Odyssey 2016
21-24 Jun 2016, Bilbao, Spain

I^{2}R Submission to the 2015 NIST Language Recognition I-vector Challenge

Hanwu Sun, Trung Hieu Nguyen, Guangsen Wang,
Kong Aik Lee, Bin Ma, Haizhou Li

Institute for Infocomm research, A*STAR, Singapore

{hwsun,thnguyen,wang-g,kalee,mabin,hli} @i2r.a-star.edu.sg

Abstract

This paper presents a detailed description and analysis of I^{2}R submission, which is among the top performing systems, to the 2015 NIST *language recognition i-vector machine learning challenge*. Our submission is a fusion of several sub-systems based on linear discriminant analysis (LDA), support vector machine (SVM), multi-layer perceptron (MLP), deep neural network (DNN), and multi-class logistic regression. Central to our work presented in this paper is a novel out-of-set (OOS) detection scheme for selecting i-vectors from an unlabeled development set. It consists of a *best fit out-of-set* selection followed by cluster purification. We also propose a novel empirical kernel map to be used with SVM. Experimental results show that the proposed approach achieves significant improvement on both the progress and evaluation sets defined for the i-vector challenge. Our final submission achieves 55.0% and 54.5% relative improvement over the baseline system on the progress and evaluation sets, respectively.

1. Introduction

Following the success of the *i-vector Machine Learning Challenge* for speaker recognition [1, 2, 3, 4], NIST coordinated the 2015 edition of similar challenge for language recognition (LR) [5]. The LR i-vector challenge focuses on the development of new methods using i-vectors for language recognition [6] in the context of conversational telephone and narrowband broadcast speech. The use of i-vectors allows the participants to focus on improving the performance of language recognition system without having to go through complicated front-end processing which is typical in speech processing. The aim was to make the challenge accessible to participants from a wider *machine learning* community so as to promote the application of new machine learning techniques for language recognition.

The 2015 NIST i-vector challenge focuses on open-set language identification task [1, 5]. Given a test segment in the form of i-vector, the task is to determine which language from a pre-defined set of target languages is being spoken in the test segment. Under the open-set context, the actual language of the test segments may come from any of the out-of-set (OOS) languages, which are different from that of the target set. Specific to the 2015 NIST i-vector challenge is the set of 50 target languages used, which is the largest set compared to previous LREs. Training data was provided for all the 50 target languages. Nevertheless, the set of OOS languages remains unknown and no labeled training data was given.

The primary scoring metric takes into account the incorrect decision percentage across all target and OOS languages. The final cost function gives a higher weight to the detection of OOS compared to individual language in the target set. This set the stage whereby OOS modeling and detection becomes an important component in order to achieve a good performance. In this regard, NIST provided an unlabeled development set which consists of a mixed of i-vectors extracted from the target as well as some unknown languages. Data for modeling the OOS class have to be drawn from this unlabeled development set. Also, the unknown languages in the development set may not overlap completely with the OOS languages appear in the evaluation set.

Our contribution in this paper is two-fold. First, we propose a hybrid method to identify OOS i-vectors from the unlabeled development set. The selected i-vectors were then used to train a model representing the OOS class. Inclusion of an additional model representing the OOS class leads to a significant improvement for the language identification task. Second, we propose an empirical kernel that maps input i-vectors to score supervectors with dimension equal to the size of training data points. These score supervectors were then fed into support vector machine (SVM) [7] for language classification.

The paper is organized as follows. Section 2 describes the training, development, and evaluation datasets provided by NIST for the 2015 i-vector challenge, as well as the LR evaluation cost function. Section 3 describes the strategy for identifying OOS segments from the unlabeled development data. Section 4 presents the classifiers used. Finally, the experimental results on the progress set are presented in Section 5. Section 6 concludes the paper.

2. Dataset, Baseline, and Performance Metric

2.1. Dataset

The i-vectors provided by NIST for the 2015 i-vector challenge were extracted using the system developed by the Johns Hopkins University and MIT Lincoln Laboratory [8]. Each i-vector has a dimension of 400. The challenge consists of a pre-defined set of 50 target languages. Each target language has a set of 300 training segments, each specified by a single i-vector. A total of 15,000 i-vectors were provided for the 50 target languages. Besides the training set for the 50 target languages, an unlabeled development set consisting of 6,431 i-vectors was provided for general system development purposes. These unlabeled data consists of a mixed of i-vectors extracted from the target as well as the unknown languages. This unlabeled set serves as the source of data from which we draw data for OOS modeling as detailed in Section 3.

