This document is part of the Research and Innovation Action "European Live Translator (ELITR)".
This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 825460.

Deliverable D2.2

Report 2 on ASR Systems

Tuan-Nam Nguyen (KIT), Christian Huber (KIT)

Dissemination Level: Public

Final (Version 1.0), 31st March, 2022

Grant agreement no.	825460
Project acronym	ELITR
Project full title	European Live Translator
Type of action	Research and Innovation Action
Coordinator	doc. RNDr. Ondřej Bojar, PhD. (CUNI)
Start date, duration	1st January, 2019, 39 months
Dissemination level	Public
Contractual date of delivery	31st March, 2022
Actual date of delivery	31st March, 2022
Deliverable number	D2.2
Deliverable title	Report 2 on ASR Systems
Type	Report
Status and version	Final (Version 1.0)
Number of pages	31
Contributing partners	KIT, CUNI
WP leader	KIT
Author(s)	Tuan-Nam Nguyen (KIT), Christian Huber (KIT)
EC project officer	Luis Eduardo Martinez Lafuente
The partners in ELITR are:	▪ Univerzita Karlova (CUNI), Czech Republic ▪ University of Edinburgh (UEDIN), United Kingdom ▪ Karlsruher Institut für Technologie (KIT), Germany ▪ PerVoice SPA (PV), Italy ▪ alfatraining Bildungszentrum GmbH (AV), Germany
Partially-participating party	▪ Nejvyšší kontrolní úřad (SAO), Czech Republic

For copies of reports, updates on project activities and other ELITR-related information, contact:

doc. RNDr. Ondřej Bojar, PhD., ÚFAL MFF UK
Malostranské náměstí 25
118 00 Praha, Czech Republic
bojar@ufal.mff.cuni.cz
Phone: +420 951 554 276
Fax: +420 257 223 293

Copies of reports and other material can also be accessed via the project's homepage:

`http://www.elitr.eu/`

Contents

1 Executive Summary

In this deliverable we describe the final automatic speech recognition (ASR) systems that we created for the ELITR speech translation system.

In Section 2 we describe our work. In the subsections 2.1 and 2.2 we address tasks T2.1 (Robustness of Acoustic Model) and T2.2 (Unsupervised / Semi-Supervised Adaptation of Language Models), respectively. We developed an online low-latency automatic speech recognition system with super-human performance. Furthermore, we investigated multilingual ASR systems. In subsection 2.3 we address tasks T2.2 (Unsupervised / Semi-Supervised Adaptation of Language Models) and T2.3 (Life-Long Learning during Deployment). We started with one of our high performance ASR systems and developed a system which is able to adapt itself with little data to new domains and new words.

2 Types of systems investigated

2.1 Super-human performance speech recognition

For many decades, researchers have worked on increasingly difficult tasks in order to achieve superhuman capability in speech recognition. We present a system **?** that achieves superhuman performance, i.e. a WER of 5.0 % on the Switchboard conversational benchmark, with a word-based latency of less than one second behind a speaker's speech. The system combines attention-based encoder-decoder networks with a novel low-latency incremental inference method.

In this work, we examined the delay that users experience while interacting with an online speech recognition system and suggest a method for measuring it using two distinct terms: computation latency and confidence latency. While computation latency represents the standard real-time factor (RTF), confidence latency is the amount of time required for an online recognizer to confidently decide its output. We demonstrate that with the support of modern computer hardware (such as GPUs), the computation latency of S2S models is reasonably minimal (even for large models), however the confidence latency is a more crucial criteria that we completely address for the first time. For optimizing confidence latency, we considered online processing of S2S models as an incremental speech recognition problem. We proposed an incremental inference approach with two stability detection methods to convert an S2S model to be used in online speech recognition and to allow to trade-off between latency and accuracy. Our experimental results show that it is possible to use a popular Long Short-Term Memory (LSTM) or self-attention based S2S ASR model for streaming speech recognition

without any model modification. With a delay of 1.8 seconds in all output elements, all the experimental models retain their state-of-the-art performance when performing offline inference. On the Switchboard benchmark, our best online system, which effectively utilizes three S2S models in a low-latency way, achieves a word-error-rate (WER) of 5.0 %. This online accuracy is, to the best of our knowledge, on par with state-of-the-art offline performance. We also show that human performance may be achieved while delivering output with extremely low latency. An incremental inference and a stability detection are two important components of our approach.

Incremental Inference: The incremental inference component waits for a chunk of acoustic frames to arrive with a predetermined duration before sending them to the inference component. The inference component needs to accumulate all the chunks received so far and extend the current stable hypothesis to produce a set of new unstable hypotheses. This unstable set is then sent to the stability detection component, which uses it to find a longer stable hypothesis. Because the detection of stability is handled individually, we may use different models for inference to increase identification accuracy. S2S models with various architectures or language models trained on various text sources might be involved. Using the ensemble approach, all of these models could be evenly combined.

Stability Detection: We investigate a combination of two stability detection conditions for incremental S2S speech recognition: shared prefix in all hypotheses and best-ranked prefix with reliable endpoint. First condition happens when all the active hypotheses in the beam-search share the same prefix. In the second condition, we follow the approach in **?** for the estimation of a prefix endpoint. After that, we can cut out the part of audio with a stable hypothesis, consider the remain part of audio as input to be to fed to our model.

We also participated with this system (see at **?**) in the Offline Speech Translation Task for IWSLT 2021. The ensemble of LSTM-based and Conformer-based sequence-to-sequence model provider the best results, which are 2.4 and 3.9 WERs for Libri and TED-LIUM test sets respectively.

2.2 Multilingual end-to-end Sequence-to-sequence ASR

Since our speech recognition system reached a super-human performance in English, we expanded our model to include a variety of languages. End-to-end multilingual speech recognition entails training a single model on a compositional speech corpus including several languages, culminating in a single neural network capable of transcribing multiple languages. It is feasible to use a single neural network to capture common characteristics across various datasets of different languages. This strategy has been widely utilized to assist under-resourced languages in gaining knowledge from their richer counterparts.

Because each language in the training data has its own characteristics, the shared network may struggle to optimize for all of them at the same time. We present a novel multilingual architecture that focuses on the core operation in neural networks: linear transformation functions. We propose a multilingual architecture that utilizes a factorization approach that is both efficient and scalable with the number of languages involved **?**. Furthermore, as long as matrix-vector multiplication is the dominant operation, this approach may be used to any neural design. The method's main concept is to assign fast weight matrices for each language by decomposing each weight matrix apart into a shared component and a language-dependent component. The latter is then factorized into vectors using rank-1 assumptions to reduce the amount of parameters per language, allowing us to use the original low-latency network's optimized implementation, which reduces our approach's the computational cost. This approach has been tested on two widely used architectures: Long Short-Term Memories (LSTM) and Transformers which indicate that weight factorization can help both types of networks in multilingual ASR. In two multilingual settings with 7 and 27 languages, this efficient factorization approach reduced word error rates by 26% and 27%, respectively.

2.3 Domain Adaptation / New Word Adaptation

As described above, neural sequence-to-sequence systems deliver state-of-the-art performance for automatic speech recognition (ASR). When using appropriate modeling units, e.g., byte-pair encoded characters, these systems are in principle open vocabulary systems. In practice, however, they often fail to recognize words not seen during training, e.g., named entities, numbers or technical terms.

To alleviate this problem, **?** proposed to supplement an end-to-end ASR system with a word/phrase memory and a mechanism to access this memory to recognize the words and phrases correctly. After the training of the ASR system, and when it has already been deployed, a relevant word can be added or subtracted instantly without the need for further training. They demonstrated that through this mechanism the system is able to recognize more than 85% of newly added words that it previously failed to recognize compared to a strong baseline.

Within the ELITR project, we extended this system and extensively evaluated it in a dynamically changing environment. The results (see appendix E) are submitted to Interspeech 2022.

3 Conclusion

This deliverable described the final state of the ASR systems developed for the ELITR project. We developed online low-latency automatic speech recognition system with super-human performance, through incremental inference and stability detection, multilingual end-to-end sequence-to-sequence ASR models, using weight factorization, and a method to add new words instantly the such end-to-end sequence-to-sequence ASR models.

A High Performance Sequence-to-Sequence Model for Streaming Speech Recognition from Interspeech 2020

INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

High Performance Sequence-to-Sequence Model for Streaming Speech Recognition

Thai-Son Nguyen, Ngoc-Quan Pham, Sebastian Stüker, Alex Waibel

Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology
thai.nguyen@kit.edu

Abstract

Recently sequence-to-sequence models have started to achieve state-of-the-art performance on standard speech recognition tasks when processing audio data in batch mode, i.e., the complete audio data is available when starting processing. However, when it comes to performing run-on recognition on an input stream of audio data while producing recognition results in real-time and with low word-based latency, these models face several challenges. For many techniques, the whole audio sequence to be decoded needs to be available at the start of the processing, e.g., for the attention mechanism or the bidirectional LSTM (BLSTM). In this paper, we propose several techniques to mitigate these problems. We introduce an additional loss function controlling the uncertainty of the attention mechanism, a modified beam search identifying partial, stable hypotheses, ways of working with BLSTM in the encoder, and the use of chunked BLSTM. Our experiments show that with the right combination of these techniques, it is possible to perform run-on speech recognition with low word-based latency without sacrificing in word error rate performance.
Index Terms: sequence-to-sequence, online, streaming

1. Introduction

Sequence-to-sequence (S2S) attention-based models [1, 2] have become increasingly popular for end-to-end speech recognition. Several advances [3, 4, 5, 6] have been proposed to the architecture and the optimization of S2S models to achieve superior recognition performance. In offline scenarios, i.e., batch processing of audio files, the S2S models in [7, 8] have already shown state-of-the-art performance on standard benchmarks. However, methods for employing S2S models in online speech recognition, i.e., run-on recognition with low latency, still needs to be researched, to obtain the desired accuracy and latency.

[9, 10, 11] pointed out early that the shortcoming of an attention-based S2S model used in online condition lies in its attention mechanism, which must perform a pass over the entire input sequence for every element of the output sequence. [10, 11] proposed a so-called monotonic attention mechanism that enforces a monotonic alignment between the input and output sequence. Later on, [12, 13, 14] have addressed the latency issue of bidirectional encoders, which is also an obstacle for online speech recognition. In these studies, unidirectional and chunk-based encoder architectures replace the fully-bidirectional approach to control the latency.

In this work, we analyze the alignment behavior of the attention function of a high-performance S2S model and propose an additional constraint loss to make it capable of streaming inference. By discussing the problems that occurred when adapting a S2S model to be used for a streaming recognizer, we additionally show that the standard beam-search has no guarantee for low-latency inference results, and needs to be modified for providing partial hypotheses. Besides, we argue that the common real-time factor is not a proper choice for measuring the user-perceived latency in online and streaming setup, and propose a novel and suitable technique for the replacement.

In contrast to the earlier works in the literature, our experimental results proved that a bidirectional encoder could be combined with suitable inference methods to produce high accuracy and low latency speech recognition output. With a delay of 1.5 seconds in all output elements, our streaming recognizer can achieve the ideal performance of an offline system with the same configuration. To the best of our knowledge for the first time, a S2S speech recognition model can be used in online conditions without scarifying accuracy [1].

2. Sequence-to-Sequence Model

In this work, we modify the LSTM-based sequence-to-sequence encoder-decoder model proposed in [8] to perform high-accuracy online streaming ASR with very low latency. Our model can be decomposed using a set of neural network functions as follows:

$$enc = LSTM(CNN(spectrogram))$$
$$emb = LSTM(Embedding(subwords))$$
$$ctx, attn = SoftAttention(emb, enc, enc)$$
$$y = Distribution(ctx + emb)$$

In principle, the functions are designed to map a sequence of acoustic features into a sequence of sub-words and can be grouped into two parts: encoder and decoder. In the encoder, acoustic vectors are down-sampled with two convolutional layers and then fed into several bidirectional LSTM layers to generate the encoder's hidden states enc. In the decoder, two unidirectional LSTM layers are used to embed a sub-word unit into a latent representation emb. The multi-head *soft-attention* function proposed in [15] is used to model the relationship between enc and emb, which results in a context vector ctx. All the functions are jointly trained via the sequence cross-entropy loss by plugging a softmax distribution on top of ctx and emb.

As shown in [8], this S2S model can achieve highly-competitive offline performance on the Switchboard speech recognition task. However, the model encounters latency issues when being used in online conditions since both, the attention function and the bidirectional encoder network, require the entire input sequence to achieve their optimal performance.

3. Streaming S2S ASR

In this section, we describe our modifications that enable the S2S model to perform online streaming speech recognition with

[1]The project ELITR leading to this publication has received funding from the European Unions Horizon 2020 Research and Innovation Programme under grant agreement No 825460.

http://dx.doi.org/10.21437/Interspeech.2020-1863

low latency and without loss in performance. The modifications include an additional loss to control the uncertainty of the attention function and search algorithms to infer high-accuracy partial hypothesis.

3.1. Discouraging Look-ahead Attention

The core of S2S models is the mechanism that autoregessively generates a context vector *ctx* for the prediction of the next token. For the model described in Section 2, *ctx* is computed as a sum of all the encoder's hidden states weighted by the *attention scores* which are calculated by the attention function. The attention scores calculated for a specific token typically reveals the positions within the encoder states (or spectral frames) corresponding to the token. So, the attention function can be considered as an *alignment model*. However, this unsupervised alignment does not resemble traditional forced-alignments (or human alignments) in speech recognition. As illustrated in Figure 1a, during a particular inference process, the attention scores produced for many tokens (e.g., #3, 8, 9, 16) are dominated by the start and end frames, which are not the proper alignments. In this case, the inference still produces the correct transcript, and so the attention function works as it is expected. The mismatch between the attention-based alignment and regular alignment reveals uncertainty that the attention function may have while being optimized with the sequence training likelihood. Although this uncertainty may not lead to inference errors, the attention function always employs all the encoder's hidden states, which hinders the model from being used in streaming inference. It is preferable for a streaming inference that for the prediction of a token S, the attention function only considers past frames until a particular time S_t (the endpoint) and disregards all future frames.

To build such a S2S model for streaming, we investigated the incorporation of an additional loss which discourages the attention function from using future frames during training. Specifically, given token S which belongs to word W in label sequence L, we find a region $R_s = (W_t, \infty)$ in which W_t is the end time of W provided by a Viterbi alignment. The attention-based constraint loss is computed as the sum of all attention scores within the region R_s for all S in L:

$$\mathcal{L}_{attn} = \alpha \sum_{S} \sum_{x}^{|R_s|} Ascore_S^x$$

The tuneable parameter α adjusts the influence of the constraint loss to the maximum likelihood loss of the label sequence during training. By minimizing both losses simultaneously, we expect that the attention function learns to produce *close-to-zero* scores for the constraint regions for all label tokens while still minimizing the main loss.

