Advances in Speaker Recognition for Multilingual Conversational Telephone Speech: The JHU-MIT System for NIST SRE20 CTS Challenge

Jesús Villalba\(^1,2\), Bengt J. Borgstrom\(^3\), Saurabh Kataria\(^1,2\), Jaejin Cho\(^1\), Pedro A. Torres-Carrasquillo\(^3\), Najim Dehak\(^1,2\)

\(^1\)Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD, USA  
\(^2\)Human Language Technology Center of Excellence, Johns Hopkins University, Baltimore, MD, USA  
\(^3\)MIT Lincoln Laboratory, Lexington, MA, USA

\{jvillalba,ndehak\}@jhu.edu, \{jonas.borgstrom,ptorres\}@ll.mit.edu

Abstract

We present a condensed description of the joint effort of JHU-CLSP/HLTCOE and MIT-LL for NIST SRE20. NIST SRE20 CTS consisted of multilingual conversational telephone speech. The set of languages included in the evaluation was not provided, encouraging the participants to develop systems robust to any language. We evaluated x-vector architectures based on ResNet, squeeze-excitation ResNets, Transformers and EfficientNets. Though squeeze-excitation ResNets and EfficientNets provide superior performance in in-domain tasks like VoxCeleb, regular ResNet34 was more robust in the challenge scenario. On the contrary, squeeze-excitation networks over-fitted to the training data, mostly in English. We also proposed a novel PLDA mixture and k-NN PLDA back-ends to handle the multilingual trials. The former clusters the x-vector space expecting that each cluster will correspond to a language family. The latter trains a PLDA model adapted to each enrollment speaker using the nearest speakers—i.e., those with similar language/chain. The k-NN back-end improved Act. Cp by 68% in SRE16-19 and 22% in SRE20 Progress w.r.t. a single adapted PLDA back-end. Our best single system achieved Act. Cp=0.110 in SRE20 progress. Meanwhile, our best fusion obtained Act. Cp=0.110 in the progress—8% better than single—and Cp=0.087 in the eval set.

1. Introduction

The National Institute of Standards and Technology (NIST) regularly conducts speaker recognition evaluations (SRE) to assess the state-of-the-art of the technology [1]. These evaluations focus on the speaker detection task, i.e., given one or more enrollment recordings and a test recording, we need to decide whether the enrollment speaker is also present in the test. Over the years, SRE has evolved from telephone speech [2], to far-field microphone [3, 4], to non-English telephone speech [5, 6, 7]. While previous evaluations focused on one or two languages (Tagalog, Cantonese, Tunisian Arabic), the recent NIST SRE20 CTS Challenge\(^1\) proposes a challenging multilingual scenario, where the participants do not know which languages are involved, nor how many.

In this paper, we analyze the JHU-MIT submission to NIST SRE20 CTS. This is the joint effort of teams at Johns Hopkins CLSP/HLTCOE and MIT Lincoln Laboratory. We built on the knowledge acquired working on previous evaluations [8, 9, 10]. We moved from TDNN x-vector embeddings to residual network variants like ResNets\([11]\), SE-ResNets \([12]\) and EfficientNets \([13]\); and transformers \([14]\). The network embeddings were scored using different flavors of PLDA \([15]\) back-ends. First, we propose a PLDA mixture model where each mixture component is expected to model a different language group. Second, we propose k-Nearest Neighbour(kNN) PLDA, where we compute a different PLDA adapted to each enrollment side. The training data for each model are the k speakers nearest to the enrollment segments. These back-ends significantly improved the performance for all development and evaluation sets w.r.t. the single baseline back-end adapted to non-English audios.

2. Datasets

2.1. Training datasets

We used the following training datasets:

- **Switchboard phase1-3 and cellular1-2**[16].
- **NIST SRE04-12** telephone data [2, 4].
- **MIXER6** telephone phonecalls (MX6-tel) [3].
- **NIST SRE16 Dev**: It contains 668 recordings from 10 Mandarin speakers and 659 recordings from 10 Ceubano speakers [5].
- **NIST SRE16 Eval 60%**: This set contains 60% of the speakers in the NIST SRE16 Eval set. We kept the remaining 40% for development. This set includes 3299 recordings from 60 Cantonese speakers and 2904 recordings from 61 Tagalog speakers.
- **NIST SRE18 Dev/Eval**: This set contains 15192 recordings from 213 Tunisian Arabic speakers [6].
• **Fisher Spanish**: This set contains 1638 recordings from 136 Spanish speakers with several accents.

