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

Language Recognition for Telephone and Video Speech: The JHU HLTCOE Submission for NIST LRE17

Alan McCree, David Snyder, Gregory Sell, and Daniel Garcia-Romero

Human Language Technology Center of Excellence
Johns Hopkins University, Baltimore, MD, USA

alan.mccree@jhu.edu, david.ryan.snyder@gmail.com, gsell@jhu.edu, dgromero@jhu.edu

Abstract

This paper presents our newest language recognition systems developed for NIST LRE17. For this challenging limited data multidomain task, we were able to get very good performance with our state-of-the-art DNN senone and bottleneck joint i-vector systems by effective utilization of all of the available training and development data. Data augmentation techniques were very valuable for this task, and our discriminative Gaussian classifier combined with naive fusion used all of the development data for system design rather than holding some out for separate back-end training. Finally, our newest research with discriminatively-trained DNN embeddings allowed us to replace i-vectors with more powerful x-vectors to further improve language recognition accuracy, resulting in very good LRE17 performance for this single system, our JHU HLTCOE site fusion primary submission, and the JHU MIT team submission.

1. Introduction

The 2017 Language Recognition Evaluation (LRE17) continues a long tradition of NIST conducting formal evaluations of language recognition technology to foster research progress in the field. Like the previous evaluation in 2015, this one focused on differentiating closely-related languages and featured a core requirement of a *fixed* training condition, where only specified speech material could be used to train systems, while an alternative *open* condition allowed participants to explore the benefits of more extensive training sets [1]. Unlike previous LREs, however, LRE17 introduced the evaluation of performance for speech extracted from videos in addition to the more traditional conversational telephone speech material. Test data from telephone speech was selected from 3, 10, and 30 second durations, while video speech used entire recordings.

LRE17 used fourteen languages from five clusters: Arabic (Egyptian, Iraqi, Levantine, Maghrebi), Chinese (Mandarin, Min Nan), English (British, General American), Slavic (Polish, Russian), and Iberian (Caribbean Spanish, European Spanish, Latin American Continental Spanish, Brazilian Portuguese). The official performance metric $C_{primary}$ was based on average Bayes detection cost as in previous LREs, but this time averaged over two target priors (0.1 and 0.5) and data sources (telephone and video).

Most current research in language recognition revolves around i-vectors [2], which have provided the best performance in recent NIST evaluations of both speaker and language recognition. In this approach, built on techniques originally developed for subspace modeling of Gaussian Mixture Models (GMMs) using Joint Factor Analysis, the GMM for each speech cut is assumed to differ from a Universal Background Model (UBM) by only a low-dimensional offset of the mean supervector. The Maximum a Posteriori (MAP) estimate of this offset, called an *i-vector*, is generated for each cut and treated as an input feature for classification.

Recent research progress has used Deep Neural Networks (DNNs) trained as acoustic models for speech recognition to improve modeling power in one of two ways: using bottleneck (BN) features from DNNs as inputs to a GMM/UBM [3, 4] or replacing the GMM/UBM frame alignment process with a supervised frame alignment into clustered phone states (senones) [5, 6, 7]. For these DNN approaches, language recognition systems also use counts of phone state posteriors to capture phonetic content in a *phonotactic* method [8]. For this LRE, we found good performance with our approach of combining both acoustic and phonotactic information with a single *joint* i-vector extraction [9].

Alternatively, DNNs can be directly optimized to discriminate between languages or to produce embeddings from bottleneck layers [10]. In this paper we describe a high-performing x-vector DNN language embedding system using the temporal pooling idea successfully developed for speaker recognition in [11].

For LRE17, the research challenge was to achieve high performance using DNNs but under the fixed training condition constraints of limited telephone data in the languages and channels of interest. This paper describes our submission to LRE17, including system design, DNN i-vector and x-vector embedding systems, use of limited and augmented training data, and development and evaluation set performance. The layout of this paper is as follows. In Section 2, we discuss the algorithms and system designs used in the JHU HLTCOE submissions; then, in Section 3, we detail the usage of training data for these systems. Finally, Section 4, discusses experimental results on the LRE17 development and evaluation sets, as well as providing official NIST submission scores.

