I-VECTOR-BASED SPEAKER ADAPTATION OF DEEP NEURAL NETWORKS FOR FRENCH BROADCAST AUDIO TRANSCRIPTION

Vishwa Gupta, Patrick Kenny, Pierre Ouellet, Themos Stafylakis

Centre de recherche informatique de Montréal (CRIM)

{Vishwa.Gupta, Patrick.Kenny, Pierre.ouellet, Themos.Stafylakis}@crim.ca

ABSTRACT

State of the art speaker recognition systems are based on the i-vector representation of speech segments. In this paper we show how this representation can be used to perform blind speaker adaptation of hybrid DNN-HMM speech recognition system and we report excellent results on a French language audio transcription task. The implemenation is very simple. An audio file is first diarized and each speaker cluster is represented by an i-vector. Acoustic feature vectors are augmented by the corresponding i-vectors before being presented to the DNN. (The same i-vector is used for all acoustic feature vectors aligned with a given speaker.) This supplementary information improves the DNN's ability to discriminate between phonetic events in a speaker independent way without having to make any modification to the DNN training algorithms. We report results on the ETAPE 2011 transcription task, and show that i-vector based speaker adaptation is effective irrespective of whether cross-entropy or sequence training is used. For cross-entropy training, we obtained a word error rate (WER) reduction from 22.16% to 20.67% whereas for sequence training the WER reduces from 19.93% to 18.40%.

Index Terms— Deep Neural Networks, HMM, i-vectors, speaker adaptation, speech recognition.

1. INTRODUCTION

Speaker adaptation in speech recognition tries to reduce mismatch between the training and the test speakers. Reducing this mismatch results in significant improvement in recognition accuracy for the test speakers. This problem has mostly been studied in the GMM-HMM (Hidden Markov Models using Gaussian mixtures) speech recognition scenario. Perhaps the most popular method is to adapt the acoustic features to the acoustic phonetic models through a feature transform (fMLLR or CMLLR [1]). Another way is to adapt the acoustic phonetic models to the test speaker using a MLLR transform [1], or through MAP adaptation [2]. In vocal tract length normalization (VTLN) mapping [3], a frequency warping function or transform normalizes speech features for different speakers.

Recently, the hidden Markov models with deep neural nets (DNN-HMM) have been shown to outperform hidden Markov models with Gaussian mixtures (GMM-HMM) [4] [5]. This raises the question of how to do speaker adaptation for DNN-HMM systems. One approach currently being used is to compute a feature transform (fMLLR) using a GMM-HMM system and then to use the transformed features as input to the DNN-HMM system for both training and recognition [6]. This requires building a GMM-HMM system to compute the fMLLR transform for each speaker before the DNN-HMM system decoding step.

There have been a few attempts to modify the DNN architecture to do speaker adaptation. For example, Yao et al [7] adapt the affine transform of the top hidden layer or the softmax layer weights in the DNN to the speaker. They also use a feature-space discriminative linear regression (fDLR) method with the affine transformation of the input layer to adapt to the new speaker. This adaptation of the test speaker is done in a supervised mode: 200 utterances from the test speaker are used to estimate the weights and transformations. The resulting weights and transformations are then used to recognize other utterances from the same speaker.

Similarly, for DNN-HMM adaptation, Abdel-Hamid and Jiang [8] replace the input layer of the DNN by a small neural net to learn a speaker code and a feature transform. Results are reported on TIMIT phoneme recognition. During training, the speaker code for each speaker is unique, while the adaptation neural net is the same for all the speakers and its weights are trained jointly. The speaker code for each test speaker is learned from a small set of labelled adaptation utterances. The resulting speaker code is then used to recognize other utterances from the same speaker.

In this paper we show how an i-vector characterizing a speaker [9] can be used as an additional input to the feature layer of the DNN in order to adapt the DNN to the speaker. This is possible since i-vectors are a fixed dimensional representation of speech segments (the dimensionality is independent of the segment duration) [10]. During training and recognition, one i-vector per speaker is computed as an additional input to the DNN. All the frames corresponding to this speaker have the same i-vector appended to them. Speaker adaptation during decoding is completely unsupervised but a diarization step is needed in order to extract an i-vector for each speaker in the audio file. The computational overhead incurred in extracting the i-vectors is minimal.

