

The IWSLT 2021 BUT Speech Translation Systems

Hari Krishna Vydana, Martin Karafiát, Lukáš Burget, “Honza” Černocký
Brno University of Technology, Faculty of Information Technology, Speech@FIT, Czechia
harivydana@gmail.com

Abstract

The paper describes BUT’s English to German offline speech translation (ST) systems developed for IWSLT2021. They are based on jointly trained Automatic Speech Recognition-Machine Translation models. Their performances is evaluated on MustC-Common test set. In this work, we study their efficiency from the perspective of having a large amount of separate ASR training data and MT training data, and a smaller amount of speech-translation training data. Large amounts of ASR and MT training data are utilized for pre-training the ASR and MT models. Speech-translation data is used to jointly optimize ASR-MT models by defining an end-to-end differentiable path from speech to translations. For this purpose, we use the internal continuous representations from the ASR-decoder as the input to MT module. We show that speech translation can be further improved by training the ASR-decoder jointly with the MT-module using large amount of text-only MT training data. We also show significant improvements by training an ASR module capable of generating punctuated text, rather than leaving the punctuation task to the MT module.

1 Introduction

Speech Translation (ST) systems are intended to generate text in target language from the audio in source language. The conventional ST systems are cascade ones, including (in the most popular form) three blocks i.e., an ASR, punctuation/segmentation module and an MT model (Ngoc-Quan Pham, 2019; Pham et al., 2020b; Jan et al., 2019; Ansari et al., 2020). Both Automatic Speech Recognition system (ASR) and Machine Translation (MT) models are independently trained, and the MT model processes the ASR output text (ASR hypotheses) to generate translations. In a cascade system, the advance-

ments in ASR and MT can be directly extended to ST. These models can also leverage on the availability of large ASR and MT data-sets, and some of the state-of-the art ST systems are still cascade ones.

Recently, End-to-End ST systems have become widely popular. An End-to-End ST can directly generate text in target language from the audio in source language. These models are simpler in structure and they are more suitable for operating in streaming fashion. Most End-to-End speech translation systems are variants of encoder-decoder architecture with attention models (Bahdanau et al., 2015; Di Gangi et al., 2019; Zhao et al., 2020). This category includes the popular Transformer models, which have been adapted for training End-to-End ST in (Di Gangi et al., 2019). In (Inaguma et al., 2020), a better performance of ST was achieved by initializing the encoder and decoder modules from pre-trained ASR and MT systems, respectively. Very-deep transformer models have been trained with stochastic depth for training End-to-End ST models in (Pham et al., 2019). The use of relative positional embeddings has also improved the performance of transformer (Pham et al., 2020a).

One major drawback or end-to-end ST is the data availability, i.e., paired speech-to-translation data is scarce compared to ASR or MT data. Data augmentations and use of synthetic data have been explored in (Bahar et al., 2019, 2020) to mitigate the issue. Unlike End-to-End ST systems, the data for training cascade systems is easily available and less costly.

A brief survey of existing approaches and their principal limitations are discussed in (Sperber and Paulik, 2020). Despite multiple advantages, the cascade systems suffer from a major drawback: propagating erroneous early decisions into MT models, which then cause degradation in the trans-

lation performance. To mitigate this degradation, rather than passing a single ASR output sequence to MT model, other forms such as lattices, n-best hypotheses and continuous representations have been explored in (Anastasopoulos and Chiang, 2018; Zhang et al., 2019; Sperber et al., 2019; Vydana et al., 2021; Dong et al., 2020).

In this work, we use our jointly trained Automatic Speech Recognition-Machine Translation (Joint-ASR-MT) model previously described in (Vydana et al., 2021). Joint-ASR-MT model is a cascade system, but it has a differentiable path between ASR and MT modules. To create such differentiable path, the continuous hidden representations (corresponding to each output token) from the ASR decoder are passed to the MT-Model. The hidden continuous tokens corresponding to each output token are the attention-weighted value vectors in the last layer of the transformer decoder. We refer to these continuous representations as “context vectors” as proposed in (Sperber et al., 2019).

Existing large separate ASR training data and MT training data can be used to pre-train these modules; then, the pre-trained modules are jointly optimized using a small amount of speech translation data. The joint optimization mitigates the degradation in performance due to erroneous early decisions.

In this paper, we generate German translation from English speech, and we focus on two main contributions: (1) We train different MT models that can translate normalized text or punctuated text. It is known that MT-models translating punctuated text provide superior performance, therefore, we propose to train an ASR system that can generate the punctuated text. We confirm that such ASR system provides superior performance in ASR-MT pipeline. (2) We use the internal continuous representations from the ASR-decoder as the input to MT module. In section 6, we show that speech translation can be further improved by adapting ASR-decoder to the MT module. This is achieved by training the ASR-decoder jointly with the MT-module using a large amount of text-only MT training data.

