The XMUSPEECH System for the AP19-OLR Challenge

Zheng Li[1], *Miao Zhao*[2], *Jing Li*[2], *Yiming Zhi*[2], *Lin Li*[1], *Qingyang Hong*[2]

[1]School of Electronic Science and Engineering, Xiamen University, China
[2]School of Informatics, Xiamen University, China

lilin@xmu.edu.cn, qyhong@xmu.edu.cn

Abstract

In this paper, we present our XMUSPEECH system for the oriental language recognition (OLR) challenge, AP19-OLR. The challenge this year contained three tasks: (1) short-utterance LID, (2) cross-channel LID, and (3) zero-resource LID. We leveraged the system pipeline from three aspects, including front-end training, back-end processing, and fusion strategy. We implemented many encoder networks for Tasks 1 and 3, such as extended x-vector, multi-task learning x-vector with phonetic information, and our previously presented multi-feature integration structure. Furthermore, our previously proposed length expansion method was used in the test set for Task 1. I-vector systems based on different acoustic features were built for the cross-channel task. For all of three tasks, the same back-end procedure was used for the sake of stability but with different settings for three tasks. Finally, the greedy fusion strategy helped to choose the subsystems to compose the final fusion systems (submitted systems). $Cavg$ values of 0.0263, 0.2813, and 0.1697 from the development set for Task 1, 2, and 3 were obtained from our submitted systems, and we achieved rank $3rd$, $3rd$, and $1st$ in the three tasks in this challenge, respectively.

Index Terms: AP19-OLR, language identification, x-vector, multi-task learning

1. Introduction

Language Identification (LID) refers to identifying language categories from utterances, and it is usually presented at the front-end of speech processing systems, such as in Automatic Speech Recognition (ASR). To encourage the improvement of LID technologies and to tackle the real challenge existing in LID tasks, the oriental language recognition challenge has been organized annually since 2016, attracting dozens of teams around the world [1, 2]. The XMUSPEECH team has attended the OLR challenge since the AP17-OLR.

The AP19-OLR challenge [3] included three tasks. Task 1 involved short-utterance LID from test utterances that were only one second long, which was the same task format as in the past two challenges. Task 2 was comprised of cross-channel LID, which revealed the real-life, practical demands of speech technology. In Task 3, zero-resource LID, no resources were provided for training before inference; only several utterances of each language were offered for language reference. All tasks were evaluated and ranked separately. The principle evaluation metric was $Cavg$, which was defined as the average of the pairwise performance of test languages, given $P_{target} = 0.5$ as the prior probability of the target language.

We submitted the final results of the three tasks with required test conditions in this challenge. Our developed systems consisted of multiple front-end extractors, including i-vector, x-vector, extended x-vector, multi-feature x-vector, and multi-task x-vector. After language embeddings were extracted, the same back-end processing was used for all three tasks, but with different settings given the results of the development sets. The fusion strategy consisted of equal weight fusion for the subsystems selected from the greedy fusion algorithm.

In this paper, we introduce the details of the XMUSPEECH system for AP19-OLR, and the rest of this paper is organized as follows. Section 2 describes the data preparation process, and Section 3 introduces the methods used to build the systems. The experimental settings and results of the subsystems for development sets and evaluation sets are shown in Section 4. Finally, the conclusion is given in Section 5.

2. Data Preparation

In this AP19-OLR challenge, additional training materials were forbidden to participants, and the permitted resources were several specified data sets, including AP16-OL7, AP17-OL3, AP17-OLR-test, AP18-OLR-test, and THCHS 30 [4]. Those data sets included ten languages, which were Mandarin, Cantonese, Indonesian, Japanese, Russian, Korean, Vietnamese, Kazakh, Tibetan, and Uyghur. The detailed descriptions of the data sets that we used are listed in Table 1.

