BAT System Description for NIST LRE 2015

Oldřich Plchot[1], *Pavel Matějka*[1], *Radek Fér*[1], *Ondřej Glembek*[1], *Ondřej Novotný*[1], *Jan Pešán*[1], *Karel Veselý*[1], *Lucas Ondel*[1], *Martin Karafiát*[1], *František Grézl*[1], *Santosh Kesiraju*[1], *Lukáš Burget*[1], *Niko Brümmer*[2], *Albert Swart*[2], *Sandro Cumani*[3], *Sri Harish Mallidi*[4], *Ruizhi Li*[4]

(1) BUT, Speech@FIT, Czech Republic
(2) AGNITIO, South Africa
(3) Politecnico di Torino, Italy
(4) Johns Hopkins University, USA

iplchot@fit.vutbr.cz[1], niko.brummer@gmail.com[2]
sandro.cumani@polito.it[3], mallidi.harish@gmail.com[4]

Abstract

In this paper, we summarize our efforts in the NIST Language Recognition (LRE) 2015 Evaluations which resulted in systems providing very competitive performance. We provide both the descriptions and the analysis of the systems that we included in our submission. We start by detailed description of the datasets that we used for training and development, and we follow by describing the models and methods that were used to produce the final scores. These include the front-end (i.e., the voice activity detection and feature extraction), the back-end (i.e., the final classifier), and the calibration and fusion stages. Apart from the techniques commonly used in the field (such as i-vectors, DNN Bottle-Neck features, NN classifiers, Gaussian Back-ends, etc.), we present less-common methods, such as Sequence Summarizing Neural Networks (SSNN), and Automatic Unit Discovery. We present the performance of the systems both on the Fixed condition (where participants are required to use predefined data sets only), and the Open condition (where participants are allowed to use any publicly available resource) of the NIST LRE 2015.

1. Introduction

Four years have passed since the last NIST LRE in 2011 and researchers in the field have been developing and advancing language recognition technology, often utilizing the data released by NIST and LDC as well as other data from various sources. As the test sets of NIST LREs usually serve as a common benchmark in scientific publications, researchers often tune their systems to perform well on these tasks and one of the key elements in the system design is definitely a good training and calibration set. The large influence of the data on system performance is the reason that almost every research group has eventually developed its own database for training language recognition systems.

Composition of the training data can certainly have a positive or negative effect on the performance of any particular method for language identification (LID) and it is therefore very important to keep this in mind when comparing different systems for LID. One can certainly achieve impressive improvements with a small training dataset given a common baseline system and a particular method, while other may not be able to replicate his success while using his own much larger dataset.

We believe that the problem of data engineering was one of the reasons for a new scheme in NIST LRE 2015 [1]. We welcome the introduction of the *Fixed* condition where participants are required to use only predefined datasets for system development, which partially eliminates one of the differentiating factor between the published research systems. At the same time we want to be able to quantify the effect of various datasets on the performance, and this is where we also welcome the *Open* condition which is in line with the previous LRE's, where the participants could use any public resources. As we have participated in various language and speaker recognition evaluations, we are in a position to build a rather large database for LRE and present an analysis with its composition and use.

In comparison to LRE11 [2], the state-of-the-art has changed especially in the front-end. Instead of relying on standard acoustic features (such as MFCC) or outputs from various phoneme recognizers, bottleneck features (BN) [3, 4], extracted from Deep Neural Networks (DNN) trained to classify phoneme states, are used with great success. The core research regarding BN-DNNs is happening mainly in the field of Automatic Speech Recognition (ASR), where recently, a variant with multilingual training – that literally calls for being used in LRE [5] – has been developed [6]. It is obvious that we need annotated data for training of such DNNs and we show that utilizing the data from multiple languages to train a multilingual DNN can bring substantial improvements.

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0013. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government. This work was supported by the DARPA RATS Program under Contract No. HR0011-15-C-0038. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. The work was also supported by Czech Ministry of Interior project No. VI20152020025 "DRAPAK"

Having participated in both *Fixed* and *Open* conditions allows us to present not only the newly established state-of-the-art feature extraction for LID, but also the analysis of using different data and architectures for training both the feature extractors and the whole systems.

As the change in preferred feature extraction has been substantial since the last evaluation, the general system architecture has not changed much. Most of the systems being presented in this work are based on the i-vectors [7], which have de-facto became a standard for speaker recognition and have grown in popularity also for language recognition [8, 9, 10, 11]. As a classifier on top of the i-vectors, we use mainly generative Gaussian models [9, 11] that have been shown to provide results that are similar or better than those of discriminative classifiers based on Support Vector Machines (SVM) or multiclass logistic regression [9]. To compensate for poor i-vector estimates given very short test segments, we extend the Gaussian linear model from [9] by taking into account an i-vector uncertainty [12, 13, 14].

Given the nature of LRE15 and the division of the data into clusters of close languages, we were also experimenting with the architecture based on the average of cluster-dependent systems. We show that such approach turned out to be beneficial.

We also briefly present methods that can be seen as less conventional given the dominance of i-vectors in the field. We have experimented with Sequence summarizing DNNs, training them to directly classify language given the sequence of spectral-based features. Another technique that is briefly introduced is the system for automatic unit discovery in the acoustic data. The motivation behind this system was to train a bottleneck-based system without the need of any data transcriptions and train it directly on the LRE15 data. This system is in fact used as a source for automatic data tokenization and the units themselves were used as targets for training the BN-DNN.

