TalTech Systems for the OpenASR20 Challenge

Tanel Alumäe
Laboratory of Language Technology
Tallinn University of Technology
Tallinn, Estonia
tanel.alumae@taltech.ee

***Abstract*—This paper describes the Tallinn University of Technology systems built for the OpenASR20 challenge. We participated in the constrained data track for all ten languages of the challenge. Our models use the hybrid DNN-HMM architecture. For all languages, the final systems rely of two acoustic models that use the Kaldi "chain" architecture, trained using LF-MMI. We also use adapted maximm entropy language models and recurrent neural network language models for rescoring the results from the first pass. For most languages, we use IARPA BABEL data for language modeling.**

***Index Terms*—OpenASR20, speech recognition**

I. Introduction

Th OpenASR (Open Automatic Speech Recognition) Challenge [1] is organized by NIST. The goal of the challenge is to evaluate speech recognition technologies under low resource constraints.

The 2020 edition of the challenge (OpenASR20) consists of speech recognition tasks for 10 languages. There are two training conditions for all languages: Constrained and Unconstrained. The Constrained condition restricts the acoustic training data only to the provided 10 hour subset of the Build dataset. Additional text data from any public sources may be used for training. The Unconstrained training condition allows using additional publicly available speech and text training data from any language for training the models.

The Tallinn University of Technology (TalTech) team participated in the Constrained training condition of all 10 languages. Our systems for different languages are very similar. Our models are based on the hybrid DNN-HMM approach. We use Kaldi [2] for training acoustic models. For each language, we use two acoustic models, each using the CNN-TDNNF architecture. One acoustic model is trained using multi-condition training and the other using SpecAugment [3]. The lattices generated using the two acoustic models are rescored using adapted maximum entropy language models and two recurrent neural network models, one running in forward direction and the other running in backward direction. The rescored lattices originating from the two acoustic models are finally combined and decoded to one-best hypotheses. We use IARPA BABEL data for additional language model training data for most languages, with the exception of Somali.

II. Detailed Description

A. Training data

Since we participated in the Constrained training condition, the only acoustic data that we used was the 10-hour subset of the Build dataset provided for the language being processed.

TABLE I
IARPA BABEL language packs used for additional textual training data.

Language	Language pack	LDC ID
Amharic	IARPA-babel307b-v1.0b	LDC2019S22
Cantonese	IARPA-babel101b-v0.4c	LDC2016S02
Guarani	IARPA-babel305b-v1.0c	LDC2019S08
Javanese	IARPA-babel402b-v1.0b	LDC2020S07
Kurmanji-Kurdish	IARPA-babel205b-v1.0a	LDC2017S22
Mongolian	IARPA-babel401b-v2.0b	LDC2020S10
Pashto	IARPA-babel104b-v0.4bY	LDC2016S09
Tamil	IARPA-babel204b-v1.1b	LDC2017S13
Vietnamese	IARPA-babel107b-v0.7	LDC2017S01

For language modeling, we used additional training data for all languages. For all languages except Somali, we use the speech transcripts from IARPA BABEL language packs as additional textual training data (see Table I), together with the pronunciation lexicon in the language packs.

Since there is no BABEL language pack for Somali, we used the Somali Web Corpus [4] as additional textual training data for this language. We used the corpus only to extend the language model vocabulary by 500 000 most frequent words. Using the corpus for training the actual language models did not improve language model performance.

B. Decoding pipeline

Figure 1 gives a visual representation of the decoding pipeline. Some certain rescoring steps are omitted for some of the languages (see below).

C. Acoustic Modelling

We uses the hybrid DNN-HMM approach using Kaldi as the main software tool.We trained two acoustic models with identical architecture for each language. The models use the TDNN-F topolology [5] and are trained according to the Kaldi "chain" model training approach [6]. There are three input feature streams: 40-dimensional filterbank features, 100-dimensional i-vectors (updated every 10 milliseconds) and 3-dimensional pitch features. The i-vector and pitch features are transformed into spatial 40-dimensional planes (five for i-vector features, one for pitch features) using learned linear layers and combined with the filterbank features. The resulting seven $40 \times T$ planes (where T corresponds to the time dimension) are first processed using six 3×3 convolutional layers, each using the ReLU activation function. The output

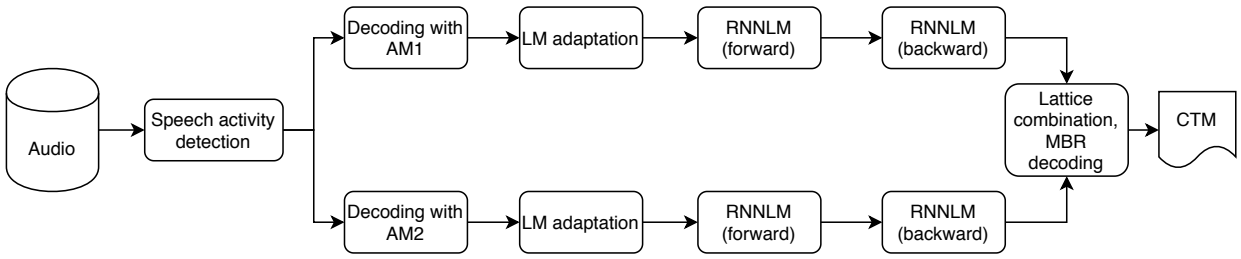

Fig. 1. Workflow of the decoding pipeline.

from the convolutional block is processed by nine TDNN-F layers. We used i-vectors extracted in online mode for training and decoding.

