INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Native Language Detection using the I-Vector Framework

Mohammed Senoussaoui[1], *Patrick Cardinal*[1], *Najim Dehak*[2], *Alessandro L. Koerich*[1]

[1]École de Technologie Supérieure, Montréal, Canada
[2]Johns Hopkins University, Baltimore, USA

mohammed.senoussaoui.1@ens.etsmtl.ca, patrick.cardinal@etsmtl.ca
ndehak3@jhu.edu, alessandro.koerich@etsmtl.ca

Abstract

Native-language identification is the task of determining a speaker's native language based only on their speeches in a second language. In this paper we propose the use of the well-known i-vector representation of the speech signal to detect the native language of an English speaker. The i-vector representation has shown an excellent performance on the quite similar task of distinguishing between different languages. We have evaluated different ways to extract i-vectors in order to adapt them to the specificities of the native language detection task. The experimental results on the 2016 ComParE Native language sub-challenge test set have shown that the proposed system based on a conventional i-vector extractor outperforms the baseline system with a 42% relative improvement.
Index Terms: native language, i-vector, computational paralinguistics, ComParE.

1. Introduction

The task of recognizing the native language of a speaker or his mother tongue is called Native language identification (NLI). This task can be considered similar to the Language Identification task (LID), which consists of identifying the language spoken in a given speech segment. It is well known that even if a multilingual speaker is able to speak correctly a second language (L2), his native language (L1) can still influence the pronunciation of the second one. This speaking behavior can be seen as soft biometrics information and it can be helpful for identifying the speaker identity. An automatic NLI system can be useful for several other speech applications such as speech and speaker recognition. In such applications, the speaker?s accent is considered as a nuisance that needs to be compensated.

Several levels of information extracted from speech signal have been studied for NLI. Acoustics features and prosodic cues [1] are considered the most popular characteristics for NLI. Phonetic characteristics such as phonemes frequent appearance and their duration also provide a good indication of speaker?s mother tongue [2]. Note that the same task can be achieve by analysing the text instead of the speech signal [3].

Since seven years ago, the i-vector framework became the state-of-the-art speech representation for text-independent speaker recognition as well as for many other speech related fields such as language recognition [4, 5, 6, 7] and speaker adaptation for automatic speech recognition [8]. The i-vector is an elegant way to represent speech segments of variable lengths in the same space of moderate dimension (typically in the range of hundreds) [9]. In this space, several kind of machine learning approaches can be applied to solve different kind of audio classification problems. Therefore, the i-vector will be our main speech representation investigated in this NLI challenge.

In this paper we propose two different i-vector extraction strategies to the problem of native language detection: language-independent and language-dependent. In the former, a unique language-independent i-vector extractor is estimated for all the native language classes. The latter strategy consists of training one native language-dependent i-vector for each native language class [10]. Furthermore, we also evaluated three normalization strategies for the language-dependent i-vector representation. The different i-vector implementations are evaluated and compared within the scoop of the ninth edition Computational Paralinguistics Challenge (ComParE) [11, 12].

The remainder of this paper is organized as follows. Section 2 reviews the conventional i-vector speech representation as well as its related intersession compensation methods. Section 3 describes the language-dependent i-vector representation and the three related intersession compensation strategies. In Section 4, all the experiments and the results are presented and discussed. Finally, in the last section we present some future research directions as well as the conclusions of this work.

2. Conventional i-vector framework

The i-vector framework [9] is based on modeling speech segments using a universal background model (UBM), which is typically a large Gaussian Mixture Models (GMM) where the UBM is usually trained on a large quantity of data to represent general feature characteristics. This model plays the role of a prior on how all sounds look like. The i-vector approach is a powerful technique that summarizes all the updates happening during the adaptation of the UBM mean components to a given speech recording. All this information is modeled in a low dimensional subspace referred to as the total variability space. In the i-vector framework, each speech utterance can be represented by a GMM supervector, which is assumed to be generated as follows:

$$\boldsymbol{M} = \boldsymbol{m} + \boldsymbol{T}\boldsymbol{x} \tag{1}$$

where $\boldsymbol{m}$ is the language and dialect independent supervector (which can be taken to be the UBM supervector), $\boldsymbol{T}$ is a rectangular matrix of low rank and $\boldsymbol{x}$ are the factors that best describe the utterance-dependent mean offset. The vector $\boldsymbol{x}$ is treated as a latent variable with the i-vector being its maximum-a-posteriori point estimate. The subspace matrix $\boldsymbol{T}$ is estimated using maximum likelihood on a large training set [9]. Figure 1 summarizes the steps used to extract the i-vectors.