3.2. Inference for Partial Stable Hypothesis

Beam search is the most efficient approach for the inference of S2S models. Its basic idea is to maintain a search network in which network paths are extended with new nodes with the highest accumulated scores and then pruned away to keep only a set of active paths (or hypotheses). Typically, the most probable hypothesis for an utterance X is found and guaranteed when the search space constructed from the entire acoustic signals of X is supplied to the search. However, waiting for the complete acoustic signals of X to output inference results is not efficient for a streaming setup. A streaming recognizer must be able to produce partial output while processing partial input. In this section, we describe our search algorithm applied to the proposed S2S model to produce partial output while retaining high accuracy.

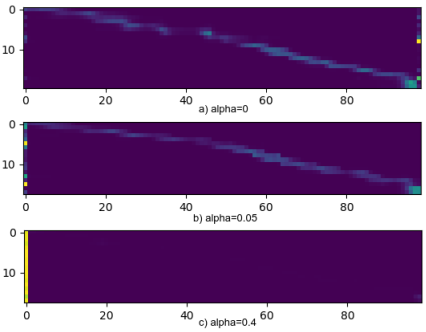

Figure 1: *Attention-based alignments provided by a) the regular attention function and b) c) the attention function trained with the constraint loss during the inference of an utterance of 4-seconds length (down-sampling of 4 frames after encoder's layers). The alignments for the tokens 3, 8, 9, 16 are dominated by both start and end frames in a), and dominated by start frames only in b).*

Assume that in a streaming setup, at time t we use the proposed S2S model to perform inference for t audio frames. Given a context sequence C, the attention function is used to generate t attention scores for the prediction of the next token. We find a time $t_c <= t$ such that the sum of all attention scores from the *covering* window $w = [0, t_c]$ is equal to a constant $\theta = \sum_x^{|t_c|} Ascore^x$. When $\theta = 0.95$, w covers all dominant attention scores and the context vector generated from w is almost the same as from $[0, t]$. If t_c is observed to be unchanged when t keeps growing, then we consider t_c as the endpoint of C. During stream processing, we use a term Δ to determine if endpoint t_c finally gets fixed as $t_c < t - \Delta$.

We then incorporate the information of endpoints into the beam search to find a partial stable hypothesis. Assume that our beam search can always perform in real-time for t audio frames to produce N considered hypotheses. If all N hypotheses share the same prefix sequence C and the endpoint of C is determined, then we consider C to be an *immortal* part that will not change anymore in the future. When more audio frames are available in the stream, C will be used as the prefix for all search hypotheses, and we repeat this step to find a longer stable hypothesis. Except the condition on endpoints, the idea of finding *immortal prefix* is similar to the partial trace-back [16, 17] used in HMM-based speech recognizers.

In addition to the immortal prefix, we also investigated a more straightforward method in which we only consider the *best-ranked* hypothesis and decide on a stable part C based solely on the term Δ. The inspiration comes from the incremental speech recognition approach proposed in [18].

3.3. Bidirectional Encoder

To achieve high performance, bidirectional LSTM have been the optimal choice for the encoder of LSTM-based S2S models. However, due to the backward LSTM, bidirectional LSTM are *not* suited to provide partial and low-latency output as needed for streaming recognizers. The addition of acoustic input will affect all of the encoder's hidden states, which then makes all

Table 1: *WER performance of the S2S model with bidirectional encoder trained with different scales of the constraint loss.*

Model	α	SWB	CH	Hub5'00
6x1024 BLSTM		5.9	11.8	8.9
	0.4	6.2	12.2	9.2
	0.2	6.1	12.1	9.1
	0.05	5.8	12.0	8.9

partial inference results unstable. This effect leads to the fact that stable output can be confidently inferred only when the input is complete. Therefore, earlier works [10, 19, 20] switched to unidirectional LSTM in their online models.

In this work, we try to utilize bidirectional LSTM for high-performance speech recognition in a streaming scenario. In the first setting, we investigated the use of the S2S model with a fully bidirectional encoder. First, we train the S2S model to achieve optimal parameters for the offline setup, and with the attention-based constraint loss proposed in Section 3.1. Then, during inference, we update the encoder's hidden states from all available acoustic input before performing the search approaches in Section 3.2 to find stable hypotheses. As will be shown later, the use of a bidirectional LSTM as this way is possible since the proposed inference methods rely on the determination of endpoints, and the update of encoder's hidden states leads to stabilizing this determination.

In addition to fully bidirectional LSTM, we also experimented with a chunk-based BLSTM approach. During training, we divide input sequences into many non-overlapping blocks of a fixed size of K, and then use a BLSTM to compute each block sequentially. To benefit from long-range contextual learning, we initialize the forward LSTM with its last hidden states after processing the previous chunk. The initialization of the backward LSTM can either be a constant or from the previous chunk. By doing so, the encoder's hidden states can be computed incrementally and efficiently as for unidirectional LSTM. This chunk-based approach is different from [21] and the latency-controlled BLSTM [22, 23] that adopt constant initialization of both directions.

4. Experiments

4.1. Experimental Setup

Our experiments were conducted on the Fisher+Switchboard corpus consisting of 2,000 hours of telephone conversation speech. The Hub5'00 evaluation data was used as the test set. All the experimental models use the same input features of 40 dimensional log-mel filterbanks to predict 4,000 BPE sub-word units generated with the SentencePiece [24] toolkit from all the training transcripts. The models with bidirectional encoder employ six layers of 1024 units while it is 1536 for the unidirectional encoders. We used only 1-head for the attention function in all setups. All models were trained with a dropout of 0.3. We further used the combination of two data augmentation methods *Dynamic Time Stretching* and *SpecAugment* proposed in [8] to reduce model overfitting. We use Adam [25] with an adaptive learning rate schedule to perform 12,000 updates during training. The model parameters of the 5 best epochs according to the perplexity on the cross-validation set are averaged to produce the final model.

For beam search, we use neither length normalization nor any language model. With a beam size of 8, the experimental models typically achieve their optimal accuracy.

Table 2: *Latency and accuracy of the S2S model with bidirectional encoder on Hub5'00 test set.*

Method	Beam Size	Δ	WER	Latency
Force-Alignment	8		8.9	0.60
	4		9.1	0.60
	2		9.3	0.60
Immortal Prefix	8	20	**8.9**	**0.93**
	8	30	8.9	0.93
	4	20	9.2	0.86
	4	30	**9.1**	**0.87**
	2	20	12.6	0.74
	2	40	10.1	0.79
	2	60	9.5	0.83
	2	80	9.3	0.86
1st-Ranked Prefix	8	30	11.2	0.75
	8	50	9.6	0.80
	8	70	9.3	0.84
	4	30	11.3	0.75
	4	50	9.6	0.80
	4	70	9.3	0.84
	2	10	25.8	0.62
	2	30	11.4	0.75
	2	50	9.7	0.80
	2	70	9.3	0.84
Combination	8	20-70	9.2	0.83
	4	30-70	**9.4**	**0.81**
	2	60-70	9.5	0.83

4.2. Latency Measure

Neither the commonly used *real-time factor* (RTF) nor commitment latency [26, 27] are sufficient to measure user-perceived latency for a streaming recognizer. For example, the transcript outputs for an 11-seconds sentence can appear 10 seconds later than a 1-second sentence, but the RTFs measured in two cases can be similar. In this study, we propose to use a different method for measuring streaming latency. Assume that a recognizer processes a sentence S of T seconds in streaming fashion and it outputs N token s_1, s_2,.. s_n at different timestamps t_1, t_2,.. t_n. And assume the inference time is always a small constant, then timestamp t_i is just when the recognizer is confident of producing s_i. The latency of recognizing S with regard to the transcript s_1, s_2,.. s_n is calculated as the average of all token latencies $t_i \in [1, n]$ normalized by the duration of S: $\sum t_i/(n * T)$. With this measure, the latency of an offline system is always 1 – as the offline system is only confident for all transcripts until end-of-sentence. In the same way, we simulate the latency of an *instant* recognizer by using a forced-alignment to find t_i for s_i.

5. Results

5.1. Effect of the Constraint Loss

In this section, we evaluate the influence of the constraint loss proposed in Section 3.1 on the training of the S2S model. We started by using a high value for α and exponentially decreased it to train several systems for comparison. As observed during training, the constraint loss gets small quickly to a stable value depended on α. Joint training slows down the convergence of the main loss but does not have a significant impact on the final performance. As shown in Table 1, WERs are slightly worse with high α and can be similar to the regular training when α is

small (e.g., 0.05). Different from that, the constraint loss may largely change the behavior of the attention function. For example, in Figure 1b, the attention function moves the high scores of the mismatched alignment to start frames, instead of end frames as in the regular training. We also found an extreme case when $\alpha = 0.4$. The attention-based alignment does not correspond at all to the proper alignment as illustrated Figure 1c.

Using the model trained with $\alpha = 0.05$, we follow the approach in Section 3.2 to extract the endpoints for all prefixes found during the inference of the evaluation set. We could verify that the extracted endpoints in all sentences match the expectation for streaming inference described in Section 3.1. So we keep this model for further experiments.

5.2. Latency on Various Conditions

Using the S2S model with a bidirectional encoder trained with the constraint loss scale $\alpha = 0.05$, we performed several experiments with the inference approaches described in Section 3.2. In the experiments, the streaming scenario is simulated by repeatedly feeding an additional audio chunk of 250 ms to the experimental systems for incremental inferences. All the inferences were performed on a single Nvidia Titan RTX GPU, which produced an average RTF of 0.065 with a beam size of 8. The RTF result shows that real-time capacity is not a bottleneck problem in this setup. So we focus on the latency measure proposed in Section 4.2.

For baselines, we computed the offline WER performance with beam sizes 8, 4, 2, and then used a force-alignment system to produce the *ideal* latency from the offline transcripts. The ideal latency is always 0.6. If we shift the time alignment of the transcripts with 250 ms (i.e., all the outputs have a delay of 250 ms), 500 ms, 1 second, and 1.5 seconds, then we obtained a latency of 0.71, 0.78, 0.86 and 0.91 respectively.

Table 2 presents the accuracy and the latency we achieved when using the *immortal prefix* and *1st-ranked prefix* inference methods with several settings of Δ. Overall, the two methods are consistent with the observations in the HMM-based systems [16, 17, 18]. Using the *immortal prefix* condition, the final accuracy can be guaranteed as for the offline inference for large beam sizes, e.g., 8 and 4. For a smaller beam size, this condition is not strong enough to deal with unstable partial results – probably due to the changes of the encoder's hidden states. In the *1st-ranked prefix* approach, increasing Δ allows for a flexible trade-off between the accuracy and the latency. The offline accuracy can also be achieved if a very large Δ is applied. These results consolidate our findings in two aspects. First, the integration of Δ is reliable and crucial for the streaming inferences to work efficiently. And second, the use of the bidirectional LSTM for the encoder is possible and results in high accuracy.

To achieve 8.9% WER (the offline accuracy), the system needs to delay outputs with an average duration of about 1.5 seconds. To obtain a lower latency of 1 second, the WER increases to 9.2%, e.g., by using the *immortal prefix* method with $\Delta = 20$ and $beamsize = 4$. The *combination* of both methods is efficient if we want to reach a latency of 0.81, which is closer to the average delay of 0.5 seconds.

5.3. Performance of Different Encoders

The bidirectional requires additional re-computation of the entire encoder's hidden states for every addition of input signal in the stream. In this section, we investigate two additional network architectures, *unidirectional* LSTM and *chunk-based* BLSTM described in Section 3.3, that improve the computational efficiency of the encoder. For chunk-based, we experimented with $K = 80$ and $K = 200$, as the chunk sizes of 800 ms and 2 seconds. We constantly found that initializing the backward LSTM from the last hidden states of the previous chunk is better than a constant, so we only present the results of this approach. We evaluated two types of encoders in two categories: the best accuracy and the accuracy that the systems retain when maintaining an average delay of 1 second. To do so, we use the same *immortal prefix* inference and experiment with different settings of beam size and Δ.

Table 3: *Latency and accuracy of the S2S models with unidirectional and chunk-based encoders using immortal prefix.*

Encoder	Beam Size	Δ	WER	Latency
Unidirectional	8	30	12.7	0.94
	8	∞	12.6	1.00
	2	20	13.6	0.82
	2	30	13.2	0.85
	2	∞	13.1	1.00
Chunk-based *K=80*	8	30	10.5	0.91
	8	∞	10.4	1.00
	2	20	11.1	0.80
	2	30	10.9	0.82
	2	∞	10.8	1.00
Chunk-based *K=200*	8	30	10.3	0.89
	8	∞	10.0	1.00
	2	30	11.3	0.79
	2	60	10.8	0.85
	2	∞	10.7	1.00

As shown in Table 3, there is a big gap between the best WER of the unidirectional and bidirectional encoders (12.6% vs. 8.9%). The chunk-based encoder, however, is closer to the performance of the bidirectional one when a large chunk size is used. As the encoder's states are fixed early, the inferences are already stable when $\Delta = 30$ for all beam sizes. To achieve 1-second delay, all the approaches need to trade-off for an accuracy reduction of 5% relatively. In term of latency, the chunk-based approach with $K = 80$ and $beamsize = 2$ and $\Delta = 30$ is the best setting in this setup.

6. Related work

[10, 11] pointed out the problems of the *soft-attention* mechanism on acquiring the entire encoder's states and proposed a trainable *monotonic* attention function to train sequence-to-sequence models for online application. Given a prefix, the monotonic attention function allows finding an encoder position [10] or the endpoint of a chunk [11] used for prediction of the next token. In our study, we showed that endpoints can also be estimated precisely and efficiently via the regular soft-attention function by controlling its uncertainty. We further showed that there are more issues to be addressed for high-performance online speech recognition, such as finalizing partial results of the beam search and the use of a bidirectional encoder, and proposed effective methods for addressing theses issues.

7. Conclusion

We have proposed and evaluated several techniques for applying S2S attention-based models to streaming speech recognition. Our results show that with these techniques it is possible to produce low latency online recognition results on the Switchboard task without a significant decrease in performance.

8. References

[1] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015.

[2] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[3] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *ICASSP 2018*.

[4] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," *Proc. Interspeech 2018*.

[5] C. Weng, J. Cui, G. Wang, J. Wang, C. Yu, D. Su, and D. Yu, "Improving attention based sequence-to-sequence models for end-to-end english conversational speech recognition," *Proc. Interspeech 2018*.

[6] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Muller, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *Proc. of Interspeech 2019*.

[7] 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," *Proc. of Interspeech 2019*.

[8] T.-S. Nguyen, S. Stueker, J. Niehues, and A. Waibel, "Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation," *arXiv preprint arXiv:1910.13296*, 2019.