2. System Description

All systems in the JHU HLTCOE submissions used embeddings based on DNNs. Classification was done with discriminatively-trained Gaussian classifiers [12]. In more detail, each system followed the same recipe:

- Speech activity detection
- Extraction of embedding
- Whitening and length normalization [13]
- Estimation of class means and shared within-class and across-class covariances

10.21437/Odyssey.2018-10

- Dimension reduction by Linear Discriminant Analysis (LDA)
- Refinement of Gaussian parameters using discriminative training (MMI).

Individual systems differed in their use of DNNs and types of embeddings. Two types of acoustic DNNs were trained: one generated bottleneck features for subsequent GMM modeling, and the other estimated clustered phone state (senone) posteriors for supervised estimation of sufficient statistics [5, 6, 7, 14]. We experimented with three types of embeddings: acoustic i-vectors [2], joint i-vectors [15, 9], and DNN x-vectors [11]. The remainder of this section presents more details of these algorithms and system designs.

2.1. Input Bandwidth

All systems used narrowband input features throughout. We did some preliminary experiments with wider bandwidths for microphone data but found we would need more wideband data to build an effective system.

2.2. Speech Activity Detection

We trained a bi-directional long short-term memory (BLSTM) for speech activity detection (SAD) using the CURRENNT toolkit[1]. Input features for this SAD system were 13 mel-frequency cepstral coefficients (MFCCs) appended with deltas and double-deltas. The MFCCs, which were computed every 10 ms for 25 ms Hamming windowed segments, were all centered and normalized with statistics learned from the training corpus. The simple architecture connects a 39-dimensional input layer to a single hidden layer of 32 bi-directional memory cells. The outputs of the hidden layer are then mapped to a single logistic output node that estimates probability of speech. The BLSTM outputs were further smoothed with a 500 ms median filter prior to converting to speech marks.

The x-vector systems use a simple energy-based SAD which is currently the default SAD for speaker recognition recipes in the Kaldi speech recognition toolkit. Decisions are made at the frame-level, classifying a frame as speech or non-speech based on the average log-energy in a given window centered around the current frame. No training data is used.

2.3. I-vector Systems

We developed both acoustic and joint i-vector systems using both DNN bottleneck features and DNN senone statistics with shifted delta cepstra input features.

2.3.1. Acoustic DNNs

For the i-vector systems, we trained two acoustic DNNs using the Kaldi speech recognition toolkit. One DNN is trained to compute senone posteriors and the other to extract bottleneck features. The labels (i.e. frame alignments to senones) for the DNNs are obtained from a standard tied-state triphone GMM-HMM system trained with maximum likelihood on the Fisher English material provided by NIST. The senone set is obtained by clustering the states using a decision tree and the number of total Gaussians was set to 200K. The number of senones after the clustering was 7591. Input to the senone DNN is MFCCs, and it uses p-norm nonlinearities. The BN system used PLP+pitch as input features and sigmoid nonlinearities to produce an 80-dimensional feature output. No fMLLR or i-vectors are used for speaker adaptation.

2.3.2. Acoustic i-vectors

In an acoustic i-vector system [2], each audio cut is assumed to have been generated by a GMM which only differs from the UBM by a linear subspace offset:

$$\mathbf{m}_i = \mathbf{m}_0 + \mathbf{T}_m \mathbf{z}_i \qquad (1)$$

where $\mathbf{m}_i$ is a Gaussian supervector representing the stacked GMM means for cut i, the matrix $\mathbf{T}_m$ represents the total variability subspace over all such GMM models, and the subspace offset for cut i is $\mathbf{z}_i$. If the GMM alignments are given by the UBM and the subspace vector $\mathbf{z}_i$ has a standard normal prior, then the MAP estimate of the subspace offset is referred to as an i-vector. The matrix $\mathbf{T}_m$ is estimated using maximum likelihood (ML) over a large set of speech audio cuts with an EM algorithm.

