The Royal Flush System for AP20-OLR Challenge

Ding Wang, Shuaishuai Ye, and Xinhui Hu

Hithink RoyaFlush AI Research Institute, Zhejiang, China

{wangding2,yeshuaishuai,huxinhui}@myhexin.com

Abstract

This paper describes our royal flush system for AP20-OLR challenge. The challenge this year contains three tasks: (1) cross-channel Language identification (LID), (2) dialect identification, and (3) noisy LID. We leveraged the system pipeline from three aspects, including the data preparation, modeling method, and fusion strategy. Firstly, we performed the data augmentation strategy by applying the speed and volume perturbation on the training set. Secondly, the traditional speech denoising method based on WebRTC was used in the test set for task 3. As for the model building, we developed LID systems on Kaldi, Pytorch, and ESPnet, in which different optimization methods are available. For task 1 and task 3, the extended x-vector architecture and the i-vector system were used to extract corresponding embedding features. In order to classify different languages using the embedding features, we used the Logistic Regression , a supervised backend classifier, in our systems of task 1 and task3. For task 2, we used knowledge transfer learning to train an end-to-end LID model from another end-to-end automatic speech recognition (ASR) model built by using a large corpus. Finally, the greedy fusion strategy helped us choose the subsystems for the final fusion system of tasks 1 and 3 (the submitted systems).

Index Terms: language identification, x-vector, end-to-end

1. Introduction

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, 3]. The AP20-OLR challenge [4] includes three tasks: cross-channel LID (task 1), where the subset recordings were recorded using different devices in different environments; dialect identification (task 2), which was designed for the dialect identification task; and noisy LID (task 3) with the test data set recorded under noisy environment (low SNR). All tasks are evaluated and ranked separately. We participated all the three tasks, and submitted the results of these tasks according to the required test conditions by the challenge.

In this paper, we introduce the Royal Flush systems for AP20-OLR in detail. The remainder of this paper is organized as follows. Section 2 describes the data preparation. Section 3 introduces the approaches adopted for our systems. The experimental settings and results of subsystems on the development set are shown in Section 4. Finally, the conclusion is given in Section 5.

2. Data Preparation

In this AP20-OLR challenge, except for the data sets specified by the challenge committee, the other training materials and non-speech data are forbidden to participants. The permitted resources are several specified data sets, including AP16-OL7, AP17-OL3, AP17-OLR-test, AP18-OLR-test, AP19-OLR-test and THCHS 30 [1, 2, 3, 4]. The data sets used in this challenge are listed in Table 1 and Table 2, and their detailed descriptions will be given in following parts.

2.1. Training Set

For task 1 and task 3, the AP16-OL7, AP17-OL3, AP17-OLR-test, AP18-OLR-test-task2, and THCHS 30, namely *ap20_task_1_train_with_thchs30_aug*, constitute the training set for Kaldi [5] and Pytorch [6] based systems.

For task 2, the AP16-OL7, AP17-OL3, and THCHS 30, which were named *ap20_task2_e2e_asr_train*, constituted the training set for the end-to-end ASR model, and the *AP20-OLR-dialect* was used for knowledge transfer learning.

In task 1 and task 3, we adopted the data augmentation methods, including speed and volume perturbation, to increase the training data's amount and diversity. We applied the random speed factor of 0.9 or 1.1 to slow down or speed up the original recordings for speed perturbation and volume factor to modify the volume of original recordings. Finally, two augmented copies of the original recording were added to the original data set to obtain a 3-fold training set. In task 2, we only applied speed perturbation with the speed factor of 0.9 and 1.1 to increase the training data's amount and diversity.

2.2. Enrollment Set

The enrollment sets for task 1 and task 3 are subsets of the *ap20_task_1_train_with_thchs30* without data augmentation, namely *ap20_task1_back_end_train* and *ap20_task3_back_end_train*. For better matching the test sets, the *ap20_task1_back_end_train* contains only 6 target languages, and the *ap20_task3_back_end_train* contains only 5 target languages, which is also the best way to select the enrollment set for each task.

2.3. Training Set for Bottleneck Feature

For task1 and task 3, the THCHS 30 was used for training the ASR bottleneck featuring (BNF) [7] network, namely *ap20_task_1_bnf_train*.We chose the BNF feature as one of the acoustic features to train the x-vector and i-vector models for the LID tasks, because we thought that the feature based on speech recognition was more helpful for the extraction of language information than the other features.