This speaker adaptation scenario results in significant reduction in word error rate (WER) for ETAPE 2011 French broadcast audio transcription task. With cross entropy training, a WER of 22.16% without i-vector adaptation goes down to 20.67% with i-vector adaptation. After sequence training, the WER of 19.93% (without i-vector adaptation) goes down to 18.40% after i-vector adaptation. Sequence training alone without i-vector adaptation reduces the WER by 1.35%, while sequence training with i-vector adaptation reduces this WER by another 1.53%. The absolute reduction in WER due to i-vectors (1.53%) is higher than WER reduction due to sequence training without i-vector adaptation (1.35%).

2. ACOUSTIC TRAINING AND TEST DATA

The ETAPE training data consisted of training data for ESTER 2 evaluation[1] and the more recently transcribed audio for ETAPE eval-

[1]The data set resulting from the ESTER 2 evaluation campaign comprises about 250 hours of radio broadcast transcribed by human listeners, as well as the newspaper corpus Le Monde from 1987 to 2003. These datasets are

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

uation [11] for a total of 300 hours of audio. We also had 178 hours of internally transcribed audio from French TV broadcasts in Quebec. Overall, we had 478 hours of transcribed audio for training. In all the training audio, speaker segments were manually labeled in order to facilitate speaker-adapted training.

The ETAPE development set consisted of 15 files of 10 minutes to 1 hour in duration for a total duration of 8.6 hours. They were recorded from French radio and TV programs. The programs contained both broadcast news and talk shows.

3. BASELINE DNN-HMM SYSTEM

A detailed description of how we optimized the DNN-HMM system for recognizing French broadcast audio is given in [12]. We will just outline the entire training in brief here. We used the Kaldi toolkit for these experiments [13]. For training the deep neural network (DNN) using back propagation, we divided the 478 hours of training audio into 475 hours for training and 3 hours for validation. We got only small improvements with DNNs having more than 7 layers. So we carried out all the experiments with a 7 layer DNN. For input features, we experimented with filter-bank and MFCC features. We tried both the TRAP (TempoRAl Pattern) features [14] and the features with delta and delta-delta added to them as input to the neural net. The TRAP features extracted from filter-bank or MFCC features gave lower WER than the filter-bank or the MFCC features enhanced with delta and delta-delta coefficients.

To compute the TRAP features, we first normalize the 23-dimensional filterbank features to zero mean per speaker. Then 31 frames of these 23-dimensional filterbank features (15 frames on each side of current frame) are spliced together to form a 713-dimensional feature vector. This 713-dimensional feature vector is transformed using a hamming window (to emphasize the center), passed through a discrete cosine transform and the dimensionality is reduced to 368. This 368-dimensional feature vector is globally normalized to have zero mean and unit variance. This normalized 368-dimensional feature vector is then input to the 7-layer DNN. The feature vector is advanced by one frame every time. The 5 hidden layers have 2048 neurons each, and the output layer has 4148 outputs corresponding to the 4148 states in the HMM. The 7-layer DNN has a total of 26 million weights. The alignments and output labels for training the DNN come from the GMM-HMMs with 4148 states and 400k Gaussian means (trained with MLE only).

Several sources were used in the development of the language model [15]. They included broadcast news transcriptions from EPAC and ESTER campaigns, training transcripts from ETAPE, 350,000 sentences selected from French Gigaword database and Google 4-grams. The resulting trigram search LM has 1.8M trigrams and the quadgram LM for rescoring lattices has 20M quadgrams. The final model perplexity of the trigram search LM on the Etape development text is 113.

The initial vocabulary was selected by taking words with the highest frequency count weighted inversely with source size until a vocabulary size of 100 000 words was obtained. To this we added 30,000 words from the training set. As a last step, we added proper names found to occur at least twice in ETAPE, EPAC and ESTER sources, as well as French departments, Paris metro stations, and French acronyms taken from the Web. The final vocabulary was approximately 140,000 words.