[9] N. Jaitly, Q. V. Le, O. Vinyals, I. Sutskever, D. Sussillo, and S. Bengio, "An online sequence-to-sequence model using partial conditioning," in *Advances in Neural Information Processing Systems*, 2016, pp. 5067–5075.

[10] C. Raffel, M.-T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2837–2846.

[11] C.-C. Chiu and C. Raffel, "Monotonic chunkwise attention," *arXiv preprint arXiv:1712.05382*, 2017.

[12] R. Fan, P. Zhou, W. Chen, J. Jia, and G. Liu, "An online attention-based model for speech recognition," *Proc. Interspeech 2019*, pp. 4390–4394, 2019.

[13] H. Miao, G. Cheng, P. Zhang, T. Li, and Y. Yan, "Online hybrid ctc/attention architecture for end-to-end speech recognition," *Proc. of Interspeech 2019*, pp. 2623–2627, 2019.

[14] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Towards online end-to-end transformer automatic speech recognition," 2019.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017.

[16] P. Brown, J. Spohrer, P. Hochschild, and J. Baker, "Partial traceback and dynamic programming," in *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 7. IEEE, 1982, pp. 1629–1632.

[17] E. O. Selfridge, I. Arizmendi, P. A. Heeman, and J. D. Williams, "Stability and accuracy in incremental speech recognition," in *Proceedings of the SIGDIAL 2011 Conference*. Association for Computational Linguistics, 2011, pp. 110–119.

[18] S. Wachsmuth, G. A. Fink, and G. Sagerer, "Integration of parsing and incremental speech recognition," in *9th European Signal Processing Conference (EUSIPCO 1998)*. IEEE, 1998, pp. 1–4.

[19] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[20] A. Narayanan, R. Prabhavalkar, C.-C. Chiu, D. Rybach, T. N. Sainath, and T. Strohman, "Recognizing long-form speech using streaming end-to-end models," *arXiv preprint arXiv:1910.11455*, 2019.

[21] K. Audhkhasi, G. Saon, Z. Tüske, B. Kingsbury, and M. Picheny, "Forget a bit to learn better: Soft forgetting for ctc-based automatic speech recognition," *Proc. Interspeech 2019*, pp. 2618–2622, 2019.

[22] R. Fan, P. Zhou, W. Chen, J. Jia, and G. Liu, "An online attention-based model for speech recognition," *Proc. Interspeech 2019*, 2019.

[23] S. Xue and Z. Yan, "Improving latency-controlled blstm acoustic models for online speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5340–5344.

[24] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] T. S. Nguyen, J. Niehues, E. Cho, T.-L. Ha, K. Kilgour, M. Muller, M. Sperber, S. Stueker, and A. Waibel, "Low latency asr for simultaneous speech translation," 2020.

[27] B. Li, S.-y. Chang, T. N. Sainath, R. Pang, Y. He, T. Strohman, and Y. Wu, "Towards fast and accurate streaming end-to-end asr," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6069–6073.

B Super-Human Performance in Online Low-latency Recognition of Conversational Speech

Super-Human Performance in Online Low-latency Recognition of Conversational Speech

Thai-Son Nguyen[12], *Sebastian Stüker*[12], *Alex Waibel*[12]

[1] Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology
[2] Karlsruhe Information Technology Solutions — kites GmbH

firstname.lastname@kit.edu

Abstract

Achieving super-human performance in recognizing human speech has been a goal for several decades as researchers have worked on increasingly challenging tasks. In the 1990's it was discovered, that conversational speech between two humans turns out to be considerably more difficult than read speech as hesitations, disfluencies, false starts and sloppy articulation complicate acoustic processing and require robust joint handling of acoustic, lexical and language context. Early attempts with statistical models could only reach word error rates (WER) of over 50% which is far from human performance with shows a WER of around 5.5%. Neural hybrid models and recent attention-based encoder-decoder models have considerably improved performance as such contexts can now be learned in an integral fashion. However, processing such contexts requires an entire utterance presentation and thus introduces unwanted delays before a recognition result can be output. In this paper, we address performance *as well as* latency. We present results for a system that can achieve super-human performance, i.e. a WER of 5.0% on the Switchboard conversational benchmark, at a word based latency of only 1 second behind a speaker's speech. The system uses multiple attention-based encoder-decoder networks integrated within a novel low latency incremental inference approach.

Index Terms: ASR, Sequence-to-sequence, Online, Streaming, Low Latency, Human Performance

1. Introduction

Sequence-to-sequence (S2S) attention-based models [1, 2] are a currently one of the best performing approaches to end-to-end automatic speech recognition (ASR). A lot of research has already been dedicated to boost the performance of S2S models. Several works [3, 4, 5, 6, 7] have successfully pushed up the state-of-the-art performance records on different speech recognition benchmarks and proved the superior performance of S2S models over conventional speech recognition models in an offline setting. As so, the next research trend is to apply S2S speech recognition in practice. Many practical applications need to work ASR systems in real-time run-on mode with low-latency [8, 9].

Early studies [10, 11, 12] pointed out that the disadvantage of an S2S model used in online condition lies in its attention mechanism, which must perform a pass over the entire input sequence for every output element. [11, 12] have dealt with this disadvantage by proposing a so-called monotonic attention mechanism that enforces a monotonic alignment between the input and output sequence. Later on, [13, 14, 15] have additionally resolved the latency issue of bidirectional encoders by using efficient chunk-based architectures. More recent works [16, 17, 18, 19, 20, 21] have addressed these latency issues for different S2S architectures.

While most of the studies focus on model modifications to make S2S models capable of online processing with minimal accuracy reduction, they lack thoughtful research on the latency aspect. In this work, we analyze the latency that the users suffer while interacting with an online speech recognition system, and propose to measure it with two separate terms *computation latency* and *confidence latency*. While computation latency reflects the common real-time factor (RTF), confidence latency corresponds to the time an online recognizer needs to confidently decide its output. We show that with the support of new computing hardware (such as GPUs), the computation latency of S2S models is relatively small (even for big models), and the confidence latency is a more critical criterion which, for the first time, we address thoroughly.

To optimize for confidence latency, we consider the online processing of S2S models as an incremental speech recognition problem. We propose an *incremental inference* approach with two stability detection methods to convert an S2S model to be used in online speech recognition and to allow to trade-off between latency and accuracy. Our experimental results show that it is possible to use a popular Long Short-Term Memory (LSTM) [22] or self-attention based S2S ASR model for run-on recognition without any model modification. With a delay of 1.8 seconds in all output elements, all the experimental models retain their state-of-the-art performance when performing offline inference. Our best online system, which successfully employs three S2S models in a low-latency manner, achieves a word-error-rate (WER) of 5.0% on the Switchboard benchmark. To the best of our knowledge, this online accuracy is at par with the state-of-the-art offline performance. We also demonstrate that it is possible to achieve human performance as measured in [23, 24] while producing output at very low latency.

2. Sequence-to-sequence Based Low-latency ASR

In this section, we first describe different sequence-to-sequence ASR architectures investigated in this paper. We then present the proposed incremental inference with two stability detection methods.

2.1. Models

There have been two efficient approaches for making S2S ASR systems. The first approach employs LSTM layers in both encoder and decoder networks, while the second follows the Transformer architecture [25] which uses solely self-attention

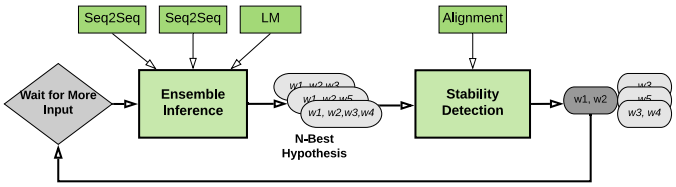

Figure 1: *Incremental inference for low-latency S2S ASR*

modules to construct the whole S2S network. In this work, we investigate both of the S2S architectures for the online low-latency setting.

Our LSTM-based S2S model employs two time-delay neural network (TDNN) layers [26, 27] with a total time stride of four for down-sampling followed by several bidirectional LSTM layers to encode the input spectrogram. In the decoder, we adopt two layers of unidirectional LSTMs for modeling the sequence of sub-word labels and the multi-head *soft-attention* function proposed in [25] to generate attentional context vectors. In detail, the LSTM-based model works as the following neural network functions:

$$\begin{aligned} enc &= LSTM(TDNN(spectrogram)) \\ emb &= LSTM(Embedding(symbols)) \\ ctx &= SoftAttention(emb, enc, enc) \\ y &= Softmax(ctx + emb) \end{aligned}$$

In the Transformer model, the down-sampling is handled by a linear projection layer on four consecutive stacked feature vectors. The rest of the model architecture is similar to the Transformer approach proposed for speech recognition in [6]. We also adopt the layer stochastic technique to efficiently employ more self-attention layers in both encoder and decoder.

For more details of the model architectures and offline evaluations, we would refer the readers to [7] and [6].

2.2. Incremental Inference

Figure 1 illustrates our proposed architecture that allows S2S models to produce incremental transcriptions on a speech stream. In the architecture, we handle the two tasks of inference and stability detection by two separate components in a processing pipeline. The first step in the pipeline is to wait for a chunk of acoustic frames with a predefined length (i.e., 200ms), which is then sent to the inference component. The inference component needs to accumulate all the chunks received so far and extend the current *stable* hypothesis to produce a set of new *unstable* hypotheses. This unstable set is then provided to the *stability detection* component for detecting a longer stable hypothesis.

As the stability detection is handled separately, we are able to involve multiple models for the inference to improve recognition accuracy. The involved models can be S2S models with different architectures or language models trained on different text data. All of these models can be uniformly combined via the ensemble technique.

2.3. Stability Detection

Stability detection is the key to make the system work in the incremental manner and to produce low latency output. For an HMM based speech recognition system, stability conditions can be determined incrementally during the time-synchronous Viterbi search [28, 29, 30]. Due to lack of time alignment information and unstable internal hidden states (e.g., of a bidirectional encoder), it is not straightforward to apply the same idea to S2S models. In this work, we investigate a combination of two stability detection conditions for incremental S2S speech recognition:

- **Shared prefix in all hypotheses**: Similar to the *immortal prefix* [28, 30] in HMM ASR, this condition happens when all the active hypotheses in the beam-search share the same prefix. However, different from HMM ASR, this condition may not strongly lead to an *immortal* partial hypothesis due to the unstable search network states in S2S beam-search.
- **Best-ranked prefix with reliable endpoint**: Since it may require a long delay for a *shared prefix* to happen, we also consider a different approach to improve the latency. We make use of the observation from [29] for HMM ASR, that the longer a prefix remains to be part of the most likely hypothesis, the more stable it is. Applied to S2S models, we need a method to align a prefix with audio frames, and so be able to find its endpoint in time. We follow the approach in [18] for the estimation of a prefix endpoint. First, this approach requires to train a single-head attention LSTM-based S2S model with the attention-based constraint loss [18]. Then, the endpoint of a prefix C is estimated during incremental inference by finding a time t_c such that the sum of all attention scores from the covering window $[0, t_c]$ is at least 0.95. After that, we can measure Δ as the difference between the estimated endpoint and the end of the audio stream. Δ will be used as the single input to decide the prefix C is reliable enough and considered to output.

3. Measure of Latency

Latency is one of the most important factors that decide the usability of an user-based online ASR system. A latency measure needs to reflect the actual delay that the users perceive so that the improvement of latency can lead to better usability. Strictly, the latency observed by a user for a single word is the time difference between when the word was uttered and when its transcript appeared to the user and will never be changed again. We formulate this complete latency as follows.

Let's assume a word w has been uttered, i.e., completely pronounced, at time U_w. Let C_w be the time that the ASR system can start to process the audio of w and that the ASR system can *confidently* infer w after a *delay* of D_w, the time needed to perform the inference. The user-perceived latency with regard to w is then:

$$Latency_w = C_w + D_w + T_w - U_w$$

where T_w presents the transmitting time for audio and text data. T_w is usually small and can be omitted.

For a speech utterance S consisting of N words w_1, w_2,.. w_N, we are interested in the average latency:

$$\begin{aligned} Latency_S &= \sum_i^N (D_{w_i} + C_{w_i} - U_{w_i})/N \\ &= \sum_i^N D_{w_i}/N + \sum_i^N C_{w_i}/N - \sum_i^N U_{w_i}/N \\ &= \sum_i^N D_{w_i}/N + \sum_i^N C_{w_i}/N - \sum_i^N (U_{w_i} - \delta)/N + \delta \\ &= D_{avg} + C_{avg} - U_{avg-\delta} + \delta \end{aligned}$$

In the final equation, the first term represents the computational delay. If we normalize this term by length of the decoding audio segments, then we obtain the real-time factor of the ASR system. The second term indicates how much acoustic evidence the model needs to confidently decide on its output. This latency term makes the difference in calculating the latency for online vs. offline processing. For offline processing, it is always a constant for a specific test set, since all the offline transcripts are output at the end of the test set.

To estimate the third term, we usually need to use an external time alignment system, e.g. a Viterbi alignment using an Hidden Markov Model (HMM) based acoustic model. It is inconvenient to re-run the time alignment for every new transcript. To cope with this issue, [18] introduced a fixed delay δ for all the outputs, and proposed to pre-compute a set of $U_{avg-\delta}$ for different δ. Later on, only the calculation of C_{avg} is required as the average delay can be found by comparing C_{avg} with the pre-computed set.

The latency improvement requires the optimization of both terms D_{avg} and C_{avg} which we refer to as *computation latency* and *confidence latency*. While computation latency can be improved by faster hardware or improved implementations for the search, confidence latency mainly depends on the recognition model.

4. Experimental Setup

Our experiments were conducted on the Fisher+Switchboard corpus consisting of 2,000 hours of telephone conversation speech. The Hub5'00 evaluation data was used as the test set, reporting separate performance numbers for the Switchboard (SWB) and CallHome (CH) portions.

All our models use the same input features of 40 dimensional log-mel filterbanks to predict 4,000 byte-pair-encoded (BPE) sub-word units. During training, we employ the combination of two data augmentation methods *Dynamic Time Stretching* and *SpecAugment* [7] to reduce model overfitting. Adam with an adaptive learning rate schedule is used to perform 200,000 updates. The model parameters of the 5 best epochs according to the perplexity on the cross-validation set are averaged to produce the final model.