The test set consists of 6,500 i-vectors, which was split into two subsets [5]: a *progress set*, and an *evaluation set*. The progress set contains the random selected 30% of those 6,500 test i-vectors, and used to monitor progress on the leader board. The remaining 70% of these i-vectors forms the evaluation set and used to establish the final scores at the end

doi: 10.21437/Odyssey.2016-45

of the challenge. Table 1 summarizes the data provided by NIST for the 2015 i-vector challenge [1, 5]:

Table 1: *Dataset available for the 2015 NIST i-vector challenge.*

	Number of i-vectors	Languages
Development set	6,431	Mixed of target and unknown languages
Training set	15,000	Fifty target languages
Test set	6,500	Both target and OOS languages

2.2. Language identification baseline

NIST provided a bare minimum baseline using i-vector with cosine-scoring for the i-vector challenge [5]. This also serves as the baseline to benchmark our progress during the i-vector challenge. We describe below the baseline system and how it is used for language identification:

- *Length normalization* [9]: all i-vectors (both training and test) are centred and whitened with respect to a global mean and covariance estimated from the unlabelled development set. This is followed by normalizing each vector to have unit norm.
- *Cosine scoring*: represent each language as the average of its training i-vectors. For a given test i-vector, scores are computed by taking the cosine distant between the language mean vectors and the test i-vector, as follows

  $$S_{i,k} = \frac{\phi_k^{\mathrm{T}}\phi_i}{\|\phi_k\|\cdot\|\phi_i\|} \text{ for } k = 1,2,\dots,K \quad (1)$$

  where ϕ_k represents the averaged i-vector of the k-th language class and ϕ_i is the i-th testing i-vectors.
- *Language identification*: select the language hypothesis that gives the highest score.

2.3. Performance metric

The primary performance metric for the LR i-vector challenge is the identification error rate averaged across the fifty target languages and the OOS set [5] defined as

$$Cost = \frac{(1-P_{oos})}{K} \times \sum_{k=1}^{K} P_{error}(k) + P_{oos} \times P_{error}(oos) \quad (2)$$

where $P_{error}(k)$ and $P_{error}(oos)$ are the error rates of test segments not assigned to the correct k-th language or OOS set, respectively. The cost function in (2) serves as the primary metric used in maintaining i-vector challenge leaderboard with the value $K = 50$ and $P_{oos} = 0.23$. Substituting these values into the cost function, whereby

$$Cost = \sum_{k=1}^{50} 0.0154 \times P_{error}(k) + 0.230 \times P_{error}(oos)$$

we found that the cost in making an error in identifying an OOS segment as inset is much larger than the other way round. As such, OOS modeling and detection become an important part in reducing the total cost.

3. Unspecified Language I-vector Clustering

Given the training data for target languages, there are many well-established researches of how these classes can be modeled. However, determining whether a language is out-of-set is less obvious. In this paper, it is hypothesized that multiple OOS languages can be represented by a single model. By doing so, the task of open-set language recognition simply becomes the task of close-set language identification. The critical question here is how to obtain the training samples to estimate the OOS model.

In this challenge, besides the training data, an additional development set of unlabeled data containing both the target and OOS languages is also provided. Therefore, if sufficient OOS samples can be identified from the development set, a reliable OOS model can be estimated. To this end, this section investigates several strategies for extracting OOS samples from the unlabeled development data. To begin with, section 3.1 discusses the process of data preparation to enable objective comparison of various strategies. Section 3.2 and section 3.3 present the two detection strategies. Finally, section 3.4 introduces a purification algorithm to further improve the detection results.

3.1. Data preparation

Firstly, the training data with 50 target languages is split into two groups:

- Target group: contains 40 target languages
- OOS group: contains the remaining 10 languages

In the target group, the data is then further divided into two subsets: *training set* includes the first 250 i-vectors of each language, and *validation set* includes the remaining 50 i-vectors. This validation set is subsequently merged with the OOS group to form the *open validation set*. To sum up, the *training set* has 10000 samples from 40 target languages. The *open validation set* has 5000 samples containing 2000 samples from the 40 target languages and 3000 samples from the OOS languages (the OOS group).

The *open validation set* is then used to simulate the condition of the unlabeled development data (6431 i-vectors) except that the OOS labels are known to allow the evaluation and fine-tuning of different OOS detection strategies. Now, the task is how to best discriminate the 3000 OOS samples from the 2000 samples of 40 target languages.

Before the OOS's i-vectors clustering, a LDA [9] transform was estimated using the target languages i-vectors from the 40 languages. The LDA transform was then applied on the raw target language i-vectors to reduce the i-vector dimension from 400 to 39. The 39-dimensional LDA transformed i-vectors were then whitened and normalized before they were used as the input for modelling and clustering purpose.

3.2. Least fit target

The *least fit target* (LFT) strategy is initiated by training a multi-class SVM classifier on the set of target languages from the labeled data. Given the trained classifier and its support vectors, the posterior probability of a sample belonging to each target class can be estimated according to Wu et al. [10].

Denoting P_{ik} as the posterior probability of sample i $(1 \le i \le N)$ of language k $(1 \le k \le K)$, where N and K are the number of samples and number of target languages respectively. The distance from sample i to the best-matched class is then defined as:

$$P_i = \max_{1 \le k \le K} P_{ik} \tag{3}$$

Intuitively, P_i will be closed to 1 if sample i is indeed the target class. We can also perform a OOS classification task by imposing a threshold θ: then the sample i is classified as the out-of-set language if $P_i < \theta$.

