Submission for cross-channel task of oriental language recognition 2020 challenge

Peng Shen [1], *Xugang Lu*[1], *Takuma Yoshimoto*[12], *Ryoichi Takashima*[2], *Hisashi Kawai*[1]

[1]NICT, Japan
[2]Kobe University, Japan
peng.shen@nict.go.jp

Abstract

In this report, we describe our language identification system for the cross-channel task of oriental language recognition 2020 challenge. We firstly present the data and feature preparation. Then, front-ends and back-ends are introduced. For front-ends, we evaluated extended TDNN and ResNet network configurations with softmax and additive margin softmax objective loss functions. For back-ends, the multi-class logistic regression is used. In the front-end and back-end processing, we integrated our proposed unsupervised feature learning and adaptation algorithms which significantly improved the performance. Finally, a greedy fusion method was used to obtain the final score for submission.

Index Terms: spoken language identification, cross channel, OLR challenge

1. Datasets and feature extraction

The oriental language recognition (OLR) 2020 challenge includes three tasks: (1) cross-channel LID, (2) dialect identification, and (3) noisy LID. In this work, we focus on the cross-channel LID tasks, which was a very challenging task because the target channel data are not available during the system preparation.

1.1. Data preparation

We used the AP16-OL7, AP17-OL3, AP17-OLR-test, AP18-OLR-test, and THCHS30 as the training data to build our systems. AP19-OLR-dev and AP19-OLR-test were used as the development data [1]. We followed the training data preparation of the baseline x-vector system supplied by the OLR 2020 challenge organizer.

For the training data, we first combined the AP16-OL7, AP17-OL3, AP17-OLR-test, and AP18-OLR-test data. Then, we used several data augmentation methods to increase the amount and diversity of the training data. One is the method similar to the official baseline system, which was speed and volume perturbation. For speed perturbation, we applied a speed factor of 0.9 or 1.1 to slow down or speed up the original recording. And for volume perturbation, a random volume factor was applied. We also used the noisy and speech data from THCHS30 as noise and background speech to increase the amount of the training data. Recently, the mixup-based method showed its effectiveness in speech processing tasks [2, 3]. In this work, we implemented an audio-based mixup by randomly choose a pair of audio samples from different languages to generate new audio. The shorter one was repeated to match the length of the longer one. The new length-aligned audio was used for model training.

1.2. Feature extraction

Three types of acoustic features were applied, i.e., the Mel-frequency cepstral coefficient (MFCC), perceptual linear predictive cepstrum (PLP), a log Mel-filter bank (FBANK). MFCC features were computed using 30 Mel-filter banks. The PLP analysis computed 20-order PLP-cepstra. FBANK features were estimated using 40 and 60 Mel-filter banks. The feature extraction was progressed with a frame window of 25 ms and a shift of 10 ms. The frames of silence were removed with energy-based voice activity detection (VAD) after doing feature extraction.

2. Front-ends

One i-vector and several x-vector-based front-ends were built in this work.

2.1. i-vector system

The i-vector baseline system was constructed by using the MFCC feature that was augmented by their first- and second-order derivatives. The UBM involved 2,048 Gaussian components and the dimensionality of the i-vectors was 600.

2.2. x-vector system

Recently, x-vector showed its effectiveness for both speaker and language recognition tasks [4, 5, 6]. The model for extracting language embedding representations, i.e., x-vector, consists of three modules: a frame-level feature extractor, a statistics pooling layer, and utterance-level representation layers. In this work, by fixing the statistics pooling layer and utterance-level representation layers, we investigated the frame-level feature extractor with two neural networks for extracting the language embedding and different settings of the objective loss function.

2.2.1. Network

TDNN is the most commonly used for x-vector extraction [4]. The TDNN network includes three time-delay layers and two fully connected layers. There are 512 channels except for the last one, which has 1500 channels. The kernel sizes are 5, 3, and 3; and dilation factors are 1, 2, and 3 for time-delay layers, respectively.

An extended TDNN architecture (E-TDNN) has been shown its effectiveness for extracting x-vectors [7]. Compared with TDNN, E-TDNN consists of one more time-delay layer and three fully-connected layers. The new fully connected layers are inserted into every two time-delay layers. The kernel sizes are 5, 3, 3, and 3; and dilation factors are 1, 2, 3, and 4, respectively. Therefore, the temporal context of E-TDNN is wider than that of TDNN. And E-TDNN has more parameters.