We trained the 7-layer DNN using back propagation with cross-entropy as the objective function (CE) from 475 hours of audio. This was followed by two iterations of alignment of training data with the resulting DNN-HMM and retraining of the DNN using the aligned data (CE 2 iter) as suggested by Su et al [16]. The resulting DNN was then used for sequence training. Sequence training used the MMI criterion (no boosting or MPE or sMBR criteria) [6]. The word error rate on the ETAPE Dev set for the various steps is shown in Table 1.

Table 1. *WER for the ETAPE Dev set using cross-entropy training (CE), followed by 2 iterations of alignment and cross-entropy training (CE 2 iter), followed by sequence training (Sequence).*

CE	CE 2 iter	Sequence
22.16%	21.28%	19.93%

4. I-VECTOR EXTRACTION

When we are given a diarized audio file and we wish to transcribe it with a DNN-HMM speech recognition system, we represent each speaker in the audio file by an i-vector and use these i-vector features to perform blind speaker adaptation of the speech recognition system.

I-vectors arose as a byproduct of eigenvoice modeling [17, 10]. Suppose we are given recordings each consisting of the speech of a single speaker. Each recording is assumed to be represented by a Gaussian mixture model (GMM) and a principal components analysis is applied to the GMM supervectors. Thus the basic assumption is that all utterance supervectors are confined to a low dimensional subspace of the GMM supervector space so that each utterance supervector is specified by a small number of coordinates. For a given recording, these coordinates of the corresponding supervector define the i-vector representation.

In probabilistic eigenvoice modeling, i-vectors are assumed to have a standard normal prior and point estimates of i-vectors are obtained by a posterior probability calculation [17, 10]. Using $\boldsymbol{i}$ to denote i-vectors and $\boldsymbol{s}$ to denote utterance supervectors, probability model for supervectors is

$$\boldsymbol{s} = \boldsymbol{m} + \boldsymbol{T}\boldsymbol{i}, \quad \boldsymbol{i} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

where $\boldsymbol{m}$ is the supervector defined by a universal background model (UBM) and the columns of the matrix $\boldsymbol{T}$ are the eigenvoices. For each mixture component c, denote the UBM mean vector and covariance matrix by $\boldsymbol{m}_c$ and $\boldsymbol{\Sigma}_c$ and, for a given utterance, denote the corresponding zero and first order Baum-Welch statistics by N_c and $\boldsymbol{F}_c$. Then the posterior covariance matrix $\boldsymbol{C}$ and the posterior expectation $\boldsymbol{i}$ are given by

$$\begin{aligned} \boldsymbol{C} &= \left(\boldsymbol{I} + \sum_c N_c \boldsymbol{T}_c^* \boldsymbol{\Sigma}_c^{-1} \boldsymbol{T}_c \right)^{-1} \\ \boldsymbol{i} &= \boldsymbol{C} \sum_c \boldsymbol{T}_c^* \boldsymbol{\Sigma}_c^{-1} (\boldsymbol{F}_c - N_c \boldsymbol{m}_c). \end{aligned} \quad (1)$$

We trained the i-vector extractor (that is, the matrix $\boldsymbol{T}$) and the underlying UBM using the same 473 hours of broadcast audio used for training the DNN. This training audio contains 99018 utterances from 6597 speakers. The UBM configuration has 512 diagonal Gaussians and 60 dimensional acoustic feature vectors (Gaussianized MFCCs and their first and second derivatives).

distributed by ELDA and by the DGA.

We extracted i-vectors of dimension 100, 200 and 400 in order to test speaker adaptation with these three different i-vector dimensions. The main computing in generating the i-vectors is in extracting the first order Baum-Welch statistics. The total computing in extracting the i-vectors is around 1% of real time. This is much smaller than 20% of real-time for extracting fMLLR transforms per speaker using a small HMM-GMM model [12].

For the training speakers, the audio segments correspond to the speaker labels in the manually transcribed transcription files. For the ETAPE Dev set used for testing, each show was automatically diarized using a multistage segmentation and clustering system [18]. This diarization system segmented the 8.6 hours of the ETAPE Dev set into 2099 audio segments and 271 speaker clusters. Each of the 15 audio files in the Dev set was diarized separately. There was no cross-file diarization. The overall diarization error rate for this system was approximately 14%. For each speaker cluster in the Dev set, we estimated one i-vector.