• **VoxCeleb 1+2**: This dataset contains 7365 speakers audio from video [17]. The original distribution of VoxCeleb splits each video into multiple short excerpts. We concatenated all excerpts from the same video into one file. This makes the dataset more appropriate for PLDA training and helps balance each video’s weight in the embedding training. After concatenation, we obtain 173088 recordings. We applied GSM and AMR-NB telephone codes to this data using SoX.

• **NIST LRE**: This set includes telephony samples from NIST LRE11-19, which contain more than 5 seconds of active speech [18]. The set was randomly downsampled to a size of 20k.

We trained our x-vectors on the combination of the datasets above (except LRE) with a total of 304k recordings from 13466 speakers. For x-vector training, we augmented speech on the fly with noise and reverberation. Impulse responses for augmentation were obtained from the Aachen impulse response database (AIR)\(^3\). The noises were acquired from the MUSAN corpus\(^3\). We used the same SNR levels as in the Kaldi recipes.

For PLDA back-end training, we used NIST SRE04-18 and Fisher Spanish. We did not use any data augmentation.

We also tried to add other multilingual datasets to x-vector and PLDA training. Still, they did not improve the results on the SRE20 progress set: IARPA Babel, NIST LRE17, Mozilla Common-Voice [19], CN-Celeb [20] and Multilingual LibriSpeech (MLS) [21].

### 2.2. Development datasets

We prepared three datasets for development:

- **NIST SRE16 Eval YUE/TGL40%**: This set contains 40% of the speakers (40 YUE and 40 TGL speakers) in the NIST SRE16 evaluation set [5]. We used the same trial list as in the original SRE16 but kept only the trials involving those 80 speakers. In total, there are 158k YUE and 174k TGL trials.

- **NIST SRE19 Eval**: This set contains 2.6M trials from Tunisian Arabic speakers [7].

These three sets were used for fusion. NIST SRE16 YUE40% was used for individual system calibration and final calibration of fusions. We observed that NIST SRE16 YUE produced the best calibration on the SRE20 progress set.

### 3. Embeddings

#### 3.1. Acoustic features and VAD

The acoustic features were 64 log-Mel-filter banks for all our systems. These features were short-time mean normalized with a 3 seconds window. Silence frames were removed using Kaldi energy VAD.

#### 3.2. Architectures

All the x-vector architectures follow the x-vector scheme [22, 23]. In essence, the embedding network consists of an encoder that extracts frame-level discriminant embeddings, a pooling mechanism and a classification head. We tried several encoder architectures and used either statistics pooling (mean+stddev) [22] or channel-wise attentive statistics pooling [24]. The network is trained to minimize the categorical cross-entropy loss of the predicted speaker posteriors. We used additive angular margin softmax loss [25] in all our networks. We describe the encoder architectures in the following paragraphs.

#### 3.2.1. ResNet34

This encoder is based on the original ResNet34 architecture proposed in [11]. ResNet34 has an input stem layer followed by 16 2D convolutional residual blocks. This architecture downsamples the feature maps \(3 \times 3\) with a stride of \(2\) (8x total downsampling), at the same time as it duplicates the number of channels in the convolutions. The output of this network is a four-dimensional tensor \((B, C, F/8, T/8)\), where \(B\) is batch-size, \(C\) is the number of channels, \(F\) is the number of Mel filters, and \(T\) is time. Channel and frequency dimensions are flattened to \((B, C \times F/8, T/8)\) before passing the features to the pooling layer [26].

#### 3.2.2. ResNet34-IN

We replaced the Batch-Normalization [27] in ResNet34 by Instance Normalization [28]. We also replace the classification head Batch-Normalization with Layer Normalization [29]. Hence, the normalization parameters do not depend on the training batch-size. This enables us to train with smaller batches and longer chunks.

#### 3.2.3. TSE-ResNet34

This encoder adds squeeze-excitation (SE) [12] blocks to ResNet34. The original SE, in 2D convolutions performs a pooling operation in both time and frequency dimensions (spatial dimensions in image). Then, it applies a channel-dependent scaling to the feature maps. However, the scaling is the same for all the frequency dimensions. We observed that standard SE did not provide significant gains for speaker recognition. In [30, 31], we proposed temporal squeeze-excitation (TSE). TSE applies pooling only in the temporal axis and applies a scaling, which is different for each channel and frequency dimension.