2.3.3. Joint i-vectors

For both senone and BN DNNs, we extracted joint i-vectors to model both acoustic and phonotactic variability [9]. This joint i-vector allows the GMM for cut i to differ from the UBM in both means and weights:

$$\begin{aligned} \mathbf{m}_i &= \mathbf{m}_0 + \mathbf{T}_m \mathbf{z}_i & (2) \\ \mathbf{w}_i &= \text{softmax}\left(\log(\mathbf{w}_0) + \mathbf{T}_w \mathbf{z}_i\right) & (3) \end{aligned}$$

This can be viewed as a modified form of the subspace GMM, and this i-vector can be extracted using Newton's method with updates similar to [15]. To reduce complexity and improve convergence, we use the following extensions to the joint i-vector extraction algorithm. First, we use MAP estimation, rather than ML, for more robust estimation with short durations and improved numerical stability. Second, we initialize the joint i-vector with the acoustic closed-form solution instead of a zero vector. Finally, we use a diagonal approximation to the Hessian matrix to greatly reduce the computation per iteration, utilizing an adaptive stepsize logic to ensure convergence, as in [16]. Without these modifications, 10 iterations of Newton's method require an order of magnitude increase in complexity of the i-vector extraction as compared to an acoustic i-vector; with them, the joint i-vector presents only a minor computation increase.

Joint estimation of the two matrices $\mathbf{T}_m$ and $\mathbf{T}_w$ is done with an EM-like algorithm. To improve convergence, we initialize $\mathbf{T}_m$ with PCA over a subset of the training data and start with $\mathbf{T}_w$ set to zero. We find similar convergence of this algorithm to the traditional acoustic case, and the additional computation to update $\mathbf{T}_w$ is small since this count matrix is much smaller than $\mathbf{T}_m$.

2.4. X-vectors

Our x-vector system is an extension of the work we have been doing for speaker recognition [11]. Two key differences here are that the x-vector DNN was trained with a language recognition objective function and the input is acoustic DNN bottleneck features.

2.4.1. DNN Bottleneck Features

The x-vector acoustic input was 60-dimensional linear bottleneck features extracted from a speech recognition DNN built

[1] https://sourceforge.net/projects/currennt

Layer	Layer context	Tot. context	In x out
frame1	$[t-2, t+2]$	5	$5F$x512
frame2	$\{t-2, t, t+2\}$	9	1536x512
frame3	$\{t-3, t, t+3\}$	15	1536x512
frame4	$\{t\}$	15	512x512
frame5	$\{t\}$	15	512x1500
stats pooling	$[0, T)$	T	1500Tx3000
segment6	$\{0\}$	T	3000x512
segment7	$\{0\}$	T	512x512
softmax	$\{0\}$	T	512xL

Table 1: The standard x-vector DNN architecture. X-vectors are extracted at layer *segment6*, before the nonlinearity. The input layer accepts F-dimensional features. The L in the softmax layer corresponds to the number of training languages.

using the nnet2 library in Kaldi. Besides the bottleneck layer, it uses the same architecture and training recipe as the system described in Section 2.2 of [17]. Its input features are 40 MFCCs with a frame-length of 25 ms. Cepstral mean subtraction is performed over a sliding window of 6 seconds. The DNN has six layers, and a total left-context of 13 and a right-context of 9. The hidden layers use the p-norm (where $p = 2$) nonlinearity and have an input dimension of 3500 and an output dimension 350. The penultimate layer is a 60 dimensional linear bottleneck layer. No fMLLR or i-vectors are used for speaker adaptation. For the fixed condition, this bottleneck DNN was trained using Fisher English with 5297 triphone states, while the open condition used a multilingual version trained with IARPA Babel data with 4300 to 4900 triphone states depending on the language.

