The RoyalFlush System Description for AP21-OLR Challenge

Ding Wang[1], *Shuaishuai Ye*[1], *Xinhui Hu*[1], *Sheng Li*[2]

[1]Hithink RoyalFlush AI Research Institute, Zhejiang, China
[2]National Institute of Information and Communications Technology (NICT), Kyoto, Japan

[1]{wangding2,yeshuaishuai,huxinhui}@myhexin.com
[2]sheng.li@nict.jp

Abstract

This paper describes our RoyalFlush system for the AP21-OLR challenge. The challenge this year contains four tasks: (1) constrained Language identification (LID), (2) unconstrained LID, (3) constrained multilingual ASR, (4) unconstrained multilingual ASR. Because the challenge of this year has two main tasks: LID and multilingual ASR, we adopted different modeling schemes for different tasks and optimized the systems from three aspects, including the data augmentation, modeling method, and fusion strategy. Firstly, We constructed one training set and two development sets for LID and ASR tasks respectively. Secondly, we developed our systems on Kaldi, Pytorch, ESPnet and Wenet, in which different optimization methods are available. For LID tasks, x-vector system and end-to-end (E2E) system were both adopted. For ASR tasks, we only used E2E system because of its better performance and flexibility. Finally, the greedy fusion strategy helped us choose the LID subsystems for the final fusion system of task 1 and 2, and the rover toolkit was used to obtain the ASR results of the final fusion of multiple end-to-end systems for task 3 and 4.

Index Terms: LID, multilingual ASR, x-vector, end-to-end, multi-task training

1. Data Preparation

In this AP21-OLR challenge, only the data provided by the organizer can be used in the constrained tasks (task 1 and 3), and any data can be used for training and optimization in the unconstrained tasks (task 2 and 4). Moreover, the organizing committee has not designated the development set of the competition this year, and all participating teams are encouraged to build their own development set. Therefore, we sorted out the data provided by the organizer and constructed different training sets and development sets for each task, which is listed in Table 1.

1.1. Training and Development Sets

Because the total amount of data available for the LID task and ASR task is different, we have sorted out one training set and one development set for task 1 and task 3 respectively.

We arranged all available data in the *OLR2021_train* provided by the organizer and sorted out five data sets, *AP21-LID-Train*, *AP21-ASR-Train*, *AP21-LID-Dev*, *AP21-ASR-Dev*, *AP21-LID-Task2-4Lang*. The training sets of LID and ASR both contain cross channel data, while the two development sets do not contain cross channel data.

For task 1 and 3, after the progress set results can be submitted on November 1, we also use the progress set as the development set for system tuning. After November 20, we use the fusion results of three better performing subsystems on the progress set as labels to continue to use it to evaluate our system.

1.2. Data Augmentation

We adopted speed and volume perturbation data augmentation methods in both x-vector systems and end-to-end systems, to increase the amount and diversity of training set. We applied the speed factor of 0.9, 1.0 and 1.1 to the original recordings for speed perturbation and random volume factor to modify the volume of original recordings. Finally, three augmented copies of the original recording were added to the original data set to obtain a 4-fold training set. For end-to-end systems, we also used the data augmentation strategy of noise disturbance, one noise augmented copies were added to the end-to-end training set so we obtain a 5-fold training set.

2. Task 1

In the task 1, because the results of the X-vector and the End-to-end systems are complementary, we fused their results as the final result.

2.1. End-to-end LID

In the end-to-end (E2E) system, we adopted two training schemes, namely transfer learning and multi-task joint training. For transfer learning, we firstly trained a 18-layer conformer-based [6] Joint CTC/Attention end-to-end ASR model using the training set *AP21-ASR-Train*. Then, we trained the 18-layer encoder with an attention mechanism of the conformer as LID model to classify the 13 languages using knowledge transfer learning on the 18-layer encoder of the conformer-based Joint CTC/Attention ASR model, using the training data *AP21-LID-Train* [7]. The more detailed system configuration was described in the [7]. For the multi-task joint training, our E2E OLR/ASR model adopted a hybrid CTC/attention architecture that consists of three components: a shared encoder, an attention decoder, and a CTC module. We trained a 18-layer conformer-based Joint OLR/ASR end-to-end model, as shown in the figure 1. The training process is to jointly optimize the weight-sum of the loss of the attention decoder L_{att} and the CTC loss L_{ctc} shown as:

$$L = \alpha L_{ctc} + (1 - \alpha) L_{att} + \beta L_{olr} \quad (1)$$

where the hyper-parameter α represents the weight of the CTC loss, and the β is the weight of the OLR loss.