To illustrate the concept, Figure 1 and Figure 2 show an example of using LFT strategy on the data set with three target languages and one out-of-set language. Only the first two dimensions are shown for visualization purpose. From Figure 1, we can see that target samples are detected with very high probabilities with their respective classes. On the other hand, the effectiveness of LFT in detecting OOS samples is unclear since Figure 2 does not exhibit high confidence in estimating the OOS posterior probabilities.

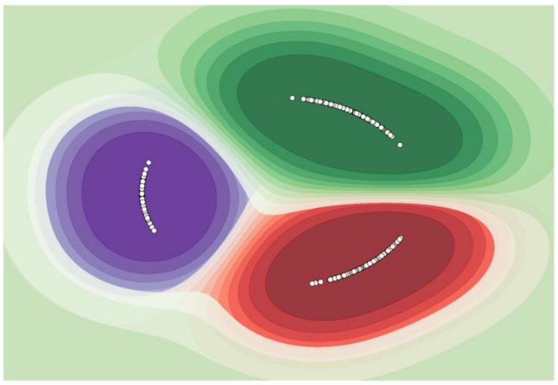

Figure 1: *Visualization of posterior probabilities of 3 target languages estimating with the LFT strategy. Higher color intensity indicates higher probabilities.*

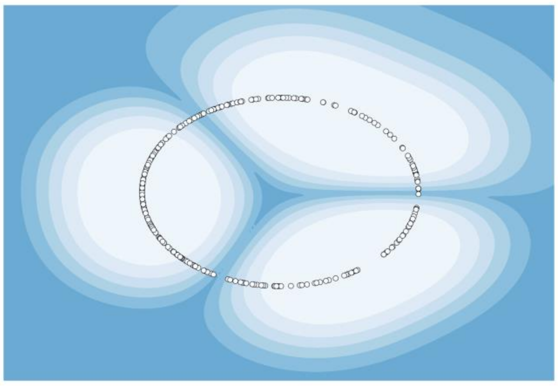

Figure 2: *Visualization of OOS posterior probabilities of OOS samples estimating with the LFT strategy. Higher color intensity indicates higher probabilities.*

3.3. Best fit out-of-set

The above mentioned strategy does not make use of any prior knowledge on the OOS languages. For this challenge, a set of unlabeled data with multiple OOS languages was also provided [5]. This additional development set may shed some insights into the prior distribution of OOS samples and may potentially benefit the OOS detection if used properly. Therefore, the *best fit out-of-set* (BFO) strategy, which incorporates OOS distribution from the unlabeled development set, is proposed.

To start with, all unlabeled samples are assumed to be OOS languages. A multi-class SVM classifier is then trained on the combined set of target samples and hypothesized OOS samples. The next step is to extract the genuine OOS samples from the hypothesized OOS set. Specifically, for each unlabeled sample i, the posterior probability P_i' of sample i belonging to the OOS class is estimated according to the pairwise coupling [10]. A threshold θ' is then applied to make the OOS decision: if $P_i' \ge \theta'$ then sample i is classified as the OOS language.

Similarly, Figure 3 and Figure 4 illustrate the BFO strategy with three target languages and one out-of-set language. In contrast to LFT strategy, BFO demonstrates potentially better performance in detecting OOS samples as more OOS samples are selected with higher probabilities as depicted in Figure 4. However, as a trade-off, the BFO's capability in detecting the target languages seems to suffer from miss detections as shown in Figure 3.

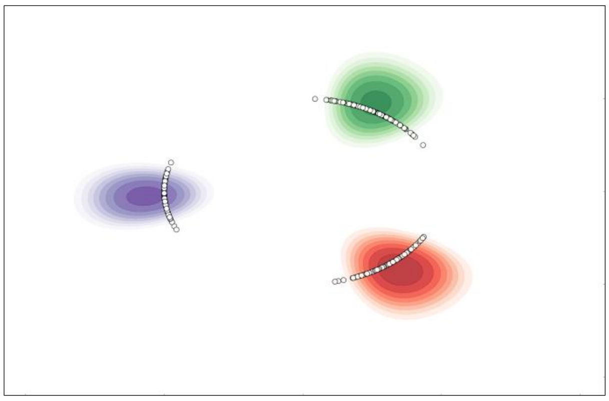

Figure 3: *Visualization of posterior probabilities of three target languages estimating with the BFO strategy. Higher color intensity indicates higher probabilities*

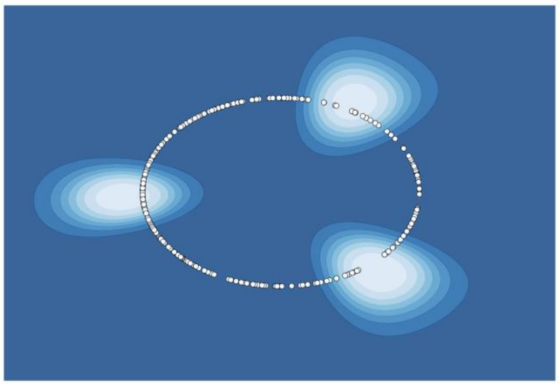

Figure 4: *Visualization of OOS posterior probabilities of OOS samples estimating with the BFO strategy. Higher color intensity indicates higher probabilities.*

Figure 5 shows the OOS's segments detection precision and recall curves using LFT and BFO strategies on the simulated dataset. We can see that the precision of the BFO method is obviously better than those of LFT method over almost all the recall positions.

