Improvements in Language Modeling, Voice Activity Detection, and Lexicon in OpenASR21 Low Resource Languages

Vishwa Gupta(✉) and Gilles Boulianne

Centre de Recherche Informatique de Montréal (CRIM), Quebec, Canada
{vishwa.gupta,gilles.boulianne}@crim.ca

Abstract. OpenASR21 evaluation was on 15 low resource languages and 3 case sensitive languages. During the evaluation, participants got significant reduction in word error rates (WER) with text downloaded from the internet for only the case sensitive languages, since the development and evaluation audio contained broadcast news. For the 15 low resource languages, participants showed only small gains for some of the languages. The reason is that the development and test set contain dialog between two people, which is very different from the primarily news texts and web pages available over the internet. Here, we show that training text translated from other OpenASR21 languages reduces the WER for many languages. During the evaluation, one team added words to the lexicon using a 3-gram phone language model, but they do not show what WER reduction they achieve. We show that adding new words in the lexicon from public text is beneficial for languages where the out-of-vocabulary rate is high, and outline conditions for reducing the WER. Adding an attention layer to the TDNN (time delay neural net) based voice activity detector reduced the WER for 17 out of the 18 languages. With all the improvements combined, we are getting lower word error rate for the development set for three languages (Farsi, Kazakh and Tamil) than the site that achieved the best error rate for all the languages during the evaluation period.

Keywords: OpenASR21 · Low-resource · Speech recognition · Language modeling

1 Introduction

The OpenASR21 (Open Automatic Speech Recognition 2021) Challenge set out to assess the state of the art of ASR technologies under low-resource language constraints [11]. The task consisted of performing ASR on audio datasets in up to 15 different low-resource languages and 3 languages with case sensitive scoring, to produce the recognized text. Ten languages were carried over from the OpenASR20 challenge [10], and five new languages were added. A case sensitive scoring was also added for three of these languages: Kazakh, Swahili and Tagalog. In case-sensitive scoring, words capitalized differently from the reference

A. Karpov et al. (Eds.): SPECOM 2023, LNAI 14339, pp. 73–86, 2023.
https://doi.org/10.1007/978-3-031-48312-7_6

transcript will not count as a match. Case-sensitive scoring is used as a proxy for evaluating ASR performance on proper nouns.

We took part in the constrained condition for all the 15 languages and the 3 languages with case sensitive scoring. In the constrained condition, only a 10-hour audio Build dataset for that language can be used for training acoustic models. Additional text data, either from the Build dataset or publicly available resources, can be used for training the language model in the constrained condition. Any such additional training text must be specified in detail in the system description. In the constrained condition, no pre-trained large models were allowed.

A good overview of OpenASR20 is given in [10]. In OpenASR20, two teams achieved very good results [1,17]. They used larger training text and lexicon from Linguistic Data Consortium (LDC) corpora for training language models (LM) and using a larger lexicon. These LMs and larger lexicon reduced the word error rate (WER) significantly for each language.

For OpenASR21 see [11] for a good overview. The team from USTC/iFlytek Research [19] achieved the lowest word error rate (WER) for all the 15 languages. Their WER was significantly lower than any other participant for all the languages. For acoustic modeling, they used TTS to generate additional audio for training either from public text or the Babel training text. This gave them an additional 1.3% average WER reduction for the 15 languages. They also interpolated language model (LM) from LDC text with LM from public text (publicly available text downloaded from internet). This interpolated LM gave them better results for 3 of the 7 languages. They also rescored the decoded lattices with bidirectional LSTMP (LSTM with a recurrent projection layer) [14] language model from public text. This LM was fine tuned with the LDC training text before rescoring. Note that, the leading teams in the OpenASR21 evaluation used hybrid DNN-HMM systems rather than end-to-end systems, since the end-to-end systems perform poorly with only 10 h of audio.

In [18], they do not use any publicly available text for either decoding or for rescoring with LSTM LM. They use a larger lexicon for the 15 languages and 3 case sensitive languages. They generate a 3-gram phone language model from the lexicon, and then generate 12 million sequences from this LM, and keep 1 million most likely sequences not in the lexicon. Words corresponding to these phone sequences are generated using G2P (grapheme to phoneme) methods to augment the lexicon. But they do not show the benefit of using such a large lexicon.

In [4], the authors use three different features (MFCCs, MFCCs+Conformer embeddings, MFCCs + voice activity detector embeddings) to generate 3 different acoustic models for later fusion. They also used publicly available text for language modeling. Instead of generating separate LM from LDC and public text and then interpolating the two [19], they first filter the public text with sentence selection to match the sentences in the LDC training text, and then generate an LM from the combined LDC + filtered public text. The filtered public text is about the same size as the LDC text. This resulted in reduced WER (word error rate) for many languages.

