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

Analysis of BUT-PT Submission for NIST LRE 2017

Oldřich Plchot[1], Pavel Matějka[1,4], Ondřej Novotný[1], Sandro Cumani[2], Alicia Lozano-Diez[1,3], Josef Slavicek[4,1], Mireia Diez[1], František Grézl[1], Ondřej Glembek[1], Kamsali Veera Mounika[1], Anna Silnova[1], Lukáš Burget[1], Lucas Ondel[1], Santosh Kesiraju[1], Johan Rohdin[1]

(1) Brno University of Technology, Speech@FIT and IT4I Center of Excellence, Czechia
(2) Politecnico di Torino, Italy
(3) Audias-UAM, Universidad Autonoma de Madrid, Spain
(4) Phonexia, Czech Republic

{iplchot|matejkap}@vut.cz[1], sandro.cumani@polito.it[2]
alicia.lozano@uam.es[3], josef.slavicek@phonexia.com[4]

Abstract

In this paper, we summarize our efforts in the NIST Language Recognition Evaluations (LRE) 2017 which resulted in systems providing very competitive and state-of-the-art performance. We provide both the descriptions and the analysis of the systems that we included in our submission. We explain our partitioning of the datasets that we were provided by NIST for training and development, and we follow by describing the features, DNN models and classifiers that were used to produce the final systems. After covering the architecture of our submission, we concentrate on post-evaluation analysis. We compare different DNN Bottle-Neck features, i-vector systems of different sizes and architectures, different classifiers and we present experimental results with data augmentation and with improved architecture of the system based on DNN embeddings. We present the performance of the systems in the Fixed condition (where participants are required to use only predefined data sets) and in addition to official NIST LRE17 evaluation set, we also provide results on our internal development set which can serve as a baseline for other researchers, since all training data are fixed and provided by NIST.

1. Introduction

It has been only two years since the previous NIST LRE 2015, where we still saw the dominance of systems based on i-vectors [1, 2, 3]. In 2015, there was a clear shift in the state of the art of Language identification (LID) front-ends. Instead of relying on standard acoustic features (such as MFCC) or outputs from various phoneme recognizers, LID field has adopted bottleneck features (BN) [4, 5], extracted from Deep Neural Networks (DNN) that are trained to classify phoneme states.

The variant of BN-DNN, which utilizes multilingual training [6] has brought additional improvements over monolingual DNNs on NIST LRE 2009 and 2015 benchmarks[7, 8], but we have not seen such improvements in NIST LRE2017 yet. We attribute this partly to the different data domains of NIST LRE2017, but we will need to conduct more research to draw some conclusions on using multilingual BN features in NIST LRE2017.

In our submission to NIST LRE2017, most of our systems are still based on i-vectors and BN features, but we also followed a general trend coming from speaker recognition [9, 10, 11], and we developed a subsystem based on DNN-embeddings. We offer an analysis with BN features, i-vector and embedding systems, but as the use of DNN embeddings is still relatively new in the field, we provide more detailed description, analysis and experimental results with our embedding system in [12, 13]. Following the research in speaker recognition, we explore and discuss the use of Non-linear transformation of i-vectors [14, 15] for language recognition and we also touch the topic of classifiers operating on top of i-vectors or embeddings by discussing two variants of generative Gaussian Linear Classifier. Finally we briefly describe our calibration and fusion which follows the standard recipe with a multi-class logistic regression.

2. Data

We have utilized all data supplied by NIST (training and development) for the primary (fixed) condition. All data were downsampled to 8kHz.

The annotated Switchboard and Fisher I and II databases were used to train the bottleneck NN. These databases were provided for all participants to allow the use of techniques requiring annotated speech corpora for system development.