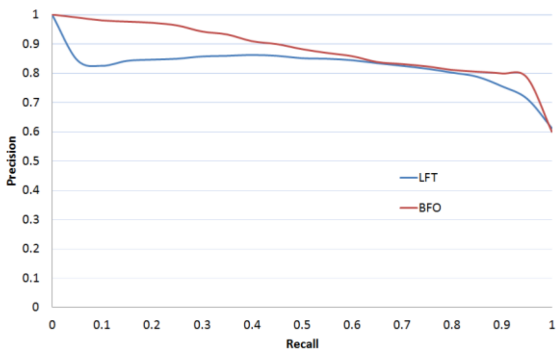

Figure 5: *OOS detection precision and recall curves using LFT and BFO strategies.*

3.4. Adaptive cluster purification

Based on the BFO's initial OOS detection, the 2nd module of our OOS identification system uses the i-vectors as the features for iterative re-clustering or cluster purification. We regard all the mixed i-vectors belonging to one of the two clusters/models: target and OOS. Target cluster has 2000 i-vectors and OOS cluster has 3000 i-vectors. This process of re-clustering has the effect of increasing the out-of-set homogeneity in order to purify the possible OOS i-vectors away from the target language i-vectors [11].

After BFO strategy training and testing, these 5000 i-vectors will have their testing scores generated from the OOS model. The i-vectors with the highest OOS model scores may be regarded as the 'out-of-set' languages segments and the i-vectors with the lowest OOS model scores are considered as the 40 targets languages.

After selecting the OOS and target i-vectors based on BFO, the dot product scores for all the i-vectors are computed against these two models i-vectors. We then re-label these two groups according to all the i-vectors scores and gradually increase the number of i-vectors from these two groups before class overlap is observed.

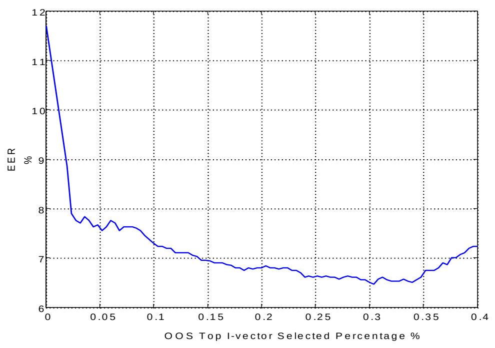

Figure 6: *The OOS and target EER rates via the adaptation percentage of the training sets.*

Figure 6 shows the targets and OOS detection EER rates via the increases of the adaptive size. The OOS i-vectors are regarded as the true scores and target languages i-vector are viewed as the imposter scores. From Figure 6, we can see that after the first BFO SVM multi-class training, the EER is about 11.6%. After the second stage purification, the clustering EER is significantly reduced from 11.6% to 6.5%. The best EER rate occurs if we select top 30% highest OOS scores as the OOS threshold

We have similarly applied this two stage OOS clustering methods to the 6431 unspecified development dataset. We have total 51 groups, including 50 target language i-vectors groups and one unspecified i-vector group.

Meanwhile we also found that the above experimental curve for choosing best percentage point cannot be directly applied to the unlabeled 6431 development dataset. It is because that our simulated mixed OOS dataset may not match the unlabeled 6431 i-vectors dataset. However, the experiments performed on the simulated dataset indicate that the OOS threshold is quite important for the cluster purification. In fact, our final OOS selection threshold is determined by submitting several submissions via the LRE challenge progress sets online [1, 5].

The algorithm for adaptive purifying the OOS is summarized as:

Algorithm 1: OOS Set Clustering Purification Algorithm.

```
Step 1. Get the unlabeled i-vector BFO initial testing scores.
Step 2. Choose $M$ top and M bottom i-vectors form OOS and target two language initial groups.
Step 3. Use these two group i-vectors to conduct the dot-product against all the mixed OOS i-vectors.
Step 4. Sort the test scores from high to low.
Step 5. Repeat Step 2 to 4 and increase size $M$ by $K$ for each iteration until no overlapped i-vectors are selected by both groups.
```

4. Description of Subsystems

This section describes I^{2}R submission which is among the top performing systems submitted to the 2015 NIST LR i-vector challenge. Serving as the baseline system is the multi-class logistic regression (MCLR) sub-system. In addition, one new component that we introduced in the i-vector challenge is the empirical kernel mapping for SVM which we describe next. We also explore the use of non-linear transformation implemented using a multi-layer perceptron (MLP) to pre-process the i-vectors. Last but not least is a DNN based classifier.