However, the reduction in WER with the added public text was small. The primary reason for this small reduction is that the public text comes mostly from news and web pages, while OpenASR21 contains conversations between two people. So the question is how can we add text corresponding to conversations. Here we introduce a method to generate synthetic conversational text and augment data for a language of interest, using a back-translation approach, and obtain small reduction in WER for that language. Although back-translation has been used to improve machine translation models [15], here it is used for the first time to improve a monolingual model for ASR (to the best of our knowledge).

During the evaluation period, we had used a voice activity detector (VAD) [4] to remove noise segments. This VAD, as outlined in Chime6 track2 speech activity detection[1] had 5 TDNN layers and 2 layers of statistics pooling [2] with an added specAugment layer [8]. We propose to add an attention layer, similar to the one outlined for ASR in [13]. The speech/non-speech segmentation with this attention layer results in small reductions in WER for most languages. The reason is that an attention layer provides longer term context and it improves discrimination between segments with long duration. Silent and speech segments fall in that category.

We also show that increasing the lexicon with new words from public text is effective for languages where OOV (out of vocabulary) rate is high.

As the NIST scoring server for the evaluation set is closed, we only show comparative results for the development set. Through all the improvements in [4], we showed that we got lower WER for the eval set for Tamil than any participant during the evaluation period. In this paper, we show that with all the improvements combined, we get lower single decode WER for development set for Farsi, Kazakh and Tamil than the best team during evaluation [19].

2 Dataset

In the constrained condition, for acoustic model training, we only used the 10-hour Build data set provided by NIST for the language being processed, with corresponding transcripts in UTF-8 encoding. Training and development lexicons were also provided by NIST.

For the 13 languages with LDC packs (all the languages except for Farsi and Somali), we used the expanded lexicon and text provided in those packs. For example, the training text in the OpenASR21 build dataset varies from 66k words for Kazakh to 126k words for Vietnamese, while the training text in the LDC packs varies from 270k words for Kazakh to 989k words for Vietnamese. Overall, the LDC training text is between 4 times and 8 times larger than the text in the OpenASR21 build. The lexicon in the LDC packs is also much larger than the lexicon in the OpenASR21 build. For example, the number of words for Vietnamese in the OpenASR21 lexicon is 3.2k, while in the LDC lexicon there are 6.4k words. For these reasons, training a language model from the training

[1] https://chimechallenge.github.io/chime6/track2software.html.

text and lexicon in the LDC packs reduces the word error rate significantly for all the 13 languages with the LDC packs.

3 ASR Approach

Our system is a hybrid HMM-DNN based on WFSTs (Weighted Finite-State Transducers) and trained with the Kaldi toolkit [12]. During the evaluation period, we trained three different acoustic models for decoding the dev and eval sets so that we could combine the multiple results after decoding [4]. In this paper, we just use one set of acoustic models with 40-dim MFCC's and 2-stream TDNN-F architecture (for multi-stream architecture see [5]) for decoding the development (dev) set (we cannot use the eval set since the scoring server at NIST is closed). These models are the same as those used during evaluation. The idea here is to test new algorithms to reduce WER using improved language modeling, better voice activity detector models, and larger vocabulary.

3.1 Enhanced Voice Activity Detector

During the OpenASR21 evaluation [4], we used two voice activity detectors to segment development and evaluation audio into speech/nonspeech segments: one based on DNN-HMM architecture, and another based on TDNN architecture [9] as outlined in Chime6 track2 speech activity detection[2]. This VAD TDNN has 40-dim MFCC features as input, followed by 3 TDNN layers, followed by 2 layers of statistics pooling [2], followed by 2 TDNN layers. During the OpenASR21 evaluation, we showed that adding a specAugment layer after the input layer to this VAD-TDNN results in lower WER for most of the languages [4]. Here we show that by adding an attention layer [13], we can reduce the WER even more for most of the languages. The attention layer is added after the 5th TDNN layer (see Fig. 1). The attention layer has 12 heads, a value dimension of 60, key dimension of 40, and the number of left and right inputs are 5 and 2 respectively. The attention layer has a wider context (108 frames of left context and 69 frames of right context) and it is able to improve speech/non-speech discrimination as speech and non-speech segments are longer in duration. All the TDNN layers have an output dimension of 256. Table 1 compares WER with/without the attention layer in one back-to-back comparison. Except for Swahili, the WER goes down for the other languages. So the attention layer consistently gives lower WER. Overall, the WER is reduced on average by 0.3% absolute. The largest WER reduction is for Cantonese and Georgian (0.7% absolute).