The LRE17 training data were used to train the i-vector system (GMM-UBM, i-vector extractor), LRE17 development data were split into two parts. The first part was used added to the classifier and DNN-embedding training data and the second part was used as a held-out set, to monitor the performance during development, and for calibration and fusion during devel-

This work was partly supported by Czech Ministry of Interior project No. VI20152020025 DRAPAK, by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science - LQ1602, by Google Faculty Research Award and by Grant Agency of the Czech Republic project No. GJ17-23870Y. Mireia Diez was supported by European Union's Horizon 2020 Marie Sklodowska-Curie grant agreement No. 748097 and Johan Rohdin was supported by European Union's Horizon 2020 Marie Sklodowska-Curie / South Moravian Region grant agreement No. 665860.

10.21437/Odyssey.2018-7

opment. For the final calibration and fusion before submission, we have re-calibrated and fused all systems on the full LRE17 development set.

Our submission for the open data condition is based on the same datasets, with the addition of 17 languages from the IARPA BABEL data used to train the multilingual BN feature extractor. As the additional data did not provide improvement, and since we put a limited effort into experimenting for the open data condition, we will concentrate our analysis only on the fixed data condition. More details regarding the data and results with multilingual BN features can be found in [16] and [17].

2.1. Training and development data for fixed condition

We did not follow the data partitioning suggested by NIST, but we rather split the LRE17 training and development data to include the newly introduced datasets (VAST and MLS14) in the training stage of our classifiers.

As the training data provided by NIST contained many rather long segments, we have decided to prepare two versions of the training data. First, we used the LRE17 training data as is without any modification (we denote this version as ***full***), and second we cut all of the files containing more than 40s of speech into cuts ranging from 2.5 to 40s (we denote this version as ***cuts***). We did not include the original long segment in the resulting training list for the **cuts** set. Cutting the data into many smaller segments turned out to be important for the much smaller LRE15 training dataset (3042 segments, 248 hours of speech, 20 languages). The LRE17 training dataset (16205 segments, 837 hours of speech, 14 languages) is, however, considerably larger and we have not seen improvements from using such dataset for training our classifiers. In this work the ***full*** data was only used for training the non-linear transformation of i-vectors described in Sections 6.2 and 8.6.

The LRE17 development set, on the contrary, is much smaller (3660 segments, 28.6 hours of speech). Therefore, we split the segments that contain more than 40s of speech into multiple short cuts ranging from 2.5 to 40 seconds of speech. We also kept the original long segments that we cut, obtaining a total of 6090 segments. We decided to put $2/3$ of this data into the training sets (both **cuts** and **full**) and leave the remaining $1/3$ as a held out **test set**. We did not attempt to detect overlapping segments (nested cuts) between the two splits of LRE17 development data, but we ensured that the short cuts created by us do not overlap with their original long segments.

It is worth mentioning, that NIST LRE17 development segments contain nested cuts of the same segment (i.e. there are segments containing part of a particular longer segment). After the workshop, the labels allowing us to exactly identify such nesting were disclosed, but in this work we decided to keep our original split. We are aware that this could have caused some over-training, especially with system based on DNN embeddings.

2.2. Training data for bottleneck features

We used Fisher English database Part 1 and 2 for training. The final training data was composed of 1800 hours of clean Fisher augmented with another 3 copies of artificially corrupted Fisher data. We used the fant tool [18] to mix reverberated speech and reverberated noise with given SNR with original clean audio file.

We generated artificial room impulse responses (IR) using Room Impulse Response Generator tool from E. Habets [1].

IRs were generated for rooms where each dimension was limited to the range of 2–22 meters and other parameters in the tool were set randomly.

Noises were added at SNRs ranging from 0dB to 45dB and the noise samples of the following types were downloaded from the Freesound.org library:

- real fan stationary noises - fan, AC, hvac, street, ventilation - 115 samples from Freesound.org
- real background transient noises - dishes, motor, workshop, doors, city, keyboard, library, office. The character is mainly transient, with some minor portion of stationary noises - 60 samples from freesound.org
- babbling noises: each created by merging speech from 100 random speakers from Fisher database using speech activity detector - 25 samples
- artificially generated noises - various spectral modifications of white noise + 50 and 100 Hz hum - 7 samples

3. Voice Activity Detection

Our VAD 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.