#### 3.2.4. Transformer

We also tried the Transformer Encoder architecture [14] as an encoder for x-vectors. We used an encoder with eight self-attention blocks. The input stem uses two 2D Conv layers that downsample time dimension \(4\times\). We also implemented a local attention procedure that limits the self-attention receptive field to 6 time steps (25 msecs) in each layer. This is similar to the Longformer architecture [32].

#### 3.2.5. EfficientNet-b4

EfficientNet architecture was proposed in [13] for images. This is a residual network that uses 2D separable convolutions to reduce the number of multiplications of the network. The work in [13] proposes a base architecture, denoted as EfficientNet-b0. Then larger networks EfficientNet-bn are obtained by scaling up the number of channels and network depth in such a way that EfficientNet-bn is \(2^n\) times more computationally expensive than b0. We found that b4 was needed to improve ResNet34.

---

\(^{3}\)http://www.openslr.org/resources/28

\(^{3}\)http://www.openslr.org/resources/17
3.3. Training procedure

All networks were first trained on 4 second chunks using an effective batch-size of 512. The actual batch size depended on the GPU memory and network size. Gradient accumulation was used to achieve the desired effective batch size. The learning rate was set to 0.01 and kept constant for 40k model updates. Afterward, it was divided by two every 10k steps until convergence. Later, the networks were fine-tuned using cyclic cosine learning rate scheduling on longer utterances (10-60 second chunks) for three periods, with an initial period of 2.5k updates, which multiplies by two after each re-start. The loss function was additive angular margin softmax \([25]\) with margin=0.3 and scale=30.

3.4. Embeddings in the submissions

The networks included in our fusions were ResNet34 with batch-norm (ResNet34), squeeze-excitation ResNet34 (TSE-ResNet34), ResNet34 with instance-norm (ResNet34-IN), ResNet34-IN with channel-wise attention pooling (ResNet34-IN-chwise-att), Transformer and EfficientNet-b4. All of them used statistics pooling except ResNet34-IN-chwise-att.

3.5. Embedding Comparison

Figure 1 compares our embeddings in terms of minimum Cprimary. All systems used the JHU kNN-PLDA-v3 back-end, described below. ResNet34 was more robust than more complex architectures. We found strong over-fitting for squeeze-excitation ResNets. Transformers also performed poorly. EfficientNets did not overtake ResNets. Instance normalization improved batch normalization in all development datasets. Res2Net [33], which was not included in any of our fusions, performed the best in SRE16 and 19. However, this improvement was not translated to the SRE20 progress set. ResNet34 with channel-wise attentive pooling [24] significantly improved all the dev. sets w.r.t. statistics pooling. It slightly improved the SRE20 progress, becoming our best single system.

4. Back-ends

4.1. JHU PLDA-v4

The pipeline for this back-end included CORAL, LDA, centering, whitening, length normalization, generative Gaussian SPLDA and adaptive S-Norm score normalization, as shown in Figure 2. For this back-end, we considered NIST SRE04-12 as out-of-domain (OUT) data (mainly in English); and SRE16-18 and Fisher Spanish as in-domain (IN) data.

The CORAL step computes a rotation that adapts the OUT data to the target domain. Thus, we applied that rotation to the OUT data, leaving the IN data untouched. Next, we pooled IN and adapted OUT data to train LDA and the Whitening step. Meanwhile, we computed different centering for OUT and IN data. The latter was the one used on the test data. Next, we applied length normalization.

PLDA was trained on IN+OUT length normalized embeddings. Following, PLDA was adapted to the IN domain data. For PLDA adaptation, the within-class and across-class covariances of the adapted model were a weighted sum of the out-of-domain \(S_{\text{out}}\) and in-domain \(S_{\text{in}}\) covariances,

\[
S_{\text{adapt}} = \alpha S_{\text{in}} + (1 - \alpha) S_{\text{out}} .
\]  

where we set \(\alpha = 0.75\).

After PLDA scoring, we applied Adaptive S-Norm using all IN+OUT data as a cohort. We used the top 500 cohort segments to compute the normalization parameters for each trial.
Figure 4: JHU kNN-PLDA back-ends. *Enr* denotes the enrollment segments, *Tst* denotes the test segment, and *All* denotes the full training data.

Figure 5: JHU back-ends comparison.

Figure 6: MIT-LL back-ends comparison.
4.2. JHU kNN-PLDA

The idea of the nearest neighbours back-ends consists of training a back-end model adapted to each trial. The motivation is that we do not know the number of domains in our eval data and also do not know if the eval. domains match any of the domains in our training and adaptation data. Thus, a PLDA mixture may not work since the eval data may not match any of the components of the mixture.