2.4. Training Set for Backend

According to the strategies of embedding extraction and backend classifier, the selection of training set for backend is very influential. So, we used the corresponding enrollment set to train its backend systems for task 1 and task 3. We used the Logistic Regression(LR) as the classifier for these two tasks, which was trained on the enrollment set of each task.

Table 1: *Data sets used in our systems*

Task	Data for Model	Data for Backend Classifier
Task 1	ap20_task_1_train_with_thchs30_aug ap20_task_1_bnf_train	ap20_task1_back_end_train
Task 2	ap20_task2_e2e_asr_train AP20-OLR-dialect	-
Task 3	ap20_task_1_train_with_thchs30_aug ap20_task_1_bnf_train	ap20_task3_back_end_train

Table 2: *The results of subsystems on dev sets*

Task	Platform	Model	Feature	Denoise	Epoch	Cavg	EER%
Task 1	kaldi	I-vector	PLP&PITCH	No	-	0.1522	25.36
		I-vector	BNF	No	-	0.1577	26.21
	Pytorch	Extended TDNN	PLP&PITCH	No	21	0.1807	30.21
Task 2	ESPnet	Transformer-12L	FBANK&PITCH	No	40	0.0351	6.337
Task 3	Pytorch	Extended TDNN	PLP&PITCH	Yes	21	-	-
		Extended TDNN	PLP&PITCH	No	21	-	-
		Extended TDNN	BNF	Yes	21	-	-
		Extended TDNN	BNF	No	21	-	-

3. System Descriptions

In this section, we will briefly describe our approaches for this challenges, including all sub-systems for LID, regressions adopted, and the strategy of system fusion.

3.1. I-vector

The baseline i-vector system [8] was used in our systems, in which the input features were acoustic features with its first and second order derivatives.

3.2. Extended X-Vector

We chose an extended TDNN [9, 10] as the x-vector system, which was mentioned in the recipes of ap-olr2020-baseline (run_kaldi_ivector.sh). Compared to the traditional x-vector, the extended TDNN x-vector structure used a slightly wider temporal context in the TDNN layers and interleaved dense layers between TDNN layers than the original x-vector architecture, which led to a deeper x-vector model. The deep structure was trained to classify the N languages using the cross entropy (CE) loss function. During the test stage, the embedding features of ¡®x-vector¡¯ were extracted from the affine component of the penultimate layer.

3.3. End-to-end LID

In the task 2, we first trained a transformer-based [11] Joint CTC/Attention end-to-end ASR model on the ESPnet platform using the training set *ap20_task2_e2e_asr_train*. Then, we trained the 6-layer encoder with an attention mechanism of the transformer as LID model to classify the three dialects using knowledge transfer learning on the 12-layer encoder of the transformer-based Joint CTC/Attention ASR model, using the training data *AP20-OLR-dialect*.

3.4. Logistic Regression

LR [12] is a classical supervised classification-regression algorithm. With the help of sigmoid function, the training samples are compressed between [0, 1] which represents a probability of significance of each sample in the discrimination space. Therefore, in order to classify different language categories, we used the LRs for task 1 and task 3 which were trained using embedding feature (e.g. x-vector or i-vector) extracted from their respective enrollment sets.

3.5. Greedy Fusion

The fusion strategy adopted in our systems was the greedy fusion [13]. The greedy fusion strategy is to weighted average the output of all subsystems to obtain the final result. According to our preliminary experiments, all subsystems were set to the same fusion weight, and the sum of fusion weight of all subsystems was 1.

4. Experimental Settings and Results

4.1. Experimental Settings

In this challenge, we built more than 20 subsystems for all the 3 tasks. Although 3 platforms (Kaldi, Pytorch, EspNet) were used to build subsystems, the feature engineering and the backend processing were all completed on the Kaldi platform. For feature engineering, two basic acoustic features: 80-dimensional FBANK and 20-dimensional PLP concatenated with 3-dimensional pitch feature respectively were used. Also, a 60-dimensional BNF was used as another acoustic feature for model training, which was trained by 12-dimensional MFCC of the THCHS30 data set.