The input features for the NN consist of 15 log-Mel filterbank outputs and 3 Kaldi-pitch features [19]. We apply per-speaker mean 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 (i.e. 16x18) are globally mean and variance normalized on the NN input.

The NN was trained on the Fisher English with labels provided from ASR alignment. 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.

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.

4. 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 2 fundamental frequency features based on [20] resulting in 26-dimensional feature vectors.

Mean subtraction is applied at the utterance level. Hamming window followed by DCT consisting of 0th to 5th base

[1] http://www.audiolabs-erlangen.de/content/05-fau/professor/00-habets/05-software/01-rir-generator/rir_generator.pdf

are applied on the time trajectory of each parameter resulting in $(24 + 2) \times 6 = 156$ 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 which has the same topology as the first stage, except that for the bottleneck layer we also evaluated a dimensionality of 30 in addition to 80. The 30 or 80 dimensional bottleneck outputs from the second NN (referred as SBN) are the final features.

We experiment with two types of SBN features:

1. **FSH-30** - trained on Fisher English corpus. First bottleneck is 80 dimensional and the second bottleneck is 30 dimensional. There are 3 hidden layers before BN layer in both NNs. The output layer has 9824 outputs (triphones). During training the BN layer is connected directly to the output layer.

2. **FSH-80** - trained on Fisher English corpus. First and second bottleneck layers are 80 dimensional. There are 2 hidden layers before BN layer in both NNs. The output layer has 9824 outputs (triphones). During training, there is one hidden layer between the bottleneck and output layer.

5. DNN embeddings

For one of our systems, we use an architecture based on DNN embeddings. It consists of a sequence summarizing DNN trained to learn a fixed-length utterance (or segment) level representation from the frame-by-frame input features. We used FSH-30 SBN features described in a previous section. The structure of the DNN was inspired by the embedding system presented in [10] for speaker verification, but instead of using the time delay feed-forward DNN, we use BLSTMs.

We can split the DNN into two parts separated by the pooling (summarizing) layer. The first part of the DNN, up to the pooling layer, works on a frame-by-frame basis, and consists of two BLSTM layers followed by a fully connected layer. These recurrent layers based on BLSTMs have proven their ability to deal with temporal information without stacking of input features (especially for very short segments in LID) [21] as they take into account the information learned from previous (and following) frames in the input sequence of features. After the BLSTM layers, we add a single fully connected layer. The size of this layer is set to 256 (for **small**) or 1500 (for **large** version of the DNN).

Subsequent pooling layer computes mean and standard deviation statistics over the framewise outputs of the previous layer, summarizing the information of a given input sequence. The output of the pooling layer is forwarded through two additional fully connected layers, whose output values will be later used as embedding representations of the input utterance. Sizes of these embedding layers are set to 512 and 300 (**small**), or 512 each (**large**).

Finally, the output is a 14-dimensional softmax layer that provides a vector of language posterior probabilities for each utterance. We used a sigmoid activation function for all hidden units. An example of this architecture is depicted in Figure 1.

5.1. Training

The DNN was trained by minimizing the categorical cross-entropy loss function via Adam optimizer. To reduce overfitting, dropout rate of 30% was used in all layers, including also dropout for gates on the recurrent BLSTM layers. The maximum number of iterations (epochs) for all experiments was set to 400, and the final model was selected according to the best accuracy on the validation set.

During training, the gradients are estimated over batches of 210 segments of 3 seconds duration. Each batch is created by selecting 70 audio files and then from each audio file, randomly selecting 3 segments. One epoch is completed when all files from the training list, which varies for the **small** and **large** architecture, have been seen by the network. The **small** DNN is trained on a balanced set derived from the **full** training dataset. This set is limited to 15 hours of clean speech per language. The **large** NN is trained on all data from our **full** training set which is further augmented by adding two noisy versions for each training segment.