5. DNN-HMM SYSTEM WITH SPEAKER ADAPTATION

For the speaker-adapted deep neural network, the input TRAP features are computed as in the baseline system. However, these TRAP features are now augmented with a 100-dimensional i-vector computed from all the audio segments of the speaker as shown in Fig. 1. The only difference between this DNN and the baseline DNN is the addition of 100 more input features corresponding to the 100-dimensional i-vector. Instead of 368 TRAP features, the DNN input layer has 368 TRAP features and 100 dimensional i-vector as input. During training, the 100 dimensional i-vector for a speaker is computed from all the audio segments in an audio file (or show) labeled with that speaker Id. Note that each training audio segment is marked with a speaker Id. No cross show marking is done.

During decoding, the audio from a show is automatically diarized into speaker clusters using a multistage segmentation and clustering system [18]. We compute one 100-dimensional i-vector for each speaker cluster. This 100-dimensional i-vector together with 368 TRAP features are input to the DNN. So for the features input to the DNN for one speaker, the 368 TRAP features vary from frame to frame. However, the 100-dimensional i-vector for this speaker is fixed and does not vary from frame-to-frame. This speaker-specific i-vector provides the speaker characteristics to the DNN, and allows the DNN to adapt itself to these speaker characteristics.

The training of the DNN augmented with the i-vector is performed in the same fashion as that for the baseline system. We train this DNN using back propagation with cross-entropy (CE) objective function and stochastic gradient descent (SGD) as outlined in [6]. Basically, we start with a learning rate of 0.008. We start halving this learning rate when the frame accuracy of a 3-hour validation set improves by less than 0.5% between successive iterations. The DNN training terminates when the frame accuracy of this validation set increases by less than 0.1% between successive iterations.

5.1. Results with speaker adaptation

We trained a 7-layer DNN augmented with the i-vector on 475 hours of French broadcast audio. We trained two separate DNN's, one with 100 dimensional unnormalized i-vectors, and the other one with 100-dimensional length normalized i-vectors. The i-vector was length normalized by dividing the i-vector by the square root of the sum of the squares of its elements. Note that the unnormalized i-vectors are approximately Gaussianized by length normalization [19]. We found that the length normalized i-vectors gave lower WER than the unnormalized i-vectors as shown in Table 2. The speaker adaptation through length normalized i-vectors reduced the WER by 1.26% absolute.

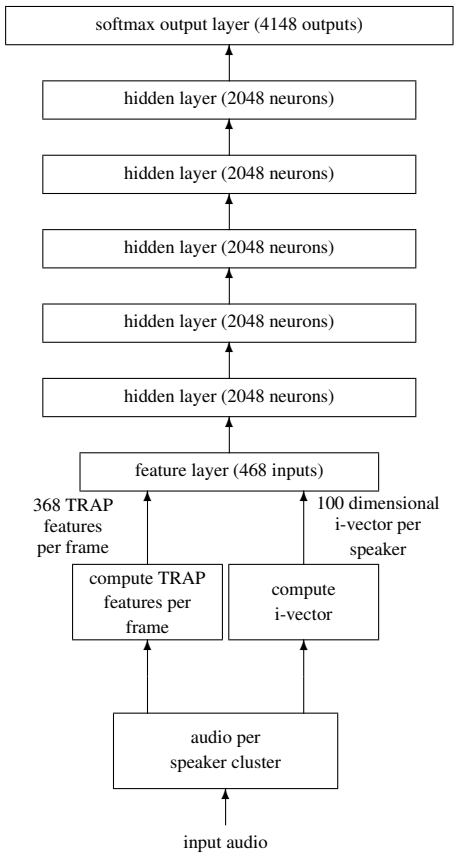

Fig. 1. *Architecture of 7-layer DNN used for speaker adaptation.*

Table 2. *Comparison of WER for the ETAPE Dev set using cross-entropy training (CE) for baseline DNN (without i-vector), for DNN augmented with unnormalized i-vector, and with length normalized i-vector.*

baseline	unnormalized i-vector	normalized i-vector
22.16%	21.62%	20.90%

To see if the i-vector adaptation is effective for both ETAPE Dev set speakers that are in the training set and for ETAPE Dev set speakers not in the training set, we compiled separate WER stats for the Dev set for speakers that are in the training set versus speakers that are not in the training set. A total of 1447 segments in the Dev set correspond to speakers in the training set, and 3103 segments that belong to speakers not in the training set. The breakdown of WER before and after i-vector speaker adaptation is shown in Table 3. As we can see from the table, the reduction in WER due to i-vector adaptation is similar for the two groups. In other words, the deep neural net can effectively adapt for both seen and unseen speakers with the help of i-vectors.