2.2. X-vector

2.2.1. *Resnet34 x-vector*

We chose an Resnet34 as the x-vector system. Compared to the traditional x-vector and the extended TDNN x-vector,ResNet structure can learn a lot of detailed temporal information. The deep structure was trained to classify the N languages using the

Table 1: *Data sets used in our systems*

Set name in our paper	Source data sets [1, 2, 3, 4, 5]	Used in wihch task	Usage in our systems	Utterances	Transcripts
AP21-LID-Train	OLR2021_Train	task 1-2	train & enroll	300999	No
AP21-ASR-Train	OLR2021_Train	task 1-4	train	99315	Yes
AP21-LID-Dev	OLR2021_Train	task 1-2	dev	6500	No
AP21-ASR-Dev	OLR2021_Train	task 3-4	dev	6500	Yes
AP21-LID-Task2-4Lang	OLR2021_Train	task 2	enroll	7762	No
Task1_3_4_Progress	OLR2021_Progress	task 1	dev	16431	No

additive angular margin (AAM) loss function. During the test stage, the embedding features of x-vector were extracted from the affine component of the penultimate layer.

In addition, the input feature of our Resnet34 x-vector system is Bottleneck feature [8] of the end-to-end system mentioned above, We use the encoder output of the end-to-end network as the bottleneck feature. Therefore, we train all BNF data as valid frames without using VAD module to filter silent frames.

2.2.2. Back-end

Logistic regression (LR) [9] 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 category, we used the LRs in our x-vector system which were trained using embedding feature extracted from respective enrollment set.

2.3. Greedy Fusion

The fusion strategy of the greedy fusion [10] was adopted in our systems. 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.

3. Task 2

In the task 2, we did not use any speech or text data due to our own conditions. So we regarded this task as a low resources task.

Firstly, we built 3 Resnet34 x-vector systems using the BNF features output by 3 different end-to-end systems (conformer-wenet, conformer-espnet2, conformer-wenet-unified). The target language of our model training is also 13 languages, as same as task 1. Secondly, in the back-end classification stage, we take the two data sets *AP21-LID-Train* and *AP21-LID-Task2-4lang*, which contain 17 languages together as the enrollment set, extract the x-vector embedding by the model, and use LR to train a classifier for 17 languages. Finally, we also adopted greedy fusion strategy to obtain the final result.

4. Task 3 & Task 4

Similar to task 2, we did not collect data for task 4 and train an ASR system under an unrestricted data set, so our systems of task 3 and task 4 are the same.

4.1. End-to-end ASR

For multilingual ASR, we also chose two methods, namely the E2E multilingual ASR [7] and the multi-task training of ASR and LID. As for E2E multilingual ASR, we adopt the same method with [7]. For the multi-task training of ASR and LID, we used the identical network architecture to that mentioned in 2.1. Our multi-task training system of ASR and LID is shown in Figure 1.

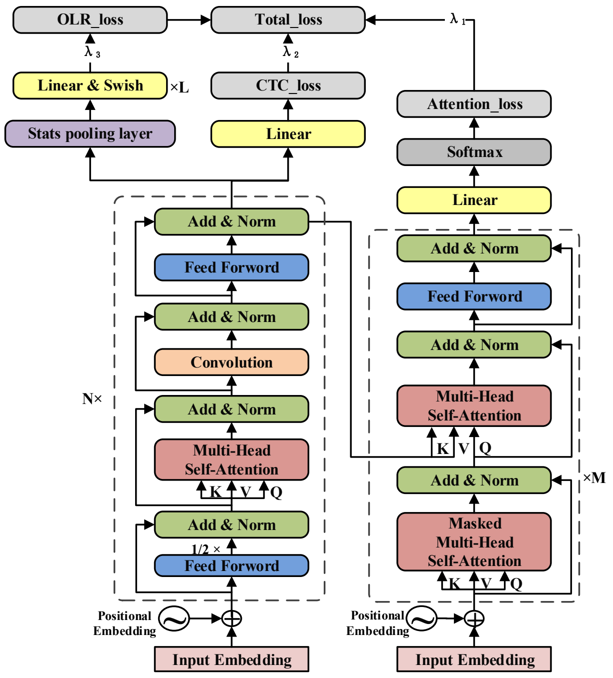

Figure 1: *Structure of our multi-task training of ASR and LID system.*

4.2. Rover Fusion

The fusion strategy used in ASR systems is Rover [11], a system developed at NIST to produce a composite ASR system output when the outputs of multiple ASR systems are available. The Rover system implements a ”voting” or rescoring process to reconcile differences in ASR system outputs. The outputs of multiple ASR systems are combined into a single, minimal cost word transition network via iterative applications of dynamic programming alignments. The resulting network is searched by an automatic ”voting” process that selects an output sequence with the lowest score.

Table 2: *LID results for systems with different settings on dev-set and progress-set. 'CE' represents cross entropy loss and 'KL' represents Kullback-Leibler divergence loss.*

System	Training schemes	OLR Loss	Platform	Dev-sets		Progress-sets[*]	
				Cavg	EER%	Cavg	EER%
Baseline x-vector	-	AM	Pytorch	-	-	0.0826	9.038
Our fusion system	-	CE & KL	Wenet & Pytorch	0.0033	0.3692	0.0060	0.767
Resnet x-vector	-	AAM	Pytorch	0.0375	3.646	0.0286	2.629
E2E-swish	Transfer	CE	Wenet	0.0019	0.2154	0.0181	1.868
E2E-swish	Transfer	KL	Wenet	0.0018	0.2000	0.0189	1.984
E2E-noswish	Transfer	KL	Wenet	0.0020	0.2154	0.0194	2.392
E2E-swish	Multi-task joint	KL	Wenet	0.0481	5.0150	0.0255	2.690
E2E-noswish	Multi-task joint	KL	Wenet	0.0428	4.4620	0.0226	2.234

[*] The performance of each system in the following table on the progress set is that after the x-vector system and end-to-end system we trained are submitted on the progress set, we got the label of the progress set according to the subsystem output voting strategy.

