Language recognition on unknown channels: the LORIA-Inria-MULTISPEECH system for AP20-OLR Challenge

Raphaël Duroselle, Md Sahidullah, Denis Jouvet, Irina Illina

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

firstname.name@loria.fr

Abstract

We describe the LORIA-Inria-MULTISPEECH submitted system for the oriental language recognition AP20-OLR Challenge. This system has been specifically designed to be robust to an unknown channel (task 1). Three sets of experiments have been carried out: training of multilingual bottleneck features, selection of robust features by evaluating language recognition performance on an unobserved channel, and design of the final models with recipes that take advantage of channel diversity within the training set. Key factors for channel robustness are data augmentation techniques, stochastic weight averaging, and regularization of TDNNs with domain robustness loss functions. The final system is the combination of four TDNNs and one GMM model. It achieve a C_{avg} of 0.0156 for matched conditions and 0.0382 for mismatched conditions, on our development sets.

Index Terms: language recognition, channel mismatch, domain generalization

1. Introduction

Language recognition [1] is the task of predicting the language used in a test speech utterance. State-of-the-art language recognition systems are based on automatic training and consequently highly depend on the training data sets. A drop in performance is expected from a domain shift of the test data, such as channel mismatch. Various techniques have been introduced to reduce the impact of domain shift over language recognition systems: design of robust features, data augmentation of the training set, feature-based adaptation in an embedding space and model-based adaptation of the classifier.

The Oriental Language Recognition (OLR) Challenge has been organized for the last five years with the goal of boosting research on language recognition technology [2, 3, 4, 5]. The AP2020-OLR edition [6] focuses on three tasks: cross channel language identification (task 1), dialect identification (task 2) and noisy language identification (task 3). The LORIA-Inria-MULTISPEECH team focused on improving robustness to unknown conditions in order to compete for task 1. Since the noisy language identification problem can also be addressed with domain robustness methods, we also submit a system for task 3. We did not participate in task 2.

Three sets of experiments have been carried out to design a robust language recognition system. First, we trained robust frame-level *bottleneck features*, with a recipe based on an end-to-end speech recognition model (Section 3). Then, in order to simulate conditions of the evaluation, we trained several utterance-level classifiers with different recipes, without using training data from the unknown channels, and evaluated their performance on the unknown channel. We call this step *optimization for system robustness*. Finally, we trained several systems on a large dataset including data from unknown channels, and merged their prediction in order to increase robustness. This step is called the *final system design*. For each step specific training and testing sets have been used, they are described in Section 2. Results for each set of experiments are presented and discussed in Section 7.

The submitted system is the same for task 1 and 3. It is constituted of the fusion of five models: one GMM model (Section 5) and four TDNNs (Section 4), trained with bottleneck features and different loss functions. We propose a new recipe for training multilingual bottleneck features from a *Conformer* model [7], artificial data augmentation techniques abiding by the rules of the challenge and stochastic weight averaging [8] to increase generalization of the model. Other key factors that have been investigated on our development sets are the combination of models with different properties and a duration-dependent calibration [9].

2. Use of data

2.1. Corpora

The training data contains 16 languages. The 10 traditional languages of the OLR evaluation are Cantonese, Mandarin, Indonesian, Japanese, Russian, Korean, Vietnamese, Kazakh, Tibetan, and Uyghur. Only these languages are provided with transcriptions in the sets AP16-OL7 and AP17-OL3. In addition, recordings for three Chinese dialects are provided: Hokkien, Sichuanese, and Shanghainese. Moreover three non target languages of previous evaluations are provided: Catalan, Greek, and Telugu.

Of key interest are the sets that contain recordings from unknown channels: AP19-OLR-dev-task2 and AP19-OLR-test-task2. They only contain 6 languages: Japanese, Russian, Vietnamese, Tibetan, Mandarin and Uyghur. These constitute the *development languages*. It means that we do not have access to recordings from unknown channels for three of the six languages of task 1: Cantonese, Indonesian and Korean. Consequently, we designed a two-step strategy to use this data first for evaluating performance on unknown channels (*optimization for system robustness*) and then to increase the diversity of the training set (*final system design*).

As mentioned before, the system has been designed in several steps. Train, validation and test datasets used for each step are described in Table 1.