4.1. Multi-class logistic regression

The multiclass logistic regression (MCLR) sub-system is based on the multi-class cross-entropy discriminative training in the score vector space. To this end, i-vectors were transformed into log-likelihood score vectors through a set of Gaussian distributions, each representing the distribution of the language class in the i-vector space. We trained a global covariance matrix where language-specific covariance could be derived with a smoothing factor of 0.1. Given a test i-vector, a score vector is obtained by concatenating the log-likelihood scores from these Gaussian distribution.

Discriminative training is further applied on the score vector. The multiclass FoCal toolkit [12] was used for this purpose.

4.2. SVM with empirical kernel mapping

4.2.1 Polynomial expansion and compensation

We first expand the 400 dimensional i-vectors to a higher dimensional space by calculating the monomials up to order two. This results in an expansion of i-vector with 400 dimensions to a supervector of 80,601 dimensions. Let $\mathbf{b}_i$ be the supervector expansion of the i-vector ϕ_i. The supervectors are first centralized with respect to a global mean and normalized to have unit norm, as follows

$$\mathbf{b}_i \leftarrow \frac{\mathbf{b}_i - \boldsymbol{\mu}_o}{\|\mathbf{b}_i - \boldsymbol{\mu}_o\|}$$

The supervectors are then subjected to nuisance attribute projection (NAP) [13]. To this end, we constructed the so-called NAP projection matrix using the labeled training set (see Table 1). The NAP projection is given as

$$\tilde{\mathbf{b}}_i = (\mathbf{I}_i - \mathbf{E}\mathbf{E}^T) \times \mathbf{b}_i \quad (4)$$

The columns of the matrix $\mathbf{E}$ are the eigenvectors of the within-language covariance matrix estimated from the training set.

4.2.2 Empirical kernel mapping

The central idea of empirical kernel mapping is to transform the training data to the score space with respect to all other data points in the training set. Let

$$\mathbf{M} = [\tilde{\mathbf{b}}_1, \quad \tilde{\mathbf{b}}_2, \quad \tilde{\mathbf{b}}_3, \quad \dots \quad \tilde{\mathbf{b}}_N] \quad (5)$$

be a big matrix consisting of *N* supervectors drawn from all the *K* = 50 target languages and the OOS set. Given an input supervector $\tilde{\mathbf{b}}_i$, the empirical kernel mapping produces an *N*-dimensional score supervector in the following form

$$S_i = \tilde{\mathbf{b}}_i \times \mathbf{M}^T \quad (6)$$

The above mapping is applied to all supervectors. Finally, the supervectors are normalized as follows

$$S_i \leftarrow \frac{S_i - m_o}{\|S_i - m_o\|} \quad (7)$$

The dimensionality of the supervector is determined by the size of the training dataset. As shown in Table 1, the number of i-vector in the training set is 15,000, which is much lesser than the length of polynomial supervector.

4.2.3 SVM classifier

Consider that the OOS set as an additional class, the number of language classes is *K* + 1, where *K* is the number of target languages. Two strategies were adopted to train the SVM model for the target languages and OOS set, namely, one-versus-rest and pair-wise training.

Let Ω_k be the set of training data (i.e., the score supervectors after the empirical kernel mapping) for the *k*-th language, and $\bar{\Omega}_k$ be the set of training data from other languages. The one-versus-all SVM training solver for the dual formulation [7] is

$$f_k \leftarrow SVM(\Omega_k || \bar{\Omega}_k) \quad (8)$$

Another training scheme is pair-wise SVM training where a target class is trained against other languages, one at a time. In our case with 50 target languages and one OOS class, there are 1,275 unique pair-wise. In particular, the SVM solver for the dual formulation for pair-wise training is

$$f_{k,j} \leftarrow SVM(\Omega_k || \Omega_j), j \neq k \quad (9)$$

During test, a test vector S_i is scored against all pair-wise models. For the *k*-th language, the final score is taken as the minimum values among its pair-wise scores.

4.3. SVM with MLP feature

In order to benefit from system fusion, many diverse features and classifiers were explored throughout the development phase. Notably, the transformed feature using MLP together with a RBF-kernel SVM classifier rendered a nice addition to the existing sub-systems.

To obtain the transformed features, the original i-vectors after whitening and length normalization are fed into a 3-layer MLP. This MLP comprises of 400 visible units, 600 hidden units and 50 output units representing 50 target language classes. Instead of the frequently used *sigmoid* function, *linear* is chosen to be the input activation to the hidden layer. The output activation function of the hidden layer is also *linear*. To begin with, the parameters of this network are initialized layer-wise following the stacked auto-encoder approach introduced in Bengio et al. [15]. The network is then trained with the standard back-propagation supervised learning and drop-out [16] is applied to prevent overfitting and improve robustness. Once converged, the outputs of the hidden layer are taken as the transformed features. In brief, this trained network acts as a generalized version of LDA with two advantages: the feature dimension is not limited to the number of classes and the feature is more robust as a result of drop-out technique.