It is well known that the i-vector space models a wide variety of variability embedded in the speech signal and this is mainly due to the fact of using unsupervised methods for the

http://dx.doi.org/10.21437/Interspeech.2016-1473

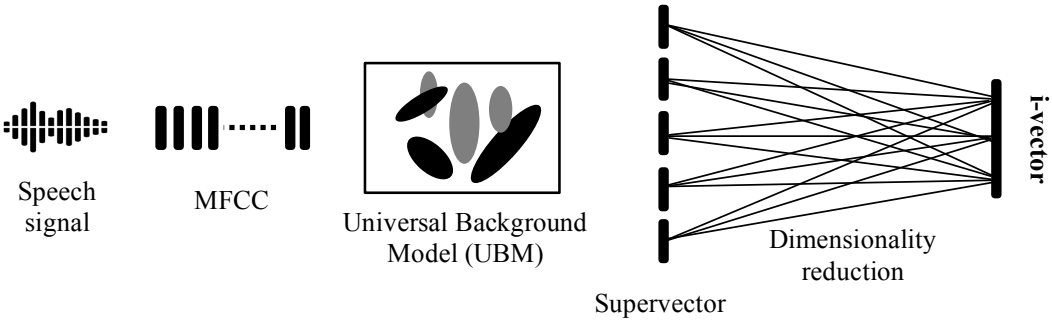

Figure 1: *Conventional i-vector feature extraction from a given speech signal.*

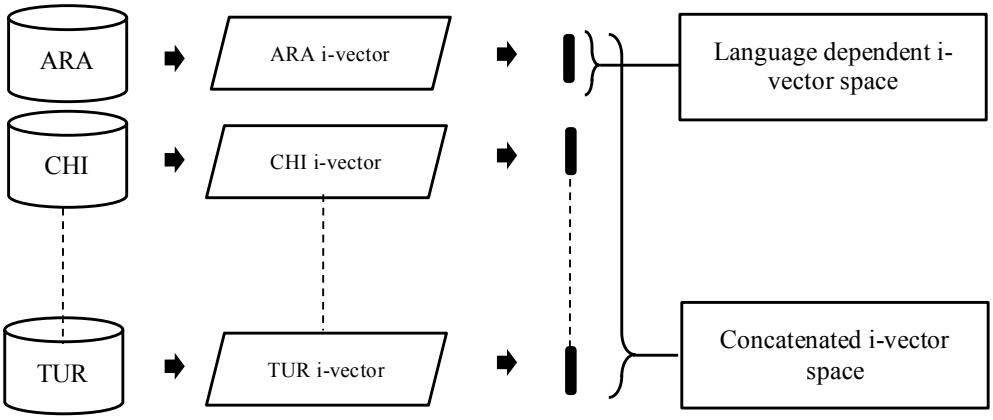

Figure 2: *Per-Native-language i-vector feature spaces.*

supervector space dimensionality reduction. In order to extract any specific information from this space (e.g. the language nativeness), task-dependent normalization (i.e. intersession compensation) needs to be performed in the raw i-vector before carrying out any recognition task. The main objective of the task-dependent normalization is the mapping of the raw i-vector into a new space that maximizes only the useful information and minimizes the effect of unwanted variability. In the i-vector context, the most known normalization procedure [13] is based on a combination of the Linear Discriminant Analysis (LDA) followed by the Within Class Covariance Normalization (WCCN) [14]. Moreover, it has been found that the normalization of the i-vector to the unit length (i.e. norm = 1) is helpful for speaker recognition systems [15]. The detailed procedure of i-vector space normalization adopted in this work is depicted in Algorithm 1.

Algorithm 1 i-vector space normalization

1: Normalize the length of the i-vector $\boldsymbol{x}$ to one.
2: Map the length normalized i-vector to the LDA subspace using the optimal dimension (in our case 10 was the best dimension).
3: Normalize again the length of the LDA normalized i-vector.
4: Rotate the length-, LDA- and length-normalized i-vector using the Cholesky decomposition of the inverse Within Class Covariance.