The model selection was performed according to the validation accuracy on the held out **test set**, which we split into 3 second sequences, resulting in 23782 samples for validation (about 19 h of speech). After the training is finished, we extract both embeddings for each utterance, but now we do not constrain the length of the segment to 3 seconds, but forward the whole segment. Finally we concatenate both embeddings and obtain a single fixed-length vector per utterance which we subsequently model as i-vectors with the Gaussian Linear Classifier (GLC) described in section 6.1. We train the GLC on the **full** training dataset.

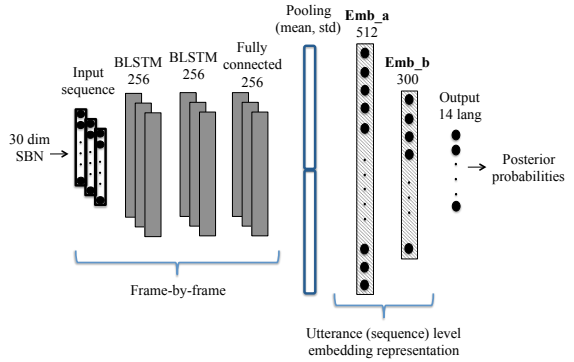

Figure 1: *Example of embedding DNN architecture.*

6. Classifiers

6.1. Gaussian linear classifier / Multi-Gaussian classifier

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 [2], 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:

$$\boldsymbol{\mu}_i \sim \mathcal{N}\left(\mathbf{m}_\ell, \boldsymbol{\Lambda}^{-1}\right) , \quad (1)$$

where $\mathbf{m}_\ell$ is a language–dependent mean vector and $\boldsymbol{\Lambda}^{-1}$ is a covariance matrix, shared among all language distributions.

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:

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

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

Since development and evaluation data comprise different, possibly mismatched, data sources, and we also observed a relevant mismatch between development data and previous LRE data, we also propose a modified Gaussian classifier, named Multi-Gaussian Classifier (MGC), able to better model these different sources. The MGC classifier assumes that i-vectors of each language-source combination are generated by a different Gaussian distribution according to

$$\boldsymbol{\mu}_i \sim \mathcal{N}\left(\mathbf{m}_{\ell,s}, \boldsymbol{\Lambda}^{-1}\right)\ , \tag{3}$$

where s denotes the source. For this evaluation, we considered three sources, namely, VAST, MSL14 and previous LRE data. At test time a language score is computed from a GMM whose components are the Gaussian distributions associated to the target language, assuming uniform weights over the data sources.

6.2. Non-Linear transformation

Following the success of Non–Linear PLDA (NL–PLDA) [14] for speaker verification, we propose to apply the technique for language recognition tasks. To this extent, we trained a NL–PLDA model using the formulations given in [14], assuming that the classes are languages rather than speakers. The model is given by

$$\begin{aligned}\boldsymbol{z} &= \boldsymbol{U}\boldsymbol{y} + \boldsymbol{x}\\ \boldsymbol{\mu}_{NL} &= f^{-1}(\boldsymbol{z})\ ,\end{aligned} \tag{4}$$

where $\boldsymbol{z}$ is the original i-vector, $\boldsymbol{y}$ represents a language factor sampled from $\boldsymbol{Y} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, $\boldsymbol{x}$ represents inter–session and residual noise sampled from $\boldsymbol{X} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}^{-1})$, and f is an invertible transformation. Following [14], function f was obtained as a composition of affine and non–linear elementwise sinh–arcsinh transformations [22, 23]. The model was trained using the EM procedure detailed in [14].

