TalTech Systems for the OpenASR21 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 OpenASR21 challenge. We participated in the constrained and constrained-plus data track for all fifteen languages of the challenge, focusing only on the case-insensitive subtracks. Our models use the hybrid DNN-HMM architecture. For all languages, the final systems of the constrained track rely on three acoustic models that use the Kaldi "chain" architecture, trained using LF-MMI. We also use adapted maximum entropy language models, transformer-based 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. In the constrained-plus track, we use an additional acoustic model that uses input features generated by the XLSR-53 wav2vec2.0 model, reduced to 80 dimensions using a language-specific LDA transform.**

***Index Terms*—OpenASR21, speech recognition**

I. Introduction

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

The 2021 edition of the challenge (OpenASR20) consists of speech recognition tasks for 15 languages. There are three training conditions for all languages: Constrained, Constrained-plus, 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 Constrained-plus condition has the same training data restrictions as the Constrained condition, but additionally allows the use of publicly available and previously existing speech pretrained models. The Unconstrained training condition allows using additional publicly available speech and text training data from any language for training the models. In general, the test data for all languages includes conversational telephone speech that is scored (where applicable) case-insensitively. For thee languages, there is also a additional multi-genre tests sets that are scored case-sensitively.

The Tallinn University of Technology (TalTech) team participated in the Constrained and Constrained-plus training condition of all 15 languages and in the Unconstrained condition for one language. We submitted the results only of the case-insensitive scoring track. Our systems for different languages are very similar. Our models are based on the hybrid DNN-HMM approach. We use Kaldi [1] for training acoustic models. For each language, we use three acoustic models – two with a CNN-TDNNF architecture and one with a CNN-BLSTM architecture. The lattices generated using the three acoustic models are rescored using adapted maximum entropy language models, a Transformer-based language model and a backward recurrent neural network model. 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 and Farsi. In the Constrained-plus track, we use an additional CNN-TDNNF acoustic model that uses input features generated by the pretraine multilingual XLSR-53 wav2vec2.0 model, reduced to 80 dimensions using a language-specific LDA transform.

II. Description of the System for the Constrained Condition

A. Training data

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

For language modeling, we used additional training data for all languages. For all languages except Somali and Farsi, 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.

For Somali, we used the Somali Web Corpus [2] 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 three acoustic models for each language in the Constrained condition. Two models use the TDNN-F topolology [3] and are trained according to the Kaldi "chain" model training approach [4]. There are three

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
Georgian	IARPA-babel404b-v1.0a	LDC2016S12
Javanese	IARPA-babel402b-v1.0b	LDC2020S07
Kazakh	IARPA-babel302b-v1.0a	LDC2018S13
Kurmanji-Kurdish	IARPA-babel205b-v1.0a	LDC2017S22
Mongolian	IARPA-babel401b-v2.0b	LDC2020S10
Pashto	IARPA-babel104b-v0.4bY	LDC2016S09
Swahili	IARPA-babel202b-v1.0d	LDC2017S05
Tagalog	IARPA-babel106-v0.2g	LDC2016S13
Tamil	IARPA-babel204b-v1.1b	LDC2017S13
Vietnamese	IARPA-babel107b-v0.7	LDC2017S01