2. Data

We have utilized all supplied training data for primary condition.

The annotated Switchboard database was used to train the bottleneck NN. This database was provided for all participants to allow the use of techniques requiring annotated speech corpora for system development.

The LRE15 training data were then split into two parts, where the first part was used to train both classifiers and parts of the i-vector system (UBM, i-vector extractor) and the second part was used as a held-out set for calibration and fusion. We denote LRE15 training data as *Fixed* set later in this document.

For the open condition, we tried to design an auxiliary dataset with an emphasis on large channel variability and amount of data. We used data from previous evaluations and other publicly available sources. We denote this as *Open* set later in this document.

2.1. Primary Data

For the primary condition, we split the LRE15 training data into two parts – train and development (dev). We have coordinated our split with MIT team to keep the possibility to experiment with cross-site combinations later in the post-evaluation period. MIT defined the split of all available data in such a way that 60% belonged to the train part and 40% to the dev. The segments belonging to the dev were further split into short cuts that contain from 3 to 30 seconds of speech. These cuts were defined by MIT as well. At this point it should be pointed out that our Voice Activity Detection (VAD) system was giving approximately 20% less speech frames than the one used by MIT for the cuts definitions.

After splitting the data and dividing the dev segments into cuts, we ended up with 3042 segments (248 hours of speech) in train set and 42295 segments (146 hours of speech) in dev set.

2.2. Data for Open condition

We designed an independent dataset for the open-data condition, where we concentrated on obtaining a large diversity in channels and a large amount of training data. We have reused data from previous NIST evaluations and part of the LRE15 training data which we believed was coming from new data collections. We have also added part of KALAKA-3 database [15] (British English, European Spanish) and human annotated part of Al Jazeera Dialectal Speech Corpus for the Arabic dialects [16].

The train part of the auxiliary data reached the size of 55142 segments (1398 hours of speech). For the dev part of this dataset, we kept the design from previous evaluations which means we were creating clusters of 3 s, 10 s and 30 s cuts from longer segments. The size of the auxiliary dev set reached 64539 segments (200 hours of speech).

Table 1: *Sources of data used for open training data condition (referred as Open dataset) and their statistics. Rows are sorted according to amounts of clean speech in Train part.*

	Train		Dev	
Database	#files	hours	#files	hours
OGI 22 languages	2270	3.5	2077	6.0
unknown broadcast news	64	3.9	72	0.2
OGI multilingual	1640	8.8	-	-
KALAKA-3	409	9.9	4774	10.7
Radio Free Europe	476	11.2	705	2.3
Foreign Accented English	4913	15.6	-	-
HKUST Mandarin	276	17.2	-	-
SpeechDat-East	6202	30.6	200	1.1
NIST LREs	1124	38.2	36391	119.2
Al Jazeera Dialectal Speech Corpus	9306	55.2	2200	7.2
NIST SREs	3657	96.0	-	-
Callfriend	838	144	4257	11.5
Callhome	-	-	1819	5.8
Levantine Arabic and Iraqi CTS[1]	3606	168	7299	26.1
Fisher English, Arabic	5804	337	1393	4.5
VOA broadcasts	14555	458	3280	5.8
Total	55142	1398	64539	200

2.3. Additional Data

We used also transcribed data from the IARPA BABEL program[2] for multilingual training of bottleneck features and Region Dependent Transform (RDT) features used in the open training data condition.

This database simulates a case of what one could collect in limited time from a completely new language. It consists

[1]LDC2007S01, LDC2006S45, LDC2005S14, LDC2005S07

[2]The IARPA Babel Program (http://www.iarpa.gov/index.php/research-programs/babel)

mainly of telephone conversational speech, but scripted recordings as well as far field recordings are present too. The data are available only to the project participants, but are released also within the Open KWS evaluations[3] organized by NIST. We used 17 languages from this program for multilingual bottleneck neural network training and first 11 languages (Year1 and Year2) for RDT features and VAD neural network training.

3. Voice Activity Detection

Our multi-lingual VAD was originally developed for the IARPA BABEL program and consists of two carefully designed parts: a neural network (NN) which produces per-frame scores, and a post-processing stage which builds the segments based on the scores. By the term "multi-lingual", we mean splitting the last softmax layer of NN into several blocks, where each block accommodates training targets from one language [6].

The NN was trained on the Year1+Year2 Full Language Pack train-sets (11 languages), which results in 842 hours of audio. The input dimension is 288, while there are 2 hidden layers, each of 400 sigmoid neurons, and the final softmax layer has 2 outputs, corresponding to the classes: speech, non-speech. The NN has 277k parameters. The training targets were prepared by mapping from the mono-lingual forced-alignments (generated with GMM/HMM model).

The input features for the NN consist of 15 log-Mel filterbank outputs and 3 Kaldi-pitch features [17]. We apply per-speaker mean and variance normalization estimated on the whole unsegmented recordings. Then we apply frame splicing with 31 frame-long context, where the temporal trajectory of each feature is scaled by a Hamming window and reduced to 16 dimensions by Discrete Cosine Transform. The final 288-dimensional features are globally mean and variance normalized on the NN input.

In the post-processing, we bypass the NN output softmax function (allowing us to interpret the outputs as log-likelihoods), then we convert the two outputs to logit-posteriors, and then we smooth the score by averaging over consecutive 31 frames. In the final step, the speech segments were extracted by thresholding the posterior at the value of -0.5.

