 2: *Schematic illustration of individual sub-systems and the combination used in our LRE submission.*

3.3.2. *Deep Neural Networks (DNN)*

A two layer feed forward neural network (DNN) was trained using the 500 dimension i-vectors and mapped to corresponding encoded class labels. There are two hidden layers of size 512, with rectified linear unit (ReLU) non-linearity and 30% dropout for regularization. The output layer has 14 nodes with softmax non-linearity and Adam optimization algorithm was used for back propagation. The DNN posteriors were normalized to remove the language priors and a logarithm operation was used to generate the scores.

3.3.3. *Transfer learning based classification*

We also experiment with a transfer learning approach [20]. The motivation for this is drawn from the fact that dialects within each broad language cluster may benefit by sharing the representations. A two hidden layer DNN was designed with ReLU non-linearities and softmax output layers (shown in System-D of Fig. 2). There are two output layers: one with 5 nodes representing the language clusters, and another with 14 nodes representing dialects. The representations learned at the first hidden layer which is common to both outputs contain dialect and the corresponding language cluster information. The language cluster information is thus "transferred" to the output layer corresponding to the dialects. The loss function of the network is a convex combination of loss functions corresponding to dialect and language cluster outputs. In our system, we weighted the two loss functions equally.

3.4. NIST Baseline System

An i-vector baseline system implemented in python was provided by NIST LRE2017 [1], which we used for performance comparison. This system uses bottleneck features to compute i-vectors using UBM-GMM model and the languages are classified using cosine similarity clustering. The results for this system (in terms of overall C_{avg}) on the development and evaluation dataset are shown in Table 2.

3.5. Systems submitted to LRE 2017

The different combinations of language features and classifiers used are listed below and illustrated in Fig. 2. The performance of these systems are reported in Table 2.

- **System A** - An i-vector extractor pipeline was trained using the BN features from the "DATASET-C" (Sec. 2). The short-term i-vectors (Sec. 3.2.2) were extracted using 1000msec windows and 200msec hop, for each ut-

Table 2: Overall performance of the LRE submission system and individual sub-systems on the Dev and Eval datasets.

Systems	Dev C_{avg}	Eval C_{avg}
NIST baseline	0.65	0.64
System A	0.38	0.32
System B	0.34	0.29
System C	0.39	0.28
System D	0.56	0.44
Contrastive I	0.30	0.27
Contrastive II	0.34	0.29
Primary	0.30	0.23

terance from "DATASET-B" and were averaged. These averaged i-vectors processed with WCCN are used as features for language recognition. The classifier is the DNN model described in Sec. 3.3.3.

- **System B** - The i-vector extractor used was same as that of system A. Then i-vectors were extracted for every utterance in "DATASET-C", and were used to train a DNN classifier.(Sec. 3.3.2).
- **System C** - The i-vector extractor used was trained on all the utterances from the train dataset (LDC2017E22). BW statistics were computed over the entire utterance for all the files in the train dataset. The extracted i-vectors were then processed with WCCN and used to train a SVM classifier. (Sec. 3.3.1).
- **System D** - The 2D-AR based features (Sec. 3.1.2) were used as the front-end features. The i-vectors were extracted, and classified using transfer learning based DNN (Sec. 3.3.3) approach that used the language cluster information.

The system combination was trained using foCAL toolkit with a logistic regression classifier (multi-class) [14]. Various system combinations were evaluated on the Dev dataset and the best performing combination was choosen for submission. The systems submitted in the evaluation are

- Primary : Combination of all four systems (A,B,C,D).
- Contrastive I : Combination of two sub-systems (A,C).
- Contrastive II : Individual system (B).

3.6. Evaluation results

The results for LRE submission are shown in Table 2. All individual systems achieved superior performance compared to baseline. The best eval results were achieved for the primary submission which improved the C_{avg} relatively by about 54% for development data and 64% for the eval data over the corresponding NIST baseline scores.

4. Post-evaluation efforts

4.1. Linear Discriminant Analysis

Linear Discriminant Analysis [21] is a dimensionality reduction technique that tries to find a linear projection of high dimensional features into a lower dimensional space. It maximizes the ratio of between class variance to within class variance. We have used it for the dimensionality reduction of i-vectors. The 500 dimensional i-vectors were reduced to 13 dimensions and were used to train a two layer DNN model (LDA-DNN) for classification. The DNN model has 13 dimension input layer and two 128 dimension hidden layers with 50% dropout for regularization. The performance in terms of C_{avg} score for development data and the evaluation data is shown in Table 4. The use of LDA provided significant improvements in C_{avg} for both the development and evaluation datasets.