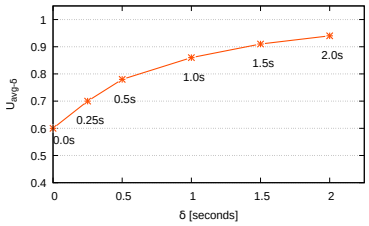

Figure 2: *Confidence latency conversion.*

4.1. Latency Evaluation

We evaluate our systems with the decomposed latency terms from Section 3. *Computation latency* is measured every time when incremental inference is performed, while for *confidence latency* we adopt a similar approach to [18] to estimate the terms C_{avg} and U_{avg}. First, we build a good HMM-based force-alignment system and use it to find time alignment for the test set transcripts. U_{avg} is calculated as the average of the ending times of all the transcript words found by the alignment system. To normalize U_{avg} between 0 and 1, all the time alignment indexes are divided by their utterance lengths. We then shift the time indexes to the right with different δ (the delay term) to compute $U_{avg-\delta}$. This results in a conversion chart illustrated in Figure 2. Later on, C_{avg} is computed the same way for the systems, and the corresponding delay is extracted from the conversion chart.

Table 1: *Experimental systems and their offline accuracy. The optimal beam size of 8 was found for all the systems.*

ID	Model Type	#Params	SWB	CH
S1	6x2 LSTM-1024	162M	5.8	11.8
S2	6x2 LSTM-1536	258M	5.3	11.5
T1	24x8 Transformer	111M	5.8	11.9
E1	S1 + S2	420M	5.3	10.9
E2	S1 + S2 + T1	531M	5.0	10.1

5. Results

5.1. Models and Offline Accuracy

We constructed two LSTM-based models with different model sizes. The smaller one uses 1-head attention and was trained with the attention-based constraint loss proposed in [18] to prevent the attention function from using future context, while the bigger uses 8-head attention and produces better accuracy. The smaller model *S1* can be used either for inference or to extract the endpoint of a hypothesis prefix following [18]. Additionally, we experiment with a transformer model which has 24 self-attention encoder layers and 8 decoder layers.

Table 1 shows the offline performance of all the investigated S2S models in this work. The big LSTM model achieved the best WER performance while the transformer performs worse. However, the transformer is very efficient to supplement the LSTM models in the combination. The ensemble of 3 models (labeled as *E3*) results in a single system that achieved a 5.0% WER on the SWB test set, which is on par with the state-of-the-art performance on this benchmark.

Table 2: *Computation and confidence latency when using shared prefix condition.*

Model	Beam Size	Comp.	Conf.	SWB
S1	8	0.10	1.50	5.8
S2	8	0.13	1.55	5.6
T1	8	0.19	1.50	5.8
T1	6	0.16	1.35	5.9
T1	4	0.12	0.70	6.6
E1	8	0.18	1.55	5.3
E2	8	0.29	1.50	5.0
E2	6	0.25	1.30	5.1
E2	4	0.20	0.80	5.7

5.2. Latency with Shared Prefix

We use an audio chunk size of 300ms to perform incremental inference with the systems in Table 1. All inferences were performed on a single Nvidia Titan RTX GPU. Table 2 shows the WERs for SWB, *computation latency* and *confidence latency* (see Section 3) for different beam sizes when only using the *share prefix* strategy for stability detection.

As can be seen, the confidence latency is much larger than the computation latency in all the experiments and shown to be a more critical factor for final latency improvement. The systems involving multiple S2S models require more computational power, however, they obtain better confidence latency and accuracy due to the reduction of model uncertainty.

When using a high beam size (e.g., 8), all the experimental systems can achieve their offline accuracy. This result reveals interesting observations for making online S2S ASR systems. First, as this condition is reliable among different S2S architectures, it shows that all S2S ASR models may share the same characteristic in which they tend not to use further context for the inference of a given prefix at a particular time. This observation is consistent with the finding in [18] for the LSTM-based S2S model. Secondly, it proves that the use of bidirectional encoders in online conditions is possible and even results in the same optimal accuracy as in offline inference. Lastly, it reveals a unified approach to build online ASR for different S2S architectures. As an attractive advantage, this approach does not require model modifications.

The best system using the shared prefix condition achieved a WER of 5.0% and suffered an average delay of 1.79 seconds which is slightly slower than the one with lowest latency.

5.3. Trade-off for Better Latency

To further improve the latency, we use both the stability detection strategies from Section 2.3. We do the combination via a logical OR which means the stability is detected as soon as one of the conditions applies. At the end, we can trade-off latency against accuracy as the function of the term Δ – the delay time needed to finalize the endpoint of a prefix. Figure 3 presents the trade-off curves for two systems, *S1* and *E2*. In both systems, the model *S1* is used for detecting the *best-ranked prefix* condition.

As can be seen, both systems can achieve much better latency (of only 1.30 seconds) with only a slight increase in WER (e.g., 0.1% absolute). The ensemble system *E2* achieves a latency of 0.85 seconds while yielding the same accuracy as *S1*. Human performance (5.5%) can be reached with an average delay of only 1 second. Note that, the WER for human performance was extracted as the average of the two studies [23] and [24].

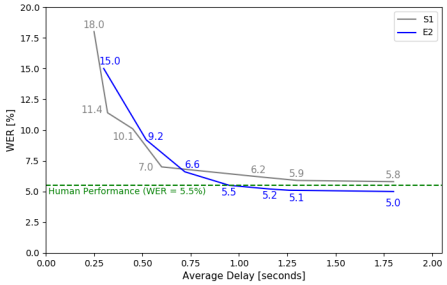

Figure 3: *Trade-off between latency and accuracy. Beam size of 8 is used for both systems.*

5.4. Compared to Other Works

Table 3 presents the WER performance from recent studies for online and offline conversational speech recognition systems. Human WER performance was obtained in 2016 with a combination of different HMM hybrid ASR systems. While until 2019 and 2020, new records on offline conversational speech was set with end-to-end sequence-to-sequence ASR systems. We found only a few attempts [31, 32] that make streaming ASR for this benchmark. In these studies, the accuracy between offline and streaming conditions was shown to be in clear margins. In a different manner, we show the offline accuracy can be possibly reached with our proposed low-latency S2S system. Our best achieved online WER is slightly behind the state-of-the-art offline performance on the Switchboard benchmark.

Table 3: *Results from other works on SWB test set.*

Model	Train. Data	Condition	WER
Hybrid [24] (2017)	SWB+Fisher	*Offline*	5.5
Hybrid [33] (2018)	SWB+Fisher	*Offline*	5.1
S2S [7] (2019)	SWB+Fisher	*Offline*	5.2
S2S [34] (2020)	SWB+Fisher	*Offline*	4.9
S2S [35] (2020)	SWB+Fisher	*Offline*	4.8
CTC [31] (2019)	SWB	*Streaming*	9.1
Transducer [32] (2020)	SWB	*Offline*	12.8
Transducer [32] (2020)	SWB	*Streaming*	17.0
Ours	SWB+Fisher	*Low-latency*	5.0

6. Conclusion

We have shown a unified approach to construct online and low-latency ASR systems for different S2S architectures. The proposed online system employing three S2S models works either in an accuracy-optimized fashion that achieves state-of-the-art performance on telephone conversation speech or in a very low-latency manner while still producing the same or better accuracy as the reported human performance.

7. Acknowledgement

The project ELITR leading to this publication has received funding from the European Unions Horizon 2020 Research and Innovation Programme under grant agreement No 825460.

8. References

[1] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015.

[2] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP 2016*. IEEE, 2016, pp. 4960–4964.

[3] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *ICASSP 2018*, 2018.

[4] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," in *Proc. Interspeech 2018*, 2018.

[5] C. Weng, J. Cui, G. Wang, J. Wang, C. Yu, D. Su, and D. Yu, "Improving attention based sequence-to-sequence models for end-to-end english conversational speech recognition," in *Proc. Interspeech 2018*, 2018.

[6] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Muller, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *Proc. of Interspeech 2019*, 2019.

[7] T.-S. Nguyen, S. Stüker, J. Niehues, and A. Waibel, "Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation," in *ICASSP 2020*. IEEE, 2020, pp. 7689–7693.

[8] T. S. Nguyen, J. Niehues, E. Cho, T.-L. Ha, K. Kilgour, M. Muller, M. Sperber, S. Stueker, and A. Waibel, "Low latency asr for simultaneous speech translation," *arXiv preprint arXiv:2003.09891*, 2020.

[9] J. Niehues, T. S. Nguyen, E. Cho, T.-L. Ha, K. Kilgour, M. Müller, M. Sperber, S. Stüker, and A. Waibel, "Dynamic transcription for low-latency speech translation," in *Proc. of Interspeech 2016*, 2016.

[10] N. Jaitly, Q. V. Le, O. Vinyals, I. Sutskever, D. Sussillo, and S. Bengio, "An online sequence-to-sequence model using partial conditioning," in *Advances in Neural Information Processing Systems*, 2016, pp. 5067–5075.

[11] C. Raffel, M.-T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," in *Proceedings of the 34th ICML*, 2017, pp. 2837–2846.

[12] C.-C. Chiu and C. Raffel, "Monotonic chunkwise attention," *arXiv preprint arXiv:1712.05382*, 2017.

[13] R. Fan, P. Zhou, W. Chen, J. Jia, and G. Liu, "An online attention-based model for speech recognition," *Proc. Interspeech 2019*, pp. 4390–4394, 2019.

[14] H. Miao, G. Cheng, P. Zhang, T. Li, and Y. Yan, "Online hybrid ctc/attention architecture for end-to-end speech recognition," *Proc. of Interspeech 2019*, pp. 2623–2627, 2019.

[15] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Towards online end-to-end transformer automatic speech recognition," *arXiv preprint arXiv:1910.11871*, 2019.

[16] H. Miao, G. Cheng, C. Gao, P. Zhang, and Y. Yan, "Transformer-based online ctc/attention end-to-end speech recognition architecture," in *ICASSP 2020*, 2020, pp. 6084–6088.

[17] N. Moritz, T. Hori, and J. Le, "Streaming automatic speech recognition with the transformer model," in *ICASSP 2020*. IEEE, 2020, pp. 6074–6078.

[18] T.-S. Nguyen, N.-Q. Pham, S. Stueker, and A. Waibel, "High performance sequence-to-sequence model for streaming speech recognition," *Proc. of Interspeech 2020*, 2020.

[19] C. Wu, Y. Wang, Y. Shi, C.-F. Yeh, and F. Zhang, "Streaming transformer-based acoustic models using self-attention with augmented memory," *arXiv preprint arXiv:2005.08042*, 2020.

[20] S. Zhang, Z. Gao, H. Luo, M. Lei, J. Gao, Z. Yan, and L. Xie, "Streaming chunk-aware multihead attention for online end-to-end speech recognition," *arXiv preprint arXiv:2006.01712*, 2020.

[21] K. Kumar, C. Liu, Y. Gong, and J. Wu, "1-d row-convolution lstm: Fast streaming asr at accuracy parity with lc-blstm," *Proc. Interspeech 2020*, pp. 2107–2111, 2020.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.

[24] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim *et al.*, "English conversational telephone speech recognition by humans and machines," *arXiv preprint arXiv:1703.02136*, 2017.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017.

[26] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, March 1989.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[28] P. Brown, J. Spohrer, P. Hochschild, and J. Baker, "Partial traceback and dynamic programming," in *ICASSP 82*, vol. 7. IEEE, 1982, pp. 1629–1632.

[29] S. Wachsmuth, G. A. Fink, and G. Sagerer, "Integration of parsing and incremental speech recognition," in *9th of the EUSIPCO 1998*, 1998, pp. 1–4.

[30] E. O. Selfridge, I. Arizmendi, P. A. Heeman, and J. D. Williams, "Stability and accuracy in incremental speech recognition," in *Proceedings of the SIGDIAL 2011 Conference*, 2011, pp. 110–119.

[31] K. Audhkhasi, G. Saon, Z. Tüske, B. Kingsbury, and M. Picheny, "Forget a bit to learn better: Soft forgetting for ctc-based automatic speech recognition." in *INTERSPEECH*, 2019, pp. 2618–2622.

[32] G. Kurata and G. Saon, "Knowledge distillation from offline to streaming rnn transducer for end-to-end speech recognition," *Proc. Interspeech 2020*, pp. 2117–2121, 2020.

[33] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5934–5938.

[34] W. Wang, Y. Zhou, C. Xiong, and R. Socher, "An investigation of phone-based subword units for end-to-end speech recognition," *arXiv preprint arXiv:2004.04290*, 2020.

[35] Z. Tüske, G. Saon, K. Audhkhasi, and B. Kingsbury, "Single headed attention based sequence-to-sequence model for state-of-the-art results on switchboard-300," *arXiv preprint arXiv:2001.07263*, 2020.

C KIT's IWSLT 2021 Offline Speech Translation System

KIT's IWSLT 2021 Offline Speech Translation System

Tuan-Nam Nguyen, Thai-Son Nguyen, Christian Huber, Maximilian Awiszus, Ngoc-Quan Pham, Thanh-Le Ha, Felix Schneider, Sebastian Stüker, Alexander Waibel
Karlsruhe Institute of Technology
firstname.lastname@kit.edu

Abstract

This paper describes KIT'submission to the IWSLT 2021 Offline Speech Translation Task. We describe a system in both cascaded condition and end-to-end condition. In the cascaded condition, we investigated different end-to-end architectures for the speech recognition module. For the text segmentation module, we trained a small transformer-based model on high-quality monolingual data. For the translation module, our last year's neural machine translation model was reused. In the end-to-end condition, we improved our Speech Relative Transformer architecture to reach or even surpass the result of the cascade system.

1 Introduction

As in previous years, the cascade system's pipeline is constituted by an ASR module, a text segmentation module and a machine translation module. In this year's evaluation campaign, we investigated only sequence-to-sequence ASR models with three architectures. The segmentation module is basically a monolingual system which translates a disfluent, broken, uncased text (i.e. ASR outputs) into a more fluent, written-style text with punctuations in order to match the data conditions of the translation system. The machine translation module's architecture is the same as the previous year's. For the end-to-end system, we improved from our last year's Speech Relative Transformer architecture (Pham et al., 2020a). As a result, the end-to-end system can produce better results on certain test sets and approach the performance on some others compared to the cascade system this year, while the end-to-end system was the dominant approach last year.

The rest of the paper is organized as followed. Section 2 describes the data set used to train and test the system. It is then followed by Section 3 providing the description and experimental results of both the cascade and the end-to-end system. In the end, we conclude the paper with Section 4.

2 Data

Speech Corpora. For training and evaluation of our ASR models, we used Mozilla Common Voice v6.1 (Ardila et al., 2019), Europarl (Koehn, 2005), How2 (Sanabria et al., 2018), Librispeech (Panayotov et al., 2015), MuST-C v1 (Di Gangi et al., 2019), MuST-C v2 (Cattoni et al., 2021) and Tedlium v3 (Hernandez et al., 2018) dataset. The data split is presented in the following table 1.