2.4.2. X-vector DNN architecture

The x-vector DNN configuration is outlined in Table 1. The input to the DNN is a sequence of T speech frames. The first 5 layers process the input at the frame-level, with a small temporal context centered at the current frame t. For example, the input layer, *frame1*, splices together the F-dimensional features at frames $t-2$, $t-1$, t, $t+1$ and $t+2$, which gives it a total temporal context of 5 frames. The input to the next layer, *frame2*, is the spliced output of *frame1* at $t-2$, t and $t+2$. This builds on the temporal context established by the previous layer, so that *frame2* sees a total context of 9 frames. This process is continued in the following layers, and results in *frame5* seeing a total context of 15 frames.

The statistics pooling layer aggregates information across the time dimension so that subsequent layers operate on the entire segment. The input to the pooling layer is a sequence of T 1500-dimensional vectors from the previous layer, *frame5*. The output is the mean and standard deviation of the input (each 1500-dimensional vectors). These statistics are concatenated together (to produce a 3000-dimensional vector) and passed through the segment-level layers and finally the softmax output layer. The nonlinearities are rectified linear units (ReLUs). Excluding the softmax output layer and *segment7* (because they are not needed after training) there are a total of 4.2 million parameters.

2.5. MMI Gaussian Classifier

Most state-of-the-art systems for i-vector language recognition use classifiers such as Gaussian models, logistic regression, or cosine scoring, followed by a multiclass back-end which provides significant performance improvement as well as producing calibrated probability outputs [18]. We prefer to use only a single step: a Gaussian classifier discriminatively trained using Maximum Mutual Information (MMI) [12]. This two-covariance model requires three parameters: the class means, a shared within-class covariance, and an across-class covariance. Initial sample covariance estimates of these parameters are used for dimension reduction using a diagonalizing form of LDA. The parameters are then refined with two-stage MMI training of first a scale factor of the within-class covariance and then the class means.

2.6. Fusion and Scoring

Since individual systems are already calibrated by discriminative training, score fusion was implemented using a simple linear fusion. A scale factor was learned for each individual system, the scaled scores were averaged, and an overall scale factor was learned for the fused system. Learning used multiclass cross-entropy.

3. Training Data

Systems for the fixed training condition were trained using the NIST-provided data: Fisher, Switchboard, LRE17 training (a subset of previously-released LRE telephony data in the 14 languages of interest), and LRE17 development (a fairly small amount of new material in these languages from both telephone and video audio).

A focus of our effort was to use as much of this limited development set for system training as possible while also using it for development decisions. A key advantage of the MMI Gaussian classifier is that it can be trained on a single training set, rather than using a separate training set for the classifier and development set for the multiclass back-end. With this in mind, we chose to combine the NIST LRE17 training and development sets together to allow the classifier to train with as much in-domain data as possible. This section describes the training data used for our systems.

For development, we split the provided development set in two halves and allocated the first set (dev1) to train while using the second (dev2) for internal testing and to train the fusion. The split of the development data into two sets was balanced, equally splitting the segments in each language/condition pair. Prior to splitting, a simple i-vector clustering was performed with the goal of grouping together overlapping segments or possibly segments from the same speaker. The clustering was only lightly evaluated by human confirmation, but the process appeared to be effective for identifying segments drawn from the same audio.

Final submissions use all of train+dev for system training while keeping the fusion weights learned from the partitioned data. Specific usage of training data is described below.

3.1. SAD

The BLSTM SAD was trained using only Switchboard data, a subset of which was also augmented using one randomly selected method from a variety of strategies:

- artificial reverberation with an impulse response selected from the Aachen Impulse Response (AIR) database[2]

[2] http://www.ind.rwth-aachen.de/AIR

- added non-vocal music or noise from the MUSAN corpus[3]
- simulated GSM-AMR phone encoding[4]
- multi-band dynamic range compression

The energy-based SAD does not require training data.

3.2. Acoustic DNNs

All fixed condition bottleneck and senone posterior DNNs were trained on Fisher English. For the open training condition x-vector system, we use multilingual BNs trained on 23 languages from the IARPA Babel dataset (Amharic, Cebuano, Guarani, Javanese, Lao, Tagalog, Tokpisin, Zulu, Assamese, Dholuo, Haitian, Kazakh, Lithuanian, Pashto, Tamil, Turkish, Cantonese, Igbo, Kurmanji, Mongolian, Swahili, Telugu, and Vietnamese).