2 Datasets and Pre-processing

The Datasets used for training various models are described in Table. 1. ASR-Train-set and MT-Train-set are used for pre-training ASR and MT

models respectively. The pre-trained models are fine-tuned using ASR-MT-Train-set. All models are evaluated using MustC-Common test set.

Table 1: Data used for training various models.

	Corpora	#Sentences	Audio	Source text	Target Text
MT-Train-set	ParaCrawl v3	31M	-	✓	✓
	OpenSubtitles 2018	12M	-	✓	✓
	Rapid 2019	1.5M	-	✓	✓
	Europarl v9	1.81M	-	✓	✓
	News Commentary	365K	-	✓	✓
	Common Crawl	2.4M	-	✓	✓
	Wikitiitles	1.3M	-	✓	✓
	WIT3	196K	-	✓	✓
	TED Talks	220K	-	✓	✓
ASR-MT-Train-set	Europarl-ST	32K	✓	✓	✓
	Must-C V2	230K	✓	✓	✓
	IWSLT2018	171K	✓	✓	✓
ASR-Train-set	Tedlium3	264K	✓	✓	-
	Librispeech	281K	✓	✓	-

2.1 Pre-processing and Feature Extraction

From audio data, 80-Dimensional Mel-Filter bank energies along with pitch features are extracted. The Moses toolkit is used for text tokenization and other standard text pre-processing. The umlauts from the German text are replaced by the special tokens. All the non ASCII characters are removed from the text data. The repetitions of the same sentences are removed from the corpora. We cleaned up the MT training data by identifying and manually removing the sentences where successive words were erroneously concatenated in to very long erroneous words. Sentence-piece models (Kudo and Richardson, 2018) are used for training BPE-tokenizers. 40M lines of text are used for training each BPE-tokenizer and all the tokenizers have a vocabulary of 20K units. Three separate tokenizers are trained using normalized English text, punctuated English text and punctuated German text. The output of MT module is always punctuated text, while input to MT (as well as ASR output) can be either normalized or punctuated text (see norm-MT and Punc-MT in sections 4).

2.2 Pruning Noisy ASR corpus

Some of the utterances in ASR-MT-Train-set (MustC, IWSLT and Europarl) sets are erroneous due to the shift in alignments between audio and text. Training an End-to-End ASR on this data

directly did not lead to convergence. To remove erroneous transcripts, a hybrid TDNN-LFMMI ASR system based on KALDI (Povey et al., 2011, 2016) was trained and this ASR system was used to decode the ASR-MT-train set. The Word Error Rate (WER) for each sentence is computed and the sentences with more than 50% WER are deleted from the ASR-MT-Train-set (Potapczyk et al., 2019). Even with this cleaning, training the ASR systems only on ASR-MT-Train-set did not lead to convergence. Pre-training the ASR models on ASR-Train-set turned out to be crucial for convergence as described in section 3.

3 Automatic Speech Recognition (ASR)

ASR systems trained in this work are built on Transformer ASR models (Dong et al., 2018; Karita et al., 2019; Vydana et al., 2021; Vaswani et al., 2017). The ASR models have 12 encoder and 6 decoder layers with 4096 feed-forward units and 1024 attention dimension with 16 heads. Models are initially trained with ASR-Train-set and are later fine-tuned with ASR-MT-Train-set. A thresholding mechanism is used for pruning away the noisy end-of-sequence (EOS) tokens from beam search (Kahn et al., 2019). Models are trained with 30K warm-up updates and a check-point is saved after every 8K updates. The training is stopped with an early stopping criterion. 8-best check-points are averaged and the averaged weights are used for decoding the hypothesis. Vectorized beam search (Seki et al., 2019) was used for decoding the ASR hypotheses with a beam size of 10. Further in this paper, ASR models described in this section are referred to as Ext.ASR models (Externally trained ASR models).

Two different ASR systems were trained for generating normalized text (Norm-ASR) and punctuated text (Punc-ASR), and their performances are reported in Table 2. It can be observed that the WER of Punc-ASR appears to be higher than Norm-ASR. Punc-ASR is a obviously more difficult task than Norm-ASR — the punctuation tokens are considered as extra words and each error in those words contributes to the WER.

ASR-LM: A Transformer language model was trained on English text (Irie et al., 2019). The model has 6 layers, with 4096 feed-forward units and 1024 attention dimension with 8 heads. The model is initially pre-trained on Librispeech LM corpus and it is later fine-tuned on English text

Table 2: Performance of trained ASR systems reported on MustC-Common set. For Punc-ASR, the errors in punctuation tokens are considered, which makes it a more difficult task.