2.1. Language Identification Training Set

Before training, we adopted two types of data augmentation, including speed and volume perturbation, to increase the amount and diversity of the training data. For speed perturbation, we applied a speed factor of 0.9 to slow down or 1.1 to speed up the original recording, and for volume perturbation, a random volume factor was applied. Finally, two augmented copies of the original recording were added to the original data set to obtain a 3-fold combined training set. We used the suffix '$-aug$' to indicate the combined training data set.

For Task 1, AP16-OL7, AP17-OL3, AP17-OLR-test, and THCHS 30 constituted the training set for all Kaldi-based [5] systems, namely $AP19\text{-}task\text{-}1\text{-}train\text{-}with\text{-}thchs30\text{-}aug$, which included ten languages with about 310,000 utterances. While the AP18-OLR-test was reserved as the development set for this task. For the training set in Pytorch platform [6], the THCHS 30 data set was not used due to the time constraints. Therefore, the training set in Pytorch based systems for Task 1 was named $AP19\text{-}task\text{-}1\text{-}train\text{-}aug$, which included ten languages with about 280,000 utterances.

For Task 2, to increase the robustness of systems for this cross-channel task, we used all data sets allowed in this challenge to train systems with the data augmentation methods mentioned above. The training data for Task 2 included AP16-OL7, AP17-OL3, AP17-OLR-test, AP18-OLR-test, and THCHS 30. The training set used in Task 2 was named $AP19\text{-}task\text{-}2\text{-}train\text{-}aug$, which included ten languages with about 360,000 utterances.

Table 1: *Data Sets Used in Systems.*

Task	Encoder Model Training	LDA Training & Centering	LR Model Training
1	$AP19$-$task$-1-$train$-$with$-$thchs30$-aug $AP19$-$task$-1-$train$-aug $Phonetic$-$training$-set	$AP19$-$task$-1-$enroll$	$AP19$-$task$-1-$enroll$
2	$AP19$-$task$-2-$train$-aug	$AP19$-$task$-2-$enroll$-aug	$AP19$-$task$-2-$enroll$-aug
3	$AP19$-$task$-1-$train$-$with$-$thchs30$-aug $AP19$-$task$-1-$enroll$ $Phonetic$-$training$-set	$AP19$-$task$-1-$enroll$	$AP19$-OLR-$test$-$task$-3-$enroll$

Task 3, which had an open set test condition, shared the exact same training set as Task 1, i.e. $AP19$-$task$-1-$train$-aug.

2.2. Phonetic Training Set

The AP16-OL7 and AP17-OL3 databases contain lexicons of all ten languages, as well as the transcriptions of all the training utterances. These resources were chosen to train ASR models and to create phonetic alignment labels for multi-task learning with phonetic information. We named this set the $Phonetic$-$training$-set.

2.3. Enrollment Set

To better match the test condition for Task 1 (short utterance), the enrollment set for Task 1 contained AP16-OL7, AP17-OL3, and AP17-OLR-test-task1, which was the short utterance test set, without data augmentation. $AP19$-$task$-1-$enroll$, included ten languages with about 280,000 utterances.

In order to generalize for the cross channel test condition, we used as much data as possible to compose the enrollment set for Task 2. The enrollment set is a subset of the training set for Task 2, in which only the target languages were remained. The enrollment set is also called $AP19$-$task$-2-$enroll$-aug.

We used the official task 3 enrollment set for Task 3, due to the open set testing condition.

2.4. Back-end Training Set

As the system procedure consisted of embedding extraction and a back-end classifier, the selection of the back-end training set was influential. We utilized the same data set as the Task 1 enrollment set to train the linear discriminant analysis (LDA) model and to conduct the centering for Task 1 and 3. The enrollment set in Task 2 was used for the back-end of Task 2. The logistic regression (LR) models were trained with the enrollment sets for the respective three tasks.

2.5. Development Set

The organizers released development sets for Tasks 2 and 3, but there were no new development sets provided for Task 1. So, the AP18-OLR-test was used as the development set for Task 1. We noted that the three target languages in the development set for Task 3 were different from the final test set, and we assumed that the channel condition in the development set for Task 2 may also differ from the final test set.

