The xmuspeech System for AP19-OLR Challenge

Zheng Li[1], Miao Zhao[2], Jing Li[2], Yiming Zhi[2] and Lin Li[1]
[1]School of Electronic Science and Engineering, Xiamen University, China
[2]School of Informatics, Xiamen University, China

***Abstract*—This paper describes our xmuspeech system for AP19-OLR challenge. The challenge this year contains three tasks, (1) short-utterance LID, (2) cross-channel LID and (3) zero-resource LID. We leverage the system pipeline from three aspects, including the data preparation, language modeling method, and fusion strategy. First, we perform the data augmentation strategy by applying the speed and volume perturbation on the training set. Then, the proposed length expanding method is used in the test set for task 1. As for model building, we develop LID systems on Kaldi and Pytorch, in which different optimization methods are available, for task 1 and task 3, respectively. The well-known x-vector extended architecture, the multi-task learning model with phonetic information and our previously proposed multi-feature integration structure are implemented for task 1 and task 3. The i-vector systems are built for the cross-channel task. For all of three tasks, the pipeline of embedding extraction with a backend classifier is used. Finally, the greedy fusion strategy helps us to choose the subsystems to the final fusion system (submitted system). The Cavg of 0.0263, 0.2813 and 0.1697 on the development set for task 1, 2, 3 are obtained for our submitted systems.**

***Index Terms*—AP19-OLR, language identification, x-vector, multi-task learning, multi-feature integration**

I. Introduction

The language identification (LID) refers to identify the language categories from utterances. Considering the challenge existing in LID tasks, the oriental language recognition challenge is organized annually since 2016 [1,2]. The xmuspeech team has attended the OLR challenge since the AP17-OLR and achieved the third and the first rank in AP17-OLR and AP18-OLR respectively.

The AP19-OLR challenge [3] includes three tasks: short-utterance (1 second) LID (task 1), the same as past two challenges; cross-channel LID (task 2), which reveals the real-life demand of speech technology; and zero-resource LID (task 3) ,in which no resources are provided for training before inference, but only several utterance of each language are provided for language reference. All tasks will be evaluated and ranked separately. We submitted the results of three tasks with required test condition in this challenge.

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

II. Data Preparation

In this AP19-OLR challenge, additional training materials are forbidden to participants and the permitted resources are several specified data sets, including AP16-OL7, AP17-OL3, AP17-OLR-test, AP18-OLR-test and THCHS 30. The detailed description of data sets used is listed in the Table 1.

A. Training Set

Before training, we adopt the data augmentation, including speed and volume perturbation, to increase the amount and diversity of the training data. For speed perturbation, we apply a speed factor of 0.9 or 1.1 to slow down or speed up the original recording, and for volume perturbation, random volume factor is applied. Finally, two augmented copies of the original recording are added to the original data set to obtain a 3-fold training set.

For task 1, the AP16-OL7, AP17-OL3, AP17-OLR-test and THCHS 30 constitute the training set for all Kaldi [4] based systems, namely ap19_task_1_train_with_thchs30_aug. As for training set in Pytorch platform [5], the THCHS 30 data set is not used for the limit of time, therefore the training set in Pytorch based systems for task 1 is named ap19_task_1_train_aug.

As for task 2, we used all data set allowed to use in this challenge to train the i-vector system with data augmentation mentioned above, including AP16-OL7, AP17-OL3, AP17-OLR-test, AP18-OLR-test and THCHS 30. The training set used in task 2 is named ap19_task_2_train_aug. The models for task 3 share the exactly same training set as task 1.

B. Phonetic Training Set

The AP16-OL7 and AP17-OL3 databases contain lexicons of all the 10 languages, as well as the transcriptions of all the training utterances. These resources are chosen to train an ASR model firstly and attain phonetic alignment labels for the multi-task learning with phonetic information. We name this set as phonetic_training_set.

C. Task 1 Test Set

The proposed method [6] introducing length expanding strategy to provide supplemental information of short-duration utterances by dithering the short duration evaluation utterances

Table 1: Data Sets Used in Systems

Task	Model	LDA	Centering	LR
task 1	ap19_task_1_train_with_thchs30_aug ap19_task_1_train_aug phonetic_training_set	ap19_task_1_back_end_train	ap19_task_1_back_end_train	ap19_task_1_back_end_train
task 2	ap19_task_2_train_aug	ap19_task_2_train_aug	ap19_task_2_train_aug	ap19_task_2_enroll_aug
task 3	ap19_task_1_train_with_thchs30_aug ap19_task_1_train_aug phonetic_training_set	ap19_task_1_back_end_train	ap19_task_1_back_end_train	AP19-OLR-test-task3-enroll

at different speeds is used for task 1 development set and evaluation set.