The MLP features, without any normalization, are then used to train a multi-class SVM with RBF kernel. This time, the SVM is trained to separate 51 classes (50 targets plus OOS). As one may expect of this system, the labeled training data should be divided into two parts: one for feature learning, and one for SVM training to improve the robustness. As a comparison, this system performs very well, it is just behind the best sub-system presented in section 4.2 and it requires much less time to run.

4.4. DNN i-vector

We also built a DNN classifier taking the i-vector as input [14]. Before training the DNN, a LDA was estimated and applied on the raw 400-dimensional i-vectors to reduce the dimension to 49. The 49-dimensional LDA transformed ivectors were then whitened and normalized before they were used as the input for the DNN training. Due to the data limitation, we trained a DNN with two hidden layers with 1500 hidden units. The target consists of 51 units including 50 target languages and one OOS language. The OOS language contains the OOS i-vector from the unspecified development set decided by the clustering method mentioned in Section 3. After the DNN training was successfully converged, a forwarding operation is used on all the development set. We have found that the experimental results on the DNN model approach is not as good as the supervector SVM approach, but it has some helps for final system fusion.

4.5. Sub-system fusion

I2R submission is based fusion of multiple sub-systems as described above. Since the training and development dataset are not well matching the test set, we choose to use simple linear fusion with equal weights for the classifier fusion. The linear fusion was applied to combine the scores from the sub-sub-systems, as follows

$$SCORE_i = \sum_{p=1}^{M} w_p \times s_{p,i} \quad (10)$$

where M is the number of sub-systems, $w_p = 1/M$ is the equal weight applied to all the sub-systems, and $s_{p,i}$ is score produced by sub-systyem p for the i-th test segment.

5. Experiment Results

In this section, we firstly compare system performances based on the i-vectors, polynomial supervectors and score level supervectors on the progress subset. Subsequently, we present the comparison results of our systems using different OOS detection strategies. Finally, the I2R fusion results for the i-vector LRE challenge are given.

5.1. Comparison of three different vectors

In this section, we compared the NIST i-vector baseline system with our SVM-based system using three different vectors. For a fair comparison, Table 2 shows the results of the four systems on the progress subset without using either OOS labels or models.

From Table 2, we can see that the NIST baseline of the cosine scoring cost is 0.3959. After training a SVM language model with the i-vectors, we can achieve about a 7.8% relative improvement over the baseline. In addition, when applying 2nd order polynomial expansion on the i-vectors, the SVM-based i-vector system performance can be further improved to a cost of 0.3405, a 14.0% improvement over the baseline. The best performance can be obtained by the proposed score-level SVM system, yielding a cost of 0.3321, a 16.1% improvement over the baseline score.

Table 2: *Comparison of DCF values without OOS detection*

System and Methods	DCF on progress set	improve vs Baseline
i-vector cosine baseline	0.3959	NA
i-vector SVM system	0.3651	7.8%
Polynomial SVM system	0.3405	14.0%
Score level SVM system	0.3321	16.1%

5.2. Different OOS detection strategies

In this section, three different OOS detections results are presented in Table 3 by using the polynomial supervector SVM system and score level supervector SVM system only. Three different OOS detection methods are:

a. Using Section 3.2 LFT method to detect the OOS i-vectors from unlabelled 6431 development dataset.
b. Using Section 3.3 BFO method to extract the OOS i-vectors from the unlabelled 6431 dataset.
c. Using the Section 3.3 and 3.4.hybrid method to extract the OOS i-vectors from the unlabelled 6431 dataset.

Table 3: *Comparison of DCF values with/without OOS detection*

	Polynomial SVM system		Score level SVM system	
OOS method	DCF on progress set	Improve over Baseline	DCF on progress set	Improve over Baseline
NA	0.3405	14.0%	0.3321	16.1%
LFT	0.2687	32.1%	0.2567	35.2%
BFO	0.2313	41.6%	0.2138	46.0%
Hybrid	0.2169	45.2%	0.2074	47.6%

In Table 3, the polynomial and score-level SVM systems have the DCF values of 0.3405 and 0.3321, respectively, without applying any OOS detection. After applying the simple LFT detection for training an OOS model, the two systems can achieve 32.1% and 35.2% improvement over the baseline. When the BFO OOS method is used, the performances of two systems are further improved by relatively 41.6% and 46.0% over the baseline, respectively. We also noticed that the best performance is obtained using the hybrid OOS's detection method for the two systems. They yield a relative improvement of 45.2% and 47.6% in terms of DCF values over the baseline, respectively.

From the overall results shown in Table 3, we also note that the results of the score-level SVM system are consistent better that those of polynomial SVM systems. Comparing Table 3 and Table 2, we can see that the OOS detection contributes the most to the overall performances.

5.3 Fusion results

In this part, we show the fusion results of the polynomial supervector and score-level SVM two systems. We also report the performance of our final submission on progress subset and evaluation subset.