We simplify the problem by assuming that enrollment and test segments belong to the same domain, as indicated in the eval plan. The method consists of training a back-end (including PCA/LDA/centering/whitening/PLDA) model using the $k$ Nearest training speakers to the enrollment segments (1 or 3) of the trial. The enrollment segments are also included in the back-end training. Hence, even if the trial’s domain is not included in the training, the corresponding back-end can be trained using the closest speakers from multiple domains.

We have two kNN-PLDA versions. Figure 4a depicts the procedure for back-end kNN-PLDA-v1. For each enrollment segment, we use cosine scoring to find the $k_1$ closest training speakers. Then, we pool the enrollment segments and all the recordings from those $k_1$ speakers, and we train a Basic back-end (PCA/LDA/centering/whitening/LNorm), denoted as $BE_1$.

We thought that this method could benefit from domain adaptation. The number of in-domain neighbors may be too small to train PLDA. Instead, we can train the back-end on a larger number of speaker neighbors $k_1$ and adapt to a smallest (closest) number of speakers neighbors $k_2$. Thus, we proposed kNN-PLDA-v3 in Figure 4a. First, we trained Basic back-end $BE_1$, as in kNN-PLDA-v1. Then, we scored the enrollment model versus the training speakers again, but this time using $BE_1$ back-end, to find a refined set of in-domain speakers $k_2 < k_1$. Finally, we used those speakers to adapt $BE_1$’s centering and PLDA and produce $BE_2$.

In this manner, we trained a back-end for each enrollment side, and used that back-end for all the trials that involve that side. These back-ends did not require S-Norm to perform well.

Figure 3 shows the intuition behind the kNN back-end. Embeddings of different domains live in a non-linear manifold, which can not be well approximated by a single PLDA model. However, in the vicinity of a target speaker, the manifold can be well approximated by a linear PLDA model.

Figure 5 compares the three JHU back-ends on the ResNet34-instance-norm embedding. Figure 5a compares the back-ends in terms of minimum Cprimary. kNN-PLDA-v1 improved SRE16 YUE and SRE19 but not SRE16 TGL and SRE20. kNN-PLDA-v3 significantly improved all sets. Figure 5b compares the calibration. Though kNN-PLDA-v1 did not improve min. Cprimary in some sets, it did improve calibration.

4.3. MITLL-1mix

The MITLL-1mix system used a variety of data sets in the back-end. The scoring pipeline was comprised of CORAL feature mapping of the out-of-domain set. LDA dimension reduction to 200 was then applied, followed by global centering and whitening. LDA was trained on SRE16-18. An out-domain PLDA model was then trained on SRE04-10, which was adapted to a partially unlabelled in-domain set. Finally, adaptive S-Norm was applied with a cohort size of 1000. In order to leverage the large unlabelled LRE set during PLDA scoring, semi-supervised adaptation [34] was used to adapt the out-of-domain model to this set, along with the labeled SRE16 Eval and SRE18 Eval sets. SRE16-18 and LRE were used to train CORAL, Cent/Whitening, adapt PLDA and S-Norm.

4.4. MITLL-8mix

The MITLL-8mix system extended the back-end scoring system from Sec. 4.3 to include mixture modeling in the adapted PLDA model. The technique proposed in [34] was generalized to allow for a mixture of PLDA models to be trained with a partially unlabelled data set. In all other respects, the MITLL-8mix was identical to the MITLL-1mix system.

Figure 6 compares MIT-LL back-ends on the ResNet34 embedding. The baseline is MITLL-1mix without LRE adaptation data (blue). Adding LRE data significantly improved SRE16 and SRE20 (orange). The PLDA mixture improved further. The PLDA mixture improved min. Cp by 16% for SRE16-19 and by 28% for SRE20 prog. w.r.t. the baseline.

5. Calibration and Fusion

5.1. JHU single system calibration

The JHU systems conditioned score calibration on the number of enrollment cuts (i.e., 1c vs. 3c). We trained a separate logistic regression mapping for each of these two conditions on the NIST SRE16 YUE40% development set. We used a target prior $P_T = 0.05$.

5.2. MIT single system calibration

The MIT systems conditioned score calibration on gender and the number of enrollment cuts (i.e., 1c vs. 3c). A separate logistic regression mapping was trained for each of the four combinations of these attributes, using the NIST SRE16
Table 2: Fusion summary.