For task 1 and task 3, our training process of i-vector system was the same as what it's in the recipes of ap-olr2020-baseline (run_pytorch_xvector.sh). The differences of the x-vector system between ours and the baselines are on our adjustments of hyper-parameters and the number of epochs. Furthermore, for task3 which is for noisy LID task, we use the traditional WebRTC-based noise reduction module to denoise the test audio. The x-vector are extracted from both the original and denoised test data for the LID task. The backend processing was almost the same in task 1 and task3. Linear discriminative analysis (LDA) trained on the enrollment set was employed to promote language-related information. The dimension of the LDA projection space was set to 100. After the LDA projection and centering, the LR trained on the enrollment set was used to compute the score of a trial on a particular language. Finally, according to the results of score-level greedy fusion on the development set, the final 3 subsystems were chosen (for fusion) for the task 1 and the final 4 subsystems were chosen for the task 3.

For task 2, we only used 6 of 10 languages to train the ASR model. The 6 languages are Cantonese, Indonesian, Japanese, Russian, Uygur, and Mandarin. We used 80-dimensional FBANK concatenated with 3-dimensional PITCH to train the ASR and LID models of the task. The transformer-based end-to-end ASR model was composed of a 12-layer encoder with 2048 units, a 6-layer decoder with 2048 units, and 4-head attention with 256 dimensions, which was trained on the premise of taking the word as the modeling unit for 50 epochs with a batch size of 64. The LID model was structured using a 6-layer encoder with 2048 units and 4-head attention with 256 dimensions, which was initialized respectively using the transformer ASR model's encoder and attention to classify 3 dialects. The LID model was trained with 40 epochs using a batch size of 32.

The results and configurations of subsystems used for fusion were presented in the Table 2.

4.2. Experimental Results

As shown in the result Table 2, for task 1, the best single system is the i-vector system based on the Kaldi platform, with the feature of PLP&PITCH. The fusion of the 3 sub-systems is used for submission.

For task 2, we found that the end-to-end system significantly outperforms the other systems. However, because the system's scores are not easy to fuse with the other systems for the task2, only the output of the end-to-end system is used for submission.

For task 3, due to the lack of development set for it, we only showed 2 Pytorch-based systems in which different features were used, and no corresponding results were presented here. In order to improve the performance for the test set, we enhanced the test set using a WebRTC-based noise reduction module. Therefore, depending on whether noise reduction is used, we totally used four sub-systems for the final submission.

5. Conclusions

In this paper, we illustrated the Royal Flush system for the AP20-OLR challenge. Many methods were investigated for three tasks. Among our experimented systems, the best single system was the i-vector for task 1 and the transformer-based encoder classifier for task 2. Furthermore, the fusion of subsystems is verified to improve the performance and robustness of the submitted systems for all three tasks. The contributions of our submitted systems can be concluded as: 1) End-to-end based modeling for the language identification systems, and 2) the optimization of different subsystems.

6. 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)*, 2017.

[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)*, 2018.

[3] Z. Tang, D. Wang, and L. Song, "Ap19-olr challenge: Three tasks and their baselines," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2019.

[4] Z. Li, M. Zhao, Q. Hong, L. Li, and C. Yang, "Ap20-olr challenge: Three tasks and their baselines," 2020.

[5] G. Boulianne, "The kaldi speech recognition toolkit," *IEEE 2011 workshop on automatic speech recognition and understanding. No. CONF. IEEE Signal Processing Society*, 2011.

[6] P. A, G. S, C. S, and et al., "Automatic differentiation in pytorch," *NIPS*, 2011.

[7] H. Sun, K. A. Lee, N. T. Hieu, B. Ma, and H. Li, "I2r-nus submission to oriental language recognition ap16-ol7 challenge," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2017.

[8] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems." in *Interspeech, Conference of the International Speech Communication Association, Florence, Italy, August*, 2011.

[9] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[10] D. Snyder, D. Garcia-Romero, G. Sell, A. Mccree, and D. Povey, "Speaker recognition for multi-speaker conversations using x-vectors," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[11] N. S. Vaswani, Ashish and et al., "Attention is all you need," in *In Advances in neural information processing systems*, 2019.

[12] D. Kleinbaum and M. Klein, *Logistic Regression*. New York: Springer, 2010.

[13] K. Kennedy, "Fast greedy weighted fusion," *International Journal of Parallel Programming*, 2001.