Table 3: Overall performance of systems with relevance weighted Baum-Welch statistics.

Threshold entropy H_i	C_{avg} Dev	C_{avg} Eval
DNN-50k	0.37	0.25
η =1.5	0.35	0.26
η =1.2	0.35	0.27
η =0.9	0.36	0.28
Soft weighting	0.35	0.27

4.2. Relevance weighted Baum-Welch statistics

We hypothesize that even after SAD some parts of the speech utterance carry more language relevant information than the other parts and we attempted to extract that relevance information using the short term 1000msec i-vectors. The estimation of "relevance" of a given short-term segment for language classification is somewhat challenging. In this work, we pose this as a confidence estimation problem of the classifier for the given input feature.

The 1000msec short term i-vectors (Sec. 3.2.2) were labeled individually according to the label of the corresponding utterance. These 1000msec i-vectors from train and development datasets were then used to train a 14 class feed forward deep neural network (DNN). The DNN has an input layer of 500 dimensions with 3 hidden layers having 1024 dimensions and rectified linear unit (ReLU) non-linearity. The output layer was 14 dimensional with softmax non-linearity and Adam optimization algorithm was used for back-propagation.

Once the DNN model is trained, for each non-overlapping short term i-vector $\boldsymbol{y}_j$ from every utterance in train, development and evaluation datasets, the information entropy H_j is computed from the posteriors obtained from the DNN as,

$$H_j = -\sum_{l=1}^{14} \mathrm{p}(l \,|\, \boldsymbol{y}_j) \log_2 \mathrm{p}(l \,|\, \boldsymbol{y}_j),$$

where j is the index of 1000msec i-vector. We have used the inverse of entropy as a measure of relevance of the current 1000msec i-vector for the language classification task. This approach is motivated by the use of phoneme posteriors in speech recognition [22] where the inverse of the entropy provides a confidence estimate of the classifier.

We have used the entropy measure H_i to compute a relevance parameter γ_i. The zero and first-order Baum-Welch statistics from Sec. 3.2.1 are now modified as:

$$N_c(s) = \sum_{i=1}^{H(s)} \gamma_i \, \mathrm{p}(c \,|\, \boldsymbol{x}_i)$$

$$\boldsymbol{F}_{X,c}(s) = \sum_{i=1}^{H(s)} \gamma_i \, \mathrm{p}(c \,|\, \boldsymbol{x}_i)(\boldsymbol{x}_i - \boldsymbol{\mu}_c)$$

In this case, the value of γ_i changes only at 1sec intervals. The value of γ_i is obtained from H_i in the following two ways

Table 4: Performance of the various post-eval systems and the NIST post-eval baseline in terms of C_{avg}.

Model Duration	Post-eval NIST Baseline	LDA-DNN	DNN-50k	LSTM	DNN-soft	LDA-DNN +LSTM	LDA-DNN +DNN-50k +LSTM	LDA-DNN +DNN soft +LSTM
Performance on development data (C_{avg})								
3sec	0.53	0.46	0.58	0.52	0.57	**0.41**	0.43	0.42
10sec	0.27	0.19	0.27	0.25	0.27	0.18	**0.18**	0.18
30sec	0.13	0.10	0.17	0.21	0.15	**0.10**	0.10	0.10
1000sec	0.54	0.46	0.51	0.56	0.47	0.43	0.44	**0.42**
Overall	0.37	0.31	0.37	0.38	0.35	0.28	0.28	**0.27**
Performance on evaluation data (C_{avg})								
3sec	0.35	0.36	0.44	0.45	0.46	0.34	**0.32**	0.34
10 sec	0.14	0.14	0.19	0.20	0.21	0.13	**0.13**	0.14
30 sec	0.07	0.08	0.11	0.14	0.11	0.08	**0.07**	0.08
1000sec	0.23	0.24	0.29	0.36	0.33	0.24	**0.22**	0.24
Overall	0.19	0.20	0.25	0.28	0.27	0.19	**0.18**	0.19

- Hard threshold: If the entropy of window i is too high, it is less relevant for the task of language recognition. The most straightforward strategy would be to discard the high entropy front-end features while computing the BW stats. In this case, the relevance γ_i is given by

$$\gamma_i = \mathrm{u}(\eta - H_i)$$

  where u is the unit step function and η is a threshold.