4. Feature extraction

Majority of the feature extraction methods in our work are well-known and described in other literature, however, we have also used some novel approaches. If not stated otherwise, only parts of utterances labeled as speech (according to the VAD as described above) were selected for further processing.

4.1. Stacked Bottleneck Features (SBN)

A bottleneck feature vector is generally understood as a by-product of forwarding a primary input feature vector through a NN and reading off the vector of values at the bottleneck layer. We have used a cascade of two such NNs for our experiments. The output of the first network is stacked in time, defining context-dependent input features for the second NN, hence the term Stacked Bottleneck Features (SBN). The NN input features are 24 log Mel-scale filter bank outputs augmented with fundamental frequency features from 4 different f0 estimators (Kaldi, Snack, and other two according to [18, 19]). Together, we have 13 f0-related features, see [5] for details. In summary, 24 log filter bank outputs and 13 fundamental frequency features form 37-dimensional feature vectors.

Mean subtraction is applied at the utterance level. Hamming window followed by DCT consisting of 0th to 5th base are applied on the time trajectory of each parameter resulting in $(24 + 13) \times 6 = 222$ coefficients on the first stage NN input.

The dimensionality of the bottleneck layer was set to 80. The dimensionality of the other hidden layers was set to 1500. The bottleneck outputs from the first NN are sampled at times $t-10$, $t-5$, t, $t+5$ and $t+10$, where t is the index of the current frame. The resulting 400-dimensional features are inputs to the second stage NN with the same topology as first stage. The 80 bottleneck outputs from the second NN (referred as SBN) are the final features.

The NN in both stages were trained on Switchboard database (audio and transcriptions). We denote these features either as *SBN-SWB1* (our in-house NN training toolkit was used) or *SBN-SWB1-KALDI* (NN trained using the Kaldi toolkit).

4.1.1. Multilingual training for open data condition

For the open training data condition, we were not limited by a requirement to use only Switchboard data to train bottleneck NN. Thus we were allowed to use additional corpora with transcriptions to train bottleneck features multilingually. This was proven to be superior to monolingually trained features [5]. We used all 17 languages from the IARPA Babel project to train multilingual stacked bottleneck NN, see Section 2.3. We denote these features as *SBN-ML17*.

For the Open set condition we also used another flavor of multi-lingual bottleneck features denoted as *MultilangRDT*. These features were directly borrowed from ASR domain [20]. They are based on feature level fusion of two feature streams: 30 dimensional SBN features trained on 10 languages from BABEL data and PLP+D+A+T projected to 39 dimensions using HLDA. This feature stream is fed to Region Dependent Transform (RDT) [21] transform performing dimensionality reduction to 69 dimensions. RDT is trained on 11 languages from BABEL data.

We adopted the multilingual training scheme with block-softmax, which divides the output layer into parts according to individual languages. During training, only the part of the output layer corresponding to the language the given target belongs to, is activated. Detailed description of multilingual training can be found in [6].

Note that we could not have used this approach for primary condition (utilizing the data for 20 languages from the training part of LRE15) because of missing transcriptions. Alternate approach, where we tried to automatically discover language independent speech units for these languages and train monolingual bottleneck NN on top of them is described in the next section.

4.2. Automatic unit discovery

The acoustic unit discovery is based on a non-parametric Bayesian approach. The acoustic units are assumed to be generated by a mixture of HMM/GMM and the mixture itself is assumed to be a realization of Dirichlet process, with a HMM/GMM as a base distribution. A similar model has been described in [22]. The set of HMMs is embedded into a phone loop where the transition probabilities are the weights of the infinite mixture model. The training procedure can be seen as simultaneously fitting a phone loop model to the data and estimating the number of phones in the looping. The training of the model was done with the Variational Bayes (VB) approach using the stan-

[3]http://www.nist.gov/itl/iad/mig/openkws.cfm

dard mean-field approximation and truncating the infinite mixture model by putting a hard limit on the maximum number of components. Number of units was set to 600 and they were estimated on pooled training data. Bottleneck NN was then trained on top of them using the Kaldi toolkit. We denote these bottleneck features as *SBN80-AUTO-KALDI*.

4.3. Shifted Delta Cepstra (MFCCSDC)

We computed 56-dimensional feature vectors from frames of size 25 ms with 10 ms shift. The feature vector is a concatenation of SDC-7-1-3-7 vector and 7 MFCC coefficients (including C0). Cepstral mean and variance normalization and RASTA filtering were applied before SDC stacking.

4.4. Phone Log-Likelihood Ratio features (PLLR)

PLLRs [23] are frame-level features, computed from the frame-by-frame phone posterior probabilities provided by the Neural Network. NN was trained on Switchboard to produce phoneme state posterior probabilities - there were 40 phonemes, each with 3 states, which gives us 120 outputs. We take logit on these posteriors and PCA to reduce dimensionality to 80 dimensions.

We did not use PLLR on its own but we performed a feature-level fusion (concatenation) with MFCCSDC features. This is further denoted as *MFCCSDC+PLLR* features.

5. i-vector model

If not stated otherwise, all the i-vector based system we used are using UBM with 2048 diagonal-covariance components trained on the balanced training set (up to 15 h of randomly selected training utterances per language). The total variability matrix for i-vector extraction was trained in 5 iterations and its rank was set to 600.