A simple linear fusion of the two SVM-based systems from Table 3 achieves a consistent improvement over all the individual systems as shown in Table 4

Table 4: *Fusion Results of Table 3 with different OOS detection strategies*

OOS method	DCF on progress subset	Improve vs Baseline
NA	0.3221	18.6%
LFT	0.2354	40.1%
BFO	0.2138	46.0%
Hybrid	0.1872	52.7%

The I2R final submission fusion scores for both the progress subset and the evaluation subset are also shown in Table 5 along with baseline system score. Here, the DNN and MLP subsystem are also used for fusion.

Form Table 5, we have used 1743 i-vectors from the unspecified development data to train the OOS model to identify 1702 OOS's segments for the 6500 testing i-vectors. Our final fusion progress subset have achieves the DCF cost value of 0.1779 or a 55.1% relative improvement over the baseline. Similarly on the evaluation set, our final DCF cost value is 0.1774, a 54.5% relative improvement over the baseline evaluation score value 0.3903.

Table 5: *Comparison of I2R final submission and the baseline system*

	Final I2R fusion score/imp.	I-vector baseline
DCF on progress Set/Imp.	0.1779/55.1%	0.3959
DCF on evaluation Set/Imp.	0.1774/54.5%	0.3903
No of detected I-vectors for OOS modelling	1734	0
No of detected OOS i-vectors in the 6500 test i-vectors	1702	0

Figure 7 shows the 51 individual languages identification errors for the i-vector baseline system and our final submission system. Our final system has achieved less than 20% OOS detection error. Meanwhile, we also notice that not all languages performances of our system are better than the baseline system.

We also show the performance progress of I2R submission system over the whole evaluation period in Figure 8. Figure 8 also indicated the five major stages for significantly changing the system performances. We can summary them as:

- Using polynomial and score level SVM systems.
- Applying simple LFT method to identify OOS i-vectors.
- Employing the BFO method to extract OOS i-vectors.
- Applying the hybrid method to purify OOS cluster i-vectors.
- Fusing subsystems.

From the results, a major conclusion we can draw is that the OOS label detection is essential for the challenge.

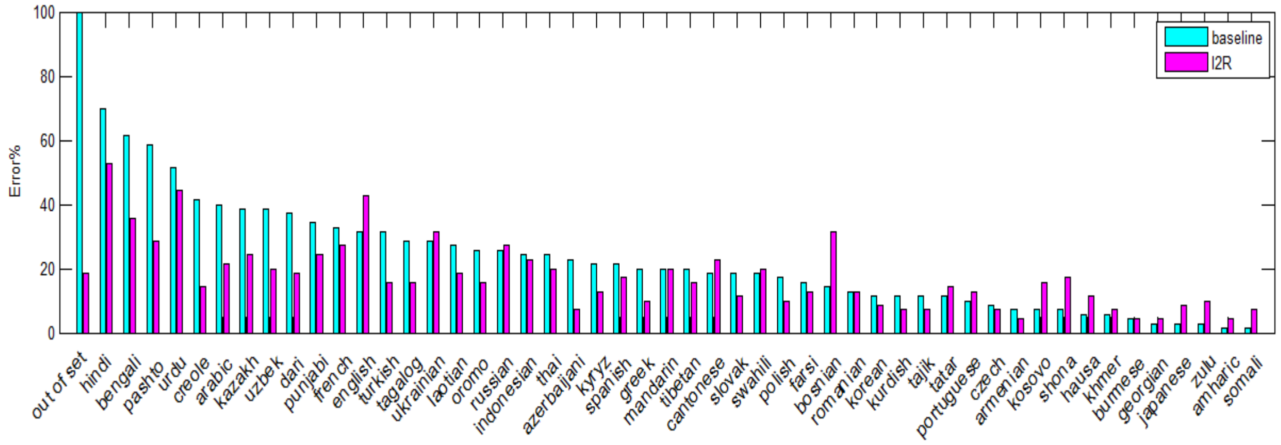

Figure 7: *51 individual languages identification errors for the baseline system and I2R final submission system on progress subset.*