3. Language-dependent i-vector

A class-dependent i-vector extractor is trained using each class dependent subset of data individually in order to model the class aspect in our native language dataset. This approach is similar to an approach that has been already successfully implemented for the problem of language Identification [10]. In this modeling, eleven i-vector extractors (including UBMs and the total variability matrices $\boldsymbol{T}$) were trained on the ComParE native language training set (i.e. the ETS Corpus Of Non-Native Spoken English) [11]. All the obtained language-dependent subspaces can be exploited in two different ways: i) Each language-dependent subspace can be processed individually; ii) All language-dependent i-vectors can be concatenated to create a larger and unique subspace (c.f. Figure 2). In this work the dimension of the language-dependent space is chosen to be 50, which means that the dimension of the concatenated space is $50 \times 11 = 550$.

We have adopted three different strategies to deal with undesirable variability in the different i-vector spaces. However, all these strategies are based on the same normalization technique as described in Algorithm 1. The first strategy consists

Table 1: Unweighted Average Recall (UAR %) for the development and test sets.

	Conventional i-vectors		Language-dependent i-vector						Baseline	
			Norm. Strategy I		Norm. Strategy II		Norm. Strategy III			
	Devel	Test	Devel	Test	Devel	Test	Devel	Test	Devel	Test
UBM 512 dim 400	**66.8**	**66.4**	-	-	-	-	-	-	45.1	47.5
UBM 256 dim 50	-	-	38.2	-	43.9	-	44.6	-		
UBM 128 dim 50	-	-	43.2	-	45.9	-	46.9	-		
UBM 64 dim 50	-	-	51.8	-	45.2	-	49.2	-		
UBM 32 dim 50	-	-	53.4	-	44.9	-	45.8	-		
UBM 16 dim 50	-	-	49.7	-	43.6	-	44.9	-		

Table 2: Confusion matrix in percent for the development set using the conventional i-vector.

	ARA	CHI	FRE	GER	HIN	ITA	JAP	KOR	SPA	TEL	TUR
ARA	52	1	6	4	3	5	2	2	4	1	6
CHI	0	62	0	1	4	0	4	10	3	0	0
FRE	3	1	51	8	2	5	2	1	6	0	1
GER	1	4	4	72	0	0	0	1	2	0	1
HIN	0	0	0	0	56	0	0	1	0	26	0
ITA	1	0	9	3	0	72	0	3	5	0	1
JAP	0	6	4	0	1	1	54	16	3	0	0
KOR	1	12	0	1	0	2	15	57	1	0	1
SPA	5	5	9	3	1	13	8	9	39	3	5
TEL	1	0	0	0	22	0	0	2	1	56	1
TUR	1	0	1	5	1	4	3	1	4	1	74

of running the space normalization procedure (c.f. Algorithm 1) directly on the concatenated language-dependent space to generate a 10-dimensional normalized i-vector. In the second strategy, we first apply separately the normalization procedure to each language-dependent subspace to obtain eleven 10-dimensional normalized i-vectors which are then stacked to form a single 110-dimensional i-vector. In the third strategy, we simply carry out the normalization procedure on the output of the second strategy (i.e. the 110-dimensional i-vector). The main objective behind this strategy is to map the concatenated space into a more homogeneous space of dimension 10.

4. Experimental Results

The performance of the proposed approach was evaluated on the ETS Corpus of Non-native spoken English which includes more than 64 hours of speech from 5,132 non-native speakers of English, with eleven different L1 backgrounds (Arabic (ARA), Chinese (CHI), French (FRE), German (GER), Hindi (HIN), Italian (ITA), Japanese (JAP), Korean (KOR), Spanish (SPA), Telugu (TEL), and Turkish (TUR)). The dataset was divided into three stratified partitions: 3,300 instances (64%) were selected as training set, 965 instances (19%) for the development set, and 867 responses (17%) used as test data. As measure of performance, we employ Unweighted Average Recall (UAR). For more details on this dataset please refer to [11].

The i-vectors have been extracted using Kaldi, a free open source toolkit for speech recognition [16]. The experiments have been conducted with support vector machine (SVM) using the sequential minimal optimization (Weka toolkit [17]) and with neural networks (NN) using the TensorFlow library [18].

4.1. Results with SVM

The fist step of the i-vector estimation process consists of extracting a sequence of short-term features (Figure 1). In this work, the Mel Frequency Cepstral Coefficients (MFCCs) augmented by their Shifted Delta Cepstral (SDC) are used as short-term features. The SDC features are able to model long-term speech temporal information which seems to be very useful for the Language Identification task [19]. The parameters used to to generate the SDC features were set as follows: The number of the cepstral coefficients was set to 20 (i.e. static MFCC); the distance between blocks was set to 3; the size of delta advance and delay was set to 1 and the number of blocks in advance of each frame to be concatenated was set to 7. This setup produces feature vectors of 160 dimensions.