D. Enroll Set

For better matching the test condition (short utterance), the enroll set for task 1 contains the AP16-OL7, AP17-OL3 and AP17-OLR-test-task1 without data augmentation, namely ap19_task_1_back_end_train.

In order to be generalized to cross channel test condition, we use as many data as possible to composes the enroll set for task 2: the enroll set is a subset of training set for task 2, including the target six languages, namely ap19_task_2_enroll_aug.

E. Backend Training Set

As the embedding extraction and backend classifier method is the strategy, the selection of backend training set is influential. We use the same data set as task 1 enroll set to train the backend for task 1 and task 3. And the enroll set for task 2 is used to train to task 2 backend. We use the Logistic Regression (LR) as the classifier for three tasks and the training set for LR is the specific enroll set of each task.

III. METHODS

In this section, the methods used in our systems are introduced briefly.

A. I-vector

The baseline i-vector is used in our systems [7], in which the input features are acoustic features with first and second order derivatives.

B. X-vector Extended

For the sake of the x-vector extended architecture significantly outperforming the baseline x-vector in the Kaldi recipes [8, 9], we choose x-vector extended to build the x-vector system. Compared to the traditional x-vector, the x-vector extended structure uses a slightly wider temporal context in the TDNN layers and interleave dense layers between the TDNN layers.

C. Multi-task Learning Model with Phonetic Information

Considering the relationship between the language and phone classification tasks, we utilize the multi-task learning to train the two tasks jointly [5]. The frame-level hidden layers are the shared part that learns the phonetic compensation information for the language task. From a view of feature space, the phonetic representation is invariant information that is not affected by differences in language and duration. The gradient descent of each task will affect the frame-level shared layers in training and the x-vector will be extracted from the penultimate segment-level layer in the language task branch.

D. Multi-feature Integration

Due to the data distribution of different features is comparatively dissimilar and the information between features is complementary, the proposed multi-feature integration structure is used to utilize complementary acoustic features into x-vector system [10]. While each branch processed one type of acoustic features on the frame level, and the outputs of the two branches for each frame were spliced together as a super vector before being input into the statistics pooling layer.

E. AM-Softmax

The AM-softmax is one of the most popular loss function in classification tasks [11]. The core concept of AM-softmax is the feature normalization and adding an additive margin into the softmax loss function. Given that the AM-softmax is not supported in the Kaldi, we use the Pytorch to complement the AM-softmax and built all of the Pytorch subsystems with AM-softmax as the loss function.

F. Logistic Regression

Logistic regression (LR) [12] is a kind of supervised classification and discrimination algorithm. With the help of Sigmoid function, the training samples are compressed between [0,1], so that each pattern sample has probability significance in the discrimination space. In our systems, all of the backend classifiers in three tasks are the LR.

G. Greedy Fusion

The fusion strategy used is the greedy fusion [13] and for the consideration of robustness, the final fusion weight is set as equal to each subsystem.

IV. EXPERIMENTAL SETTINGS AND RESULTS

A. Experimental Settings

In this challenge, we built more than 20 subsystems for three tasks. Although two platforms (Kaldi, Pytorch) were used to build subsystems, the feature engineering and the backend processing were all complemented on the Kaldi platform.

For feature engineering, three basic acoustic features: 20-dimensional MFCC, 40-dimensional FBank and 20-dimensional PLP were used with 3-dimensional pitch feature respectively.

Table 2: the Results of Subsystems

Task	Platform	Model	Feature	Epoch	LDA_dim	Cavg on dev	Cavg on eval
task1	kaldi	Multi-task.xv	plp		10	0.0475	0.1055
		Multi-task.xv	mfcc		10	0.0489	0.1076
	pytorch	extended.xv.chunk100	plp	30	256	0.0347	**0.0863**
		extended.xv.chunk100	mfcc	30	256	0.0358	0.0874
		extended.xv.chunk50	plp	30	256	0.0363	0.0924
		extended.xv.chunk50	mfcc	30	256	**0.0345**	0.0909
		extended.xv.chunk50	fbank	30	256	0.0441	0.0952
	final fusion (submitted system)					0.0263	0.0818
task2	kaldi	i-vector	plp		10	**0.2815**	**0.2713**
		i-vector	mfcc		10	0.2864	0.2848
	final fusion (submitted system)					0.2813	0.2741
task3	kaldi	Multi-task.xv	plp		512	**0.2045**	0.0228
		Multi-task.xv	mfcc		512	0.2067	0.0214
		Multi-task.xv	fbank		512	0.2220	**0.0120**
	pytorch	extended.multi-feature.xv .chunk100	mfcc&plp	10	512	0.2438	0.0302
		extended.xv.chunk100	plp	10	512	0.2530	0.0521
		extended.xv.chunk100	mfcc	10	512	0.2682	0.0549
	final fusion (submitted system)					0.1697	0.0113

The training process in Kaldi platform was the same as what's in the recipes except our adjustment of hyper-parameters.