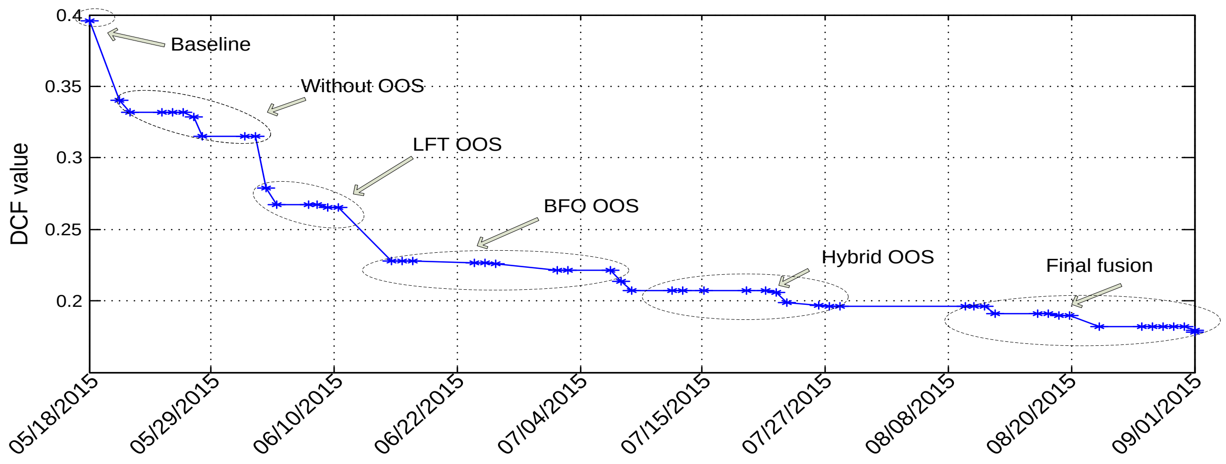

Figure 8: *The Progress on the I2R progress subset performance over the whole evaluation period.*

6. Conclusions

We presented a detailed system description and analysis of I^{2}R submission to the 2015 NIST LR i-vector challenge. We proposed a novel hybrid OOS detection strategy to select i-vectors pertaining to the OOS set from unlabeled development data. Our analysis shows that OOS detection is important to achieve good performance. The proposed hybrid OOS detection method exhibits well consistent on both *progress* and *evaluation* sets. We also proposed several sub-systems based on polynomial expansion and empirical kernel mapping for SVM. Experiment results indicate that there is an obvious benefit by applying polynomial expansion on i-vector. Further improvement can also be achieved by using score level supervectors for SVM. Our final submission achieved over 50% improvement compared to the baseline system on both progress and evaluation sets.

Nevertheless, we have not found any good strategy to detect and select i-vectors for target language i-vectors from the unlabeled development dataset, which might further improve system performance. This will be one good direction for future work.

7. References

[1] NIST, "NIST i-vector Challenge Homepage," [Online]. Available: http://www.nist.gov/itl/iad/mig/ivec.cfm.

[2] The 2013-2014 Speaker Recognition i-vector Machine Learning Challenge, http://www.nist.gov/itl/iad/mig/upload/sre-i-vectorchallenge_2013-11-18_r0.pdf.

[3] C. S. Greenberg, D. Bansé, G. R. Doddington, D. Garcia-Romero, J. J. Godfrey, T. Kinnunen, A. F. Martin, A. McCree, M. Przybocki and D. A. Reynolds,"The NIST 2014 Speaker Recognition i-vector Machine Learning Challenge", *Odyssey 2014*, pp. 224-230, Finland, June 2014.

[4] S. Novoselov, T. Pekhovsky, K. Simonchik," STC Speaker Recognition System for the NIST i-Vector Challenge", *Odyssey 2014*, pp. 231-240, Finland, June 2014.

[5] "The 2015 Language Recognition i-Vector Machine Learning Challenge', [online] Available, http://www.nist.gov/itl/iad/mig/upload/lre_i-vectorchallenge_rel_v2.pdf.

[6] H. Z. Li, B. Ma, and K. A. Lee, "Spoken language recognition: from fundamentals to practice," Proceedings of the IEEE, vol.101, pp. 1136–1159, 2013.

[7] R. Collobert and S. Bengio, "SVMTorch: support vector machines for large-scale regression problems," *Journal of Machine Learning Research*, vol. 1, pp. 143-160, 2001.

[8] N Dehak, P. A. Torres, Carrasquillo, D. Reynolds, and R. Dehak, "Language Recognition via i-vectors and Dimensionality Reduction", Proc. Interspeech 2011, Florence, Italy, Aug. 2011.

[9] D. Garcia-Romero and A. McCree, "Supervised domain adaptation for i-vector based speaker recognition," in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014, 2014, pp. 4047–4051.

[10] T. F. Wu, C. J. Lin and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling". JMLR 5:975-1005, 2004.

[11] H. Sun, B. Ma, Z. Swe. and H. Li., "Speaker Diarization System for FT07 and RT09 Meeting Room Audio," in *Proc. ICASSP*, pp.4982–4985, 2010.

[12] N. Brümmer, FoCal Multi-class: Toolkit for Evaluation, Fusion and Calibration of Multi-class Recognition Scores. Available: http://niko.brummer.googlepages.com/focalmulticlass.

[13] W. M. Campbell, D. E. Sturim, D. A. Reynolds, and A. Solomonoff, "SVM based speaker verification using a GMM supervector kernel and NAP variability compensation," in *Proc. ICASSP*, 2006, pp. 97-100.

[14] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," IEEE Signal Processing Letters, vol. 22, pp. 1671–1675, 2015.

[15] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, Greedy Layer-Wise Training of Deep Networks, in Advances in Neural Information Processing Systems 19 (NIPS'06), pages 153-160, MIT Press 2007.

[16] Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.