Let us briefly recall the definition of i-vectors. The i-vector model constrains the GMM super-vector, representing the characteristics of a given speech segment, to live in a single small-dimensional subspace according to:

$$\mathbf{s} = \mathbf{u} + \mathbf{T}\mathbf{w} \,, \tag{1}$$

where $\mathbf{u}$ is the super-vector stacking the means of the Universal Background Model (UBM), composed of C components of dimension F. $\mathbf{T}$ is a low-rank matrix spanning the i-vector subspace, and $\mathbf{w}$ is a realization of a latent variable $\mathbf{W}$, of size M, having a standard normal prior distribution. Given $\mathbf{T}$ and a set of τ feature vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\tau\}$, the posterior distribution of $\mathbf{W}$ given $\mathcal{X}$ can be computed as:

$$\mathbf{W}|\mathcal{X} \sim \mathcal{N}\left(\boldsymbol{\mu}_\mathcal{X}, \boldsymbol{\Gamma}_\mathcal{X}^{-1}\right) \,, \tag{2}$$

where

$$\boldsymbol{\Gamma}_\mathcal{X} = \mathbf{I} + \sum_{c=1}^{C} N_\mathcal{X}^{(c)} \mathbf{T}^{(\mathbf{c})^\mathrm{T}} \boldsymbol{\Sigma}^{(\mathbf{c})^{-1}} \mathbf{T}^{(c)}$$
$$\boldsymbol{\mu}_\mathcal{X} = \boldsymbol{\Gamma}_\mathcal{X}^{-1} \mathbf{T}^\mathrm{T} \boldsymbol{\Sigma}^{-1} \mathbf{f}_\mathcal{X} \,.$$

In these equations, $N_\mathcal{X}^{(c)}$ are the zero–order statistics estimated on the c-th Gaussian component of the UBM for the set of feature vectors in $\mathcal{X}$, $\mathbf{T}^{(c)}$ is the $F \times M$ sub-matrix of $\mathbf{T}$ corresponding to the c–th mixture component such that

$$\mathbf{T} = \left(\mathbf{T}^{(1)\mathrm{T}}, \ldots, \mathbf{T}^{(C)\mathrm{T}}\right)^\mathrm{T} ,$$

and $\mathbf{f}_\mathcal{X}$ is the super-vector stacking the first–order statistics $\mathbf{f}_\mathcal{X}^{(c)}$, centered around the corresponding UBM means:

$$\mathbf{f}_\mathcal{X}^{(c)} = \sum_t \left(\gamma_t^{(c)} \mathbf{x}_t\right) - N_\mathcal{X}^{(c)} \mathbf{m}^{(c)} \,, \tag{3}$$

$\boldsymbol{\Sigma}^{(c)}$ is the UBM c–th covariance matrix, $\boldsymbol{\Sigma}$ is a block diagonal matrix with matrices $\boldsymbol{\Sigma}^{(c)}$ as its entries, and $\gamma_t^{(c)}$ is the occupation probability of feature vector $\mathbf{x}_t$ for the c-th Gaussian component.

In the i-vector paradigm, an utterance is represented as the MAP point–estimate $\boldsymbol{\mu}_\mathcal{X}$ of the i-vector posterior distribution, and the term i-vector usually refers to this point–estimate. We are however also interested in exploiting the additional information conveyed by the uncertainty in the i-vector extraction process, represented by the i-vector posterior covariance $\boldsymbol{\Gamma}_\mathcal{X}^{-1}$. Thus, we will explicitly refer to $\boldsymbol{\mu}_\mathcal{X}$ as the "i-vector point–estimate", to avoid confusion with the i-vector posterior distribution. In order to increase readability, in the following we will also drop the reference to the feature set $\mathcal{X}$ from $\boldsymbol{\mu}_\mathcal{X}$ and $\boldsymbol{\Gamma}_\mathcal{X}$.

5.1. Gaussian models for language recognition

Generative modeling of i-vector point–estimates for language recognition has proven to be an effective alternative to discriminative classifiers based on Logistic Regression or Support Vector Machines. In [9], we have proposed a simple linear classifier based on Gaussian distributions which provides accuracies similar to those of linear discriminative approaches. The model assumes that, for each language, the corresponding i-vector point–estimates $\boldsymbol{\mu}_i$ are generated according to:

$$\boldsymbol{\mu}_i = \mathbf{m}_\ell + \boldsymbol{\varepsilon}_i \,, \tag{4}$$

where $\mathbf{m}_\ell$ is a language–dependent mean vector and

$$\boldsymbol{\varepsilon}_i \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Lambda}^{-1}\right) \tag{5}$$

represents a (language–independent) residual. The model parameters can be easily obtained by Maximum–Likelihood estimation. The class–conditional log–likelihood for $\boldsymbol{\mu}_i$ given language ℓ can be computed as:

$$\log P(\boldsymbol{\mu}_i|\ell) = \frac{1}{2}\log|\boldsymbol{\Lambda}| - \frac{1}{2}(\boldsymbol{\mu}_i - \mathbf{m}_\ell)^T \boldsymbol{\Lambda} (\boldsymbol{\mu}_i - \mathbf{m}_\ell) + k \,, \tag{6}$$

where k is a data–independent constant. We denote this classifier as GLC.