PLDA models (including NL–PLDA) can be directly used to compute language recognition scores. In [24] we have shown that the GLC backend provides a very good approximation of the PLDA scores, provided that the set of i–vectors for each language $\boldsymbol{M}_\ell$ is large. In particular, we have shown that, in this case, the posterior distributions of the language factors $\boldsymbol{y}|\boldsymbol{M}_\ell$ become sharp, and thus can be replaced by MAP point estimates $\hat{\boldsymbol{y}}_\ell = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{M}_\ell)$. Furthermore, the product $\boldsymbol{U}\hat{\boldsymbol{y}}_\ell$ converges to the class mean $\boldsymbol{m}_\ell$. For this reason, rather than directly using NL–PLDA for scoring, we chose to simply apply the estimated transformation on i–vectors, and to classify the transformed patterns using a GLC. This approach allowed us to also train the MGC backend on the transformed i–vectors. It is worth noting that, in contrast with our previously proposed i-vector Gaussianization approach [15], the NL–PLDA transformation is aware of language labels.

7. Calibration and fusion

After classification scores have been obtained from using the aforementioned techniques, we applied *pre-calibration* and *fusion* in score-space. Both are trained on the dev subset (NIST LRE17 dev). During development, we were calibrating and fusing on the $2/3$ of LRE17 dev that we also added into the training data. For the final submission, we calibrated and fused on the whole LRE17 dev including our short cuts. Both stages are implemented by multi-class logistic regression [25].

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 optimizations of prior-weighted multi-class cross-entropy. We used a *uniform prior* (flat) over all 14 languages for both pre-calibration and fusion.

7.1. Cluster-dependent system fusion

Some of our systems use cluster dependent subsystems [5] fused into one system by simple average of their scores, which simplifies the development and provides sufficient robustness. Such system is then denoted by **CD**.

Basically it is a simple fusion of 5 (as there are 5 language clusters) i-vector based systems, where the individual UBMs are trained only on data belonging to the particular cluster (Arabic, Chinese, English, Iberian, Slavic). The training data for T-matrix were however common for all individual cluster dependent subsystems.

8. Experiments and analysis

In this section, we will analyze and comment numerous experiments that we performed with our systems during LRE17 development and in the post-evaluation period. Primarily, we report results on the unmodified LRE17 evaluation dataset denoted as EVL. For the purpose of deeper analysis and as reference we also report results on the held out test set that we used for system development, denoted as DEV. All results are given as equalized actual cost, which is a primary evaluation metric for LRE17 [26].

8.1. Classifiers and general observations

Most of our systems are based on the 30 dimensional bottleneck features trained on Fisher English (FSH-30). We varied the topology and size of GMM and i-vector extractor, we added delta coefficients at the input, we experimented with post-processing of the i-vectors and we used different classifiers. If not stated otherwise, our classifiers (GLC, MGC) are trained on the **full** training data. During the development, we observed that unlike in LRE15 [8], including the short cuts in the training was not necessary as the amount of provided training data for LRE17 is sufficiently large.

The Table 1 summarizes the components of individual systems and points out the difference between submitted and post-evaluation architecture. We can immediately notice that the MGC classifier is most-likely overtrained since we always achieve better results with simple GLC. It is worth noting that the MGC classifier was designed specifically for the LRE17 when we expected two different data sources in the evaluation data. Its good performance on the development set compared to evaluation set suggest that we should experiment with redesigning our data-split – especially avoiding overlapping speech between training and test due to the nested segments.

We can also observe that we managed to improve our embedding system by using **large** DNN trained on more data together with augmentation. We are now actively working on

Table 1: *Results of the core systems and fusions. Comparison of GLC and MGC which reflect our submission and postevaluation results.*

Systems	Difference	Submission EVL (DEV)	Postevaluation EVL (DEV)
[1] SBN30D ClusterDep 4096/800	MGC → GLC	18.68 (14.57)	16.88 (17.94)
[2] SBN30D 4096/800 NLPLDA	MGC → GLC	19.80 (17.30)	19.30 (20.30)
[3] DNN embedings	More data + data augmentation, **large** DNN	24.07 (19.55)	19.61 (16.14)
[4] SBN80 4096/800 (iXtractor with FSH+SWB)	MGC → GLC	21.35 (18.05)	18.79 (19.41)
Primary Fusion $[1+2+3+4]$	-	16.60 (12.96)	16.78 (15.14)
Fusion of ivector systems only $[1+2+4]$	-	-	15.71(16.52)

improving this system and we have summarized our findings in [13].