3. System Descriptions

3.1. Feature Extraction

All features were extracted from 16kHz audio data. Three kinds of acoustic features were used, including 20-dimensional MFCC, 40-dimensional FBank, and 20-dimensional PLP, with 3-dimensional Kaldi's pitch features in all systems. All features had frame-lengths of 25ms, frame-shifts of 10ms, and mean normalization over a sliding window of up to 3 seconds. Voice active detection (VAD) was used to filter out non-speech frames. Note that for multi-feature training models, different kinds of acoustic features shared the same VAD based on PLP feature for the alignment of frames. The feature engineering was executed using the Kaldi platform.

3.2. Encoder Networks

3.2.1. I-vector

I-vector systems were developed based on the Kaldi SRE16 recipe [5, 7], in which input features are acoustic features with first and second order derivatives. A full covariance Gaussian mixture model-universal background model (GMM-UBM) with 2,048 components was trained, along with a 600 dimensional i-vector extractor. We also tried different i-vector configurations, but no significant improvement was achieved in this challenge.

3.2.2. Extended X-vector

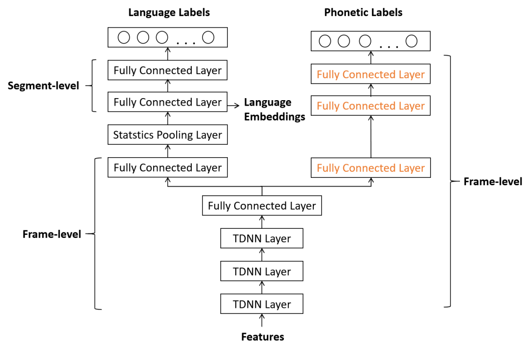
Since the extended x-vector architecture (E-TDNN) significantly outperformed the baseline x-vector (TDNN) in most cases [8, 9], we chose the extended x-vector for this challenge. Compared to the traditional x-vector, the extended x-vector structure uses a slightly wider temporal context in the TDNN layers, and it interleaves dense layers between TDNN layers, which leads to a deeper x-vector model.

3.2.3. Multi-task Learning Model with Phonetic Information

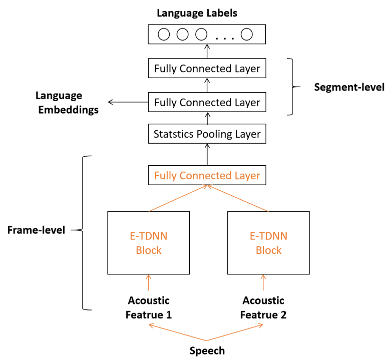
Considering the correlativity between the language identification and the phone classification tasks, we utilized multi-task learning to train those two tasks jointly [10, 11]. To achieve this goal, ASR models were trained beforehand on the $Phonetic$-$training$-set. Then, based on the ASR models, 6,832 frame-level alignment labels were obtained for the phone classification task. In this multi-task learning model, the frame-level hidden layers represent the shared part that learns the phonetic compensation information for the language task. The gradient descent of each task affects the shared layers in the frame-level during training. After training, the language embeddings were extracted from the penultimate layer in the language task branch. This model is shown in Figure 1 (a).

3.2.4. Multi-feature Learning Model

Due to fact that the data distributions of different acoustic features are comparatively dissimilar, the multi-feature integration structure is used to utilize different kinds of acoustic features into x-vector systems [12]. In this challenge, we improved the structure in [12] by replacing the TDNN blocks with the E-TDNN blocks for deeper learning abilities. In this model, as shown in Figure 1 (b), while each branch processes one type of acoustic features at the frame level, the outputs of the two branches at frame level are spliced together as a super vector



(a) Multi-task learning model with phonetic information.



(b) Multi-feature learning model.