Table 1: Summary of the English data-sets used for speech recognition

Corpus	Utterances	Speech data [h]
A: Training Data		
Mozilla Common Voice	1225k	1667
Europarl	33k	85
How2	217k	356
Librispeech	281k	963
MuST-C v1	230k	407
MuST-C v2	251k	482
Tedlium	268k	482
B: Test Data		
Tedlium	1155	2.6
Librispeech	2620	5.4

Text Corpora. We collected the text parallel training data as presented in Table 2.

3 Offline Speech Translation

We address the offline speech translation task by two main approaches, namely cascade and end-to-end. In the cascade condition, the ASR module (Section 3.1) receives audio inputs and generates raw transcripts, which will then pass through a Segmentation module (Section 3.2) to formulate well normalized inputs to our Machine Translation module (Section 3.3). The MT outputs are the final outputs of the cascade system. On the other hand,

Proceedings of the 18th International Conference on Spoken Language Translation, pages 125–130
Bangkok, Thailand (Online), August 5–6, 2021.

Table 2: Text Training Data

Dataset	Sentences
TED Talks (TED)	220K
Europarl (EPPS)	2.2MK
CommonCrawl	2.1M
Rapid	1.21M
ParaCrawl	25.1M
OpenSubtitles	12.6M
WikiTitle	423K
Back-translated News	26M

the end-to-end architecture is trained to directly translate English audio inputs into German text outputs (Section 3.4).

3.1 Speech Recognition

Data preparation and Segmentation tool After collecting all audios from all data sets mentioned in Section 2, we calculated 40 features of Mel-filterbank coefficients for ASR training. To generate labels for the sequence-to-sequence ASR models, we used the Sentence-Piece toolkit (Kudo and Richardson, 2018) to train 4000 different byte-pair-encoding (BPE). The WerRTCVAD toolkit (Wiseman, 2016) was used to segment the audio in the testing phase.

Model As in previous years (Pham et al., 2019a, 2020b), we used only sequence-to-sequence ASR models, which are based on three different network architectures: The long short-term memory (LSTM), the Transformer and the Conformer. LSTM-based models (Nguyen et al., 2020) consist of 6 bidirectional layers for the encoder and 2 unidirectional layers for the decoder, both encoder and decoder layers have 1536 units. The Transformer-based models presented in (Pham et al., 2019b) have 24 layers for the encoder and 8 layers for the decoder. The Conformer-based models (Gulati et al., 2020) comprise 16 layers for the encoder and 6 layers for the decoder. In both the Transformer-based and the Conformer-based models, the size of each layer is 512 and the size of the hidden state in the feed-forward sublayer is 2048. The speech data augmentation technique was used to reduce overfitting as described in (Nguyen et al., 2020). In order to train a deep network effectively, we also applied Stochastic Layers (Pham et al., 2019b) with a dropping layer rate of 0.5 on both Transformer-based and Conformer-based models.

3.2 Text Segmentation

The text segmentation in the cascaded pipeline serves as a normalization on the ASR output, which usually lacks punctuation marks, proper sentence boundaries and reliable casing. On the other hand, the machine translation system is often trained on well-written, high-quality bilingual data. Following the idea from (Sperber et al., 2018a), we build the segmentation as a monolingual translation system, which translates from lower-cased, without-punctuation texts into texts with case information and punctuation, prior to the machine translation module.

The monolingual translation for text segmentation is implemented using our neural speech translation framework `NMTGMinor`[1](Pham et al., 2020a). It is a small transformer architecture, consisting of a 4-layer encoder and 4-layer decoder, in which each layer' size is 512, while the inner size of feed-forward network inside each layer is 2048. The encoder and decode are self-attention blocks, which have 4 parallel attention heads. The training data for that are the English part extracted from available multilingual corpora: EPPS, NC, Global Voices and TED talks. We trained the model for 10 epochs, then we fine-tuned it on the TED corpus for 30 epochs more with stronger drop-out rate. Furthermore, to simulate possible errors in the ASR outputs, a similar model is trained on artificial noisy data and the final model is the ensemble of the two models.

The trained model is then utilized to translate the ASR outputs in a shifting window manner and the decisions are drawn by a simple voting mechanism. For more details, please refer to (Sperber et al., 2018a).

3.3 Machine Translation

For the machine translation module, we re-use the English→German machine translation model from our last year' submission to IWSLT (Pham et al., 2020b). More than 40 millions sentence pairs being extracted from TED, EPPS, NC, CommonCrawl, ParaCrawl, Rapid and OpenSubtitles corpora were used for training the model. In addition, 26 millions sentence pairs are generated from the back-translation technique by a German→English translation system. A large transformer architecture was trained with Relative Attention. We adapted to the in-domain by fine-tuning on TED talk data with

[1] https://github.com/quanpn90/NMTGMinor

stricter regularizations. The same adapted model was trained on noised data synthesized from the same TED data. The final model is the ensemble of the two.

3.4 End-to-End Model

Corpora This year, the training data consists of the second version of the MUST-C corpus (Di Gangi et al., 2019), the Europarl corpus (Iranzo-Sánchez et al., 2020), the Speech Translation corpus and the CoVoST-2 (Wang et al., 2020) corpus provided by the organizer. The speech features are generated with the in-house Janus Recognition Toolkit. The ST dataset is handled with an additional filtering step using an English speech recognizer (trained with the its transcripts with the additional Tedlium-3 training data).

Following the success of generating synthetic audio utterances, the transcripts in the Tedlium-3 corpus are translated into German using the cascade built in the previous year's submission (Pham et al., 2020b). In brief, the translation process required us to preserve the audio-text alignment from the original data collection and segmentation process. As a results, we used the Transformer-based punctuation inserting system from IWSLT2018 (Sperber et al., 2018b) to reconstruct the punctuations for the transcripts followed by the translation process that preserves the same segmentation information. Compared to the human translation from the speech translation datasets, this translation is relative noisier and incomplete (due to the segmentations are not necessarily aligned with grammatically correct sentences).

The end result of the filtering and synthetic creation process is the complete translation set, as summarised in Table 3

Table 3: Training data for E2E translation models.

Data	Utterances	Total time
MuST-C	229K	408h
Europarl	32K	60h
Speech Translation	142K	160h
Tedlium-3	268K	415h
CoVoST	288K	424h

During training, the validation data is the Development set of the MuST-C corpus. The reason is that the SLT testsets often do not have the aligned audio and translation, while training end-to-end models often rely on perplexity for early stopping.

Modeling The main architecture is the deep Transformer (Vaswani et al., 2017) with stochastic layers (Pham et al., 2019b). The encoder self attention layer uses Bidirectional relative attention (Pham et al., 2020a) which models the relative distance between one position and other positions in the sequence. This modeling is bidirectional because the distance is distinguished for each direction from the perspective of one particular position. The main models use a "Big" configuration with 16 encoder layers and 6 decoder layers, and they are randomly dropped in training according to the linear schedule presented in the original work, where the top layer has the highest dropout rate $p = 0.5$. The model size of each layer is 1024 and the inner size is 4096. We experimented with different activation functions including GELU (Hendrycks and Gimpel, 2016), SiLU (Elfwing et al., 2018) and the gated variants similar to the gated linear units (Dauphin et al., 2017). Also, each transformer block (encoder and decoder) is equipped with another feed-forward neural network in the beginning (Lu et al., 2019). Our preliminary experiments showed that GeLU and SiLU provided a slightly better performance than ReLU, and our final model is the ensemble of the three configurations that are identical except the activation functions.

First, the encoders are pretrained using the data portions containing English texts to make training SLT stable. With the initialized encoder, the networks can be trained with an aggressive learning rate with 4096 warm-up steps. Label-smoothing and dropout rates are set at 0.1 and 0.3 respectively for all models. Furthermore, all speech inputs are augmented with spectral augmentation (Park et al., 2019; Bahar et al., 2019). All models are trained for 200000 steps, each consists of accumulated 360000 audio frames. Using the model setup like above, we managed to fit a batch size of around 16000 frames to 24 GB of GPU memory.

Speech segmentation As reflected from last year's experiments, audio segmentation plays an important role in the performance of the whole system, and the end-to-end model unfortunately does not have control of segmentation, as it is a prerequisite before training one. During evaluation, we relied on the WerRTCVAD toolkit (Wiseman, 2016) to cut the long audio files into segments of reasonable length, and the tool is also able to rule out silence and events that do not belong to human speech, such as noise and music.

Overall, we improved the submission from last year (Pham et al., 2020b) using stronger models together with a more accurate segmentation tool.

3.5 Experimental Results

3.5.1 Cascade Offline Speech Translation

Speech Recognition. We tested our ASR systems on two datasets, Tedlium and Libri test set. The ensemble of LSTM-based and Conformer-based sequence-to-sequence model provide the best results, which are 2.4 and 3.9 WERs respectively for two test set Table 4.

Table 4: WER on Libri and Tedlium sets

Data	Libri	Tedlium
Conformer-based	3.0	4.8
Transformer-based	3.2	4.9
LSTM-based	**2.6**	**3.9**
Ensemble	**2.4**	3.9

Machine Translation. We do not train any new machine translation module but re-use last year's model, thus, we do not conduct experiments and comparisons with different machine translation systems. We submitted one cascased model with our audio segmentation.

3.5.2 End-to-end Offline Speech Translation

Our models are tested on two different setups. On the one hand, we evaluated the model on the tst-COMMON (2nd version) of the MuST-C corpora. Due to the incompatibility between the models and the audio data that requires resegmentation, we rely on the dev and test sets of MuST-C to evaluate the ability to translate on "ideal" conditions. As mentioned above, our ensemble managed to reach 32.4 BLEU points on this test set[2].

On the other hand, we used the testsets from 2010 to 2015 to measure the progress from last year in the condition requiring audio segmentation. In this particular comparison as shown in Table 5, we showed that using a stronger model together with better voice detection not only improves the SLT results by up to 1.9 BLEU points (in *tst2014*) but also outperforms the strong cascade in 2 different sets: *tst2013* and *tst2014*, in which the difference could be even 1 BLEU point. There is still a performance gap in the last two tests, however, a strong E2E system can now trade blow with a strongly tuned cascade. The deciding factor, in our opinion, is audio segmentation because this is the sole advantage of the cascade which can recover from badly cut segments[3].

Table 5: ST: Translation performance in BLEU↑ on IWSLT testsets (re-segmentation required). Progressive results from this year and last year end-to-end (E2E) and cascades (CD) are provided.

Testset →	**CD 2020**	**E2E 2020**	**E2E 2021**
tst2010	**26.68**	24.27	25.28
tst2013	28.60	28.13	**29.62**
tst2014	25.64	25.46	**27.32**
tst2015	**24.95**	21.82	22.13

4 Conclusion

In this year's evaluation campaign, the end-to-end model proves to be a very promising approach since it can compete or even transcend the best cascade model in offline speech translation task. As a note for future work, we would like to investigate two-stage speech translation models (Sperber et al., 2019) using transformer architectures and compare them with our recent speech translation end-to-end models.

Acknowledgments

The work leading to these results has received funding from the European Union under grant agreement n°825460 and the Federal Ministry of Education and Research (Germany)/DLR Projektträger Bereich Gesundheit under grant agreement n° 01EF1803B.

References

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. 2019. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*.

Parnia Bahar, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. On using specaugment for end-to-end speech translation. *arXiv preprint arXiv:1911.08876*.

Roldano Cattoni, Mattia Antonino Di Gangi, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2021.

[2] Unfortunately the comparison to last year tst-COMMON (30.6 is not available due to version mismatch.

[3] Changing the VAD parameters does not affect the performance of the cascade significantly, while the E2E can be badly afffected

Must-c: A multilingual corpus for end-to-end speech translation. *Computer Speech & Language*, 66:101155.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented transformer for speech recognitio. In *Proc. Interspeech 2020*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia Tomashenko, and Yannick Estève. 2018. Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation. In *International Conference on Speech and Computer*, pages 198–208. Springer.

Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerdà, Javier Jorge, Nahuel Roselló, Adrià Giménez, Albert Sanchis, Jorge Civera, and Alfons Juan. 2020. Europarl-st: A multilingual corpus for speech translation of parliamentary debates. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8229–8233. IEEE.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*.

Thai-Son Nguyen, Sebastian Stueker, Jan Niehues, and Alex Waibel. 2020. Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation. *arXiv preprint arXiv:1910.13296*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Ngoc-Quan Pham, Thanh-Le Ha, Tuan-Nam Nguyen, Thai-Son Nguyen, Elizabeth Salesky, Sebastian Stüker, Jan Niehues, and Alex Waibel. 2020a. Relative Positional Encoding for Speech Recognition and Direct Translation. In *Proc. Interspeech 2020*, pages 31–35.

Ngoc-Quan Pham, Thai-Son Nguyen, Thanh-Le Ha, Juan Hussain, Felix Schneider, Jan Niehues, Sebastian Stüker, and Alexander Waibel. 2019a. The iwslt 2019 kit speech translation system. In *Proceedings of the 16th International Workshop on Spoken Language Translation (IWSLT 2019)*.

Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Muller, and Alex Waibel. 2019b. Very deep self-attention networks for end-to-end speech recognition. *arXiv preprint arXiv:1904.13377*.

Ngoc-Quan Pham, Felix Schneider, Tuan-Nam Nguyen, Thanh-Le Ha, Thai-Son Nguyen, Maximilian Awiszus, Sebastian Stüker, and Alexander Waibel. 2020b. Kit's iwslt 2020 slt translation system. In *Proceedings of the 17th International Workshop on Spoken Language Translation (IWSLT 2020)*.

Ramon Sanabria, Ozan Caglayan, Shruti Palaskar, Desmond Elliott, Loïc Barrault, Lucia Specia, and Florian Metze. 2018. How2: a large-scale dataset for multimodal language understanding. *arXiv preprint arXiv:1811.00347*.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2019. Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation. In *Proc. ACL 2019*.

Matthias Sperber, Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, Thanh-Le Ha, Sebastian Stüker, and Alex Waibel. 2018a. KIT's IWSLT 2018 SLT Translation System. In *15th International Workshop on Spoken Language Translation 2018*. IWSLT.

Matthias Sperber, Ngoc Quan Pham, Thai Son Nguyen, Jan Niehues, Markus Müller, Thanh-Le Ha, Sebastian Stüker, and Alex Waibel. 2018b. KIT's IWSLT 2018 SLT Translation System. In *"Proceedings of the 15th International Workshop on Spoken Language Translation (IWSLT 2018)"*, Brussels, Belgium.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Changhan Wang, Anne Wu, and Juan Pino. 2020. Covost 2: A massively multilingual speech-to-text translation corpus.

John Wiseman. 2016. python-webrtcvad. https://github.com/wiseman/py-webrtcvad.