1. training of *bottleneck features*. We use recordings from the ten traditional languages with their transcriptions.
2. *optimization for system robustness*. In order to replicate the unknown channel evaluation conditions, we design systems using data from mobile channels only. We evaluate them on three datasets: test-2018 (with mobile channel data), dev-2019 and test-2019 (with unknown channel data). To focus on the channel mismatch problem, we only use the 6 *development languages*.

Table 1: *Use of datasets for each set of experiments. val. refers to validation.*

Dataset	*bottleneck features*	*optimization for system robustness*	*final system design*
AP16-OL7	train & val.	train & val.	train & val.
AP17-OL3	train & val.	train & val.	train & val.
AP17-test		train & val.	train & val.
AP18-test		test-2018	train & val. & test-2018
AP19-dev-task2		dev-2019	test-2019
AP19-dev-task3		dev-2019	train & val.
AP19-test-task1		test-2019	train & val. & test-2019
AP19-test-task2		test-2019	train & val.
AP19-test-task3		test-2019	train & val.
AP20-dialect			train & val.

3. *final system design*. At the final stage of training, once we have selected training recipes robust to channel mismatch, we increase further diversity of the training set by adding data from unknown channels. We train the system with 16 languages. This allows us to evaluate the system for three different sets of languages: *development languages*, languages of task 1 and languages of task 3. We use two test sets: test-2018 (with only mobile channel data) and test-2019 (with unknown channel data).

When an original dataset is split into several datasets for our experiments, we use speaker labels (when available) to ensure that all recordings of the same speaker belong to the same set.

2.2. Data augmentation

To improve the robustness of the systems to channel mismatch, we trained them with three data augmentation techniques. To abide by the rules of the challenge, we did not use any additional speech corpus and only used other files of the training corpus or generated artificial noise or filters. The three data augmentation methods are: addition of white noise, addition of babble noise (sampled by mixing other files of the training set) and convolution with random artificial band-pass filters. We also applied the specAugment strategy [10] during training.

3. Frame-level features

We experimented with different frame-level features during the evaluation. To compare them, we trained a standard time delay neural network (TDNN - Section 4) [11] for language recognition with the datasets defined for the *optimization for system robustness* experiments. We explored two kinds of frame-level features: spectral features and bottleneck features (BNFs).

3.1. Spectral features

Mel-filterbank features and mel-frequency cepstral coefficients (MFCCs) were evaluated. When used without any domain robustness technique, restriction to the telephone bandwidth (300-3800 Hz) was shown useful. No significant gain was achieved by adding pitch features [12].

The Gaussian mixture model (GMM) system employs shifted delta coefficients (SDCs) features computed with short-term feature MFCCs [13]. The MFCCs are extracted from speech frames of 20 ms with shift of 50% and 16 filters in mel scale. We process the extracted MFCCs with relative spectral (RASTA) processing for removing slowly varying channel effect. Then we compute SDCs with shift of three frames and context of seven to create 128-dimensional MFCC-SDC features.

For all systems, we applied energy-based speech activity detector (SAD) to discard the non-speech frames. Finally, we found utterance-level cepstral mean and variance normalization (CMVN) helpful for robustness.

3.2. Bottleneck features

Multilingual bottleneck features [14] are very efficient frame-level features for language recognition. Usually, forced alignment between frame-level acoustic features and transcriptions is performed by an automatic speech recognition system for each target language. Then simple neural networks are trained to predict for each frame the assigned phone or triphone. An embedding is extracted from an intermediate bottleneck layer of the system.

In this work, we did not invest time resources to develop an acceptable automatic speech recognition system for each target language with the goal of performing forced phone alignment. Conversely we trained a unique multilingual end-to-end speech recognition system with the connectionist temporal classification (CTC) loss [15]. We used the *Conformer* architecture [7] with an output layer specific to each target language, with 64 Mel-filterbank features as input. The ten traditional languages were used for training the *Conformer* model. A sentence piece model [16] was trained on the training transcriptions to define 2000 target tokens for each language. Finally we extracted frame-level embeddings from the output of the encoder of the *Conformer* model and used them as language recognition features. We used a small architecture with two *Conformer* blocks and four attention heads.

4. Neural network training

The segment-level classification task can be performed by a neural network. First the general training recipe was selected according to experiments with the datasets defined for *optimization for system robustness*. For this set of experiments, we kept data from unkown channels for evaluation of the system.