Table 3: *ASR results for systems with different settings on dev-set.*

System	Unified training	Platform	CER% of Dev-set													
			Total	zh-cn	Minnan	Shanghai	Sichuan	ct-cn	id-id	ja-jp	ko-kr	ru-ru	vi-vn	Kazak	Tibet	Uyghu
Baseline Transformer[*]	No	Pytorch	39.4	116.8	69.3	35.9	34.5	47.0	9.2	67.3	34.2	35.5	31.1	35.1	52.7	21.0
Our fusion system	-	Wenet & Espnet2	11.4	11.2	33.9	28.3	21.8	17.9	10.4	23.8	13.3	9.1	9.0	13.1	6.0	6.5
E2E-conformer	No	Wenet	12.7	11.9	35.0	29.0	22.9	18.9	11.4	25.1	14.1	9.8	9.5	13.8	10.9	7.0
E2E-conformer	No	Espnet2	12.3	13.1	36.1	29.6	23.0	18.4	10.4	24.8	12.8	9.1	9.9	14.3	6.0	8.7
E2E-conformer	Yes	Wenet	12.4	11.9	35.5	29.1	22.8	18.1	11.3	24.3	13.7	9.6	9.5	13.8	9.8	6.9
E2E-conformer-swish	Yes	Wenet	12.3	11.4	35.2	29.4	22.4	17.8	11.0	24.0	13.3	9.5	9.3	13.9	9.4	6.9

[*] The performance of the baseline system on the development set is copied from [5].

5. Experimental Settings and Results

5.1. Experimental Settings

In this challenge, we built more than 20 subsystems for all 4 tasks. Although 3 platforms (Kaldi, Wenet, EspNet2) were used to build subsystems, the feature engineering and the back-end processing were all completed on the Kaldi [12] platform. For feature engineering, two basic acoustic features: 80-dimensional FBANK and 80-dimensional PLP concatenated with 3-dimensional pitch feature respectively were used. Also, a 256-dimensional BNF was used as another acoustic feature for x-vector model training.

For x-vector systems in task 1 and 2, the structure of our system was the same as what it's in the recipes of olr2021-baseline (resnet-xvector.py). The differences of the x-vector system are our adjustment of hyper-parameters, and the number of epochs of all x-vector systems is set to 41 with a batch size of 64. 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 logistic regression (LR) trained on the enroll 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 the task 2 fusion.

For end-to-end systems in task 1 and 3, the conformer-based end-to-end model was composed of a 18-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 character as the modeling unit. The transfer learning LID model was structured using a 18-layer encoder with 2048 units and 4-head attention with 256 dimension followed by 1-layer stats pooling layer and 3-layer linear layers, which was initialized respectively using the encoder and attention of the conformer ASR model to classify 13 languages. The ASR part of multi-task model employed the same network configuration with the E2E ASR, and the encoder of ASR was followed by 1-layer stats pooling layer and 3-layer linear layers as a classifier of LID part.

The results and configurations of subsystems used for fusion were presented in the Table 2 and Table 3.

5.2. Experimental Results

For task 1, because the results on the progress set are complementary, we fused 1 BNF x-vector system with 5 end-to-end based systems. For task 2, all 3 subsystems we used to fuse was BNF x-vector systems, the BNF feature extraction end-to-end systems are the wenet-conformer, espnet2-conformer and wenet-conformer-unified-swish systems, which is the first, second, and fourth system mentioned in Table 3. For task 3 and 4, two sets of non multi-task joint training end-to-end systems and two sets of multi-task joint end-to-end systems are fused to obtain the submitted results.

6. Conclusions

In this paper, we illustrated the RoyalFlush system for the AP21-OLR challenge. Many methods, including our previously proposed methods, were investigated in four tasks. Among our experimented systems, the best single system was the end-to-end system. Further, the fusion of subsystems improved the performance and robustness of the submitted systems for all four tasks. Finally, the contribution rank of our submitted systems will be: 1) End-to-end based modeling for the LID and ASR systems, and 2) x-vector base LID system with bottleneck feature, and 3) the optimization of different subsystems.

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)*, 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] B. Wang, W. Hu, J. Li, Y. Zhi, and C. Yang, "Olr 2021 challenge: Datasets, rules and baselines," *IEEE*, 2021.

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

[7] D. Wang, S. Ye, X. Hu, S. Li, and X. Xu, "An End-to-End Dialect Identification System with Transfer Learning from a Multilingual Automatic Speech Recognition Model," in *Proc. Interspeech 2021*, 2021.

[8] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, 2017.

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

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

[11] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, 2002.

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