Table 1 shows the results achieved by both conventional i-vector and language dependent i-vectors. A linear kernel had been used for all experiments. The best result was achieved with the language-independent i-vector (UAR of 66.4%) which also outperforms the baseline system (UAR of 47.5%). Regarding the language-dependent i-vectors, since the amount of data per language is very limited, several experiments have been conducted with different UBM sizes. It is not clear from the different UBM size performances that there is an optimal configuration and it mainly depends on the normalization strategy. The overall results show that the language-independent approach seems to be less effective for this task than using language-independent i-vectors.

The i-vector extractor used in these experiments is based on GMMs with full covariance UBMs, which requires a large amount of data for training. In another set of experiments, we have used UBMs with diagonal covariance matrices. Unfortu-

nately, no improvement has been observed. Another possibility to explore is the reduction of the MFCC-SDC features that are quite large (160 dimensions) for a GMM.

4.2. Results with NN

Table 3 summarizes the two best results obtained with NNs. These results are very close to those obtained with a SVM with i-vector as input, with a slight improvement of 0.2%. The best architectures for the NNs were with a single hidden layer and with two hidden layers. However, both architectures achieved very close UARs. The best result was achieved by combining two neural networks. The first neural network had one hidden layer of 1024 neurons with the sigmoid activation function and 11 neurons in the softmax output layer. The output of such a NN was concatenated with the i-vector and used as input for a second NN with one hidden layer of 256 neurons. This architecture led to the best result on the development set with a UAR of 68.4%.

Table 3: Results on neural network (NN) measured by the Unweighted Average Recall (UAR %).

Description	UAR (Devel)
ivec	67.0%
ivec + dist	68.4%

4.3. Combining SVM and NN

The confusion matrix (Table 2) generated by the system using both the i-vector and the output layer of a NN with 256 neurons in a single hidden layer shows that there is a high confusion between some languages (e.g. Hindi and Telugu). The confusion matrices of other systems exhibit the same behavior, meaning that the combination should improve the results.

Therefore, the last experiments were carried out to explore the use of the NN as a feature extractor for the SVM. Table 3 shows the main results achieved with this approach. A conventional i-vector was used as input for a single hidden layer NN. Results with two or three hidden layers are not reported because they were very similar or worst than those obtained with a single hidden layer.

Table 4: Results with NN as feature extractor for the SVM (measured by the Unweighted Average Recall (UAR %)).

Description	UAR (Devel)
1024 neurons	66.5%
512 neurons	67.4%
256 neurons	67.6%
Output dist, 1024 neurons	67.2%
Output dist, 256 neurons	67.2%
ivec + dist (256)	66.9%
ivec + dist +output (256)	66.4%
ivec + dist (1024)	67.3%
ivec + dist (1024) on NN	68.4%

The experiments use the output of the hidden layer (before the activation function) as input of the SVM. The best UAR achieved with this scheme is 67.6%, which is almost 1% higher than the best result achieved by the SVM using the i-vector as input. This result shows that the NN is able to extract complementory information from the i-vector. Other experiments have explored the use of NN output layer (after the softmax) as input of the SVM. The results achieved with this schema are almost as good as those obtained with the hidden layer and better than the best result obtained with the SVM alone. Note that this result also represents a small improvement over the NN alone (67%). The third experiment reported in Table 3 has explored the combination of the i-vector and the features of NN for the SVM. This approach led to a very small improvement with a UAR of 67.3%. Finally, the last experiment was to use another NN with the concatenation of the i-vector and the NN output layer as input. This configuration achieved the best result on the development set with a UAR of 68.4%.

4.4. Results on the test set

Considering the limited number of five trials to evaluate the proposed approach on the test set for the 2016 ComParE Native language sub-challenge, we have selected the approaches that have provided the highest UAR on the development set. The first three rows of Table 5 correspond to the best approaches on the development set using the SVM classifier. The fourth row was achieved by a SVM for which the input is the concatenation of SVM output confidence scores and NN output distribution. Both classifiers have a conventional i-vector as input. Surprisingly, we did not have observed any improvement with such a fusion of scores for the test set. The fifth row of Table 5 corresponds to the approach for which we have achieved the best UAR on the development set. Unfortunately, such an approach did not provided the best result on the test set as we should expect. Compared to the other approaches, it seems that the NN was overfitted during the training process. Finally, the last row shows the UAR obtained by the official baseline system [11].