3.2 Enhanced Lexicon

We carried out many experiments to see if adding new words from publicly downloaded text will reduce word error rates. Public text was heavily filtered

[2] https://chimechallenge.github.io/chime6/track2software.html.

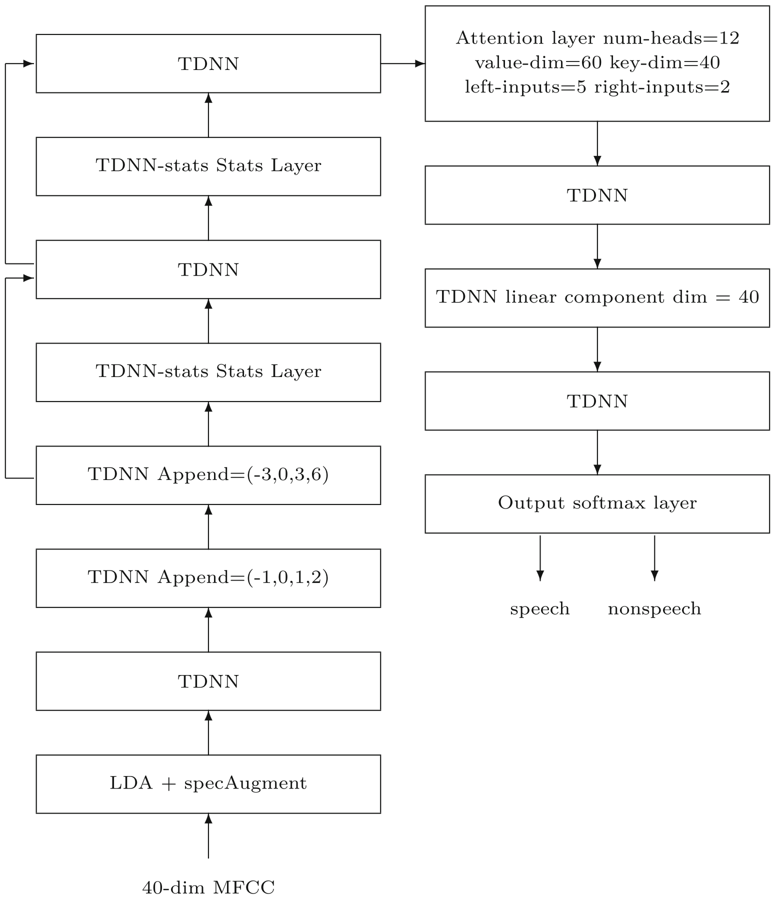

Fig. 1. *VAD TDNN with attention. All the TDNN layers have a dimension of 256.*

by sentence selection to be similar to the LDC training text [4], and the filtered text was about the same size as the LDC text (see Language Model Sec. 3.3). We added new words in this filtered public text to the lexicon. We tried adding frequency one and frequency two words (words not in LDC lexicon occurring at least 2 times in the public text) to the lexicon. To be consistent, we ran all the decoding experiments in this paper with the same acoustic models: LF-MMI (lattice free MMI) training followed by discriminative training of 2-stream acoustic models with 40-dim MFCC [4].

One question is how do we add pronunciations for the new words? We know that the training and test audio are very well transcribed, including the lexicon in the LDC build. The LSP (language specific peculiarities) file in the LDC build contains details about dialects of speakers, any special handling of spelling, character set used for orthographic transcription, romanization scheme, word boundary detection, where the transcribers are from, etc. So the text is

Table 1. WER for the dev set segmented with / without attention layer in VAD-TDNN. CSS stands for case sensitive scoring. (No LSTM LM rescoring is done.)

Lang	no att	att	Lang	no att	att
Amharic	38.0	**37.7**	Mongolian	48.0	**47.9**
Cantonese	47.3	**46.6**	Pashto	48.1	**47.9**
Farsi	52.6	**52.4**	Somali	59.2	**58.9**
Georgian	41.4	**40.7**	Swahili	**36.4**	36.5
Guarani	42.4	**42.3**	Tagalog	44.6	**44.2**
Javanese	53.3	**53.0**	Tamil	61.0	**60.9**
Kazakh	47.3	**46.9**	Kazakh css	53.2	**52.8**
Kurmanji	65.5	**65.3**	Swahili css	48.5	**48.4**
Vietnamese	48.6	**48.0**	tagalog css	47.8	**47.4**