Figure 1: *Joint learning models.*

before the statistics pooling layer.

3.3. Loss Functions

3.3.1. Loss Function for Multi-task learning

In the multi-task learning model with phonetic information mentioned in Section 3.2.3, the model includes two separate objectives: language identification and phone classification. The total loss function L is the combination of those two sub-losses L_l and L_p,with a control factor α:

$$L = L_l + \alpha L_p \tag{1}$$

The hyper-parameter of the α value for mutli-task training is empirically set to 0.2.

3.3.2. AM-Softmax

AM-softmax [13] is one of the most popular loss function in classification tasks . The core concepts of AM-softmax are feature normalization and the adoption of an additive margin into the softmax loss function. The combination of this stronger loss objective and the deeper encoder network, such as the E-TDNN x-vector, further improve the performance of subsystems. While only the softmax loss is supported in Kaldi, we implemented the AM-softmax on Pytorch and built all of the Pytorch subsystems in the fusion list with AM-softmax as the loss function.

3.4. Training Strategy

The training process using the Kaldi platform was the same as in the recipes [5] except for our adjustments to hyper-parameters.

On the Pytorch platform, we found that different settings for the training chunk size would affect the results in the short utterance test task. Thus, several Pytorch's subsystems were built, and the only difference was in the setting of chunk size, for instance 50 or 100. We used the suffix '-chunk 50' to indicate this difference, etc. Moreover, for Task 3, we assumed that fewer training epochs may be better for model's generalization, so fewer epochs were chosen for Task 3 compared with Task 1.

3.5. Back-ends

The back-ends consisted of linear discriminant analysis (LDA) dimension reduction, whitening, centering, length normalization and logistic regression (LR) for three tasks. We tuned different back-end configurations for the three tasks' respective development sets. In Task 1, the LDA dimension was 10 for Kaldi's subsystems and 256 for Pytorch's; in Task 2, the LDA dimension was 10 for Kaldi's subsystems and no Pytorch's subsystems were selected for the fusion list; and in Task 3, the LDA dimension was 512 for all subsystems.

For Task 1, an additional length expansion process was used for embedding extraction in the test set, which is introduced in the following Section 3.5.1.

3.5.1. Length Expansion for Short-duration Utterances

We proposed a length expansion method for short-duration utterances, named speed perturbation pooling (SPP) [11]. It enriches information and benefits short-duration utterances. SPP averages x-vectors of differing speeds for the same utterance to generate a new x-vector and the formulas are as follows:

$$\begin{cases} X_{sp0.9} = F\left(x_{sp0.9}\right) \\ X_{sp1.0} = F\left(x_{sp1.0}\right) \\ X_{sp1.1} = F\left(x_{sp1.1}\right) \end{cases} \tag{2}$$

$$X_{spp} = \frac{n_{sp0.9} \cdot X_{sp0.9} + n_{sp1.0} \cdot X_{sp1.0} + n_{sp1.1} \cdot X_{sp1.1}}{n_{sp0.9} + n_{sp1.0} + n_{sp1.1}} \tag{3}$$

where n is the number of frames in the corresponding utterance, sp means the speed perturbation, and $F(x)$ denotes the extractor of a neural network that maps the variable-length acoustic features x to the fixed-dimensional embeddings.

Formula (3) can be seen as the weighted integration of x-vectors X for the different speeds.

3.6. Greedy Fusion

Score-level greedy fusion was used to select the most useful subsystems for the fusion list, given the metric of $Cavg$ in development sets. In order to consider robustness in the final fusion list, the final fusion weight was set to be equal for each subsystem in the final fusion list, rather than other popular fusion approaches, such as using fusion toolkits [14, 15].

In greedy fusion processing, we first selected the subsystem with the lowest $Cavg$. Then, we evaluated all the two-system fusions and obtained the best two systems. We fixed these two systems and then added a third system, and so on. To reduce the risk of overfitting, we conducted fusions with only positive weights.