Table 5: Results on the test set by the Unweighted Average Recall (UAR %).

Description	UAR (Test)
ivec on SVM	66.4%
ivec on NN	66.6%
ivec + dist (1024) on SVM	67.4%
score fusion on SVM	67.4%
ivec + dist (1024) on NN	67.1%
Official baseline [11]	47.5%

5. Conclusions

In this paper we have explored two different implementations of the well-known i-vector representation of speech to deal with the problem of native language (L1) identification for an English (L2) speaker. The first implementation based on the conventional language-independent i-vector outperforms by far the baseline system (UAR of 66.4% vs. 47.5% on the test set) on the development and test sets of the ComParE 2016 challenge. The combination of i-vector with features extracted by a NN using the same i-vector representation as input led to UAR up to 67.4% on the test set, an improvement of 1% absolute.

6. Acknowledgments

The authors acknowledge funding from FRQNT and would like to thank Jim Glass of the CSAIL Lab for giving us access to their cluster.

7. References

[1] R. Todd, "On Non-Native Speaker Prososdy: Identifying 'Just Noticeable-Differences' of Speaker-Ethnicity," in *The 1st International Conference on Speech Prosody*, 2002.

[2] E. Shriberg, L. Ferrer, S. Kajarekar, N. Scheffer, A. Stolcke, and M. Akbacak, "Detecting nonnative speech using speaker recognition approaches," in *Odyssey Workshop*, 2008.

[3] V. Kríz, M. Holub, and P. Pecina, "Feature extraction for native language identification using languagemodeling," in *Recent Advances in Natural Language Processing, RANLP 2015, 7-9 September, 2015, Hissar, Bulgaria*, 2015, pp. 298–306.

[4] N. Dehak, P. A. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language Recognition via I-Vectors and Dimensionality Reduction," in *INTERSPEECH 2011*, Florence, Italy, Aug. 2011, pp. 857–860.

[5] D. Martnez, L. Burget, L. Ferrer, and N. Scheffer, "ivector-based prosodic system for language identification," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, March 2012, pp. 4861–4864.

[6] M. Soufifar, M. Kockmann, L. s Burget, O. Plchot, O. Glembek, and T. Svendsen, "ivector approach to phonotactic language recognition," in *INTERSPEECH*, 2011.

[7] D. Martnez, E. Lleida, A. Ortega, and A. Miguel, "Prosodic features and formant modeling for an ivector-based language recognition system," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013, pp. 6847–6851.

[8] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, Dec 2013, pp. 55–59.

[9] 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*, 2011.

[10] P. Matejka, O. Plchot, M. Soufifar, O. Glembek, L. F. D'Haro, K. Vesel, F. Grzl, J. Z. Ma, S. Matsoukas, and N. Dehak, "Patrol team language identification system for darpa rats p1 evaluation." in *INTERSPEECH*. ISCA, 2012, pp. 50–53.

[11] B. Schuller, S. Steidl, A. Batliner, J. Hirschberg, J. K. Burgoon, A. Baird, A. Elkins, Y. Zhang, E. Coutinho, and K. Evanini, "The interspeech 2016 computational paralinguistics challenge: Deception, sincerity & native language," in *Submitted to INTERSPEECH 2016*, Sept 2016.

[12] B. Schuller, A. Batliner, S. Steidl, and D. Seppi, "Recognising realistic emotions and affect in speech: State of the art and lessons learnt from the first challenge," *Speech Communication*, vol. 53, no. 910, pp. 1062 – 1087, 2011, sensing Emotion and Affect - Facing Realism in Speech Processing.

[13] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification." in *Interspeech*, vol. 9, 2009, pp. 1559–1562.

[14] A. O. Hatch, S. S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for svm-based speaker recognition." in *Interspeech*, 2006.

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

[16] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011, iEEE Catalog No.: CFP11SRW-USB.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[18] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[19] P. Torres-Carrasquillo, E. Singer, M. Kohler, R. Greene, D. Reynolds, and J. Deller, "Approaches to language identification using gaussian mixture models and shifted delta cepstral features," in *ICSLP*, Sept 2002, pp. 89–92.