5.2. Gaussian models and i-vector uncertainty

In [12], we have shown that a Gaussian model employed for closed-set LID can be interpreted as an approximation of the PLDA model. This allows us to account for i-vector uncertainty following exactly the same approach that has been used for speaker recognition [13, 14]. In particular, the i-vector uncertainty can be taken into account through the modified PLDA model:

$$\boldsymbol{\mu}_i = \mathbf{m} + \mathbf{U}\mathbf{y} + \overline{\boldsymbol{\varepsilon}}_i \,, \tag{7}$$

where the residual term $\boldsymbol{\varepsilon}_i$ has been replaced by the term $\overline{\boldsymbol{\varepsilon}}_i$, with an utterance–dependent distribution given by:

$$\overline{\boldsymbol{\varepsilon}}_i \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Lambda}_{eq,i}^{-1}\right) \,,$$
$$\boldsymbol{\Lambda}_{eq,i}^{-1} = \left(\boldsymbol{\Lambda}^{-1} + \boldsymbol{\Gamma}_i^{-1}\right) \,, \tag{8}$$

where $\mathbf{\Gamma}_i$ is the i-vector posterior precision. Model parameters can be estimated through Expectation–Maximization following the approach in [24]. For long training utterances, however, i-vector covariances can be safely neglected during training. Moreover, we are interested only in closed–set detection in LRE15, and training utterances are sufficiently long, therefore model can be simplified as:

$$\boldsymbol{\mu}_i = \mathbf{m}_\ell + \bar{\boldsymbol{\varepsilon}}_i \ . \tag{9}$$

The class–conditional log–likelihoods $\log P(\boldsymbol{\mu}_i|\ell)$ for a test i-vector mean $\boldsymbol{\mu}_i$, with associated i-vector posterior covariance $\mathbf{\Gamma}_i^{-1}$, given language ℓ, can be computed as:

$$\begin{aligned}\log P(\boldsymbol{\mu}_i|\ell) = &-\frac{1}{2}(\boldsymbol{\mu}_i - \mathbf{m}_\ell)^T \left(\mathbf{\Lambda}^{-1} + \mathbf{\Gamma}_i^{-1}\right)^{-1}(\boldsymbol{\mu}_i - \mathbf{m}_\ell) \\ &-\frac{1}{2}\log\left|\mathbf{\Lambda}^{-1} + \mathbf{\Gamma}_i^{-1}\right| + k \ , \end{aligned} \tag{10}$$

where k is a data–independent constant. We denote this classifier as FPGLC.

6. NN classifiers

6.1. Neural Network Back-end

In [25, 4], we have shown that Neural network (NN) classifier significantly outperforms Logistic Regression on noisy data. We used the Python Theano library to train NNs to map i-vectors to language posteriors. We configured all our NNs as 3-layered (input, hidden, and output), with the input layer taking the i-vectors, and with softmax activation function in the output layer, generating posteriors for all 20 language classes. Historically, we trained 3 NNs with the number of hidden nodes set to 300, 400, 500. We adopted this approach to increase robustness, accuracy, and to avoid the over fitting problem. We can see this as a variant of bootstrap aggregating (model bagging). Finally, we take the log of the arithmetic average posterior as the final score.

6.2. Sequence Summarizing Neural Network

For Sequence Summarizing setup, we use 40 Mel filter-bank features, with frame context of ± 15 frames. Stacked frames are filtered with 16 Hamming-weighted DCT bases, which gives us a vector of 496 dimensions. Utterance based CMVN is applied before input to the NN.

General form of the experimental setup is shown in Fig. 1. We were experimenting with various positions of summarizing layer. For training we further divided training set into two subsets. 2729 utterances for training and 313 utterances for cross-validation with the same proportions of languages in both subsets.

6.2.1. Training

We used two approaches for training the system for LRE15. The first, *per-cluster* method, exploits the way how the NIST LRE15 was evaluated. We trained six cluster-dependent systems and afterwards concatenated their outputs to obtain final score matrix. The second approach was traditional and system was trained for all languages in one network.

We can see that in case of *per-cluster* training, each network sees only a small fraction of the whole training set. This situation brought in the concern of over-fitting of per-cluster systems.

Final system consists of *per-cluster* trained system with one hidden layer of size 1024 and $\tanh$ non-linearity. The *summarizing* layer which computes an average of the whole utterance is placed after the hidden layer. After the summarizing layer follows a linear layer connected to a *softmax* output.

7. Calibration and fusion

After the first stage, where we performed reduction from i-vectors or sequences of acoustic features to scores, we continued with other two stages that do *pre-calibration* and *fusion* in score-space and are both trained on the dev subset of either the *Fixed* or *Open* dataset. Both stages are implemented by multiclass logistic regression [26].

Our logistic regression solutions to calibration and fusion were simple, to avoid over-training. For *pre-calibration*, an individual system has a trainable scale factor and an offset vector. In *fusion*, every system gets a single trainable scale factor, while every language gets a trainable score offset. The parameters are trained via optimizing prior-weighted multiclass cross-entropy.

We used two kinds of priors: *Uniform prior* (flat) over all 20 languages for *pre-calibration*, and a *Cluster prior* for fusion. For the data of each cluster, we used a cluster-specific prior, with zero probabilities for out-of-cluster languages and equal weights within the cluster. As an implementation, we used a modified version of the toolkit in [27].

After the fusion, the scores belonging to individual language clusters were used as language log-likelihoods to make minimum-expected-cost Bayes decisions for the cost function as prescribed by the LRE15 evaluation plan.

Later on in this document we report starred versions of the development or evaluation set scores (dev* and eval*). This means that instead of being trained on separate heldout calibration set, the calibration parameters were trained on that set itself.