D Efficient Weight factorization for Multilingual Speech Recognition

Efficient Weight factorization for Multilingual Speech Recognition

Ngoc-Quan Pham[1] *Tuan-Nam Nguyen*[1] *Sebastian Stueker*[1] *Alex Waibel*[1,2]

[1]Interactive Systems Lab, Karlsruhe Institute of Technology, Karlsruhe, Germany
[2]Carnegie Mellon University, Pittsburgh PA, USA
ngoc.pham@kit.edu

Abstract

End-to-end multilingual speech recognition involves using a single model training on a compositional speech corpus including many languages, resulting in a single neural network to handle transcribing different languages. Due to the fact that each language in the training data has different characteristics, the shared network may struggle to optimize for all various languages simultaneously. In this paper we propose a novel multilingual architecture that targets the core operation in neural networks: linear transformation functions. The key idea of the method is to assign fast weight matrices for each language by decomposing each weight matrix into a shared component and a language dependent component. The latter is then factorized into vectors using rank-1 assumptions to reduce the number of parameters per language. This efficient factorization scheme is proved to be effective in two multilingual settings with 7 and 27 languages, reducing the word error rates by 26% and 27% rel. for two popular architectures LSTM and Transformer, respectively.

Index Terms: speech recognition, multilingual, transformer, lstm, weight factorization, weight decomposition

1. Introduction

Multilingual modeling has been an important topic in applying sequence-to-squence models to language applications ranging from machine translation [1, 2] to automatic speech recognition (ASR) [3]. It is possible to employ one single neural model for multiple datasets with different languages with the goal of capturing the shared features between the languages. This method has been widely used to help under-resourced languages benefiting from the knowledge acquired from the richer counterparts.

It is noticeable that the recent multilingual neural models are based on a semi-shared mechanism in which the largest body of the network architecture is exposed to all languages, while a smaller weight subset provides a language specific bias. This was shown to be more effective in a multilingual scenario than fully sharing the whole network[1, 2] since each language has certain unique features, and the single architecture often struggles to handle a variety of languages [4].

There are two main drawbacks that are typically presented in the existing implementations of the semi-shared mechanism. On the one hand, the implementations often depends heavily on a certain architecture being popular at the time, and the given improvement is going to be diminished when a new architecture evolves. For example, the language-specifically biased attention [5] modified the self-attention architecture [6] specifically based on the assumption that each language can benefit from a bias added to the attention scores. On the other hand, the language-dependent components might require a considerable amount of parameters and struggles to scale to the number of languages. For example, the language adapters added to the Transformer layers [7] are essentially feed-forward neural network layers being similar to the counterpart already in the shared Transformer body. A scenario with 20 languages consequently generates hundreds of these layers accounting for a large amount of parameters to be optimized.

In this work, we propose a multilingual architecture using a factorization scheme that is both effective and highly scalable with the number of languages involved. Moreover, this scheme is applicable to any neural architectures as long as matrix-vector multiplication is the dominant operation. The key idea of our work is that each weight matrix in the shared architecture can be factorized into a shared component and multiple additive and multiplicative language dependent components. While each language is assigned with extra weights to learn distinctive features, simplicity and scalability are achieved by further representing those weights into as a rank-1 matrix, thus can be factored into two vectors. This method is demonstrated to be computational friendly with a minimal overhead and can be applied to a arbitrary neural architecture.

Subsequently, this weight factorization method is then evaluated on two different scenarios: one with 7 languages having similar amounts of data, and one with 27 languages with various extremely low resource data. The method is implemented on two commonly used architectures: Long Short-Term Memories (LSTM) and Transformers which show that both types of networks can benefit by weight factorization in multilingual ASR. The reduction of error rate can be up to 47% rel. in the case of low-resource languages such as Japanese[1] and 15.5% rel. on average with the moderately sized languages.

2. Methodology

A neural speech-to-text model transforms a source speech input with N frames $X = x_1, x_2, \ldots, x_N$ into a target text sequence with M tokens $Y = y_1, y_2, \ldots, y_M$. The encoder transforms the speech input into higher level feature vectors $h^X_{1\ldots N}$. The decoder jointly learns to generate the output distribution o_i based on the previous target tokens $y_1, y_2, \ldots, y_{i-1}$ while looking for the relevant inputs from the input via the attention mechanism [8, 6].

$$h^X_{1\ldots N} = ENCODER(x_1 \ldots x_N) \quad (1)$$

$$h^Y_i = DECODER(y_i, y_{1\ldots i-1}) \quad (2)$$

$$c_i = \text{ATTENTION}(h^Y_i, h_{1\ldots N}) \quad (3)$$

$$o_i = \text{SOFTMAX}(c_i + h^Y_i) \quad (4)$$

$$y_{i+1} = sample(o_i) \quad (5)$$

[1]Error is measured in characters error rate here.

Notably, there is a large variety of model architectures that implement this encoder-decoder design. The core networks in the encoder and decoder range from LSTM [9], convolution/TDNN [10, 11] to self-attention [12] or even a mix of the above [**?**]

The universal multilingual framework [1, 2] employs a single model to learn on a joint training dataset containing multiple languages, which is different than the predating multi-way encoder-decoder approach [13].

2.1. Multilingual weight composition

It can be seen that, the common ground of the aforementioned architectures is the usage of linear combinations of lower level features $X \in R^D$ which can be expressed as the matrix multiplication between input X and a weight matrix W. For example, the LSTM contains four different projections for its forget, input, output gates and candidate content [14], as can be seen in Equation 6.

$$f_t = sigmoid(W_{fx}^\top X_t + W_{fh}^\top H_{t-1} + b_f) \quad (6)$$
$$i_t = sigmoid(W_{ix}^\top X_t + W_{ih}^\top H_{t-1} + b_i) \quad (7)$$
$$\hat{c}_t = tanh(W_{cx}^\top X_t + W_{ch}^\top H_{t-1} + b_c) \quad (8)$$
$$o_t = sigmoid(W_{ox}^\top X_t + W_{oh}^\top H_{t-1} + b_o) \quad (9)$$

Similarly, the main components of the Transformer layers are self-attention layers and feed-forward layers. While the latter are fundamentally two layers of linear projections, the former is also comprised of linear projections that generate queries Q, keys K and values V from the input X:

$$Q = W_Q^\top X \quad (10)$$
$$K = W_K^\top X \quad (11)$$
$$V = W_V^\top X \quad (12)$$
$$SelfAtt(X) = softmax(QK^\top)V \quad (13)$$

The main idea here is that each matrix multiplication $Y = W^T X$ in the multilingual model can be decomposed into a function of shared weights W_S and additional language dependent weights W_{ML} and W_{BL}

$$Y = (W_S \cdot W_{ML} + W_{BL})^\top X \quad (14)$$
$$= (W_S \cdot W_{ML})^\top X + W_{BL}^\top X \quad (15)$$

Here the added weights include the first multiplicative term W_{ML} that directly change the magnitude and direction of the shared weights W_S and the biased term W_{BL} provides the network with a content-based bias depending on the input features X. Each language maintains a distinctive set of W_{ML} and W_{BL} so that the whole architecture is semi-shared.

2.2. Factorization

There is, however, an obstacle that both W_{ML} and W_{BL} require to be the same size with W_S, which makes the language dependent weights dominate the shared weights, while the intuition is the opposite. Fortunately, it is possible to use rank-1 matrices $\bar{W} \in R^{D_{in} \times D_{out}}$ that can be factorized into vectors [15, 16], for example with two vectors $r \in R^{D_{in}}$ and $s \in R^{D_{out}}$ such that $\bar{W} = rs^\top$ which reduces the number of parameters from $D_{in} \times D_{out}$ to $D_{in} + D_{out}$.

One drawback in this method is the lacking representational power of Rank-1 matrices. One solution is to modify the factorization into using k vectors per language so that there are k independent weight factors followed by a summation, which increases the rank of the additional weight matrices.

$$\bar{W} = \sum_i^k r_i s_i^\top \quad (16)$$

2.3. Computational cost

The factorization above is applied to both W_{ML} and W_{BL} to ensure that the dominated force is still the shared weights, while each language at $k = 1$ is characterized by an additional $\frac{D_{in}+D_{out}}{D_{in} \times D_{out}}$ amount of weights. In a typical network architecture with D_{in} and D_{out} being typically $512-2048$, this amounts for $0.1 - 0.3$ percents of the total network's weights per language, therefore scalable to hundreds.

On the time complexity, the amount of extra computation comes from generating the combinatory weight W from W_S and the multiplicative/bias terms W_{ML} and W_{BL}. Fortunately, this overhead coming from element-wise multiplication and addition is rather small compared to the matrix multiplication. More importantly, it is possible to utilize the optimized implementation of the original network[2] which minimizes the computational requirements of our approach.

On the same subject, [15] proved that W does not have to be explicitly computed, but their approach required to rewrite the graph operation for the core networks in popular deep learning frameworks.

3. Related works and Comparison

In the world of speech recognition, training a single recognizer for multiple languages is not a thematic stranger [3] from Hidden Markov Model (HMM) based models [17, 18], hybrid models [19] to end-to-end neural based models with CTC [20, 21] or sequence-to-sequence models [22, 5, 23, 24, 25, 26], with the last approach being inspired by the success of multilingual machine translation [1, 2]. The literature especially mentions the merits of disclosing the language identity (when the utterance is supposed to belong to a single language) to the model, whose architecture is designed to incorporate the language information.

One of the manifestations is language gating from either language embeddings [21] or language codes [20, 27] that aim at selecting a subset of the neurons in the network hidden layer. In our current approach, this effect can be achieved by factorizing further Equation 15 [15]:

$$Y = (W_S \cdot W_{ML})^\top X + W_{BL}^\top X \quad (17)$$
$$= (W_S \cdot (r_m s_m^\top)^\top X + (r_a s_a^\top)^\top X \quad (18)$$
$$= (W_S^\top (X \cdot s_m) \cdot r_m) + (r_a s_a^\top)^\top X \quad (19)$$

In Equation 17, the multiplicative matrix W_{ML} is factorized by two vectors r_m and s_m. The left hand size of Equation 19 shows us that the those vectors can be learned to gate the input vector X and the output of the linear projection

[2] such as the CUDA implementations of LSTM and Self-Attention

$(W_S^\top (X \cdot s_m)$. This intuition also suggested us to initialize r_m and s_m to one-vectors similarly to normalization techniques [28, 29]. Since layer normalization often comes before the linear projection layers in Transformers, this scheme also helps our model to generalize to assigning to each language a different normalization scale and variance [30].

On the other hand, the right hand side of Equation 19 gives us the bias to the linear projection which has been used in either language embeddings [31] and customized attention layers with language biases [5].

A different line of research involves using language code [20] to differentiate language coming from a separate classifier. The language code is often trained separately and then mixed into the ASR architecture later [27] giving the lingual bias. Our method can provide a similar effect with end-to-end training and without architectural modification. The advantage of this method is to exploit unlabeled (transcript-wise) data to gather language-specific information.

Architecture wise, [7] makes the network language aware using language-dependently adaptive feed-forward layers at the end of each Transformer block. While this method is able to be effective in translation [32] and speech recognition scenarios [5], it requires a considerable amount of parameters per language[3] and probably becomes incompatible with future architectures because it is specifically designed for Transformers.

The closest to our work is the parameter generator [4] that composes a weight matrix $W \in R^{D_{in} \times D_{out}}$ using a shared tensor $W_S \in R^{D_{in} \times D_{out} \times D_L}$ and a language embedding vector $L \in R^{D_L}$. The main disadvantage with that approach is that the amount of parameters linearly scales in the size of the language embedding D_L, and the whole body of parameters participates in every language. Our initial experiments cannot produce a reasonable result for a straight comparison, partly because the memory is quickly overwhelmed by the number of parameters.

For a larger context, weight factorization has been investigated to generate distinguishable yet cheaper copies of an existing network to allow for economical ensembles [15], Bayesian networks [33] or continual learning without catastrophe forgetting [16]. Similar ideas to use different weights for different languages have been investigated early on by [34].

4. Experiments

4.1. Datasets

The effects of the weight factorization methods are measured on datasets publicly available including Mozilla Common Voice [35] containing up to 27 languages, Euronews [36] and Europarl-ST [37] having 4 and 9 languages respectively. The preprocessing steps include converting audio into 40-dimensional feature frames, and generating BPE for each language with 256 codes each. Only Japanese and Chinese are handled at character level[4]. All of the three mentioned datasets come with the predefined validation and test partition, which are used in our experiments.

Two experimental scenarios are investigated in our work: initially we work on a set of 7 European languages: German (de), Italian (it), Spanish (es), Dutch (nl), French (fr), Polish (pl) and Portuguese (pt) each of which contain at least 60 hours of training data. The second scenario later expands to a total of 27 languages of more origin and diversity.

[3]Each feed-forward component accounts for around 25% the amount of parameters of each encoder block.

[4]Our initial experiments with joined BPE gave worse results for the 27-language dataset

4.2. Model and Training description

The experiments are conducted with two model architectures, two of which are commonly used in end-to-end speech recognition [38]: a) LSTM-based encoder-decoder networks [39] in which the LSTMs have 1024 hidden units and the encoder is downsampled using two 3×3-filter convolutional layers, and b) Transformer networks [6] with relative attention [40] with weight factorization for this multilingual setup. For the Transformer, we use the Transformer-Big configurations in [6] with model size 1024 but with 16 encoder layers with stochastic layer dropout with the same setting as in [12].

All models are trained on single GPU by grouping a maximum $45,000$ frames per mini-batch[5], and the gradients are updated every 16 mini-batches with adaptive scheduling in [6] using the base learning rate 1.5 and $4,096$ warm-up steps. The inputs are masked with SpecAugmentation [39]. Given the large configuration, we train all models up to $150,000$ updates or up to 2 weeks. It is notable that the factorized versions have minimal overhead which results in a 10 percent training speed reduction, while the adapter method requires at least 33% more time.

4.3. Baseline models

The comparison in the upcoming result section involves two previous works that were re-implemented. First, the language embedding was concatenated to the speech features and word embeddings at the encoder and decoder respectively which was used in [31]. Second, the language dependent adapters [7] were used. In this case, we use adapters in the form of feed-forward networks with $1,024$ neurons in the hidden layer. While theoretically the language embedding is a subset of our factorized network because the former is essentially a small set of weights dedicated for each language, the adapter network is fundamentally different because it requires extra layers, adding depths and nonlinearity levels to the overall architecture, while our factorization scheme keeps the interaction between inputs and weights unchanged.

4.4. Experiments with 7 languages

The word error rates for each language using two baseline models (with Transformer (TF) and LSTM), their factorized versions and the TF with adapter [7] are shown in Table 1. Averaged over the 7 languages, the error rate is reduced by 15.5% and 7.2% rel. for the Transformer and LSTM respectively, and the improvement is significant across languages, unlike the Adapter technique which manages to reduce the error rate for 4 languages but is not better for the other languages.