- Soft weighting: The relevance γ_i is defined as

$$\gamma_i = \begin{cases} 1 & \text{when } H_i < H_{\mathsf{min}} \\ \frac{H_{\mathsf{max}} - H_i}{H_{\mathsf{max}} - H_{\mathsf{min}}} & \text{when } H_{\mathsf{min}} \leq H_i \leq H_{\mathsf{max}} \\ 0 & \text{when } H_i > H_{\mathsf{max}} \end{cases}$$

The i-vectors were then extracted for the train, development and evaluation datasets using the modified Baum-Welch statistics (for different hard limits H_i) and a feed forward neural network was used to classify the i-vectors. The results are reported in Table 3. Here, DNN-50k is the neural network model trained on the original i-vectors from "DATASET-B". We can see from Table 3 that the above method of reweighing of Baum-Welch statistics improved the development performance. However, the evaluation performance is moderately deteriorated. The difference in variations of the C_{avg} for development and evaluation datasets can be attributed to the fact that the neural network used for computing entropy has seen data from both train and development datasets but couldn't generalize well for the evaluation dataset. While the current results were not totally promising, we hypothesize that more efforts may be needed in this direction to determine the relevance of short-term segments in a principled fashion.

4.3. Time sequence modeling of short term i-vectors

In our experiments with the System A (Sec. 3.5), we have averaged the sequence of short term i-vectors to get a single representation for the entire sequence, there by ignoring any time series information that may be present in the sequence. The evolution of these short term i-vectors over time may hold useful language discerning information. We have attempted the use of long short term memory (LSTM) recurrent neural networks (RNNs) to capture this time series information. The LSTMs [23, 24] are a special kind of RNNs that are better at learning long term temporal dependencies than the conventional RNNs. The LSTMs have a memory cell with input, output and forget gates which control the flow of information between and within the memory cell. These features make it efficient to train the network compared to the conventional RNNs.

The LSTM architecture that we used in this paper contains two layers with 512 memory cells in each layer. The input to the LSTM is the variable length 500 dimensional short term i-vector sequence. The results for the LSTM system by itself are reported in Table 4. The output from the last LSTM layer is concatenated to the mean of the input short term i-vectors. This 1012 $(500+512)$ dimension vector is then passed through a 512 dimension fully connected dense layer with ReLU non-linearity and a 14 dimensional output layer with softmax non-linearity and 25% dropout for regularization. The model was trained using stochastic gradient descent with exponentially decreasing learning rate. The results are tabulated in Table 4.

4.4. NIST Baseline system

The NIST LRE2017 organizers released a post evaluation system [1]. This system uses bottleneck features (Sec. 3.1.1) to extract 500 dimensional i-vectors (Sec. 3.2.1). The i-vectors are processed with WCCN and normalized to a range of [+1,-1]. The dimension of the i-vectors is reduced to 13 using linear discriminant analysis (LDA). It is then modeled using support vector machines (SVM) (Sec. 3.3.1). The results of this model - "Post-eval NIST Baseline" are reported in Table 4 with an overall C_{avg} score of 0.37 on the development data and 0.19 on the evaluation data.

4.5. Running Time

We also report the running time of the various systems submitted during LRE submission as well as the post-eval systems. All the LRE systems developed were executed on an Intel based machine with 32 cores and 256 GB of memory. The time taken for each of the systems (end-to-end run in a single-threaded mode for a file of duration 30sec) is reported in Table 5.

5. Discussion

The comparison of various results from the post evaluation experiments are shown in Table 4. The scores for ensemble systems are obtained by averaging the log likelihoods of the individual systems. The following are the major takeaways from the post evaluation analysis.

- The application of LDA significantly improves the per-

Table 5: Time taken to process a test file of duration 30 seconds.