7.1. Cluster-dependent system fusion

Some of our systems use cluster dependent subsystems [4] fused into one system by means of a simple average of their scores, which simplifies the development and provides sufficient robustness. Such system is then denoted by ”-CD” at the end of the system name.

Basically it is a simple fusion of 6 (as there are 6 language clusters) i-vector based systems, where the individual UBMs are trained only on data belonging only to the particular cluster. In this architecture, a UBM with diagonal covariance matrices performs better than its full covariance variant due to limited training data that is available per cluster. The training data for T-matrix were however common for all individual cluster-dependent subsystems.

8. Results (Fixed Data Condition)

Looking at the results in Table 2, we observe that systems based on i-vectors and the two variants of Gaussian linear classifier form the base of our submission. The best performing systems (1,2,7) are based on the state-of-the-art SBN feature extraction. In fact a single system (1) is better than the whole primary fusion. The effect of using i-vector uncertainty can be seen by comparing systems (1) and (7). System (1) which uses the uncertainty performs better on evaluation data. This may be caused by the system obtaining better calibration.

Alternative i-vector systems (3,4) perform much worse, but

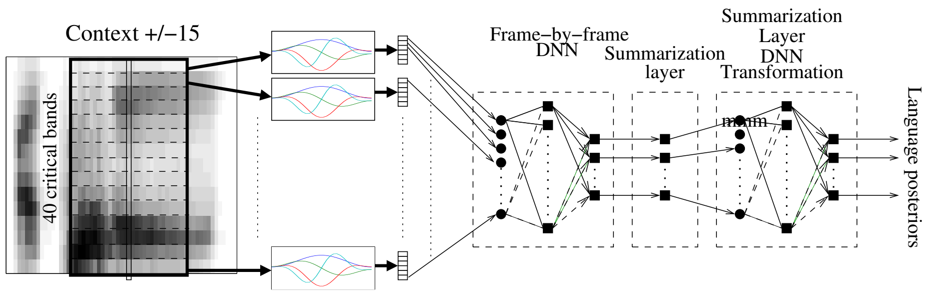

Figure 1: *Structure of Sequence Summarizing Neural Network.*

we believed that they can contribute to the fusion by bringing more diversity compared to the three very similar bottleneck-based systems. System (3) is fairly standard and is based on a feature level fusion of phonetic features with MFCC-SDC that were both shown to achieve very good results in the past. System (4) on the other hand is an attempt to bring another bottleneck system into the *fixed* training condition. Unfortunately we must conclude that at this point this approach is not even close to the classical bottleneck system and in practice, where we can usually use annotated database for DNN training, it does not have much practical use.

The NN-based systems (5,6) also provide some fresh blood into our fusion mix and especially the variant with extracting the hidden layers-related super-vectors and Gaussian linear classifier looks promising. The direct approach of classifying the language right away from the sequence of acoustic features however performs by far worse.

Analyzing the fusions in the *fixed* condition, we can see a different behavior on our development set than on actual eval. Clearly, we had problems with over-training to our development data and then obtaining poor results in fusion.

Table 2: *Results of individual systems taken to fusion for* **fixed** *training data condition. Last rows show the performances of our primary and alternate fused systems.*

		$C_{avg} \times 100$	
	System / Classifier	dev*	eval
1	SBN80-SWB1-KALDI-CD / FPGLC	2.41	16.9
2	SBN80-SWB1-CD / NN	2.80	19.9
3	MFCCSDC+PLLR-CD / GLC	4.72	22.0
4	SBN80-AUTO-KALDI / FPGLC	5.46	27.0
5	SSNN (alt. 2)	10.5	35.0
6	SSNN-layersPCA / GLC	3.41	30.4
7	SBN80-SWB1-KALDI-CD / GLC (alt. 3)	2.31	18.5
	1+2+3+4 (primary)	1.90	18.1
	1+2+3+4+5+6 (alt. 1)	1.24	19.4

8.1. Comparison of different features

In Table 3 we compare different features by training an i-vector based systems with identical parameters varying only these input features. The comparison is performed with 2048-dimensional *full*-covariance UBM, 600-dimensional i-vectors and Gaussian linear classifier.

On the development set, we can clearly see the power of SBN-based systems and a shift from the last LRE state-of-the-art. The dominance of bottlenecks is not that clear on the evaluation set, but if we compare the results considering the best calibration that we can get (eval*), then the difference is still substantial. We can also observe another step-down in error-rate when utilizing the multilingual bottleneck features.

If we look at per-cluster results of all systems (not shown here), we can clearly see that NN-based features (like PLLR and monolingual bottlenecks) perform suspiciously well on English language cluster. This is obviously caused by the fact, that they all were trained using Switchboard English database.

Table 3: *Performance of different features. Comparison was done using i-vector system (2048 full-covariance components, 600 dim. i-vectors and GLC classifier). ($^+$) means that this system violates fixed training data condition (post-eval analysis).*

	$C_{avg} \times 100$		
Features	dev*	eval	eval*
SBN80-AUTO-KALDI	5.4	28.9	24.1
MFCCSDC	6.3	23.8	21.5
MFCCSDC+PLLR	4.5	22.1	19.1
MultilangRDT$^+$	2.9	20.1	16.4
SBN80-SWB1-KALDI	2.9	20.1	16.2
SBN80-ML17$^+$	2.2	16.1	12.3

8.2. Analysis of cluster dependent system fusion

Here we will analyze a different scheme of building a LID system on top of i-vectors while taking into account presumably acoustically similar clusters of data. We train cluster-dependent UBMs to cover in detail the acoustic space of individual clusters. The baseline is the first row of Table 4, which is a single system without any cluster dependent architecture. Then we continue with the same system, just the UBM is diagonal instead of full-covariance. With this system we achieve almost the same result on development set, while getting some improvement on the evaluation data. This can be caused by less over-training on the development set given the fact that we have much less parameters in the diagonal UBM.