Model	WER
Norm-ASR	18.20
+LM	17.35
Punc-ASR	21.20

from MT-train-set and ASR-MT-train-set. An improvement in the performance is observed by shallow fusion of the ASR and language model (ASR-LM). Performances of these language models are presented in column 2 of Table. 5.

4 Machine Translation Systems(MT)

Transformer models (Vaswani et al., 2017) are also at the core of MT-systems. They have 6-encoder and 6-decoder layers with 4096 feed-forward units and 1024 attention dimensions and have 16 heads. The models are optimized with 30K warm-up updates and a check-point is saved every 8k updates. Training is stopped using an early stopping criterion. 8-best check-points are averaged and the averaged weights are used for decoding the hypotheses. The noisy EOS tokens are pruned out using (Kahn et al., 2019). Vectorized beam (Seki et al., 2019) search has been used for decoding the hypotheses with a beam size of 8. A large variance in the performance is observed w.r.t the decoding hyper-parameters such as maximum target sequence length and length-bonus. The maximum length of the target sequence is computed by multiplying the input sequence length with length-ratio: 1.2 was found as optimal on the development set. To control the length of the output sequence, the log-likelihood scores of the hypotheses are penalized by additive token insertion penalties. The optimal value for this penalty is tuned as a hyper-parameter on the development set. The hypothesis text is de-tokenized and BLEU score is evaluated using Moses Toolkit. All the BLEU scores reported in this paper are computed using the de-tokenized, punctuated German text using `multi-bleu-detok.perl`. The performances of the MT systems are reported in Table. 3. All BLEU scores reported in this paper are computed using punctuated text as reference.

In Table 3, Norm-MT, Punc-MT are MT models trained to predict punctuated German text. Norm-

Table 3: Performances of the MT systems reported on MustC-Common set.

Model	BLEU
Norm-MT	
+pretrain	27.18
+finetune	27.98
+MT-LM	28.12
Punc-MT	
+pretrain	31.02
+finetune	35.00
+MT-LM	35.04

MT uses the normalized English text as input while the Punc-MT uses the punctuated English text. Punc-MT model has performed better than Norm-MT. From Table 3, it can be observed that the punctuation tokens in the text are adding additional information for training the MT model. Fine-tuning the Punc-MT on in-domain text has improved the performance significantly. Further in this paper, MT models described in this section are referred to as Ext.MT models (Externally trained MT models).

MT-LM: A transformer language model has been trained on German text from MT-Train-set, ASR-MT-train-set. This LM is also used while decoding with the MT model (Irie et al., 2019). The architecture of the model is same as ASR-LM mentioned in section 3. A shallow fusion between the MT-model and the MT-LM Language model is performed. As shown in Table 3 and column 2 of Table 5, the additional language model (MT-LM) did not improve the performance significantly.

5 Jointly Trained ASR-MT Systems

The model has two modules: ASR and MT; their architecture is same as described in sections 3 and 4 respectively – see block diagram in Figure 1 and full description of the model in (Vydan et al., 2021). The context vectors from the final layer of the ASR-decoder are used as inputs to the MT module. Passing context vectors from ASR to MT models while training has also been explored in (Sperber et al., 2019). Both the models are jointly optimized using a multi-task cross-entropy (ASR cross-entropy and MT cross-entropy) – both losses are also shown in Figure 1. During the inference, beam search has been used to obtain the ASR hypotheses, and the corresponding context vectors obtained from the ASR model are used by

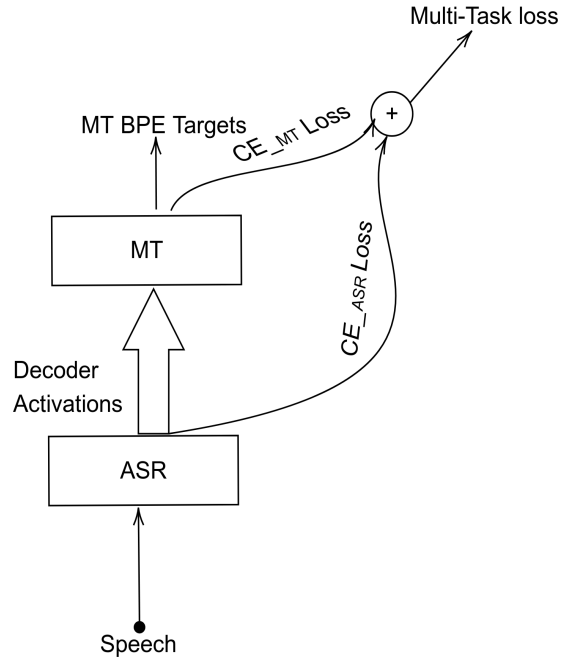


Figure 1: Joint-training of ASR-MT system using multi-task loss.