Pre-eval models	Time (secs)	Post-eval models	Time (secs)
NIST pre-eval baseline	15	NIST post-eval baseline	15
System A	49	LDA-DNN	10
System B	11	DNN-50k	10
System C	10	DNN-soft	55
System D	10	LSTM	50

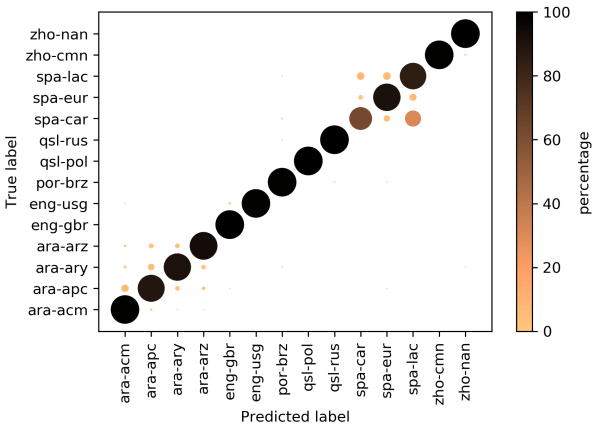

Figure 3: *Confusion matrix of the LEAP post-eval system for 30sec files from eval data*

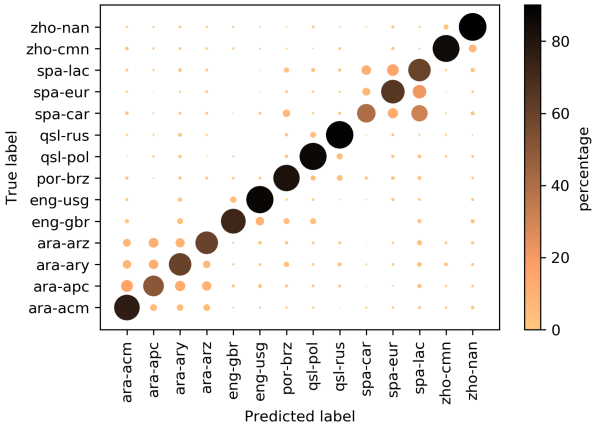

Figure 4: *Confusion matrix of the LEAP post-eval system for 3sec files from eval data*

formance of the end to end classification models as shown in second column of Table 4.

- The LSTM based models provide performances that are moderately better than the baseline models on short duration (3sec and 10sec) utterances. In the current implementation, the LDA was not applied in the LSTM based approaches.
- The combination of LSTM and LDA-DNN models resulted in a significant improvement of C_{avg} on the development dataset and a moderate improvement on the evaluation dataset over the NIST Baseline.
- The combination of LDA-DNN, DNN-50k (pre-eval system) and the LSTM model, which we refer to as the

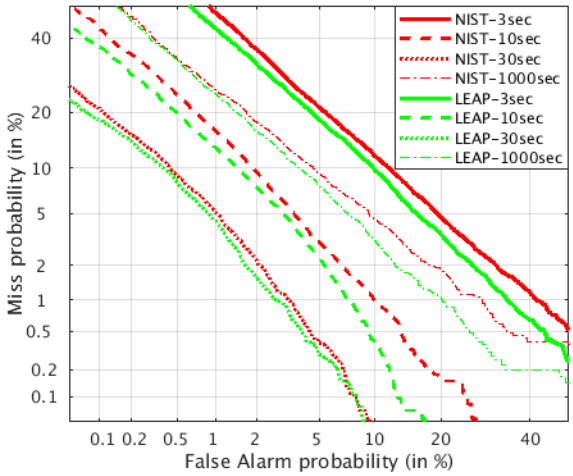

Figure 5: *Duration level DET curves comparing NIST post-eval baseline and LEAP post-eval system.*

LEAP post-eval system, gave the best evaluation performance (with C_{avg} of 0.18) as shown in column 7 of Table 4. This system improved the primary submission (reported in Table 2) relatively by about 22%.

.

In Fig. 3 and 4 we show the confusion matrix of the LEAP post-eval system on the evaluation data for 30sec and 3sec conditions respectively which are two extremes in terms of performance. As expected, the errors in the language recognition for 30sec condition are lesser, and the confusions are typically within the language clusters. However, in the case of 3sec condition the confusions are not necessarily restricted to the same language cluster. Moreover, not all dialects have the same accuracy and the Arabic and Iberian clusters contain the highest amount of the confusions while the Chinese and Slavic language clusters have the lowest error rates.