The last two systems are already cluster dependent. First is using a full-covariance UBM, which is again worse than the diagonal, but this time also on the development set. Here, we believe this is caused by a low amount of data for training individual UBMs.

Table 4: *Results for single and cluster-dependent versions of i-vector system with either full or diagonal covariance UBM. All systems use SBN80-SWB1-KALDI features.*

		$C_{avg} \times 100$		
Cluster dep. system	UBM	dev*	eval	eval*
no	full	2.87	20.09	16.21
no	diag	2.89	19.29	15.74
yes	full	2.48	19.67	15.39
yes	diag	2.31	18.48	14.86

9. Results (Open Data Condition)

There were just a few teams which participated in open training data condition. Our motivation was two-fold. Firstly, we were able to take all data from previous evaluations and plug it in. We were also aware of two new databases: KALAKA-3 and Al Jazeera Dialectal Speech Corpus, that contain some of the target languages. And secondly, besides using more data for system training, we could have used our multi-lingually trained bottleneck features (see Section 2.3).

Here we show the analysis of the gains we obtained by using additional data. Please, see Section 2.2 for detailed description of the Open training dataset.

The results of our submissions are in Table 5. As we can see, systems trained on Open set show higher error-rates on dev, but give better eval performance compared to what we have seen for the fixed data condition. We attribute this to the increased difficulty of the Open development set. It is probably closer to the LRE15 evaluation data and because of that the calibration and fusion stages work as we would expect.

The difference between our primary and alternate submission (last two rows of Table 5) is in the inclusion of the SSNN system in the fusion. The SSNN system was beneficial in fusion in preliminary experiments, so we included it in our primary submission (although these experiments have shown to be misleading later).

Table 5: *Results of individual systems present in the fusion for* **open** *training data condition. Last two rows show the performance of our primary and alternate fusions.*

		$C_{avg} \times 100$	
	System / Classifier	dev*	eval
1	ML17-SBN-CD / GLC (alt. 3)	8.8	13.9
2	MultilangRDT / GLC	10.4	13.6
3	SBN80-SWB1-KALDI-CD / GLC	10.3	17.6
4	MFCCSDC+PLLR-CD / NN	12.7	21.4
5	SNB80-AUTO-KALDI / NN	15.6	25.0
6	SSNN (alt. 2)	30.0	41.3
	1+2+3+4+5+6 (primary)	7.1	14.1
	1+2+3+4+5 (alternate 1)	7.1	14.1

9.1. Effects of additional training data

After the evaluation key was released, we have done some additional experiments to see in more detail the gains from using the Open dataset.

Because of the lack of time, we did not train the cluster dependent systems (i.e. systems 1, 3 and 4 from Table 5) properly from scratch on the Open set. Instead, we reused the frontend parts (UBM and total variability matrix) from the Fixed condition and retrained only the backend on the Open dataset. So these systems use the Open data collection only in the backend stage.

For this reason, we show in Table 6 the results of two i-vector systems trained completely (i.e. both frontend and backend) on either Fixed or Open dataset with our best bottleneck features. It is important to see the per-cluster results. For some clusters, there is improvement, while for other clusters there is a degradation. Unexpectedly, we can see a degradation for the Arabic and English cluster, while we originally thought it will perform better thanks to extra data from Al Jazeera Dialectal Speech Corpus and KALAKA-3 database. We will need to do further analysis to explain this surprise.

On the other hand, we are no longer getting almost random performance for French cluster. This shows nicely how much the state-of-the-art technology suffers in mismatched scenario. Note also the fact, that this improvement on French cluster accounts for the main part of the difference between the average C_{avg} for the two systems from Table 6: 10.5% (Fixed set training without considering French) vs. 9.9% (Open set training without considering French).

Table 6: *Effects of additional data sources for open training data condition. See Section 2.2 for detailed information about the Open training set. Comparison was performed using i-vector system (2048 full-covariance components, 600 dim. i-vectors and GLC classifier) on top of SBN-ML17 features.*

	eval $C_{avg} \times 100$						
Training set	avg	ara	chi	eng	fre	ibe	sla
Fixed	16.1	15.4	8.7	8.9	43.7	17.5	2.2
Open	11.9	15.9	4.8	9.2	21.7	17.0	2.6

10. Conclusions

In this work, we have described our efforts in the NIST LRE 2015. The most difficult part of this evaluation was to deal with limited amount of data and the results show that the proper analysis in this direction is necessary.

We have built over 20 systems for this evaluation. We have experimented with de-noising NN, automatic unit discovery, different flavors of phonotactic systems, backends, sizes of i-vector systems, feature sets, BN features or frame level language classifiers. We used up to 6 systems in the fusion. The performance of our best system reached C_{avg} of 16.9% on the fixed training data condition and 13.9% (11.9% after post-evaluation analysis) on the open training data condition.