3.3. I-vector System

For each of the i-vector systems, the GMM UBM and T matrix were trained using the NIST LRE17 training list.

3.4. X-vector System

The x-vector extractor was trained using the NIST LRE17 training set and dev1, with segmentation and augmentation. Segmentation is done by picking speech segments from 2–4 seconds long. We use augmentation to increase the amount and diversity of the x-vector DNN training data. We use a 6-fold augmentation strategy that combines the original "clean" training list with 5 augmented copies. To augment a recording, we randomly choose between one of the following:

- speed perturbation: Apply a speed factor of 0.9 or 1.1 to slow down or speed up the original recording.
- music: A single music file (without vocals) is randomly selected from MUSAN, trimmed or repeated as necessary to match duration, and added to the original signal (5-15 dB SNR).
- noise: MUSAN noises are added at one second intervals throughout the recording (0-15 dB SNR).
- reverb: The training recording is artificially reverberated via convolution with simulated RIRs[5].

To comply with the requirements of the fixed training condition of the evaluation, we only used the noises and music without vocals.

3.5. MMI Gaussian Classifier

During our development phase, the MMI Gaussian classifiers were trained on the training set plus dev1 with segmentation and augmentation in order to increase the size and diversity of the data.

Telephony files were segmented uniformly to durations of 3-30 seconds of speech (according to estimates from the SAD system described above), while video files were segmented to durations of 6-60 seconds of speech. Files with less measured speech than the maximum allowed segment (30 seconds for telephony, 60 seconds for video) were not segmented at all. As a result, the development telephony was largely left alone in this stage, as the trials were already trimmed to that length.

A subset of the resulting segments were then passed through one of a number of possible augmentations:

- artificial reverberation with an impulse response selected from AIR
- added non-vocal music or noise from the MUSAN corpus
- simulated GSM-AMR phone encoding
- added babble noise using summed files from the Fisher corpus
- added synthetic, low-frequency modulated noise
- speed perturbation with resampling

Final training lists were then created by setting a maximum number of desired trials per language per condition (telephony vs video). Clean development data was first added, and then augmented/segmented data was appended to the lists until the desired number of trials was reached for each language and condition.

The final evaluation submissions used the training set combined with the entire development set for classifier design to maximize expected performance.

Note that the segmentation and augmentations used in classifier design were different than those used for x-vector training, so that the classifier had some opportunity to learn how the x-vector embeddings reacted to new data. This effect was even more pronounced in the submission x-vector system, since the classifiers were trained using the held out portion of the development set (dev2) which were not used for x-vector training.

3.6. Fusion

The fusion coefficients were trained on the held out portion of the development set (dev2) without segmentation or augmentation. These same fusion coefficients were used in the final submissions as there was no new data to train them but they were not expected to change.

3.7. Condition-dependent Training

We found improved performance with condition-dependent training data as follows. All embeddings were condition-independent. For telephony inputs, we used a classifier training list with segmentation but not augmentation which had a higher proportion of telephone training data. We then learned fusion coefficients with the telephony subset of the dev2 set. For video inputs, the classifier training data included both segmentation and augmentation, with about a 50/50 mix of telephony and video data, and fusion coefficients were learned from the video subset of dev2.

4. NIST LRE17 Performance

In this section, we present results of these systems for the LRE17 task. The development set results are for our initial systems trained with NIST LRE17 training plus our first partition of the NIST LRE17 development set (dev1), and evaluated on the unaugmented second partition (dev2). Evaluation results are for the final systems trained with all training and development data. Our development metrics are the NIST detection cost C_{avg} for the two target priors of 0.5 and 0.1 as well as normalized cross entropy (H_{mce}/H_{max}) for both the Telephone

[3] http://www.openslr.org/17

[4] http://www.3gpp.org/ftp/Specs/archive/26_series/26.073/26073-800.zip

[5] http://www.openslr.org/resources/28

Table 2: Development Set Results: C_{avg} at P_t of 0.5 and 0.1 and normalized cross-entropy for dev2 subsets.