In order to optimize the length-normalized i-vector dimension, we trained DNN with speaker adaptation using i-vector dimensions of 100, 200 and 400. The i-vector dimension of 400 gave the lowest

Table 3. *WER for the ETAPE Dev set speakers broken down into speakers that occur in the training set versus speakers not in the training set for baseline DNN (without i-vector) and for DNN augmented with length normalized i-vector.*

speakers	baseline	normalized i-vector
in training	19.76%	18.68%
not in training	23.25%	21.91%

WER as shown in Table 4. The WER of 22.16% without adaptation goes down to 20.67% with i-vector adaptation, a reduction of 1.49% in WER. Note that with 400-dimensional i-vector, the DNN input has more features corresponding to the speaker than to the utterance (400 vs 368).

Table 4. *Comparison of WER for the ETAPE Dev set using cross-entropy training (CE) for baseline DNN (without i-vector), for DNN augmented with length normalized i-vector of dimensions 100, 200, and 400.*

i-vector dimension	WER
no ivec	22.16%
100	20.90%
200	20.76%
400	20.67%

In the baseline system, we trained the 7-layer DNN with cross-entropy (CE), followed by two more iterations of training alignment and CE training using the resulting DNN as suggested in [16]. This was followed by sequence training of the resulting DNN with MMI criteria. We tried the same scenario for training the DNN with speaker adaptation using length normalized i-vector of dimension 100. Table 5 compares the results of the baseline system with the DNN using i-vector speaker adaptation. Note that the absolute reduction in WER due to sequence training without i-vectors is 1.35%, while the reduction in WER after i-vectors is 1.53% in addition to that obtained by sequence training without i-vectors (19.93% WER without i-vectors versus 18.40% WER with i-vectors).

Table 5. *Comparison of WER for the ETAPE Dev set using baseline DNN vesus DNN with length normalized i-vector. The DNN is trained with cross-entropy (CE), 2 iterations of alignment and CE training (CE 2 iter), followed by sequence training (sequence).*

DNN	CE	CE 2 iter	sequence
no ivec	22.16%	21.28%	19.93%
100-dim ivec	20.90%	20.50%	18.40%

In order to see if MFCC's that have been transformed using an fMLLR transform per speaker provide better speaker adaptation, we trained a small GMM-HMM system as outlined in [12] and used it to generate one fMLLR feature transform for each training and test speaker. The features for both the training and test speakers were then transformed using these fMLLR transforms. The resulting transformed features are input to the neural net as in [6]. Basically, MFCCs (C0-C12) are first normalized to have zero mean per speaker. Nine frames (4 on each side of current frame) are spliced together and projected down to 40 dimensions using an LDA transform. These are then transformed using a semi-tied covariance (STC) transform to reduce the correlations between the features. The resulting features are then transformed using an fMLLR transform per speaker. The 40-dimensional LDA+STC+fMLLR features are then spliced together (4 frames on each side of current frame) and reduced to 300 dimensions by another LDA transform. These 300-dimensional features are then globally normalized to zero mean and unit variance before input to neural net. We trained a 7 layer neural net (5 hidden layers with 2048 neurons each and an output layer with 4148 outputs) with these features using CE training. The WER for the Dev set with this neural net is 22.6% compared to 20.9% with length-normalized i-vectors. Obviously, it is not necessary that we use either i-vectors or fMLLR transformed features. Maybe using the two jointly may lead to a better speaker adaptation scenario.