11. References

[1] "The 2015 NIST Language Recognition Evaluation Plan (LRE15)," http://www.nist.gov/itl/iad/mig/upload/LRE15_EvalPlan_v23.pdf.

[2] "The 2011 NIST Language Recognition Evaluation Plan (LRE11)," http://www.nist.gov/itl/iad/mig/upload/LRE11_EvalPlan_releasev1.pdf.

[3] Song et al., "I-vector representation based on bottle neck feature for language identification," in *IEEE Electronics Letters*, 2013.

[4] Pavel Matějka, Le Zhang, Tim Ng, Harish Sri Mallidi, Ondřej Glembek, Jeff Ma, and Bing Zhang, "Neural network bottleneck features for language identification," in *Proceedings of Odyssey 2014*. 2014, vol. 2014, pp. 299–304, International Speech Communication Association.

[5] Radek Fér, Pavel Matějka, František Grézl, Oldřich Plchot, and Jan Černocký, "Multilingual bottleneck features for language recognition," in *Proceedings of Interspeech 2015*. 2015, vol. 2015, pp. 389–393, International Speech Communication Association.

[6] K. Vesely, M. Karafiat, F. Grezl, M. Janda, and E. Egorova, "The language-independent bottleneck features," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*, Dec 2012, pp. 336–341.

[7] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front–end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[8] Najim Dehak, Pedro A Torres-Carrasquillo, Douglas A Reynolds, and Reda Dehak, "Language recognition via i-vectors and dimensionality reduction," in *Proceedings of Interspeech 2011*, 2011, pp. 857–860.

[9] David González Martínez, Oldřich Plchot, L. Burget, O. Glembek, and P. Matějka, "Language recognition in ivectors space," in *Proceedings of Interspeech 2011*, 2011, pp. 861–864.

[10] Niko Brummer et al., "Description and analysis of the brno276 system for LRE2011," in *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, 2012, pp. 216–223.

[11] Alan McCree, "Multiclass discriminative training of i–vector language recognition," in *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, Joensu, Finland, 2014.

[12] Sandro Cumani, Oldřich Plchot, and Radek Fér, "Exploiting i–vector posterior covariances for short–duration language recognition," in *Proceedings of Interspeech 2015*, 2015.

[13] S. Cumani, O. Plchot, and P. Laface, "On the use of i-vector posterior distributions in probabilistic linear discriminant analysis," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 846–857, 2014.

[14] S. Cumani, O. Plchot, and P. Laface, "Probabilistic Linear Discriminant Analysis of i–vector posterior distributions," in *Proceedings of ICASSP 2013*, 2013, pp. 7644–7648.

[15] Luis Javier Rodríguez-Fuentes, Mikel Penagarikano, Amparo Varona, Mireia Diez, and Germán Bordel, "Kalaka-3: a database for the assessment of spoken language recognition technology on youtube audios," *Language Resources and Evaluation*, pp. 1–23, 2015.

[16] Samantha Wray and Ahmed Ali, "Crowdsource a little to label a lot: labeling a speech corpus of dialectal Arabic," in *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*. 2015, pp. 2824–2828, ISCA.

[17] P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, "A pitch extraction algorithm tuned for automatic speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 2494–2498.

[18] David Talkin, "A robust algorithm for pitch tracking (RAPT)," in *Speech Coding and Synthesis*, W. B. Kleijn and K. Paliwal, Eds., New York, 1995, Elseviever.

[19] Kornel Laskowski and Jens Edlund, "A Snack implementation and Tcl/Tk interface to the fundamental frequency variation spectrum algorithm," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010.

[20] František Grézl and Martin Karafiát, "Adapting multilingual neural network hierarchy to a new language," in *Proceedings of the 4th International Workshop on Spoken Language Technologies for Under- resourced Languages SLTU-2014. St. Petersburg, Russia, 2014*. 2014, pp. 39–45, International Speech Communication Association.

[21] Bing Zhang, Spyros Matsoukas, and Richard Schwartz, "Recent progress on the discriminative region-dependent transform for speech feature extraction," in *in Interspeech06*, 2006.

[22] Chia-ying Lee and James Glass, "A nonparametric bayesian approach to acoustic model discovery," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, Stroudsburg, PA, USA, 2012, ACL '12, pp. 40–49, Association for Computational Linguistics.

[23] M. Diez, A. Varona, M. Penagarikano, L.J. Rodriguez-Fuentes, and G. Bordel, "On the use of phone log-likelihood ratios as features in spoken language recognition," in *Spoken Language Technology Workshop (SLT)*, Miami, Florida USA, 2012.

[24] P. Kenny, T. Stafylakis, P. Ouellet, M.J. Alam, and P. Dumouchel, "PLDA for speaker verification with utterances of arbitrary duration," in *Proceedings of ICASSP 2013*, 2013, pp. 7649–7653.

[25] Pavel Matějka, Oldřich Plchot, Mohammad Mehdi Soufifar, Ondřej Glembek, Fernando Luis D'Haro, Karel Veselý, František Grézl, Jeff Ma, Spyros Matsoukas, and Najim Dehak, "Patrol team language identification system for DARPA RATS P1 evaluation," in *Proceedings of Interspeech 2012*. 2012, vol. 2012, pp. 1–4, International Speech Communication Association.

[26] Christopher M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 2007.

[27] "MATLAB Toolkit for Albayzin 2012 LRE," http://sites.google.com/site/albayzin2012lre.