Table 2: *Results from Three Tasks of AP19-OLR.*

Task	Platform	Model	Feature	$Cavg/EER$ on Dev	$Cavg/EER$ on $Eval$
1	Kaldi	Baseline.xv [3]	MFCC	0.1271/12.37%	0.1257/12.22%
		Multi-task.xv	PLP+Pitch	0.0475	0.1055
		Multi-task.xv	MFCC+Pitch	0.0489	0.1076
	Pytorch	Extended.xv.chunk100	PLP+Pitch	0.0347	**0.0863**
		Extended.xv.chunk100	MFCC+Pitch	0.0358	0.0874
		Extended.xv.chunk50	PLP+Pitch	0.0363	0.0924
		Extended.xv.chunk50	MFCC+Pitch	**0.0345**	0.0909
		Extended.xv.chunk50	FBank+Pitch	0.0441	0.0952
	Final Fusion			**0.0263/2.63%**	**0.0818/8.65%**
2	Kaldi	Baseline.xv [3]	MFCC	0.3868/43.13%	0.3720/38.44%
		i-vector	PLP+Pitch	**0.2815**	**0.2713**
		i-vector	MFCC+Pitch	0.2864	0.2848
	Final Fusion			**0.2813/28.33%**	**0.2741/27.44%**
3	Kaldi	Baseline.xv [3]	MFCC	0.3393/34.47%	0.2027/21.94%
		Multi-task.xv	PLP+Pitch	**0.2027**	0.0228
		Multi-task.xv	MFCC+Pitch	0.2067	0.0214
		Multi-task.xv	FBank+Pitch	0.2220	**0.0120**
	Pytorch	Multi-feature.xv.chunk100	MFCC+Pitch&PLP+Pitch	0.2438	0.0302
		Extended.xv.chunk100	PLP+Pitch	0.2530	0.0521
		Extended.xv.chunk100	MFCC+Pitch	0.2682	0.0549
	Final Fusion			**0.1697/16.67%**	**0.0113/1.13%**

4. Results and Analysis

In this challenge, we built more than thirty subsystems for three tasks, while only the subsystems used in the final fusion are reported in this Section. When we developed the subsystems, the performances were measured by $Cavg$ and the Equal Error Rate (EER). However, we have omitted the EER results for the subsystems due to space constraints and the fact that the principle evaluation metric in this challenge was $Cavg$. The results and the subsystem configurations, as well as the results of the submitted final fusion systems, are all listed in Table 2.

Regarding system platforms, in Task 1, the best single system was the Pytorch based system since more flexible training strategies, such as the Adam optimizer and cosine annealing, were available and completed on Pytorch. For Task 2, Kaldi's i-vector significantly outperformed the other subsystems. As a result, the final fusion only contained two i-vector subsystems. For Task 3, the Kaldi-based multi-task learning model with phonetic information achieved the best performance.

For Task 1, the previously proposed SPP method was useful in this test condition; subsystems that used SPP in the backend procedure steadily outperformed those without SPP, and the reported subsystems in Task 1 in Table 1 all included SPP. The introduction of phonetic information in multi-task learning was helpful to this LID task; therefore, Kaldi-based subsystems without phonetic information were not selected to be part of the final fusion list. Moreover, the performances of AM-Softmax-based systems were much better than traditional cross-entropy-Softmax-based systems in development sets. So, the final fusion subsystems in Pytorch were all trained with AM-Softmax loss. The different settings for training chunk sizes led to complementary subsystem fusion, and more investigation was required to understand the reason behind this. All the subsystems achieved improvements in the Dev, but the performances were corrupted in the $Eval$. In the workshop after the evaluation, the organizer indicated that one language in the $Eval$ was slightly domain mismatched, resulting this discrepancy.

For Task 2, due to the lack of cross-channel training data, it was challenging to build a discriminative system that was more robust than the generative model. We thus built i-vector systems and all obtained significant improvements compared with the x-vector baseline. Among the subsystems, the i-vector system with the PLP feature obtained the best performance for both Dev and $Eval$.