Last two rows of Table 1 contain fusions. We can see that each of the improved post–evaluation systems achieves better performance on EVL than the corresponding system used for the original submission. On the contrary, the performance on the DEV set is, in general, worse. Surprisingly, the overall fusion is sligthly worse also on the EVL set. This is caused by a different behavior of the embedding system, which has improved in both datasets and received a large weight in the fusion compared to the best i-vector system. By removing the embedding system from the fusion we indeed obtain better results on the evaluation set. This behavior is most-likely caused by overtraining of our embedding system and reminds us that, compared to i-vectors, using these DNN-based architectures comes with a risk and requires more careful planning when designing the training lists.

8.2. Feature extraction analysis

The results in Table 2 show the performance of different bottleneck features. Stacked bottleneck features are composed of two NN in cascade. The first line belongs to the results obtained with features from the first stage of our topology (bottlenecks). The second line are the results with stacked bottlenecks, which provide approximately 3% absolute improvement. The third line shows the effect of augmenting the SBN features with their delta coefficients. The gain is again 3% absolute for 30 dimensional features, while 80 dimensional SBN suffered from a small degradation. The experiment was performed with FSH-30 and FSH-80 and i-vector system based on 2048 Gaussians, 600 dimensional i-vector, transformed by means of Within Class Covariance Normalization (WCCN) followed by length normalization, and GLC.

Table 2: *Analysis of bottleneck features, stack bottleneck features and effect of delta coefficients.*

System	BNF30 EVL (DEV)	BNF80 EVL (DEV)
BN	-	23.22 (23.94)
SBN	23.71 (26.51)	20.41 (20.84)
SBN + delta	19.78 (21.53)	20.82 (21.14)

8.3. Size of iVector system

The results in Table 3 show the analysis of the i-vector system size. We used GMM-UBMs with diagonal covariance components. There is a clear gain when using 4096 Gaussians in UBM and more than 400 dimensional ivectors. The experiments were performed with FSH-30D, WCCN+L2 and GLC.

Table 3: *Analysis of the i-vector system size.*

i-Vector dimensionality	2048 Gaussians EVL (DEV)	4096 Gaussians EVL (DEV)
400	20.26 (21.54)	19.12 (21.42)
600	19.78 (21.53)	18.68 (20.08)
800	19.57 (21.31)	18.77 (20.42)

8.4. Cluster dependent system

The results in Table 4 show approximately 2% absolute gain of a CD system over a single system. The disadvantage is that the system is 5 times slower.

Table 4: *Comparison of single system with Cluster dependent system. Systems are based on FSH-30 features.*

System	1 system EVL (DEV)	CD EVL (DEV)
2048/600	19.78 (21.53)	17.92 (19.47)
4096/800	18.77 (20.42)	16.88 (17.94)

8.5. Data augmentation

We also analyzed the effect of data augmentation for BNF extractor training. The results of the i-vector system (again based on FSH-30D, 2048 Gaussians, 600 dim. i-vectors, WCCN + L2 and GLC) in Table 6 show the performance of BNF trained with clean data only, and with clean and augmented data with 2 noisy copies, respectively. The results show a 4% absolute difference between these two systems.

A second set of experiments was performed with data augmentation for i-vector extractor and backend training. The results listed in Table 5 show little differences between the systems and we do not see clear benefits of adding augmented data to the i-vector extractor and GLC training.