probably quite consistent in transcription, and the words in the LDC lexicon are probably transcribed in a consistent manner. So we did not want to change the transcriptions in this lexicon. This lexicon is transcribed using X-SAMPA[3] phoneme set. To this LDC lexicon, we added pronunciations for the new words in X-SAMPA using two possible scenarios. In one scenario, we train a G2P [7] (grapheme-to-phoneme) from the LDC lexicon (or OpenASR21 lexicon for Farsi and Somali), and then transcribe the new words with this G2P. The second scenario is to use existing G2P from LanguageNet [6] to transcribe the new words. The advantage with LanguageNet is that there are finite state transducers (FSTs) trained for over 120 languages with a large amount of training data for each language. The problem is that some OpenASR21 languages maybe missing (Farsi is missing from LanguageNet). Another problem with LanguageNet is that words are transcribed in IPA symbols, while the OpenASR21 lexicon is transcribed in X-SAMPA. To overcome this, we transcribed the LDC lexicon with the LanguageNet G2P to get a training lexicon with IPA symbols, and then trained another G2P from it to convert IPA phone sequences into X-SAMPA phone sequences. With this conversion capability, we transcribed new words in IPA using languageNet G2P and then converted phoneme sequences in IPA to phoneme sequences in X-SAMPA. So now we had two enhanced lexicons: one from new words transcribed using G2P trained from LDC lexicon[4], and one from new words transcribed with LanguageNet G2P in IPA symbols and then converted to X-SAMPA phone sequences.

We compared the two lexicon with new words added to the LDC lexicon on Amharic and on Somali. The two lexicons gave very similar WER with a small preference for G2P trained from LDC lexicon. So we used the G2P trained from LDC lexicon for the rest of the languages.

[3] https://en.wikipedia.org/wiki/X-SAMPA.

[4] Note that in many languages, some new words had graphemes that do not occur in the LDC lexicon, so we removed these new words before transcribing with G2P.

Table 2 shows for each language the OOV (out-of-vocabulary) rate for the development (dev) set, and the WER (word error rate) with/without the enhanced vocabulary. The last column in Table 2 shows the vocabulary size with/without enhancement. For Amharic, Cantonese and Guarani, we used words occurring 2 times or more in the public text for adding to the existing LDC or OpenASR21 lexicon. For the rest of the languages, we added all the new words occurring 1 time or more in the public text to the LDC or OpenASR21 lexicon. From the table we see that the major benefit from adding the new words in the lexicon is for languages where the OOV rate is high: Farsi, Somalese, Kazakh css (case sensitive scoring), Swahili css, and Tagalog css. The reason for high OOV rate for Farsi and Somali is that there is no LDC pack for them, so the lexicon derived from the small training text is small. For the case sensitive scoring languages also there is no LDC pack, so the training text is from only 10 h of audio. But the OOV rate is higher also because words in training text are capitalized where necessary, so many words are OOV if the correct case does not occur in the lexicon.

The decoding scenario for Table 2 is that a 4-gram LM is computed with SRILM toolkit from the LDC training text (where available) with/without the enhanced lexicon. The WER shows decoding results with this LM (without rescoring with LSTM LM).

As we can see from Table 2, decoding with larger lexicon results in WER reduction for languages where OOV rate is high: Farsi, Somali, Kazakh CSS, Swahili CSS, and Tagalog CSS. Why should the error rate go down for languages with larger OOV rate and not for languages with small OOV rate, even though the likelihood for OOVs in the language model is small (since they are assigned a low unigram probability as they do not occur in the LDC text)? For large OOV rate, the words in the enhanced lexicon may come up as top choice despite small LM likelihoods and reduce WER. While for languages with low OOV rate, there are many more new words not in the development set, and if these words show up as top choice, then they will increase the WER. That is why the biggest gain is for the case sensitive scoring (CSS) languages as their OOV rate is the highest (between 17.8% and 23.7%), and the OOV rate reduction with enhanced lexicon is the largest (between 7.6% and 9.7%), as shown in Table 2. Also, the audio for case sensitive languages is from broadcast audio, and so there is a better match between publicly available text and the training text.

3.3 Language Model

We will first outline what we have done in language modeling in the past [4] for OpenASR21 evaluation, and then describe the new improvements. For language modeling (LM) in the constrained condition, we could use any text publicly available over the internet. For 13 of the 15 languages, we used the LDC IARPA Babel language packs from 2016 to 2020 (there were no language packs for Farsi and Somali). The larger LDC training text together with the larger lexicon reduced the WER for all the languages. The LDC text is from conversational speech, and it reduces WER for the dev set significantly as can be seen in [4,19].

Table 2. OOV rate before and after enhancement, and WER for dev set for each language without/with enhanced lexicon. The last column shows vocabulary size without/with enhancement.