For each language, we trained two acoustic models: one on clean speed-perturbed data and the other on noise-augmented data. SpecAugment was used during the training of the first model, and the model was trained for 20 epochs. For noise augmentation, we used the standard multi-condition training approach implemented in Kaldi [7]: four copies were made from the clean speed-perturbed data, and the copies were reverberated, mixed with background noise, music, or babble noise, respectively. Noises from the MUSAN corpus [8] were used for augmentation. Since there was now four times more training data, the second model was trained for only five epochs.

For the nine languages that had IARPA BABEL language packs, the pronunciation lexicons are produced from the corresponding language pack lexicons. We used the provided lexicons with the following minor changes:

- Amharic: all the labialized consonants were split into two: the main phoneme and a labial pseudo-phoneme (e.g., $p^w \rightarrow p\ w$);
- Kurmanji-Kurdish: we removed the distiction between phonemes in stressed and unstressed syllables.

For Somali, we used the provided pronunciation lexicon in the Build dataset for training a grapheme-to-phoneme model using Phonetisaurus [9] which was then used for generating pronunciations for words from the external text corpus.

D. Language Modelling

Language model used for decoding is a maximum entropy 4-gram model trained using SRILM [10], [11].

The lattices from the first decoding pass are rescored using language models adapted to the given conversation side. This is done similarly to the method described in [12]. The language model training data is divided into "documents", representing one conversation side. Then, the hypotheses from the first decoding pass are used to find conversations in the training data that are most useful for increasing the unigram perplexity of the first pass hypotheses of the individual conversations. In other words, we look for such documents in the training corpus that produce a language model that improve perplexity of the conversation transcripts, when applied in interpolation with the background model. For each conversation side being processed, we select all such documents from the training corpus that increase the perplexity of the first pass transcripts. The set of selected documents is then used for adapting the "background" unadapted maximum entropy 4-gram language model for each conversation side. The adaptation is performed as described in [13]: during optimization of the parameters for a certain conversation, the parent model was taken as a prior. This method encourages the domain-specific models to have feature weights close to the prior model using regularization, if there is little evidence to change them. This was done using the SRILM extension for maximum entropy language models [11].

The lattices that rescored using adapted language models are further rescored using two recurrent neural network language models (RNNLMs), one running in forward direction and the other running in backward direction. The RNNLMs are trained using Kaldi [14] and consist of two LSTM layers with the cell dimensionality of 200. Word embedding dimensionality is also 200. The so-called Backstitch optimization method [15] was used for training RNNLMs, with the exception of some languages (Mongolian and Vietnamese) for which Backstitch caused the RNNLMs not to converge. For those languages, RNNLMs were trained without Backstitch. For Amharic, the RNNLMs did not improve the recognition results at all on development data and we didn't use them for rescoring evaluation data.

E. Speech Activity Detection

Since the evaluation data is not segmented into utterances, we trained a speech activity detection model to detect regions of speech in the test data, using the implementation in Kaldi. This approach first trains a GMM speech recognition model on the provided clean training data, and then combines the labels from the alignment with the GMM model with default non-speech labels for unlabeled regions of the training data. The resulting training data is augmented with reverberation and noise perturbation, and the final TDNN-based speech activity detection model is trained. The model uses statistics pooling for incorporating long-range information.

F. Runtime Performance

We performed all the training on a single server with 44 CPUs, 384 GB of RAM and seven NVidia P100 GPUs. Training all the models for a single language is done in around 10 hours, and we never use more than 3 GPUs in parallel.

TABLE II
WORD ERROR RATES (%) FOR ON GUARANI AND JAVANESE DEVELOPMENT SETS AFTER INDIVIDUAL DECODING PHASES.

	Guarani		Javanese	
Acoustic model	SpecAugment	Multi-condition	SpecAugment	Multi-condition
1st pass	44.0	44.1	56.2	56.1
+ Language model adaptation	43.6	43.9	55.7	55.7
+ RNNLM (forward)	42.1	42.3	54.9	54.9
+ RNNLM (backward)	41.8	41.9	55.2	55.2
Combination	40.3		53.7	

It should be possible to complete the training in only a few hours after modifying the training pipeline to train all models in parallel.