Then several systems were trained using the datasets defined for the *final system design*, with regularization loss functions that benefit from training data from unknown channels. Experiments were performed with the toolkits Pytorch and Keras.

4.1. Architecture

The standard time delay neural network (TDNN) architecture for language recognition [11] was used in this work.

4.2. Training recipe

The final training recipe includes:

- the cross-entropy (CE) loss function
- stochastic gradient descent with dropout [17]
- specAugment [10]
- the three data augmentation techniques described in Subsection 2.2
- stochastic weight average (SWA) [8]. It consists of averaging parameters of the model along the trajectory of the gradient descent.

4.3. Exploring other loss functions

At the final stage, we used the *final system design* datasets with training data from unknown channels. Consequently, we trained systems with different loss functions to reduce the domain mismatch:

- additive angular margin softmax (AAM) [18] as an alternative classification loss function
- regularization of the cross-entropy with maximum mean discrepancy (MMD) between mobile channel and unknown channels [19], we compared different weights λ for the regularization loss functions
- regularization of the cross-entropy with n-pair loss [20]

5. Gaussian mixture model training

As an alternative utterance-level model, we chose the GMM-based model as this gives promising language recognition accuracy for short test utterances [1]. This statistical approach was also traditionally used in NIST language recognition evaluations (LREs) and could be helpful by providing complementary information to the neural network based methods. Language-dependent GMMs are trained using 4096 mixture components and ten iterations of the *expectation-maximization* (EM) algorithm. During scoring, we compute the log-likelihoods corresponding to each target language models separately.

6. Fusion and calibration

Standard linear multi-class calibration and score fusion are performed for each task with the FoCal toolkit [21]. This step is performed on the validation set defined for the *final system design*. The final scores are log-likelihood ratio.

For the selected final combination of systems, we implemented a duration-dependent calibration procedure [9]. For three ranges of duration (inferior to 2s, between 2 and 4s, superior to 4s), we learned specific fusion and calibration parameters. For test utterances, the duration is estimated thanks to the energy-based SAD module mentioned in Section 3.

7. Experiments

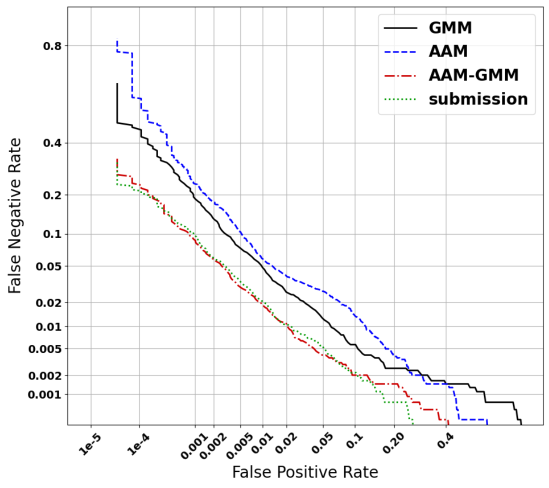
In this section, we report the main results that lead to the design of the submitted system.

7.1. Bottleneck features training

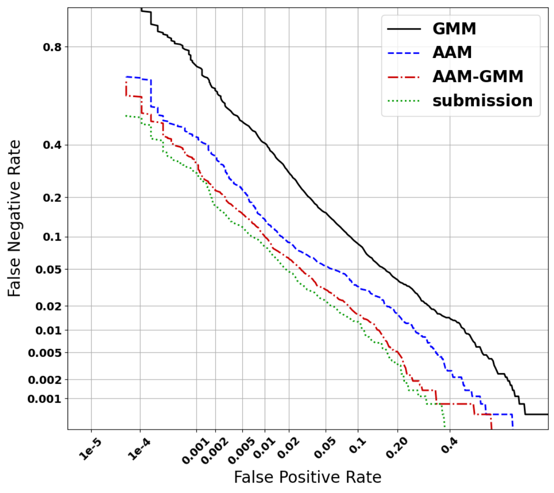
We used the *bottleneck features* datasets. Models are compared with the average CTC loss for the 10 target languages on the validation set, cf. Table 2. We observe that the combination of cepstral mean and variance normalization (CMVN), specAugment and data augmentation allows to train better speech recognition features. We use these features in the final system.