In Pytorch training, we found that different settings of the chunk size leaded to different and complementary results. Thus, several Pytorch's subsystems are just different in the setting of chunk size. Further, for task 3, we assumed that less epoch may be better for the improvement of model's generalization, so we used less epoch for task 3.

The backend processing was almost same in three tasks and the main difference was the dimension of LDA which are shown in the Table 1. First, the Backend Training Set was used to train the LDA model. Then the centering matrix was obtained from the LDA-processed Backend Training Set. Thirdly, a normalization matrix was got from the centered-LAD-processed Backend Training Set. Finally, the enroll set was processed through LDA, centering and normalization, then the LR model was trained by the backend-processed enroll set.

Finally, from the score-level greedy fusion on the development set, the final 7 subsystems were chosen for the task 1 fusion, the final 2 subsystems for task 2 fusion and final 6 subsystems for task 3 fusion. The results and configurations of subsystems for fusion and the results of final submitted systems are in the Table 2.

B. Results and Analysis

For task 1 and task 3, the best single systems were both the Pytorch based system. For task 2, the traditional i-vector significantly outperformed other systems, causing the final fusion only contained two i-vector subsystems.

For task 1, the performance of AM-softmax based systems were much better than traditional softmax based systems. The introducing of phonetic information was helpful to the LID tasks, as the Kaldi's subsystems without phonetic information were not chosen into the final fusion list. For the sake of the limit of time, the multi-task learning model with AM-softmax was not complemented but we assume it should perform well.

For task 2, due to the lack of cross channel training data, it was hard to build a more robust system than the i-vector system.

For task 3, given the zero-resource testing condition, the introducing of phonetic information into language modeling network may be useful to improve the robustness. In this task, subsystems using AM-softmax may be overfitting, even if we reduced the number of epoch.

In this challenge, after the key was released, we found that it is difficult to estimate our systems on the development set for obtaining consistent results on the evaluation set. Especially in task 1, given the official baseline obtained the almost the same level of results on the development set and the evaluation set, we can hardly speculate the reasons behind this. The fusion of subsystems improved the performance of three tasks on the development set but not in the evaluation set. The results of subsystems on the evaluation set are also listed on the Table 2.

V. CONCLUSION

In this paper, we illustrate the xmuspeech system for AP19-OLR challenge. For AP19-OLR challenge, a great deal of methods, including our previously proposed methods, were investigated in three tasks. The best single system obtained in task 1 was the x-vector extended with AM-softmax, in task 2 was i-vector and in task 3 was the multi-task learning x-vector. Further, the fusion of subsystems improved the performance and improve the robustness of the submitted systems of three tasks. Finally, the contribution rank of our submitted systems will be: 1. appropriate and novel methods of modeling language identification systems. 2. the optimization of subsystems.

REFERENCES

[1]Tang, Zhiyuan, et al. "AP17-OLR challenge: Data, plan, and baseline." 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). IEEE, 2017.

[2]Tang, Zhiyuan, Dong Wang, and Qing Chen. "AP18-OLR Challenge: Three Tasks and Their Baselines." 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). IEEE, 2018.

[3]Tang, Zhiyuan, Dong Wang, and Liming Song. "AP19-OLR Challenge: Three Tasks and Their Baselines." arXiv:1907.07626 (2019).

[4]Povey, Daniel, et al. "The Kaldi speech recognition toolkit." IEEE 2011 workshop on automatic speech recognition and understanding. No. CONF. IEEE Signal Processing Society, 2011.

[5]Paszke A, Gross S, Chintala S, et al. Automatic differentiation in pytorch[J]. 2017.

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

[7]Garcia-Romero D, Espy-Wilson C Y. Analysis of i-vector length normalization in speaker recognition systems[C]//Twelfth Annual Conference of the International Speech Communication Association. 2011.

[8]Snyder, David, et al. "X-vectors: Robust DNN embeddings for speaker recognition." 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018.

[9]Snyder, David, et al. "Speaker Recognition for Multi-speaker Conversations Using X-vectors." ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019.

[10]Z. Li, H. Lu, J. Zhou, L. Li, and Q. Hong, “Speaker embedding extraction with multi-feature integration structure,” in 2019 APSIPA ASC. IEEE, 2019, p. (accepted).

[11]Yu, Ya-Qi, Lei Fan, and Wu-Jun Li. "Ensemble Additive Margin Softmax for Speaker Verification." ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019.

[12]Kleinbaum D G, Dietz K, Gail M, et al. Logistic regression[M]. New York: Springer-Verlag, 2002.

[13] Ken Kennedy. 2000. Fast greedy weighted fusion. In Proceedings of the 14th international conference on Supercomputing (ICS '00). ACM, New York, NY, USA, 131-140. DOI=http://dx.doi.org/10.1145/335231.335244