Excepted the E-TDNN network, we also implemented the ResNet configuration, we replaced the TDNN network with a ResNet34 network [8]. A channel average pooling was applied to the output of the final layer of the ResNet. The dimension of the average pooling was 512. Then, the statistic pooling and utterance-level representation were processed.

2.2.2. Objective loss function

In conventional speaker embedding training, softmax-based categorical cross-entropy is commonly used as the objective loss function. In this work, besides softmax, we also evaluated additive margin softmax (AMSoftmax)-based function [9].

2.2.3. Model training

We prepared server types of x-vector models by using E-TDNN, ResNet configuration and SpecAugmentation, skip connection, and softmax and AMSoftmax. A feature-based mixup was also implemented. For detailed information, please refer to asv-subtools [1].

3. Back-ends

With the extracted embedding vectors, we firstly applied in-domain global mean subtraction on training and test data. Then, linear discriminant analysis (LDA) was used to select the most language relevant feature and reduce the dimension of the original x-vector. Finally, length normalization was applied to the language discriminant vectors.

We evaluated several classifiers, such as support vector machine, Gaussian mixture model, and multi-class logistic regression. And our final submission is based on the multi-class logistic regression. The LDA dimension was selected as 200.

4. Investigations on channel adaptation

To reduce the channel mismatch problems, we investigated unsupervised methods by using the test data. Firstly, we prepared the predicted label for the test data with a greedy fusion algorithm. Then, the adaptation data were selected based on the fusion result. With the selected adaptation data, the x-vector models were fine-tuned with a learning rate of 0.0001. Finally, the classification was done by using the output (posterior probability) of the network. Moreover, we integrated our proposed unsupervised feature learning and adaptation algorithms which explicitly adapt the model for the testing conditions. As our experiments showed that the adaptation process improved the performance significantly.

5. Greedy fusion

Similar to our work on speaker verification task [10], we implemented a greedy fusion algorithm to obtain the final submission. The basic algorithm is as follows: Firstly, all the subsystems are evaluated to obtain Cavg and EER values. Then, the top N best subsystems are selected as the candidate list. After that, we prepare new lists by adding a new subsystem to the candidate list. The linear logistic regression with the Bosaris toolkit [11] is used to fuse and evaluate the new lists. Then, the candidate list is updated by selecting the top N best lists. The final submission is obtained when there is no further improvement. In this work, N was set to 3.

[1] https://github.com/Snowdar/asv-subtools

In the first round of fusion, we used the AP19-OLR-test as the development data to train the logistic regression model. Then, we selected the data from the test data based on the output score of the previous fusion step. To overcome overfitting, we split the subsystems into two groups. The selected data from group one were used for the fusion of group two, and data from group two were used for the fusion of group one. Rather than using Cavg and EER, we used softmax loss to pick up the final results of the greedy fusion.

6. References

[1] Z. Li, M. Zhao, Q. Hong, L. Li, Z. Tang, D. Wang, L. Song, and C. Yang: AP20-OLR Challenge: Three Tasks and Their Baselines, submitted to *APSIPA ASC* 2020.

[2] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond Empirical Risk Minimization," in arXiv:1710.09412, 2017.

[3] Yingke Zhu, Tom Ko, Brian Mak, Mixup Learning Strategies for Text-independent Speaker Verification, in Proc *Interspeech* 2019.

[4] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in Proc. *Interspeech*, 2017.

[5] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in Proc. *ICASSP*, 2018.

[6] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker Recognition for Multi-Speaker Conversations Using X-Vectors," in Proc. *ICASSP*, 2019.

[7] D. Snyder, et al., "The JHU Speaker Recognition System for the VOiCES 2019 Challenge," in Proc. *Interspeech*, 2019.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. *CVPR*, 2016.

[9] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," in Proc. *Interspeech*, 2019.

[10] P. Shen, X. Lu, and H. Kawai, "Investigation of NICT submission for short-duration speaker verification challenge 2020," in Proc. *Interspeech*, 2020.

[11] N. Brummer and E. De Villiers, "The BOSARIS Toolkit: Theory, Algorithms and Code for Surviving the New DCF," in Proc. *NIST SRE11 Speaker Recognition Workshop*, Atlanta, Georgia, USA, 2011.