Table 2: *Automatic speech recognition performance in terms of CTC loss for different training recipes*

System name	Training recipe	CTC-loss on validation set
Baseline	Mel filterbank features	3.94
	+ CMNV	3.69
	+ specAug	3.52
Final	+ data augmentation	**3.02**



(a) *Test 2018: known channel*



(b) *Test 2019: unknown channels*

Figure 1: *DET curves of final systems for task 1. We compare four systems including the final submission.*

7.2. Optimization for system robustness experiments

During the *optimization for system robustness* experiments, we train language recognition systems without using training data from unknown channels. Performance is measured with the equal error rate (EER) on three test sets: test-2018 (mobile channel), dev-2019 and test-2019 (which contain recordings from unknown channels), with the 6 *development languages*, cf. Table 3.

Our experiments demonstrate the superiority of the trained bottleneck features over mel filterbanks and MFCCs. BNFs trained with a recipe aiming at robustness (final BNFs) are superior to baseline BNFs for the challenging dev-2019 set. Finally, for all models, SWA was superior to model selection based on the best validation loss.

7.3. Final system design experiments

To design the final system we trained various models with a large dataset containing data from 16 languages, including recordings from unknown channels. We calibrated them on the

Table 3: *Language recognition performance for domain robustness experiments. Equal error rate (%) on test sets.*

input features	Training recipe	test-2018	dev-2019	test-2019
Mel filterbanks	data augmentation	11.95	25.48	29.54
MFCCs	data augmentation	4.75	37.78	24.34
Baseline BNF	data augmentation	3.91	19.11	**11.08**
Final BNF	data augmentation	3.43	17.24	13.68
Final BNF	data augmentation + SWA	**2.97**	**16.92**	12.78

Table 4: *Comparison of final systems* - $C_{avg} \times 100$

	models	Task 1			Task 3		
		validation	2018	2019	validation	2018	2019
individual models	CE, *optimization for system robustness* datasets	3.94	5.60	10.30	3.48	5.91	9.25
	CE, *system design* datasets	**2.50**	4.69	7.26	**2.04**	4.34	7.16
	AAM	3.68	3.14	**5.34**	3.87	3.20	**6.25**
	n-pair	3.22	5.73	7.25	2.58	5.14	7.36
	MMD $\lambda = 1$	3.18	5.39	7.87	2.93	4.99	8.31
	MMD $\lambda = 100$	4.01	5.89	7.60	3.76	5.24	7.29
	GMM	3.31	**2.53**	10.82	3.00	**2.69**	13.97
combination of models	AAM - GMM	1.37	1.54	4.37	1.35	**1.45**	5.73
	AAM - GMM - MMD 100	0.98	1.57	3.91	1.00	1.60	4.96
	AAM - GMM - MMD 100 - n-pair	0.94	**1.50**	3.77	0.99	1.66	4.57
	AAM - GMM - MMD 100 - n-pair - MMD 1	**0.93**	1.53	**3.67**	**0.97**	1.60	**4.54**
	duration-dependent calibration		1.56	3.82		1.61	4.60

corresponding validation datasets and evaluated them on two datasets in terms of C_{avg} [6]: test-2018 (with mobile channel data only), test-2019 (with some unknown channel recordings). Evaluation is performed for two sets of languages corresponding to task 1 and task 3. Results are presented in Table 4.

For comparison, we present performance of a TDNN trained with cross-entropy loss with the *optimization for system robustness* datasets. All other models are trained with the *final system design* datasets and achieve better performance. The best performance is achieved by the TDNN trained with AAM loss. TDNNs trained with regularization loss functions (MMD and n-pair) do not improve on the test sets but the performance gap between known and unknown channels is reduced. Finally, the GMM model achieves the best performance on matched conditions but is very sensitive to channel mismatch.

Then, starting with the model with best performance, we greedily add models to the final system. Fusion is performed with a linear fusion of the scores. We select the combination of five systems: four TDNNs based on BNFs trained with additive angular margin softmax, regularization with maximum mean discrepancy with low and high weight values (λ), regularization with n-pair loss and one GMM model using MFCCs.

The interest of combining different systems can be understood thanks to the DET curves presented in Figure 1. For known conditions (Figure 1a), the best performance of individual systems is achieved by the GMM model. Fusion with the TDNN trained with AAM helps to improve performance but the addition of three other TDNNs trained with domain robustness objectives does not improve further. Conversely, for unknown conditions (Figure 1b), the TDNN trained with AAM is better than the GMM model. The fusion of both is also helpful and the fusion with three other robust models allows an additional gain in performance.