The detection error trade-off (DET) for all durations of the LEAP post-eval system are compared with the NIST post-eval baseline in Fig. 5. As seen here, the post evaluation system improves the NIST post-eval baseline on all conditions. The DET curve also highlights that the improvements are consistent in the low FA regions which may be required in many operation scenarios.

6. Summary

In this paper, we have detailed the LEAP LRE submission system. The submission systems used i-vector based approaches built using DNN bottleneck features and 2DAR features. The post evaluation efforts concentrated on dimensionality reduction (using LDA) of the i-vectors as well as the modeling of sequence of short-term (1sec) i-vectors. The LDA approach improved the evaluation submission and the LSTM based modeling of short term i-vectors shows promising results for short durations (3 and 10sec conditions). The application of post-evaluation approaches provided an improvement of 22 % relatively over the evaluation submission. In future, we plan to pursue the experiments of using short-term i-vectors using other modeling methods like time delay neural networks and attention based LSTMs. Also, the relevance based approaches will be investigated with various methods for classifier confidence estimation.

7. References

[1] Seyed Omid Sadjadi et al., "The 2017 NIST language recognition evaluation," in *Proc. Odyssey*, Les Sables dÓlonne, France, June 2018.

[2] M. A. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 1, pp. 31, 1996.

[3] Jiri Navratil, "Spoken language recognition-a step toward multilinguality in speech processing," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 6, pp. 678–685, 2001.

[4] N. Brummer, S. Cumani, O. Glembek, M. Karafiat, and P. Matejka, "Description and analysis of the Brno system for LRE2011," *Odyssey 2012-The Speaker and Language Recognition Workshop*, 2012.

[5] Mhamed Faouzi BenZeghiba, Jean-Luc Gauvain, and Lori Lamel, "Phonotactic language recognition using MLP features.," in *Interspeech*, 2012.

[6] J. Ma, B. Zhang, S. Matsoukas, S. Mallidi, F. Li, and H. Hermansky, "Improvements in language identification on the RATS noisy speech corpus," *Interspeech*, 2013.

[7] Fred Richardson, Douglas Reynolds, and Najim Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, 2015.

[8] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1695–1699.

[9] Najim Dehak, Pedro A Torres-Carrasquillo, Douglas Reynolds, and Reda Dehak, "Language recognition via i-vectors and dimensionality reduction," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[10] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[11] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[12] David Snyder, Daniel Garcia-Romero, and Daniel Povey, "Time delay deep neural network-based universal background models for speaker recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 92–97.

[13] Sriram Ganapathy, Samuel Thomas, and Hynek Hermansky, "Feature extraction using 2-d autoregressive models for speaker recognition," in *Odyssey 2012-The Speaker and Language Recognition Workshop*, 2012.

[14] Niko Brümmer, "Focal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scorestutorial and user manual," *Software available at http://sites. google. com/site/nikobrummer/focalmulticlass*, 2007.

[15] Gregory Sell and Daniel Garcia-Romero, "Speaker diarization with PLDA i-vector scoring and unsupervised calibration," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 413–417.

[16] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, number EPFL-CONF-192584.

[17] Jongseo Sohn, Nam Soo Kim, and Wonyong Sung, "A statistical model-based voice activity detection," *IEEE Signal Processing Letters*, vol. 6, pp. 1–3, 1999.

[18] Sriram Ganapathy, Sri Harish Mallidi, and Hynek Hermansky, "Robust feature extraction using modulation filtering of autoregressive models," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 8, pp. 1285–1295, 2014.

[19] Andrew O Hatch, Sachin Kajarekar, and Andreas Stolcke, "Within-class covariance normalization for svm-based speaker recognition," in *Ninth international conference on spoken language processing*, 2006.

[20] Yoshua Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 17–36.

[21] Ronald A Fisher, "The use of multiple measurements in taxonomic problems," *Annals of human genetics*, vol. 7, no. 2, pp. 179–188, 1936.

[22] Hemant Misra, Hervé Bourlard, and Vivek Tyagi, "New entropy based combination rules in hmm/ann multi-stream asr," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*. IEEE, 2003, vol. 2, pp. II–741.

[23] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins, "Learning to forget: Continual prediction with LSTM," *IET*, 1999.