Lang	OOV%	no enh	with enh	vocab no enh/with enh
Amharic	2.0 / 0.7	**37.7**	38.4	36.9k / 46.5k
Cantonese	1.4 / 0	**46.6**	46.6	19.9k / 21.2k
Farsi	9.5 / 6.0	52.4	**52.3**	3.7k / 14.7k
Georgian	1.1 / 0	**40.7**	42.4	35.2k / 130.7k
Guarani	2.0 / 1.1	**42.3**	42.5	28.0k / 30.9k
Javanese	1.0 / 0.6	**53.0**	53.9	15.5k / 44.0k
Kazakh	0.5 / 0.3	**46.9**	48.4	22.3k / 71.4k
Kurmanji	1.1 / 0.2	**65.3**	65.6	14.4k / 63.5k
Mongolian	0.5 / 0.5	**47.9**	48.9	23.9k / 66.9k
Pashto	0.3 / 0	**47.9**	48.0	18.7k / 77.2k
Somali	9.4 / 7.4	58.9	**58.8**	9.6k / 29.3k
Swahili	3.7 / 0.7	**36.5**	37.7	25.2k / 56.9k
Tagalog	1.0/0.8	**44.2**	45.1	22.6k / 59.1k
Tamil	0.5 / 0.2	**60.9**	62.0	58.4k / 104.0k
Vietnamese	0.3 / 0.2	**48.0**	48.1	6.4k / 26.7k
Kazakh css	23.7/14.0	52.8	**48.0**	15.8k / 69.0k
Swahili css	18.6 / 10.9	48.4	**46.3**	11.3k / 58.1k
Tagalog css	17.8 / 10.2	47.4	**43.9**	11.2k / 59.2k

In [4], we also downloaded a significant amount of publicly available monolingual texts from NewsCrawl[5] and CommonCrawl[6], but this text was primarily from news and web sources, and not from conversational speech. So this text significantly increased the perplexity on the development set for every language. We had to resort to sentence selection [16] to use only sentences close to the training text. Through strong sentence selection, we were able to reduce the overall WER after LSTM LM rescoring for 8 of the 15 languages, and all the case sensitive scoring languages (the audio for CSS languages is from broadcast news) [4].

The team with the lowest WER in the evaluation [19] did not filter the downloaded text, but they interpolated the 4-gram LM from this downloaded text with the 4-gram LM from the LDC training text using the SRILM toolkit[7]. They were able to reduce WER for 3 of 7 languages after decoding with this interpolated 4-gram LM.

So the real question is whether we can get additional conversational text that can reduce the perplexity of most of the languages after 4-gram training? In this

[5] https://www.aclweb.org/anthology/W19-5301.
[6] https://data.statmt.org/cc-100/.
[7] https://www.sri.com/platform/srilm/.

paper, we have experimented with Back-translation in order to get additional independent conversation language model training text for each language. Back-translation is used in machine translation [15] to generate synthetic data in a language with scarce resources. Here, since all languages have conversational speech between two speakers on cell phone, we translated the training text in each language to the text in all the other languages. We then combined all translations to a language of interest for generating a 4-gram language model for that language.

For translation we used the Google translation interface[8] in batch mode. We were able to get translations for 14 of the 15 languages (Google translates Mandarin but not Cantonese), thus augmenting LM training data 13-fold for each of these 14 languages. To prepare transcription texts for translation, we merged utterances from the different channel recordings by concatenating them in chronological order, to recreate the full conversation. The transcribed text does not have punctuation marks. We also removed silence and noise markers from this text. At first we considered each utterance as a sentence, but translation results in English made more sense when we concatenated all utterances of a conversation in a single document. Since punctuation marks are added during translation, we used them to split the translated document back into short utterances, then removed the punctuation marks, to make the text similar to the training transcripts.

We have some examples in English (below) that we generated as spot checks for languages. For example, from Vietnamese to English (before punctuation removal):

> hello, hello sister Kieu, oh baby, I haven't eaten yet, have you scratched there, how do you study and only study is normal? what is that dish? my husband just let it go so slowly wear the pink shirt and wear it while it's okay oh my god what is it oh my god what is it my god what is it that makes me laugh everyone studied at this time but didn't go to school at night, did you go to school tonight,

Similarly, translation from Guarani to English is shown below:

> we'll hear we can talk hello hello hello dear brother what's up what's up brother what are you doing right my friend what's the door i aka dehkansándo yeah and here I'll correct and you what are you doing yeah correct what' e here here I'm looking at the movies