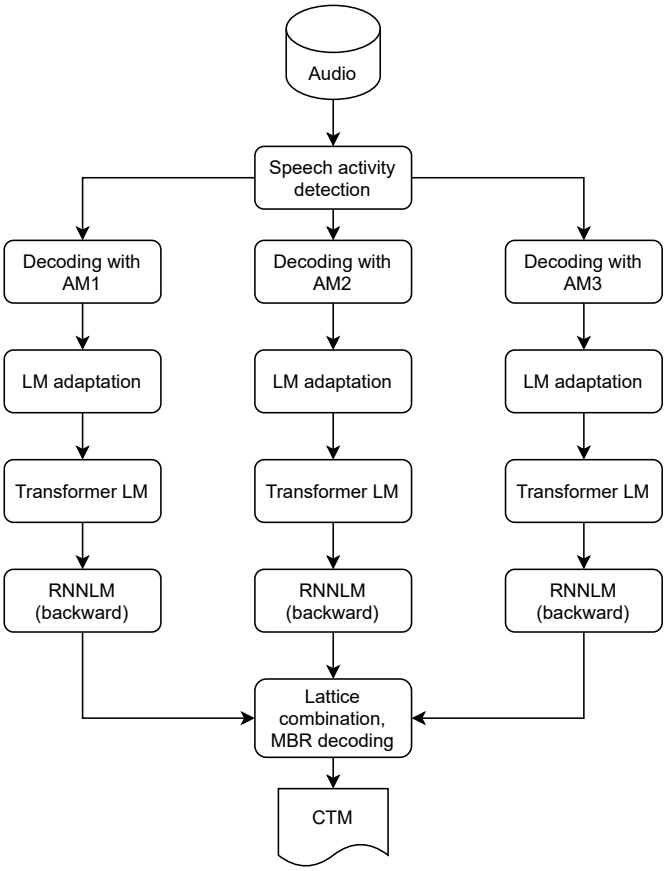

Fig. 1. Workflow of the decoding pipeline.

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 from the convolutional block is processed by nine TDNN-F layers. We used i-vectors extracted in online mode for training and decoding. The "backstitch" regularization method [5] was used during training.

For each language, we trained two CNN-TDNNF acoustic models: one on clean speed-perturbed data and the other on noise-augmented data. SpecAugment [6] was used during the training of the first model, and the model was trained for 30 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 eight epochs.

The third acoustic model (CNN-LSTM) replaces TDNN-F blocks with two LSTM layers and an attention-based local pooling layer, each separated by three fully connected layers. The CNN-LSTM model is trained on noise augmented training data for four epochs.

For the 13 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. For Farsi, we used the pronunciation lexicon that was provided with the Build dataset.

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 neural network language models. The first one used the Transformer encoder architecture and is trained using Pytorch. It constists of 6 Transformer layers. Word embedding and Transformer hidden layer dimensionalities are set to 512. The number of "heads" in the multi-head attention layers is two. Dropout (0.3) and label smoothing (0.1) are used for regularization. The other neural network LM is a recurrent neural network language model trained using Kaldi [14] and it is applied in backward direction. It consist of two LSTM layers with the cell dimensionality of 200. Word embedding dimensionality is also 200. The "backstitch" regularization method [5] 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 neural network LMs 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 of the Constrained track for a single language is done in around 15 hours, and we never use more than 3 GPUs in parallel. 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 (Swahili) to two hours (Somali), measured in wall clock time. The total CPU time per decoding run ranged from 10 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 Georgian. For most languages, the trend was similar to this language: first pass decoding using either of the CNN-TDNNF acoustic models resulted in similar WERs, although the absolute WERs across languages were very different. The CNN-LSTM model resulted in consistently higher WERs than CNN-TDNNF on all languages, yet it improved the combined results. 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, together with our results from the 2020 challenge. 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 Georgian and Swahili giving the best results and Tamil and Farsi giving worst results.

The improvement with regard to our 2020 results range from 0.9 (Tamil) to 2.7 (Cantonese) percentage points. The main differences of our current system to the 2020 system are the use of three acoustic models (instead of two), the application of backstitch regularization when training acoustic models, and the replacement of a forward RNNLM with a Transformer LM.

IV. Description of the System for the Constrained-plus Condition

The Constrained-plus training condition allows the use of publicly available pretrained models, given that they were not trained on labelled speech of the target language. We experimented with the XLSR-53 wav2vec2.0 model [15]. XLSR-53 is a large pretrained model trained on unlabeled multilingual data. The model is trained by jointly solving a contrastive task over masked latent speech representations and learning a quantization of the latents shared across languages. The model contains a convolutional feature encoder that maps raw audio to latent speech representations which are fed to a Transformer network that outputs context representations. XLSR-53 is pretrained on 56 000 hours of speech data from

TABLE II
WORD ERROR RATES (%) FOR GEORGIAN AFTER DIFFERENT DECODING PHASES

Acoustic model	CNN-TDNNF	CNN-TDNNF	CNN-LSTM
Augmentation	SpecAugment	Multi-condition	
1st pass	41.3	41.7	44.3
+ Language model adaptation	40.9	41.4	43.9
+ Transformer LM (forward)	40.7	41.2	43.8
+ RNNLM (backward)	40.6	40.9	43.5
Combination		38.5	

TABLE III
FINAL WORD ERROR RATES (%) FOR DEVELOPMENT AND EVALUATION DATA OF DIFFERENT LANGUAGES. OUR 2020 WERS ARE GIVEN FOR COMPARISON.