Regarding the number of parameters, the Transformer and its factorized variation has twice as many parameters as the LSTMs, thus possibly explaining the improvement regarding performance. While this seems to contradicts the large number of parameters for the ADT model that needs 42% more space than the factorized TF, the ADT actually adds more depth (2 per TF block). This is a significant change to the architecture because with layer normalization, all languages share the same layer mean and variance at each level, while this is not changed with the adapter.

[5]speech inputs are often longer than their transcriptions, so grouping mini-batches by frames is more efficient

4.5. Experiments with 27 languages

Under this condition, the factorization method maintains the improvement across all languages, with overall 26% rel. WER reduction in average for Transformer and 27.2% rel. for LSTM, as summarized in Table 2. Importantly, the factorized models are effective while using only 15% more parameters, while the ADT Transformer needs almost 1 billion parameters to achieve a 21.2% rel. improvement, due to each language requiring 2 more layers per block.

While the most resourceful languages such as German, Italian, Spanish and French observe the similar improvement compared to the 7-language experiment, the lower resource counterparts are often improved significantly compared to the baseline, regardless of the model architecture. The error rates on Japanese and Latvian testsets were decreased by 48% rel. compared to the base Transformer, and multiple languages were improved by 30% rel. including Arabic, Br, Cnh, Cv and Ta. The only language that remains a relative high error rate is Dhivehi, in this case staying over 60% regardless of the architecture. One explanation for the large improvements regarding lower resource languages is that, the language weights are only learned to optimize for those particular languages, while the shared weights are frequently changed attempting to optimize all different language/task losses. This problem is often alleviated using learnable and weighted sampling [41] to help the gradients remain stable for the less frequently visited languages.

A direct comparison between two Transformer variations shows that the factorization is consistently better in 21 languages and the adapters yielded better results in 6, given the same time and computational constraints. While it is also possible for the adapters to obtain better performances by longer training, the presented results provide evidences that our proposed factorization scheme is able to outperform both the baseline and the deeper language adapter network without extensive tuning and with reasonable resources.

Table 1: *Comparison on the 7-language dataset (WER↓). Our baseline models include the Transformers (TF), LSTM and their factorized (FTR) variations respectively. The last column is the Transformer with Adapter (ADT) [7].*

Language	TF	+FTR	LSTM	+FTR	ADT
# Params	335M	350M	167M	172M	497M
de	15.78	14.62	15.75	15.53	14.71
es	16.06	13.47	14.66	14.09	14.81
fr	17.34	16.26	17.35	16.44	16.76
it	18.62	15.82	16.65	15.63	17.58
nl	26.61	22.33	24.18	22.57	31.84
pl	20.4	15.7	16.39	15.28	20.65
pt	25.8	19.3	23.21	19.49	25.19
Mean	20.08	**16.97**	18.31	17.00	20.2

5. Conclusion

In this work, we proposed a method to decompose and factorize weights enabling multilingual end-to-end ASR models to learn more efficiently. While the main results are promising and the method can be applied to arbitrary neural architectures, we are also aware that method requires the utterance to contain a single language and thus is limited to such scenarios. Future work will investigate the usage in a code-mixing scenario and incorporating unlabeled data for language-specific feature learning.

Table 2: *Comparison on the 27-language dataset. The models being shown include Transformers (TF), LSTM (TF) and their factorized versions (FTR). WER↓ .*

Language	TF	+FTR	LSTM	+FTR	ADT
# Params	355M	416M	177M	194M	980M
(ar)	26.2	**17.81**	28.73	20.02	16.56
(br)	51.85	**34.69**	71.53	40.49	40.21
(cnh)	52	**38.33**	62.19	36.59	55.18
(cv)	53.88	**33.11**	61.61	39.6	38.40
(de)	16.89	**15.62**	19.89	16.59	16.35
(dv)	71.63	**63.72**	80.18	64.82	65.23
(es)	16.05	**14.53**	18.41	14.82	15.27
(et)	33.95	30.43	39.63	34.26	**28.12**
(fr)	18.61	**17.24**	20.86	17.43	17.87
(ia)	49.86	33.24	48.39	**31.96**	42.40
(id)	28.78	**17.28**	32.9	20.22	22.79
(it)	20.76	**18**	21.99	18.07	19.60
(ja)	39.17	**20.44**	38.92	23.79	27.55
(lv)	66.17	**34.3**	66.66	37.93	43.57
(ky)	22.08	**17.17**	18.68	21.46	12.86
(mn)	42.03	**35.03**	46.42	38.5	34.12
(nl)	27.54	**23.75**	29.44	23.93	28.30
(pl)	21.81	17.8	19.92	**17.19**	18.75
(pt)	25.16	21.38	27.13	**21.37**	22.82
(ro)	39.39	32.15	34.7	**26.73**	41.71
(sah)	57.47	50.47	69.04	**49.2**	55.27
(sl)	49.73	22.01	48.92	29.66	**20.77**
(ta)	33.1	22.34	18.87	28	**16.36**
(tr)	6.04	5.16	4.99	8.29	**2.40**
(tt)	24.96	22.12	38.03	24.07	**21.83**
(zh)	24.05	**22.53**	33.01	23.54	25.99
Mean	35.4	**26.2**	38.5	28.0	27.78

6. Acknowledgements

We thank Jan Niehues for suggesting the similarity between the multiplicative weights and layer normalization.

Parts of this work were realized within the project ELITR which has received funding from the European Unions Horizon 2020 Research and Innovation Programme under grant agreement No 825460.

Parts of this work were realized within a project funded by the Federal Ministry of Education and Research (BMBF) of Germany under the number 01IS18040A.

7. References

[1] T.-L. Ha, J. Niehues, and A. Waibel, "Toward multilingual neural machine translation with universal encoder and decoder," *arXiv preprint arXiv:1611.04798*, 2016.

[2] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado *et al.*, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 339–351, 2017.

[3] A. Waibel, H. Soltau, T. Schultz, T. Schaaf, and F. Metze, *Multilingual Speech Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

[4] E. A. Platanios, M. Sachan, G. Neubig, and T. Mitchell, "Contextual parameter generation for universal neural machine translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018.

[5] Y. Zhu, P. Haghani, A. Tripathi, B. Ramabhadran, B. Farris, H. Xu, H. Lu, H. Sak, I. Leal, N. Gaur, P. J. Moreno, and Q. Zhang, "Multilingual Speech Recognition with Self-Attention Structured Parameterization," in *Interspeech*, 2020.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[7] A. Bapna, N. Arivazhagan, and O. Firat, "Simple, scalable adaptation for neural machine translation," *arXiv preprint arXiv:1909.08478*, 2019.

[8] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint*, 2014.

[9] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016.

[10] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017.

[11] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4845–4849.

[12] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, and A. Waibel, "Very Deep Self-Attention Networks for End-to-End Speech Recognition," in *Proc. Interspeech 2019*, 2019, pp. 66–70.

[13] O. Firat, K. Cho, and Y. Bengio, "Multi-way, multilingual neural machine translation with a shared attention mechanism," *arXiv preprint arXiv:1601.01073*, 2016.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] Y. Wen, D. Tran, and J. Ba, "Batchensemble: an alternative approach to efficient ensemble and lifelong learning," *arXiv preprint*, 2020.

[16] J. Yoon, S. Kim, E. Yang, and S. J. Hwang, "Scalable and order-robust continual learning with additive parameter decomposition," *arXiv preprint arXiv:1902.09432*, 2019.

[17] L. Burget, P. Schwarz, M. Agarwal, P. Akyazi, K. Feng, A. Ghoshal, O. Glembek, N. Goel, M. Karafiát, D. Povey *et al.*, "Multilingual acoustic modeling for speech recognition based on subspace gaussian mixture models," in *ICASSP*, 2010.

[18] H. Lin, L. Deng, D. Yu, Y.-f. Gong, A. Acero, and C.-H. Lee, "A study on multilingual acoustic modeling for large vocabulary asr," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 4333–4336.

[19] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *ICASSP*, 2013.

[20] M. Müller, S. Stüker, and A. Waibel, "Neural language codes for multilingual acoustic models," *arXiv preprint*, 2018.

[21] S. Kim and M. L. Seltzer, "Towards language-universal end-to-end speech recognition," in *ICASSP*, 2018.

[22] S. Toshniwal, T. N. Sainath, R. J. Weiss, B. Li, P. Moreno, E. Weinstein, and K. Rao, "Multilingual speech recognition with a single end-to-end model," in *ICASSP*, 2018.

[23] S. Zhou, S. Xu, and B. Xu, "Multilingual end-to-end speech recognition with a single transformer on low-resource languages," *arXiv preprint arXiv:1806.05059*, 2018.

[24] O. Adams, M. Wiesner, S. Watanabe, and D. Yarowsky, "Massively multilingual adversarial speech recognition," in *Proceedings of the Conference of the NAACL: Human Language Technologies*, 2019.

[25] A. Kannan, A. Datta, T. N. Sainath, E. Weinstein, B. Ramabhadran, Y. Wu, A. Bapna, Z. Chen, and S. Lee, "Large-scale multilingual speech recognition with a streaming end-to-end model," *arXiv preprint arXiv:1909.05330*, 2019.

[26] B. Li, Y. Zhang, T. Sainath, Y. Wu, and W. Chan, "Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes," in *ICASSP*, 2019.

[27] M. Müller, S. Stüker, and A. Waibel, "Neural codes to factor language in multilingual speech recognition," in *ICASSP*, 2019.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[29] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[30] B. Zhang, P. Williams, I. Titov, and R. Sennrich, "Improving massively multilingual neural machine translation and zero-shot translation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Jul. 2020.

[31] V. Pratap, A. Sriram, P. Tomasello, A. Hannun, V. Liptchinsky, G. Synnaeve, and R. Collobert, "Massively multilingual asr: 50 languages, 1 model, 1 billion parameters," *arXiv*, 2020.

[32] J. Philip, A. Berard, M. Gallé, and L. Besacier, "Language adapters for zero shot neural machine translation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4465–4470.

[33] M. Dusenberry, G. Jerfel, Y. Wen, Y. Ma, J. Snoek, K. Heller, B. Lakshminarayanan, and D. Tran, "Efficient and scalable bayesian neural nets with rank-1 factors," in *International conference on machine learning*. PMLR, 2020, pp. 2782–2792.

[34] J. B. Hampshire II and A. Waibel, "The meta-pi network: Building distributed knowledge representations for robust multisource pattern recognition," *IEEE Computer Architecture Letters*, vol. 14, no. 07, pp. 751–769, 1992.

[35] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019.

[36] R. Gretter, "Euronews: a multilingual speech corpus for asr." in *LREC*, 2014, pp. 2635–2638.

[37] J. Iranzo-Sánchez, J. A. Silvestre-Cerdà, J. Jorge, N. Roselló, A. Giménez, A. Sanchis, J. Civera, and A. Juan, "Europarl-st: A multilingual corpus for speech translation of parliamentary debates," in *ICASSP*, 2020.

[38] A. Zeyer, P. Bahar, K. Irie, R. Schlüter, and H. Ney, "A comparison of transformer and lstm encoder decoder models for asr," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 8–15.

[39] 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," *arXiv*, 2019.

[40] N.-Q. Pham, T.-L. Ha, T.-N. Nguyen, T.-S. Nguyen, E. Salesky, S. Stüker, J. Niehues, and A. Waibel, "Relative Positional Encoding for Speech Recognition and Direct Translation," in *Proc. Interspeech 2020*, 2020, pp. 31–35. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-2526

[41] X. Wang, Y. Tsvetkov, and G. Neubig, "Balancing training for multilingual neural machine translation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

E Short-Term Word-Learning in a Dynamically Changing Environment

Short-Term Word-Learning in a Dynamically Changing Environment

Christian Huber[1], *Rishu Kumar*[2], *Ondřej Bojar*[2] *and Alexander Waibel*[1,3]

[1]Interactive Systems Lab, Karlsruhe Institute of Technology, Karlsruhe, Germany
[2]Charles University, Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics, Prague, Czech Republic
[3]Carnegie Mellon University, Pittsburgh PA, USA

christian.huber@kit.edu, lastname@ufal.mff.cuni.cz, alexander.waibel@cmu.edu

Abstract

Neural sequence-to-sequence automatic speech recognition (ASR) systems are in principle open vocabulary systems, when using appropriate modeling units. In practice, however, they often fail to recognize words not seen during training, e.g., named entities, numbers or technical terms. To alleviate this problem, [1] proposed to supplement an end-to-end ASR system with a word/phrase memory and a mechanism to access this memory to recognize the words and phrases correctly. In this paper we study, a) methods to acquire important words for this memory dynamically and, b) the trade-off between improvement in recognition accuracy of new words and the potential danger of false alarms for those added words. We demonstrate significant improvements in the detection rate of new words with only a minor increase in false alarms (F1 score 0.30 $\rightarrow$ 0.80), when using an appropriate number of new words. In addition, we show that important keywords can be extracted from supporting documents and used effectively.

Index Terms: speech recognition, one-shot learning, new-word learning

1. Introduction

Neural sequence-to-sequence systems deliver state-of-the-art performance for automatic speech recognition (ASR). When using appropriate modeling units, e.g., byte-pair encoded characters, these systems are in principle open vocabulary systems. In practice, however, they often fail to recognize words not seen during training, e.g., named entities, numbers or technical terms.

To alleviate this problem, [1] proposed to supplement an end-to-end ASR system with a word/phrase memory and a mechanism to access this memory to recognize the words and phrases correctly. After the training of the ASR system, and when it has already been deployed, a relevant word can be added or subtracted instantly without the need for further training.

This is achieved by, a) a memory-attention layer which predicts the availability and location of relevant information in the memory, and b) a memory-entry-attention layer which extracts the information of a memory entry.

In this paper we study, a) methods to acquire specialized words for this memory and, b) the trade-off between improvement in recognition accuracy of new words and the potential danger of false alarms for those added words. Therefore, we extensively evaluate this system in an online low-latency setup.

The ASR model described above outputs uncased text without punctuation. Therefore, we run a casing and punctuation model afterwards which reconstructs the casing of each word and inserts the punctuation. This model consists of a transformer encoder [2, 3] which is run after the beam-search and outputs for each word if the word should be uppercased and if any punctuation should be emitted after the word or not.

To use the model in an online low-latency setup [4], we do the following: The model waits for a chunk of acoustic frames with at least a predetermined duration to arrive. Then beam search is run with this input chunk. The beams (called unstable hypotheses) are then given to a stability detection component which returns a stable hypothesis, e.g. the common prefix of all hypotheses. After that, the part of audio corresponding to the stable hypothesis is cut out (via an alignment) and the model waits for more audio frames.