6. CONCLUSIONS

We have shown that deep neural networks (DNN) can adapt to speaker characteristics if we augment standard acoustic features by appending i-vectors to them. This method is simple to implement and the computational overhead is minimal. It was motivated by the success of i-vectors in speaker recognition where the i-vector representation has been found to be very useful because it can serve to characterize speakers by vectors of low, fixed dimension. If other representations having this property prove to be equally effective in speaker recognition, then they can probably be used to perform blind speaker adaptation of DNN-HMM systems in the same way as we have described here.

We observed that i-vector based speaker adaptation produced substantial reductions in word error rates on our French language broadcast audio testbed with both cross entropy and sequence training. With cross entropy training, the 22.16% WER without i-vector adaptation goes down to 20.67% with i-vector adaptation. After sequence training, the WER of 19.93% without i-vector adaptation goes down to 18.40% after i-vector adaptation. Sequence training alone without i-vector adaptation reduces the WER by 1.35%, while i-vector adaptation reduces this WER by another 1.53%. The absolute reduction in WER due to i-vectors (1.53%) is higher than WER reduction due to sequence training (1.35%). We also saw that i-vector based speaker adaptation was effective with previously unseen speakers as well as with speakers encountered in the course of training.

Finally we remark that relatively high dimensional i-vectors (e.g. 400 dimensions) gave the lowest word error rates. Thus it appears that there is no danger of overfitting in DNN training.

7. REFERENCES

[1] M. Gales,"Maximum likelihood linear transformations for HMM-based speech recognition",'*Computer Speech and Language*, vol.12, 1998.

[2] J. Gauvain, C. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains",'*IEEE Trans. Speech and Audio Proc.*, vol.2, no.2, pp. 291–298, 1994.

[3] L. Lee and R. Rose, "Speaker normalisation using efficient frequency warping procedure", Proc. ICASSP, volume 1, pages 353–356, 1996.

[4] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic Modeling Using Deep Belief Networks", *IEEE Trans. Audio Speech Lang Proc.*, vol 20, No 1, Jan 2012, pp. 14–22.

[5] G. Dahl, D. Yu, L. Deng and A. Acero, "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition", *IEEE Trans. Audio Speech Lang Proc.*, vol 20, No 1, Jan 2012, pp. 30–42.

[6] K. Veselý, A. Ghosal, L. Burget, D. Povey, "Sequence-discriminative training of deep neural networks", Proc. Interspeech 2013, pp. 2345–2349.

[7] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, Y. Gong, "Adaptation of Context-dependent deep neural networks for automatic speech recognition", Proc. Spoken Language Technology Workshop 2012, pp. 366–369.

[8] O. Abdel-Hamid, H. Jiang, "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code", Proc. ICASSP 2013, pp. 7942–7946.

[9] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 4, pp. 788–798, May 2011.

[10] P. Kenny, "A small footprint i-vector extractor", Proc. Odyssey 2012, Singapore.

[11] G. Gravier, G. Adda, N. Paulsson, M. Carré, A. Giraudel, O. Galibert,"The ETAPE corpus for the evaluation of speech-based TV content processing in the French language", Proc. LREC 2012, Istanbul, Turkey.

[12] V. Gupta, G. Boulianne, "comparing computation in gaussian mixture and neural network based large-vocabulary speech recognition", Proc. Interspeech-2013.

[13] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hanneman, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer and K. Vesely, "The Kaldi Speech Recognition Toolkit", IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, 2011.

[14] F. Grézl, "TRAP-based Probabilistic Features for Automatic Speech Recognition", Doctoral Thesis, dept. Computer Graphics & Multimedia, Brno Univ of Technology, Brno 2007.

[15] V. Gupta, G. Boulianne, F. Osterrath, and P. Ouellet, "CRIM's French Speech Transcription System for ETAPE 2011", Proc. WOSSPA-2013, pp. 318–323.

[16] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription", Proc. ICASSP-2013, pp. 6664–6668.

[17] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Trans. Speech Audio Processing*, vol. 13, no. 3, pp. 345–359, May 2005.

[18] V. Gupta, G. Boulianne, P. Kenny, P. Ouellet, P. Dumouchel, "Speaker Diarization of French Broadcast News", Proc. ICASSP-2008, pp. 4365–4368.

[19] D. Garcia-Romero, and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems", Proc. Interspeech-2011.