Although we cannot actually confirm it, anecdotal evidence suggests that Google translation uses English as the pivot language[9]. We can see that the translation from one language to English is poor, and when English is translated back to another language, it may still be poorer. But we found that in terms of language modeling, this text reduced perplexity more than using the

[8] https://cloud.google.com/translation-hub.
[9] https://www.teachyoubackwards.com/extras/pivot/.

downloaded news related text from the internet even after strong filtering. We generated three 4-gram's from three sources: LDC alone (LDC), translated text from 13 languages combined (Trans), and the downloaded and filtered text from the internet (Sel). We then generated two interpolated language models: LDC interpolated with Trans (LDC·Trans), and LDC interpolated with Trans and Sel (LDC·Trans·Sel). The "·" symbol is used here for interpolation. The optimal interpolation weights are found by the iterative E-M (expectation-maximization algorithm) estimation of SRILM. The perplexity and word error rate (WER) for the three 4-grams is shown in Table 3. So for example, in Table 3, perplexity for Amharic dev set with language model trained from LDC is 404, and the WER is 37.7%, perplexity with interpolated LDC·Trans is 394, and the WER is 37.6%, while perplexity with the interpolated LDC·Trans·Sel is 393, and the WER is 37.6%. As can be seen from the Table, we were able to reduce WER for 10 of the 14 languages.

Even though differences reported in Table 3 seem small, the test sets are large samples, between 60K and 112K words for each language, so that confidence intervals[10] for these results range from ±0.20% to ±0.28%. The differences are significant at a 95% level for Farsi, Pashto, Swahili and Vietnamese while for the other languages the differences are not so significant.

Table 3. Perplexity (PPL) and word error rate (WER) for the dev set for each language for 4-gram LMs from LDC, LDC·Trans, LDC·Trans·Sel. No LSTM LM rescoring is done.

Lang	LDC	LDC·Trans	LDC·Trans·Sel	Interp. weights
	PPL / WER	PPL / WER	PPL / WER	LDC Trans Sel
Amharic	404 / 37.7%	394 / 37.6%	**393 / 37.5%**	0.885, 0.073, 0.042
Farsi	231 / 52.3%	221 / 52.0%	**221 / 52.0%**	0.823, 0.153, 0.024
Georgian	477 / 40.7%	466 / 40.6%	**464 / 40.6%**	0.884, 0.070, 0.046
Guarani	251 /42.3%	**249** / 42.3%	249 / **42.2%**	0.945, 0.036, 0.019
Javanese	271 / **53.0%**	269 / 53.1%	**269** / 53.2%	0.946, 0.039, 0.015
Kazakh	267 / 46.9%	257 /46.8%	**257 / 46.7%**	0.874, 0.097, 0.029
Kurmanji	174 / 65.3%	**170 / 65.3%**	170 / 65.3%	0.904, 0.096, n/a
Mongol	169 / 47.9%	166 / 47.8%	**164 / 47.7%**	0.897, 0.058, 0.045
Pashto	163 / 47.9%	162 / 47.5%	**161 / 47.4%**	0.931, 0.034, 0.035
Somali	279 / **58.8%**	**261** / 59.0%	261 / 59.0%	0.800, 0.179, 0.021
Swahili	319 / 36.5%	306 / 36.3%	**305 / 36.2%**	0.852, 0.094, 0.054
Tagalog	155 / **44.2%**	152 / 44.5%	**152** / 44.4%	0.899, 0.067, 0.034
Tamil	769 / 60.9%	765 / 60.8%	**763 / 60.8%**	0.951, 0.016, 0.033
Vietnam	144 / 48.0%	143 / 48.0%	**140 / 47.8%**	0.885, 0.013, 0.102

[10] We use an 83% confidence interval computed with the Wilson score binomial interval, so that non-overlapping intervals represent a 95% significant difference between the two results [3].

Due to the poor quality of the translation (language → English → another language), the improvement with the translated text is small. However, if we can somehow find two way conversations in English, then we should do much better. It just happens that the switch board[11] data, the call home[12] data, and the Fisher corpus[13] is just that data. These corpus contain millions of words of text. Maybe translating them may lead to appropriate conversational text that can lead to significant reduction in WER. We can even filter this text to be close to the conversations in the training text, and still have a significant amount of additional training text left over. However, we have not exploited this avenue yet.