8.6. Non-Linear PLDA

The final set of experiments compares the GLC classifier with a system based on NL–PLDA. NL–PLDA estimates both the PLDA parameters and a non–linear transformation of i–vectors,

Table 5: *Effect of data augmentation in iVector and GLC training.*

System	iXtractor EVL (DEV)	GLC EVL (DEV)	iXtractor +GLC EVL (DEV)
No augmentation	19.78 (21.53)	19.78 (21.53)	19.78 (21.53)
+ 2 x data augment. noise (0-8dB, 8-20dB)	19.69 (19.77)	20.01 (22.35)	19.69 (20.94)
+ 2 x data augment. tempo (0.9, 1.1)	19.73 (19.38)	20.06 (21.58)	19.90 (19.03)
+ 4 x data augment. noise + tempo	19.84 (20.28)	19.95(21.77)	19.84 (20.28)

Table 6: *Effect of data augmentation in BNF training.*

Data	EVL (DEV)
clean	23.72 (23.34)
clean + 2 copies (submission)	19.78 (21.53)

thus training the model requires a larger amount of data. Since the number of original segments was limited, it was not possible to reliably estimate NL–PLDA models only from the **full** dataset. The NL–PLDA model was therefore estimated from the **cuts** training set. Table 7 shows the results of applying the NL–transformation on i–vectors for GLC and MGC classifiers. The experiments were performed with FSH-30D, 4096 Gaussians and 800 dimensional i-vectors.

Comparing the results in the first two rows, we can observe that the NL–PLDA transformation allows improving the results on both the EVAL and DEV sets, especially when combined with MGC. Surprisingly, the transformed i-vectors do not incur in the same degradation as the original i-vectors when the GLC backend is replaced by the MGC classifier. On the other hand, GLC models trained with the **full** training set achieve slightly better results, probably due to a better fit between the distribution of the training and test i-vectors. Finally, the last row shows that the a simple average of the scores of non–linear and linear models allows further improvement of the actual cost on both datasets, with a negligible impact on testing time.

Table 7: *Results of system with Non-Linear PLDA transformation and GLC/MGC.*

Transformation	GLC EVL (DEV)	MGC EVL (DEV)
[1] Linear (cuts)	19.8 (21.3)	22.7 (19.8)
[2] Non linear (cuts)	19.3 (20.3)	19.8 (17.3)
[3] Linear (full)	18.7 (20.0)	20.7 (16.8)
[2] + [3] (Same weight)	18.2 (19.2)	19.4 (15.8)

9. Conclusion

In this paper, we have summarized and documented part of our efforts for the NIST LRE 2017. We have, in detail, analyzed the architecture of our baseline i-vector systems which still formed the basis of our submission. Our analysis with DNN models (BNF and DNN embeddings) and data augmentation has confirmed the current trend of augmenting still small training sets to obtain better performance.

Compared to few years back, having a larger amount of training data has perhaps caused the improvement we saw with the i-vector system, where we doubled its usual size to 4096 Gaussians in UBM and 800 dimensional i-vectors.

Participation in NIST LREs is always a great research opportunity when teams of researchers work on a common goal, try new techniques and improve the state-of-the-art. This time, we have followed the trend from speaker recognition and developed a DNN-embedding system that is showing promising results and is currently being actively developed. Another interesting technique that comes from speaker recognition and is presented here is the non-linear transformation of i-vectors that has shown robust results for LID and can be perhaps applied also to DNN embeddings.

During our work on LRE17, we have also realized that our rather complex and well-tuned bottleneck features can be very useful for the research community and we have decided to describe them in more detail and make the trained models and extraction scripts available [17].

10. References

[1] 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.

[2] 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.

[3] N. Dehak, P. A. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language Recognition via Ivectors and Dimensionality Reduction," in *Proc. of Interspeech 2011*. International Speech Communication Association.

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

[5] 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.

[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] Radek Fer, Pavel Matejka, Frantisek Grezl, Oldrich Plchot, Karel Vesely, and Jan Honza Cernocky, "Multilingually trained bottleneck features in spoken language recognition," *Computer Speech & Language*, vol. 46, no. Supplement C, pp. 252 – 267, 2017.