System	Telephone	Video
[1] Senone joint	0.11/0.33/0.22	0.12/0.33/0.23
[2] BN joint	0.10/0.27/0.19	0.13/0.34/0.24
[3] BN x-vector	0.09/0.25/0.17	0.11/0.34/0.21
[1,2] Fusion	0.09/0.26/0.14	0.10/0.29/0.20
[1,2,3] Fusion	0.08/0.20/0.14	0.09/0.28/0.18
[4] ML BN x-vector	0.08/0.21/0.14	0.10/0.27/0.18
[1,2,4] Fusion	0.07/0.19/0.13	0.09/0.24/0.17

and Video subsets. For the evaluation set, we also report the official NIST primary metric $C_{primary}$ which is the average of C_{avg} over the four conditions [1].

4.1. Development Set Performance

Our initial development experiments for LRE17 helped us define our data usage and most promising systems. We used a straightforward acoustic i-vector system for early experiments, but quickly moved to the DNN systems reported here as they were much better in all conditions. We also found that joint i-vectors provided a consistent small improvement over acoustic versions for both senone and bottleneck DNNs. Results for our best systems on our development set are shown in Table 2. For the fixed training condition (first five rows), our key conclusions from these results are:

- The DNN senone joint i-vector system (Senone joint) provides competitive performance with DNN bottleneck i-vector system, particularly for video speech inputs.
- The bottleneck x-vector (BN x-vector) system performs best in most conditions, particularly for telephone.
- Fusion of the two joint i-vector systems performs about as well as the x-vector system, but combining them all provides a clear and consistent improvement.

Our only open condition system, an x-vector system with multilingual bottleneck inputs (ML BN x-vector), provides significant improvement for the open training condition. Even with this system, fusion with the monolingual i-vector systems is still helpful. Based on these results as well as other experiments, we chose to submit the following four systems to the NIST LRE17 evaluation for the JHU HLTCOE site:

- **Primary fixed submission:** fusion of senone joint i-vector, bottleneck joint i-vector, and bottleneck x-vector systems.
- **Alternate fixed submission:** bottleneck x-vector system (best single system)
- **Primary open submission:** fusion of senone joint i-vector, bottleneck joint i-vector, and multilingual bottleneck x-vector systems.
- **Alternate open submission:** multilingual bottleneck x-vector system (best single system)

4.2. Evaluation Set Performance

Results for the evaluation set are shown in Table 3. Note primarily that all of our systems performed better in the evaluation than on our development set; this was expected since they were exposed to all of development set as training data instead of just a single partition. This was a key benefit of our discriminative classifier approach. Otherwise, the relative performance of these systems on the evaluation set was generally similar to the the development set.

Table 3: Evaluation Set Results: C_{avg} at P_t of 0.5 and 0.1 and normalized cross-entropy for LRE17 evaluation subsets.

System	Telephone	Video
[1] Senone joint	0.10/0.28/0.19	0.13/0.31/0.23
[2] BN joint	0.09/0.25/0.17	0.12/0.30/0.21
[3] BN x-vector	0.08/0.22/0.15	0.09/0.26/0.18
[1,2] Fusion	0.08/0.22/0.15	0.10/0.27/0.19
[1,2,3] Fusion	0.06/0.18/0.12	0.09/0.24/0.17
[4] ML BN x-vector	0.07/0.19/0.13	0.08/0.22/0.16
[1,2,4] Fusion	0.06/0.17/0.11	0.08/0.22/0.17

Table 4: Official LRE17 NIST results $C_{primary}$ for four JHU HLTCOE site submissions and JHU-MIT team submission.