The final issue is whether all the above improvements (voice activity detector, enhanced lexicon, and translated text) result in significant reduction in WER after LSTM LM rescoring compared to the previous results with LSTM LM rescoring where LSTM LM was trained from LDC + public text or LDC text alone [4]. What we found was that decoding with LDC·Trans·Sel LM followed by rescoring with LSTM LM trained from LDC + translated + filtered public text resulted in lowest WER for Amharic and Farsi. For Guarani and Kazakh, decoding with LDC·Trans LM and rescoring with LSTM LM trained from LDC + filtered public text gave the lowest WER. For other languages, decoding with LDC LM and rescoring with LDC + public text lead to the lowest WER for the dev set. For the case sensitive languages, decoding with 4-gram LM from LDC + public text with enhanced vocabulary, and rescoring with LSTM LM from LDC + public text resulted in the lowest WER. The best WER on dev set in [4] and with all the improvements in this paper is shown in Table 4. For Kurmanji and Swahili, there is no improvement because enhanced VAD, increased vocabulary, translated text and filtered public text did not contribute to WER reduction. So the conversational LDC text for these languages is probably quite different from the text for other languages, and the translation maybe of poor quality. In Table 4, we also compare our results on the dev set for single decode with those of [19] (Table 1, column 1). We can see that we got lower WER for the dev set for three languages: Farsi, Kazakh, and Tamil. For many other languages, our WER for the dev set single decode is close to that in [19].

We also computed confidence intervals for our improvements in WER in Table 4 in a similar manner as for Table 3. The results after the improvements are 95% significant for 13 out of 18 languages[14] (Amharic, Cantonese, Farsi, Georgian, Kazakh, Pashto, Somali, Tagalog, Tamil, Vietnamese, Kazakh css, Swahili css and Tagalog css). When we make a similar comparison of our best WER with the WER in [19], our WER is better than or same as in [19] for 6 out of 15 languages (Farsi, Javanese, Kazakh, Pashto, Tagalog and Tamil) and worse for 9 out of the 15 languages.

[11] https://catalog.ldc.upenn.edu/LDC97S62.

[12] https://catalog.ldc.upenn.edu/LDC97T14.

[13] https://catalog.ldc.upenn.edu/LDC2004T19.

[14] We assumed confidence intervals in the same range for Cantonese and the three case-sensitive languages as the other languages since they have similar test sizes.

We also tried to fine tune the LSTM LM language model with LDC training text using a small learning rate. But in each case, we only achieved a 0.1% reduction in WER. The major effect was whether LSTM LM was trained with LDC + translated + filtered public text, or LDC + filtered public text. Another important factor was whether the decoded lattices for rescoring with LSTM LM were generated from LM trained with LDC alone, or from LDC·Trans·Sel, or from LDC + public text with enhanced vocabulary (as described in the previous paragraph).

Table 4. WER for the dev set before and after all the improvements in this paper. CSS stands for case sensitive scoring. Numbers in bold show whether WER before or after was significantly lower. Numbers in underline show that the WER in ref [19] was significantly lower.

Lang	before	After	from ref [19]	Lang	Before	After	from ref [19]
Amharic	37.2	**36.1**	<u>35.0</u>	Mongolian	46.4	46.3	<u>45.4</u>
Cantonese	45.6	**45.0**	<u>42.3</u>	Pashto	45.7	**45.3**	45.2
Farsi	51.7	**50.8**	52.4	Somali	58.6	**57.4**	<u>55.9</u>
Georgian	40.3	**39.2**	<u>37.5</u>	Swahili	34.6	34.7	<u>32.3</u>
Guarani	40.9	40.8	<u>39.0</u>	Tagalog	42.8	**42.3**	42.1
Javanese	52.0	51.9	51.9	Tamil	60.3	**59.4**	61.0
Kazakh	45.9	**45.2**	46.1	Kazakh css	51.9	**46.0**	
Kurmanji	64.1	64.1	<u>63.7</u>	Swahili css	47.6	**44.2**	
Vietnamese	47.0	**46.3**	<u>43.9</u>	Tagalog css	46.3	**41.4**	

4 Conclusion

We participated in all the 15 low resource languages and the three languages with case sensitive scoring in the OpenASR21 Challenge for the constrained condition. In the past, use of downloaded public text has shown small reductions in word error rate (WER) primarily due to mismatched domains (conversational speech versus news sources). We show that we can achieve small reductions in WER by translating training text from other languages in OpenASR21 to the target language. The small improvement is possibly due to the quality of the translation. Translation is the way to possibly improve the language models in low resource languages for conversational speech, since a significant amount of conversational text is available in English (for example, switchboard, call home, Fisher corpus etc.).

We show that we can reduce the WER for a DNN-based voice activity detector by adding an attention layer to the DNN architecture. We also show that increasing the vocabulary for languages in OpenASR21 with high out-of-vocabulary rate reduces the WER significantly.