Finally, we learn the fusion and calibration parameters for each task for three different duration ranges. At test time, we use the fusion parameters corresponding to the speech duration of the test utterance. Even though this technique was not beneficial for our test sets, we suspect that it is because we mainly have short duration utterances in 2018 and 2019 test data. Since we do not know the duration distribution of the test sets, we calibrate our system for different durations.

8. Conclusion

The Oriental Language Recognition AP20-OLR Challenge was a good opportunity to evaluate the domain generalization ability of training recipes of language recognition systems. In this work, we propose a combination of two ideas: selecting robust features by evaluating their performance on an unknown channel and using the different channels of the training set to enforce invariance of the models to a change of channel, with the hope of generalizing to new unknown channels. In the first step, we selected multilingual bottleneck features extracted from a *Conformer* speech recognition model trained with data augmentation methods. In the second step, we showed the effectiveness and complementarity of different loss functions to train language recognition models.

The difficulty of task 1 is generalization to a new channel. By definition of the task, we were not able to perform a reliable estimation of our performance since we do not have access to development data from the target channels. We are looking forward to the evaluation results in order to evaluate our two-step strategy which aims at leveraging diversity within the training data. We plan to validate the methods explored during this evaluation in a more controlled scenario with other language recognition corpora, to understand better the impact of unbalanced data for different channels and languages.

9. Acknowledgements

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

10. References

[1] E. Ambikairajah, H. Li, L. Wang, B. Yin, and V. Sethu, "Language identification: A tutorial," *IEEE Circuits and Systems Magazine*, vol. 11, no. 2, pp. 82–108, 2011.

[2] D. Wang, L. Li, D. Tang, and Q. Chen, "Ap16-ol7: A multilingual database for oriental languages and a language recognition baseline," in *Proc. APSIPA*. IEEE, 2016, pp. 1–5.

[3] Z. Tang, D. Wang, Y. Chen, and Q. Chen, "Ap17-olr challenge: Data, plan, and baseline," in *Proc. APSIPA*. IEEE, 2017, pp. 749–753.

[4] Q. C. Zhiyuan Tang, Dong Wang, "Ap18-olr challenge: Three tasks and their baselines," in *Proc. APSIPA*. IEEE, 2018.

[5] Z. Tang, D. Wang, and L. Song, "Ap19-olr challenge: Three tasks and their baselines," in *Proc. APSIPA*. IEEE, 2019, pp. 1917–1921.

[6] 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," *arXiv preprint arXiv:2006.03473*, 2020.

[7] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint 2005.08100*, 2020.

[8] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," in *Proc. Conference on Uncertainty in Artificial Intelligence*, 2018, pp. 876–885.

[9] M. Mclaren, M. K. Nandwana, D. Castán, and L. Ferrer, "Approaches to multi-domain language recognition." in *Proc. Odyssey*, 2018, pp. 90–97.

[10] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[11] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken language recognition using x-vectors." in *Proc. Odyssey*, 2018, pp. 105–111.

[12] D. Talkin and W. B. Kleijn, "A robust algorithm for pitch tracking (rapt)," *Speech coding and synthesis*, vol. 495, p. 518, 1995.

[13] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, and J. R. Deller Jr, "Approaches to language identification using Gaussian mixture models and shifted delta cepstral features," in *Proc. Interspeech*, 2002.

[14] R. Fér, P. Matějka, F. Grézl, O. Plchot, and J. Černockỳ, "Multilingual bottleneck features for language recognition," in *Proc. Interspeech*, 2015.

[15] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, pp. 369–376.

[16] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *CoRR*, vol. abs/1808.06226, 2018.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proc. CVPR*, 2019, pp. 4690–4699.

[19] R. Duroselle, D. Jouvet, and I. Illina, "Unsupervised regularization of the embedding extractor for robust language identification," in *Proc. Odyssey*, 2020.

[20] ——, "Metric learning loss functions to reduce domain mismatch in the x-vector space for language recognition," in *Proc. Interspeech*, 2020.

[21] N. Brümmer, "The FoCal toolkit," in *website https://sites.google.com/site/nikobrummer/focalmulticlass*, 2007.