2. Experiments and Results

We extended the model proposed in [1] with the ability to correctly do the casing of the new words supplied through the new words list. This is done by adapting the casing and punctuation model by using internals of the ASR model, namely the attention over the memory entries. For each word the beam-search has outputted, all the predicted memory entries (from the memory-attention layers) are compared with the word. If there is a match, the casing from the new words list is taken. Therefore, whenever a new word is recognized by the model, the correct casing from the new words list is outputted.

Furthermore, we evaluate the system in two different scenarios: First, when an operator adds new words to the system, and second, when new words are extracted from other sources, e.g. slides.

2.1. Data

For the first scenario, we use eight talks from the ELITR testset [5] with a total length of 3.7 hours. For the second scenario, we use ten talks from the EMNLP 2020 conference[1] with a total length of 1.6 hours. Along with the EMNLP talks, the papers and the slides of the talks are available. The text from the papers (excluding the references) is extracted with pdftotext[2], the text from the slides is extracted with Tesseract[3], since we only had access to screenshots of the slides. We also cleaned the transcripts of all talks from typos, so that they could serve as a reliable reference.

2.2. Extraction of the New Words List

We tried different methods to extract a list of new words from the document, and we ended up taking all the words of the document which are not in the training data of the ASR model.

[1]https://2020.emnlp.org
[2]https://en.wikipedia.org/wiki/Pdftotext
[3]https://github.com/tesseract-ocr/tesseract

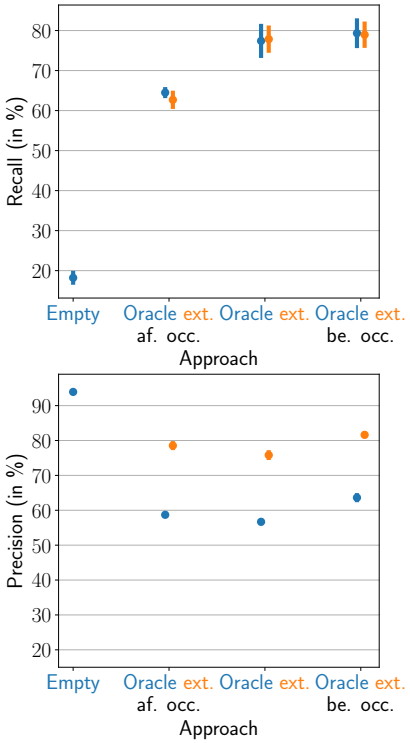

Figure 1: *Recall and precision for the evaluation of the memory ASR-worker on the ELITR testset.*

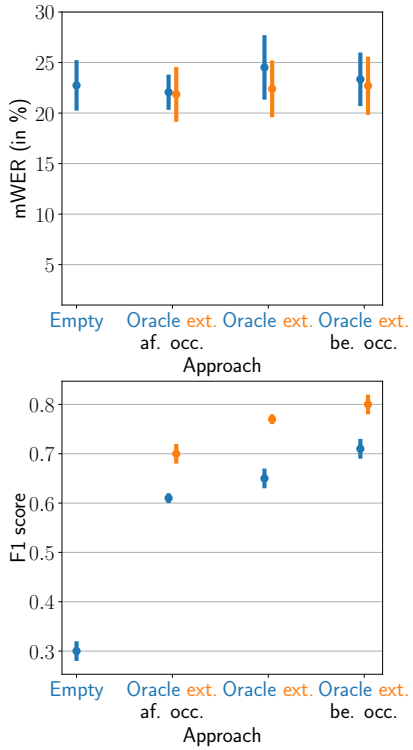

Figure 2: *mWER and F1-score for the evaluation of the memory ASR-worker on the ELITR testset.*

This method is simple and performed well when looking at the output words[4]. For example in the talk `rehm_long`,[5] we have extracted the following list of new words: pipelining, Friem, iAnnotate, MQM, LSPs, eServices, semantification, Aljoscha, Cortana, workflows, DFKI, annotating, NLP.

With this method, we extracted 134 terms from the ELITR testset reference transcripts, 148 from the EMNLP testset reference transcripts, 865 from the EMNLP papers and 584 from the EMNLP slides. Note that, in contrast to the new-words testset in [1], the list of new words is created automatically and might differ from a list of new or rare words a human might select.

2.3. Evaluation with the Help of an Operator

In this scenario, we evaluate the system, when the list of new words is extracted from the reference transcript (Oracle). This simulates an operator introducing new misrecognized words in the memory. We use eight talks of the ELITR testset for the evaluation.

[4]Furthermore this method is very effective in finding errors in the transcript.

[5]`https://github.com/ELITR/elitr-testset/tree/master/documents/rehm-language-technologies`

Dependent on the approach used (x axis in Figures 1 and 2), either an empty list of new words (Empty), i.e. the baseline, or the full list of new words (Oracle) is used. For the approaches "* af. occ." and "* be. occ.", a new word is added to the list of new words *after* the first *occurrence* or *before* the first *occurrence* of that word, respectively. These two approaches simulate an operator correcting the output and either the corrected segment is reevaluated or not. The first condition reflects the case when the operator can quickly react to errors but cannot fix them once the segment has been shipped, the second condition represents the situation when either the shipping is delayed a little to give the operator a chance to introduce the correction, or when the overall system setup allows updating previous outputs. The second situation is common e.g. in re-translating systems such as ELITR [6].

Furthermore, we noticed, that sometimes false positives occurred, i.e. a word in the new words list is confused with a common word and outputted even though it is not in the audio and the reference transcript at that point. In the approaches marked with "* ext. *", we therefore *extended* the list of new words and added these common words also to the new words list. This should help the model to distinguish between the common word and the new word. These approaches simulate

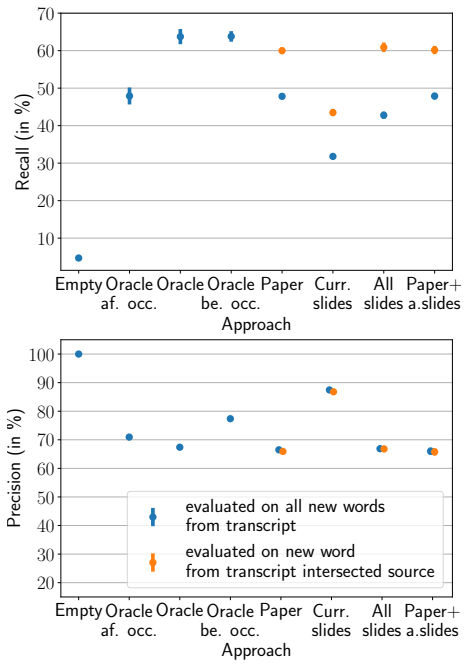

Figure 3: *Recall and precision for the evaluation of the memory ASR-worker on the EMNLP testset.*

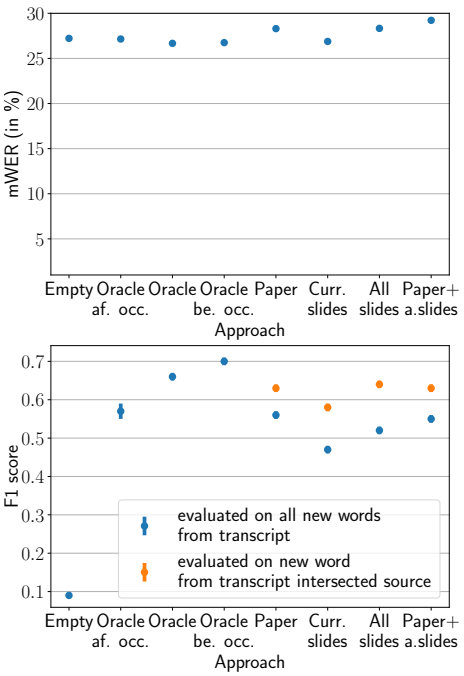

Figure 4: *mWER and F1-score for the evaluation of the memory ASR-worker on the EMNLP testset.*

an operator adding common words to the memory when a false positive is observed.

The mWER segmenter [7] is used to align the output segments with the reference segments. In Figures 1 and 2, we can see the results. The recall, precision and F1 score is measured on the new words. The baseline model with empty memory performs poorly. For the other approaches, we see that it certainly helps to add the new word before the occurrence. The approaches with extended memory list produce substantially fewer false positives with the approach "Oracle ext. be. occ." reaching an F1 score of 0.80 ± 0.02. Furthermore, we see that all approaches have similar word error rates (mWER) suggesting that the word error rate is not an appropriate measure for evaluating if important words are correctly recognized. Note that the talks are challenging and the transcript is not very clean and therefore the word error rates are relatively high.

We noticed that the performance of the ASR worker when used in online low-latency mode is not deterministic. This happens because the packets of audio are sent over the network and they can, dependent on the network latency, arrive earlier or later. Therefore, when the predetermined duration of audio is reached, the model can start processing a slightly smaller or longer audio input sequence. Thus, as described above, the stable hypothesis found by the stability detection is not deterministic and therefore the same holds true for the ASR output. Therefore, we report mean and standard deviation performance over 16 runs as shown in Figures 1 and 2.

2.4. Evaluation with Additional Sources

In this scenario, we evaluate the system when the list of new words is extracted from additional sources. We conducted experiments on ten talks from the EMNLP 2020 conference. The list of new words was extracted from the paper or the slides (with optical character recognition), respectively, with the same method described above. For the approach "Curr. slides", the new words list was extracted from the previous, current and the upcoming slide.

In Figures 3 and 4, we see similar results for the approaches that were already evaluated on the ELITR testset (e.g. F1 scores of 0.66 ± 0.01 vs. 0.65 ± 0.02 for the oracle approaches). For the approaches "Paper" and "* slides", we evaluate the performance on all the new words of the transcript and on the new words of the transcript intersected with the new words from the source. We differentiate between these two evaluation methods to show the effect of the new uttered word being actually available in the source or not. The performance on the new words of the approach "Paper" is not much worse than the oracle approach (F1 scores of 0.66 ± 0.01 vs. 0.56 ± 0.01), especially when considering the evaluation only of new words present in the paper (0.63 ± 0.01), compared to an F1 score of 0.09 ± 0.00 for the approach "Empty".

The performance when extracting the new words list from the slides is worse when evaluating on all new words from the transcript, possibly due to the used optical character recogni-

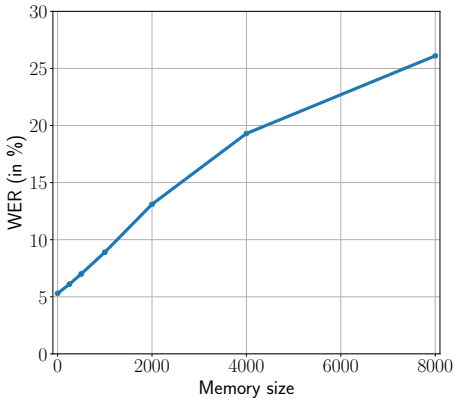

Figure 5: *Evaluation of the memory ASR-worker on the tedlium testset with a large number of random words from the training dataset in the memory.*

tion. When evaluating only on new words present in the slides, the performance is better than the corresponding approach of extracting the new words from the paper. Combining the new words from the paper and the slides did not yield improvements. The word error rate (mWER) stays almost the same for all approaches, however one can see that for the approaches with many words in the memory, the word error rate is a bit higher. We investigate this phenomenon further in section 2.5.

2.5. Additional Experiments

We investigated the usage of a large memory and took a certain number of random words from the training dataset as memory. Then we decoded the tedlium testset [8]. The results can be seen in Figure 5: a huge number of words in the memory result in a drastically worse word error rate. This happens since a lot of false positives are occurring. Therefore, our approach is best used with a small number of new words the model should focus on.

As the final experiment, we examined the situation when a new word in the memory is not found (by the memory-attention layers). This can happen if the pronunciation of the word differs considerable from the "common pronunciation" as learned by the general model. To help the model recognize the new word anyway, we propose not to search for the new word but for the word the model outputs instead. So for example if we want to recognize "his name is ron weasly" but the model would output "his name is ron weesley" even if the word "weasly" is in the memory, we would use the confused form "weesley" in the memory-attention layer which searches through the memory and eventually use "weasly" for memory-entry-attention layer which extracts information from from the memory.

Note that in this case, the scenario for the practical use is severely different. Instead of the information that a novel word will probably occur somewhere in the transcript, one now needs to have the information that at a specific point a word is wrong and another word is correct at that location.

We went through the false negatives of the new words testset from [1] and applied this approach. As a result, we obtained that the accuracy on the new words testset increased from 90.4% to 94.1%.

3. Conclusion

We demonstrated an efficient method of acquiring a new words list given a source such as supplementary paper or slides and evaluated the trade-off between improving the recognition accuracy of new words and the occurrence of too many false positives in an online low-latency environment. We documented that standard WER does not reflect the success of recognition of these typically very important words. We obtained an F1 score of up to 0.80 evaluated on the recognition of new words.

4. Acknowledgements

We want to thank SlidesLive,[6] who provided us with the transcripts of the EMNLP talks.

The projects on which this paper is based were funded by the European Union under grant agreement No 825460 (ELITR), the Federal Ministry of Education and Research (BMBF) of Germany under the numbers 01IS18040A (OML) and 01EF1803B (RELATER) and the Czech Science Foundation under the grant 19-26934X (NEUREM3).

5. References

[1] C. Huber, J. Hussain, S. Stüker, and A. Waibel, "Instant one-shot word-learning for context-specific neural sequence-to-sequence speech recognition," *arXiv preprint arXiv:2107.02268*, 2021.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[3] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *Proc. Interspeech 2019*, pp. 66–70, 2019.

[4] T.-S. Nguyen, S. Stueker, and A. Waibel, "Super-human performance in online low-latency recognition of conversational speech," 2021.

[5] E. Ansari, O. Bojar, B. Haddow, and M. Mahmoudi, "SLTev: Comprehensive Evaluation of Spoken Language Translation," in *Proc. of EACL Demo Papers*. Kyiv, Ukraine: ACL, 2021.

[6] O. Bojar, D. Macháček, S. Sagar, O. Smrž, J. Kratochvíl, P. Polák, E. Ansari, M. Mahmoudi, R. Kumar, D. Franceschini, C. Canton, I. Simonini, T.-S. Nguyen, F. Schneider, S. Stüker, A. Waibel, B. Haddow, R. Sennrich, and P. Williams, "ELITR multilingual live subtitling: Demo and strategy," in *Proc. of EACL System Demonstrations*. ACL, 2021, pp. 271–277. [Online]. Available: https://aclanthology.org/2021.eacl-demos.32

[7] E. Matusov, G. Leusch, O. Bender, and H. Ney, "Evaluating machine translation output with automatic sentence segmentation," in *Proceedings of the Second International Workshop on Spoken Language Translation*, 2005.

[8] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko, and Y. Estève, "Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation," in *International Conference on Speech and Computer*. Springer, 2018, pp. 198–208.

[6] `https://library.slideslive.com/`