For Task 3, under the zero-resource testing condition, the introduction of phonetic information into the language modeling network was extremely useful to improve the robustness. However, the performance trends in Dev and $Eval$ were reversed in three multi-task learning subsystems; we assumed it might be the over-fitting issue, though further analysis is required. On the Pytorch platform, strong robustness was revealed in the mutli-feature learning model. The performance gap of the multi-feature model between Dev and $Eval$ was considerably smaller than the subsystems based on single features, as subsystems using only one kind of acoustic feature were probably overfitted, even if we reduced the number of epochs.

Overall, from the three tasks, some conclusions have been found: (1) the PLP feature was more effective than other acoustic features for LID; (2) data augmentation was beneficial in both discriminative and generative models; and (3) score-level fusion with an appropriate fusion strategy yielded further improvements.

5. Conclusions

In this paper, we illustrate the details of XMUSPEECH systems for the AP19-OLR challenge. In this challenge, many methods, including our previously proposed methods such as multi-feature learning and length expansion for short-duration utterances, were investigated in three tasks. The final submitted systems were fusion of multiple subsystems, which improved the performances and the robustness in the three tasks. In the future, we will analyze the influence of training strategies (chunk size, etc.), and explore solutions for cross-channel tasks under the condition of no additional training materials.

6. Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No.61876160).

7. References

[1] Z. Tang, D. Wang, Y. Chen, and Q. Chen, “AP17-OLR challenge: Data, plan, and baseline,” in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2017, pp. 749–753.

[2] Z. Tang, D. Wang, and Q. Chen, “AP18-OLR challenge: Three tasks and their baselines,” in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2018, pp. 596–600.

[3] Z. Tang, D. Wang, and L. Song, “AP9-OLR challenge: Three tasks and their baselines,” in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2019.

[4] D. Wang and X. Zhang, “Thchs-30: A free Chinese speech corpus,” *arXiv preprint arXiv:1512.01882*, 2015.

[5] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, “The Kaldi speech recognition toolkit,” in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

[6] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

[7] D. Garcia-Romero and C. Y. Espy-Wilson, “Analysis of i-vector length normalization in speaker recognition systems,” in *Twelfth annual conference of the international speech communication association*, 2011.

[8] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, J. Borgstrom, F. Richardson, S. Shon, F. Grondin *et al.*, “State-of-the-art speaker recognition for telephone and video speech: the JHU-MIT submission for NIST SRE18,” *Proc. Interspeech 2019*, pp. 1488–1492, 2019.

[9] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, “X-vectors: Robust DNN embeddings for speaker recognition,” in *ICASSP*. IEEE, 2018, pp. 5329–5333.

[10] Y. Liu, L. He, J. Liu, and M. T. Johnson, “Speaker embedding extraction with phonetic information,” *arXiv preprint arXiv:1804.04862*, 2018.

[11] M. Zhao, R. Li, S. Yan, Z. Li, H. Lu, L. Li, and Q. Hong, “Phone-aware multi-task learning and length expanding for short-duration language recognition,” in *APSIPA 2019*. IEEE, 2019.

[12] Z. Li, H. Lu, J. Zhou, L. Li, and Q. Hong, “Speaker embedding extraction with multi-feature integration structure,” in *APSIPA 2019*. IEEE, 2019.

[13] F. Wang, J. Cheng, W. Liu, and H. Liu, “Additive margin softmax for face verification,” *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[14] N. Brümmer and E. De Villiers, “The Bosaris toolkit: Theory, algorithms and code for surviving the new DCF,” *arXiv preprint arXiv:1304.2865*, 2013.

[15] N. Brümmer, “Focal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scorestutorial and user manual,” *Software available at http://sites. google. com/site/nikobrummer/focalmulticlass*, vol. 33, p. 39, 2007.