Overall, for 13 of 18 languages, we reduced the WER for the single decode of the dev set when we combine the three enhancements. For Kazakh css by 5.9% (absolute), for Tagalog css by 4.9%, for Swahili css by 3.4%, for Somali by 1.2%, for Amharic and Georgian by 1.1%, for Farsi and Tamil by 0.9%, for Kazakh and Vietnamese by 0.7%, for Cantonese by 0.6%. These WER reductions are significant in the evaluation scenario.

Acknowledgments. The authors would like to thank Ministry of Economy and Innovation (MEI) of the Government of Quebec for the continued support.

References

1. Alumäe, T., Kong, J.: Combining hybrid and end-to-end approaches for the OpenASR20 challenge. In: Proceedings of the Interspeech, pp. 4349–4353 (2021)
2. Ghahremani, P., Manohar, V., Povey, D., Khudanpur, S.: Acoustic modelling from the signal domain using CNNs. In: Proceedings of the Interspeech, pp. 3434–3438 (2016)
3. Goldstein, H., Healy, M.J.R.: The graphical presentation of a collection of means. J. Roy. Stat. Soc.: Ser. A: Appl. Stat. **158**, 175–177 (1995)
4. Gupta, V., Boulianne, G.: CRIM's speech recognition system for OpenASR21 evaluation with conformer and voice activity detector embeddings. In: Prasanna, S.R.M., Karpov, A., Samudravijaya, K., Agrawal, S.S. (eds.) Speech and Computer, pp. 238–251. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-20980-2_21
5. Han, K.J., Pan, J., Tadala, V.K.N., Ma, T., Povey, D.: Multistream CNN for robust acoustic modeling. In: Proceedings of the ICASSP, pp. 6873–6877 (2021)
6. Hasegawa-Johnson, M., Rolston, L., Goudeseune, C., Levow, G.-A., Kirchhoff, K.: Grapheme-to-phoneme transduction for cross-language ASR. In: Espinosa-Anke, L., Martín-Vide, C., Spasić, I. (eds.) SLSP 2020. LNCS (LNAI), vol. 12379, pp. 3–19. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59430-5_1
7. Novak, J.R., Minematsu, N., Hirose, K.: Phonetisaurus: exploring grapheme-to-phoneme conversion with joint n-gram models in the WFST framework. Nat. Lang. Eng. **22**(6), 907–938 (2016). https://doi.org/10.1017/S1351324915000315
8. Park, D.S., et al.: SpecAugment: a simple data augmentation method for automatic speech recognition. In: Proceedings of the Interspeech, pp. 2613–2617 (2019)
9. Peddinti, V., Povey, D., Khudanpur, S.: A time delay neural network architecture for efficient modeling of long temporal contexts. In: Proceedings of the Interspeech, pp. 3214–3218 (2015)
10. Peterson, K., Tong, A., Yu, Y.: OpenASR20: an open challenge for automatic speech recognition of conversational telephone speech in low-resource languages. In: Proceedings of the Interspeech, pp. 4324–4328 (2021)
11. Peterson, K., Tong, A.N., Yu, J.: OpenASR21: The second open challenge for automatic speech recognition of low-resource languages. In: Proceedings of the Interspeech (2022)
12. Povey, D., et al.: The Kaldi speech recognition toolkit. In: Proceedings of the ASRU (2011)
13. Povey, D., Hadian, H., Ghahremani, P., Li, K., Khudanpur, S.: A time-restricted self-attention layer for ASR. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5874–5878 (2018). https://doi.org/10.1109/ICASSP.2018.8462497

14. Sak, H., Senior, A.W., Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling (2014)
15. Sennrich, R., Haddow, B., Birch, A.: Improving Neural Machine Translation Models with Monolingual Data. In: Proceedings of ACL, pp. 86–96 (2016)
16. Sethy, A., Georgiou, P.G., Ramabhadran, B., Narayanan, S.: An iterative relative entropy minimization-based data selection approach for n-Gram model adaptation. IEEE Trans. Audio Speech Lang. Process. **17**(1), 13–23 (2009)
17. Zhao, J., et al.: The TNT team system descriptions of cantonese and mongolian for IARPA OpenASR20. In: Proceedings of the Interspeech, pp. 4344–4348 (2021)
18. Zhao, J., et al.: The THUEE system description for the IARPA OpenASR21 challenge. In: Proceedings of Interspeech, pp. 4855–4859 (2022). https://doi.org/10.21437/Interspeech.2022-649
19. Zhong, G., et al.: external text based data augmentation for low-resource speech recognition in the constrained condition of OpenASR21 challenge. In: Proceedings of Interspeech, pp. 4860–4864 (2022). https://doi.org/10.21437/Interspeech.2022-649