Running the decoding pipeline on the 10-hour evaluation sets of different languages took from 32 minutes (Guarani) to one hour and 35 minutes (Vietnamese), measured in wall clock time. The total CPU time per decoding run ranged from 8 to 40 hours. Those numbers however are not exactly comparable since there were other processes running on the same server. Nevertheless, the large differences turn out to be mostly caused by the different densities of the first pass decoding lattices: the lattices of the Guarani evaluation data are more than two times smaller than the Vietnamese lattices. The differences in lattice densities have a large effect on the complexities of the rescoring steps, resulting different execution speeds. The large differences in lattice densities are probably caused by multiple factors, such as the choice of language modeling units (e.g., Vietnamese transcripts in the BABEL data use a tokenization scheme with short morpheme-like units) and acoustic conditions of the test data. The maximum memory consumption per process during decoding run was around 4.6 GB. GPUs were not used during decoding.

III. RESULTS

A. Impact of Individual Decoding Steps

Table II shows the word error rate (WER) after each decoding step for two languages, Gurarani and Javanese. For most languages, the trend was similar to those two languages: first pass decoding using either of the two acoustic models resulted in similar WERs, although the absolute WERs across languages were very different. Lattice rescoring and combination resulted in 5-10% relative improvement, with regard to the first pass results.

B. Final results

Table III lists WERs for development and evaluation sets across all languages. Development set results are taken from our internal decoding and scoring runs, while the results on the evaluation data originate from the official leaderboards.

It can be seen that the absolute differences between the WERs of the individual languages are big, with Amharic and Vietnamese giving the best results and Tamil and Kurmanji-Kurdish giving worst results.

TABLE III
FINAL WORD ERROR RATES (%) FOR DEVELOPMENT AND EVALUATION DATA OF DIFFERENT LANGUAGES.

Language	Dev	Eval
Amharic	37.0	45.1
Cantonese	47.2	45.4
Guarani	40.3	46.6
Javanese	53.7	53.8
Kurmanji-Kurdish	63.5	65.3
Mongolian	48.0	47.3
Pashto	43.6	45.7
Somali	57.1	59.1
Tamil	62.5	65.1
Vietnamese	45.2	45.1

IV. CONCLUSION

This paper described the TalTech systems developed for the OpenASR20 challenge. We participated only in the Constrained training conditions of all challenge languages. For most languages, we used the IARPA BABEL language packs as additional sources of language modeling data. For decoding, two CNN-TDNNF based Kaldi "chain" models are used, trained with different data augmentation strategies. Language model adaptation, rescoring with forward and backward RNNLMs and combining the lattices from two decoding runs is found to improve the first pass results of a single acositic model by 5 to 10% relative.

REFERENCES

[1] "OpenASR20 challenge evaluation plan," https://www.nist.gov/system/files/documents/2020/10/21/OpenASR20_EvalPlan_v1_5.pdf, accessed: 2020-11-23.

[2] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *ASRU*, 2011.

[3] 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," *Interspeech*, pp. 2613–2617, 2019.

[4] V. Suchomel and P. Rychlý, "Somali web corpus," 2016, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. [Online]. Available: http://hdl.handle.net/11234/1-2591

[5] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks," *Interspeech*, 2018.

[6] H. Hadian, D. Povey, H. Sameti, J. Trmal, and S. Khudanpur, "Improving LF-MMI using unconstrained supervisions for ASR," in *SLT*, 2018.

[7] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *ICASSP*, 2017, pp. 5220–5224.

[8] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv e-prints*, 2015.

[9] J. R. Novak, N. Minematsu, and K. Hirose, "Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the WFST framework," *Natural Language Engineering*, vol. 22, no. 6, pp. 907–938, 2016.
[10] A. Stolcke, J. Zheng, W. Wang, and V. Abrash, "SRILM at sixteen: Update and outlook," in *ASRU*, vol. 5, 2011.
[11] T. Alumäe and M. Kurimo, "Efficient estimation of maximum entropy language models with n-gram features: An SRILM extension," in *Interspeech*, 2010.
[12] T. Alumäe and K. Kaljurand, "Maximum entropy language model adaptation for mobile speech input," in *Interspeech*, 2012.
[13] C. Chelba and A. Acero, "Adaptation of maximum entropy capitalizer: Little data can help a lot," *Computer Speech & Language*, vol. 20, no. 4, pp. 382–399, 2006.
[14] H. Xu, K. Li, Y. Wang, J. Wang, S. Kang, X. Chen, D. Povey, and S. Khudanpur, "Neural network language modeling with letter-based features and importance sampling," in *ICASSP*, 2018, pp. 6109–6113.
[15] Y. Wang, V. Peddinti, H. Xu, X. Zhang, D. Povey, and S. Khudanpur, "Backstitch: Counteracting finite-sample bias via negative steps," in *Interspeech*, 2017.