	Constrained				Constrained-plus	
	2020		2021		2021	
Language	Dev	Eval	Dev	Eval	Dev	Eval
Amharic	37.0	45.1	35.5	43.2	33.2	41.5
Cantonese	47.2	45.4	43.6	42.7	42.1	41.0
Georgian			38.5	41.6		38.9
Farsi			52.3	81.9		79.1
Guarani	40.3	46.6	39.7	45.2		43.7
Javanese	53.7	53.8	52.2	52.7	50.0	50.6
Kazakh			45.2	53.0		50.6
Kurmanji-Kurdish	63.5	65.3	63.4	64.4	61.1	62.1
Mongolian	48.0	47.3	45.6	45.3		43.7
Pashto	43.6	45.7	43.5	45.3		42.5
Somali	57.1	59.1	56.5	58.7	54.7	56.9
Swahili			33.2	35.0		38.9
Tagalog			40.5	41.9		40.2
Tamil	62.5	65.1	61.7	64.2		62.8
Vietnamese	45.2	45.1	43.5	43.1		40.1

53 languages. Pretraining data includes 650 hours of BABEL training data from 10 languages (Bengali, Canotonese, Kazakh, Haitian, Kurmanji-Kurdish, Pashto, Tamil, Turkish, Tok Pisin and Vietnamese).

Usually, XLSR-53 is used in speech recognition by finetuning it on labelled training data using the CTC objective. We took an alternative approach: the 1024-dimensional outputs of the non-finetuned XLSR-53 model were first transformed to 80 dimensions, using an LDA transform trained on a very small subset (1000 utterances) of training data together with the corresponding senone alignments. Then, the 80-dimensional XLSR-53 features of the noise-augmented ans speed-perturbed training data were dumped to disk and a Kaldi CNN-TDNNF model was trained on top of the features. This model didn't use i-vectors.

The lattices generated based on the XLSR-53 features were postprocessed by the same rescoring steps as in our Constrained system. Finally, the lattices from the CNN-TDNNF models of the Constrained system were combined with the lattice of the XLSR-53 based system, using a weight of 0.5 for the latter.

The results of our submission in the Constrained-plus condition are also listed in Table III.

The decoding using the XLSR-56 based model is done on the CPU. The computationally most expensive part of this process is the extraction of XLSR-53 features which runs in roughly realtime speed with regard to the duration of an utterance. The overhead of using the XLSR-53 based model, in comparison to the Constrained system, is thus around 15 hours in total CPU time, given the evaluation set of 10 hours.

V. DESCRIPTION OF THE SYSTEM FOR THE UNCONSTRAINED CONDITION

In the Unconstrained training condition, we submitted a result only for Amharic. The system is described in detail in [16]. It is a combination of two models trained on the BABEL Amharic data: a Kaldi CNN-TDNNF system, together with language model adaptation and rescoring steps, and the XLSR-53 model, finetuned to Amharic using CTC. The output is obtained by optimized N-best list combination of the two systems. It resulted in a WER of 28.9% on the development set and 35.2% on the evaluation data.

VI. CONCLUSION

This paper described the TalTech systems developed for the OpenASR21 challenge. We participated in the Constrained and Constrained-plus 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, three Kaldi "chain" models (2x CNN-TDNNF and 1x CNN-LSTM) were used, trained with different data augmentation strategies. Language model adaptation, rescoring with Transformer and (backward) RNNLM and combining the lattices from three decoding acoustic models was found to improve the first pass results of a single acoustic model by 5 to 10% relative.

VII. ACKNOWLEDGEMENTS

The work was partly done within the project "ICT programme", supported by the European Union through the European Social Fund. The authors acknowledge the TalTech supercomputing resources made available for conducting the research reported in this paper.

REFERENCES

[1] 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.

[2] 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

[3] 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.

[4] H. Hadian, D. Povey, H. Sameti, J. Trmal, and S. Khudanpur, "Improving LF-MMI using unconstrained supervisions for ASR," in *SLT*, 2018.
[5] Y. Wang, V. Peddinti, H. Xu, X. Zhang, D. Povey, and S. Khudanpur, "Backstitch: Counteracting finite-sample bias via negative steps," in *Interspeech*, 2017.
[6] 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.
[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] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," *arXiv preprint arXiv:2006.13979*, 2020.
[16] T. Alumäe and J. Kong, "Combining hybrid and end-to-end approaches for the OpenASR20 Challenge," in *Interspeech*, 2021.