System	$C_{primary}$
Fixed Primary: [1,2,3] Fusion	0.14
Fixed Alternate: BN x-vector	0.16
Open Primary: [1,2,4] Fusion	0.13
Open Alternate: ML BN x-vector	0.14
Fixed Primary: Team JHU-MIT Fusion	0.13

For the final primary fixed submission, the JHU HLTCOE site also combined with the MIT LL/JHU CLSP site to form the JHU MIT team. Since each site was separately calibrated, combination was done with a blind averaging of log likelihood scores. In Table 4 we report the official NIST metric $C_{primary}$ for our final site and team submissions. Note that the team submission provided an additional gain from site fusion. In addition, not only was our team a top-performing team, but also both our HLTCOE site and our best x-vector system were among the top performers overall.

5. Conclusion

This paper has presented the JHU HLTCOE LRE17 language recognition system. For this challenging fixed data condition, we were able to get very good performance with our state-of-the-art DNN senone and bottleneck joint i-vector systems by effective utilization of all of the available training and development data. Data augmentation techniques were very valuable for this task, and our discriminative Gaussian classifier combined with naive fusion used all of the development data for system design rather than holding some out for separate back-end training. Finally, our newest research with discriminatively-trained DNN embeddings allowed us to replace i-vectors with more powerful x-vectors to further improve language recognition accuracy, resulting in excellent LRE17 performance for this single system, our JHU HLTCOE site fusion primary submission, and the JHU MIT team submission.

6. References

[1] "The NIST year 2017 language recognition evaluation plan," https://www.nist.gov/file/388781, 2017.

[2] N. Dehak, P. Kenny, R. Dehak, P. Ouellet, and P. Dumouchel, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, pp. 788–798, May 2011.

[3] Pavel Matejka, Le Zhang, Tim Ng, HS Mallidi, Ondrej Glembek, Jeff Ma, and Bing Zhang, "Neural network bottleneck features for language identification," *Proc. of IEEE Odyssey*, pp. 299–304, 2014.

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

[5] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[6] P. Kenny, V. Gupta, T. Stafylakis, P. Ouellet, and J. Alam, "Deep neural networks for extracting Baum-Welch statistics for speaker recognition," in *Odyssey: The Speaker and Language Recognition Workshop*, 2014.

[7] D. Garcia-Romero and A. McCree, "Insights into deep neural networks for speaker recognition," in *Interspeech*, 2015.

[8] Y. Lei, L. Ferrer, A. Lawson, M. McLaren, and N. Scheffer, "Application of convolutional neural networks to language identification in noisy conditions," in *Proc. Odyssey*, 2014, pp. 287–292.

[9] A. McCree, G. Sell, and D. Garcia-Romero, "Augmented Data Training of Joint Acoustic/Phonotactic DNN i-vectors for NIST LRE15," in *Proc. of IEEE Odyssey*, 2016.

[10] Ignacio Lopez-Moreno, Javier Gonzalez-Dominguez, David Martinez, Oldřich Plchot, Joaquin Gonzalez-Rodriguez, and Pedro J Moreno, "On the use of deep feedforward neural networks for automatic language identification," *Computer Speech and Language*, vol. 40, no. C, pp. 46–59, 2016.

[11] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[12] A. McCree, "Multiclass discriminative training of i-vector language recognition," in *Proc. Odyssey*, 2014, pp. 166–172.

[13] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. Interspeech*, 2011, pp. 249–252.

[14] D. Garcia-Romero, X. Zhang, A. McCree, and D. Povey, "Improving speaker recognition performance in the domain adaptation challenge using deep neural networks," in *IEEE Spoken Language Technology Workshop (SLT)*, 2014.

[15] D. Povey et al., "The subspace Gaussian mixture model–A structured model for speech recognition," *Computer Speech & Language*, vol. 25, no. 2, pp. 404–439, 2011.

[16] A. McCree and D. Garcia-Romero, "DNN Senone MAP Multinomial i-vectors for Phonotactic Language Recognition," in *Proc. Interspeech*, 2015, pp. 394–397.

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

[18] D. Martinez, O. Plchot, L. Burget, O. Glembek, and P. Matejka, "Language recognition in ivectors space," in *Proc. Interspeech*, 2011, pp. 861–864.