[8] Oldřich Plchot, Pavel Matějka, Radek Fér, Ondřej Glembek, Ondřej Novotný, Jan Pešán, Karel Veselý, Lucas Ondel, Martin Karafiát, František Grézl, Santosh Kesiraju, Lukáš Burget, Niko Brummer, Preez du Albert Swart,

Sandro Cumani, Harish Sri Mallidi, and Ruizhi Li, "Bat system description for nist lre 2015," in *Proceedings of Odyssey 2016*. 2016, International Speech Communication Association.

[9] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 4052–4056.

[10] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Proceedings of Interspeech*, 2017.

[11] G. Bhattacharya, J. Alam, and P. Kenny, "Deep Speaker Embeddings for Short-Duration Speaker Verification," in *Interspeech 2017*, 08 2017, pp. 1517–1521.

[12] Alicia Lozano-Diez, Oldřich Plchot, Pavel Matějka, and Joaquin Gonzalez-Rodriguez, "DNN based embeddings for language recognition," in *Proceedings of ICASSP*, April 2018.

[13] Alicia Lozano-Diez, Oldřich Plchot, Pavel Matějka, Ondřej Novotný, and Joaquin Gonzalez-Rodriguez, "Analysis of DNN-based Embeddings for Language Recognition on the NIST LRE 2017," in *Submitted to Odyssey 2018*, June 2018.

[14] S. Cumani and P. Laface, "Joint estimation of plda and nonlinear transformations of speaker vectors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 10, pp. 1890–1900, Oct 2017.

[15] S. Cumani and P. Laface, "Nonlinear i-vector transformations for plda-based speaker recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 4, pp. 908–919, April 2017.

[16] Pavel Matějka, Oldřich Plchot, Ondřej Novotný, Sandro Cumani, Alicia Lozano-Diez, Josef Slavíček, Mireia Sánchez Diez, František Grézl, Ondřej Glembek, Mounika Veera Kamsali, Anna Silnova, Lukáš Burget, Lucas Ondel, Santosh Kesiraju, and A. Johan Rohdin, "BUT- PT System Description for NIST LRE 2017," http://www.fit.vutbr.cz/research/view_pub.php?id=11655.

[17] Anna Silnova, Pavel Matějka, František Grézl, Oldřich Plchot, Ondřej Novotný, Petr Schwarz, Lukáš Burget, Ondřej Glembek, and Jan "Honza" Černocký, "BUT/Phonexia Bottleneck Feature Extractor," in *Submitted to Odyssey 2018*, June 2018.

[18] H. Gnter Hirsch and Harald Finster, "The simulation of realistic acoustic input scenarios for speech recognition systems," in *Proceedings of Interspeech 2005*, 2005.

[19] 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.

[20] 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.

[21] Javier Gonzalez-Dominguez, Ignacio Lopez-Moreno, Hasim Sak, Joaquin Gonzalez-Rodriguez, and Pedro J. Moreno, "Automatic language identification using long short-term memory recurrent neural networks," in *Proceedings of Interspeech*, September 2014.

[22] M. C. Jones and A. Pewsey, "Sinh–arcsinh distributions," *Biometrika*, vol. 96, no. 4, pp. 761–780, 2009.

[23] J. F. Rosco, M. C. Jones, and A. Pewsey, "Skew t distributions via the sinh-arcsinh transformation," *TEST*, vol. 20, no. 3, pp. 630–652, 2011.

[24] Sandro Cumani, Oldřich Plchot, and Radek Fér, "Exploiting i-vector posterior covariances for short-duration language recognition," in *Proceedings of Interspeech 2015*. 2015, vol. 2015, pp. 1002–1006, International Speech Communication Association.

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

[26] "NIST 2017 Language Recognition Evaluation Plan," https://www.nist.gov/sites/default/files/documents/2017/06/01/lre17_eval_plan-2017-05-31_v2.pdf.