<table>
<thead>
<tr>
<th>Submission</th>
<th>Date</th>
<th>Num. Sys.</th>
<th>Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.4-4</td>
<td>2020/10/29</td>
<td>4</td>
<td>ResNet34 × MITLL-1mix + (ResNet34-IN + ResNet34+Transformer) × JHU-v4-SNorm</td>
</tr>
<tr>
<td>v1.8.1-4</td>
<td>2020/11/16</td>
<td>4</td>
<td>ResNet34 × MITLL-1mix + (ResNet34-IN + ResNet34+TSE-ResNet34) × JHU-kNN-v3</td>
</tr>
<tr>
<td>v1.17-4</td>
<td>2020/12/18</td>
<td>4</td>
<td>ResNet34 × MITLL-8mix + (ResNet34-IN + Transformer + EfficientNet-b4) × JHU-kNN-v3</td>
</tr>
<tr>
<td>v1.25-5</td>
<td>2021/04/13</td>
<td>5</td>
<td>ResNet34 × MITLL-8mix + (ResNet34-IN + Transformer + EfficientNet-b4 + ResNet34-IN-chwise-att) × JHU-kNN-v3</td>
</tr>
</tbody>
</table>

Table 3: Results of Submitted fusions on SRE16-19 dev sets and SRE20 progress/eval set

<table>
<thead>
<tr>
<th>System</th>
<th>SRE19 Eval</th>
<th>SRE16 YUE40%</th>
<th>SRE16 TGL40%</th>
<th>SRE16-9 AVG</th>
<th>SRE20 Prog.</th>
<th>SRE20 Eval.</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.4-4</td>
<td>2.64</td>
<td>0.146</td>
<td>0.188</td>
<td>4.4</td>
<td>0.291</td>
<td>0.32</td>
</tr>
<tr>
<td>v1.8.1-4</td>
<td>2.39</td>
<td>0.125</td>
<td>0.137</td>
<td>1.33</td>
<td>0.084</td>
<td>0.084</td>
</tr>
<tr>
<td>v1.17-4</td>
<td>2.33</td>
<td>0.121</td>
<td>0.146</td>
<td>1.27</td>
<td>0.077</td>
<td>0.078</td>
</tr>
<tr>
<td>v1.25-5</td>
<td>2.3</td>
<td>0.118</td>
<td>0.139</td>
<td>1.23</td>
<td>0.075</td>
<td>0.075</td>
</tr>
</tbody>
</table>

Figure 7: Cprimary of submitted fusions on SRE16-19 dev sets and SRE20 progress/eval set.
YUE40% data set. The target prior $P_T = 0.05$ was used for each. To obtain gender labels, a gender classifier based on linear discriminant analysis was trained on the SRE04-10 set.

5.3. Fusion

To select the best fusion combination, we used the same greedy fusion scheme as in the previous evaluation [8]. The fusion was trained on the SRE16v19 sets. Following, we recalibrated the fused scores on NIST SRE16 YUE.

6. Submissions

Table 1 summarizes the results for the single systems that were part of our fusions. As we said before, ResNet34 with channel-wise attention pooling (ResNet34-IN-chwise-att) was our best system in SRE20 prog. Table 2 summarizes the fusions that appear on the SRE20 Eval leader-board. The first three are four-system fusions. The first one used baseline PLDA back-ends. Afterward, we switched to kNN PLDA and PLDA mixtures. The last one added the ResNet34-IN-chwise-att system. Table 3 and Figure 7a show the results on the dev, progress and eval set. We observed a significant improvement from our first to last submission on our dev and SRE20 progress sets. However, the improvement on the SRE20 Eval was less significant. Figure 7b compares the calibration of the submissions. We see that calibration improved when we moved from the baseline back-end to kNN back-ends.

7. Conclusions

We analyzed the JHU-MIT systems for the SRE20 CTS Challenge. Regarding datasets, we concluded that VoxCeleb and labeled CTS data helped to obtain good results on SRE20. However, unlabeled CTS (except LRE in MIT back-end) and labeled non-CTS multilingual data did not help to improve SRE20. Regarding neural embeddings, ResNet34 performed better than more complex architectures, and fine-tuning with long utterances improved the results. Also, channel-wise attention pooling improved results on SRE20 and dev sets w.r.t. statistics pooling. Regarding back-ends, we proposed a novel kNN PLDA and PLDA mixture model, which significantly improved performance. The recipes for JHU systems are publicly available in the Hyperion toolkit.

8. References