the MT model for generating translations. The MT model also uses a beam search, and the final ST hypotheses is obtained by a coupled search (Vydan et al., 2021) using the joint-likelihood from ASR and MT:

$$\begin{aligned}
 y^* &= \arg \max_y \sum_{z \in \hat{Z}(x)} P(y|z)P(z|x) \\
 &\equiv \arg \max_y \arg \max_{z \in \hat{Z}(x)} (\log(P(y|z)) \\
 &\quad + \log(P(z|x))), \tag{1}
 \end{aligned}$$

where x is the speech and z, y are the source and target sequences respectively. \hat{Z} is the n -best source sequence and y^* is most likely decoded hypothesis. In this equation, y^* is always a discrete sequence, while z is a discrete sequence when we are using Ext.MT and a continuous one when using Joint-MT. Note that similar coupled search was used in (Tu et al., 2017), where the back translation likelihoods are used for re-scoring the hypothesis of the MT-system.

6 Adapting ASR decoder to the MT module

Joint-ASR-MT models are jointly optimized by having an end-to-end differentiable path from speech to translations. The internal continuous representations from the ASR-decoder are used

as the input to MT module. Speech translation can be further improved by adapting ASR-decoder to the MT module. This is achieved by training the ASR-decoder jointly with the MT-module using large amount of text-only MT training data. The weights for the model are initialized from trained Joint-ASR-MT model. Speech translation data (ASR-MT-Train-set) is used to fine-tune Joint-ASR-MT model using a multi-task loss. Apart from that, the data from the MT-Train-set is used to jointly train the ASR-decoder and the MT-module of Joint-ASR-MT model. We alternately update the model using multi-task loss described in section 4 and the adaptation loss as described in this section.

A block diagram describing this training is presented in Figure 2. The input text sequence is given to the ASR-decoder and a sequence of zeros is considered as the encoder output sequence of the ASR model (i.e., H_{ASR} in Figure 2). The context vectors computed from these two sequences are used for training the MT-module. Note that similar method has been adopted in (Potapczyk et al., 2019) for improving the performance of ASR system using only text data. This training further improves the performance as will be shown in section 7.

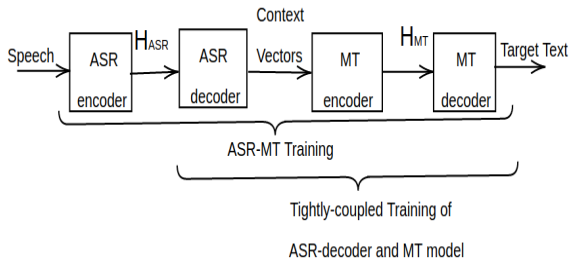


Figure 2: Adaptation of ASR-decoder to the MT-module in the Joint-ASR-MT model.

7 Speech Translation Results

Results for the various configurations of speech translation systems are given in Table 4. First, we focus on column A, where the Joint-ASR-MT models are trained using ASR-MT-Train-set (only speech translation data) with a multi-task loss as described in section 5. Note, however, that Ext.ASR and Ext.MT systems are trained on large amounts of data and finetuned to ASR-MT-Train-set as described in sections 3 and 4 respectively. For systems in column-A, normalized (unpunctuated) text is passed from ASR to MT model.

Row 1 corresponds to the conventional cascade system, where the Ext.ASR systems generates the n-best hypotheses of discrete token sequences and an Ext.MT uses these token sequences for generating the translations as described in Eq. 1. We consider this system achieving BLEU 23.20 as a baseline.

Usually, transformer-ASR decoder uses the partial output hypothesis and extends it by a new token with every autoregressive decoding step. For the system in row 2, Ext.ASR generates the complete hypothesis and ASR module from Joint-ASR-MT is “asked” to extend it by one more token. As a byproduct “context vectors” (the continuous representations) are generated for the whole sequence — these are then passed to the MT-module in joint-ASR-MT model to generate translation. Compared to row 1 of column A, we see a degradation in performance (BLEU-20.19). This can be attributed to having only small amount of speech translation training data, which is not sufficient for robustly training the Joint-ASR-MT systems.

For the systems in row 3, Ext.ASR generates the ASR hypotheses which are used by Ext.MT similar to the system described in row 1; the hypotheses from Ext.ASR are used by Joint-MT similarly to the system described in row 2. To generate the translation, the hypotheses from both models are ensembled as follows: For each output token, a weighted average of Log-softmax outputs from the two MT models is computed. This weighted average is used in the beam-search to compute the n-best partial hypotheses. These partial hypotheses are further extended by both the models to generate the Log-softmax outputs for next tokens. We can see that this ensembling system achieves a BLEU score of 24.02 and outperforms the cascaded baseline.

The systems in rows 4-6 are essentially the same as the ones in rows 1-3, respectively, except that now, the ASR module from joint-ASR-MT system is directly used to produce the n-best ASR hypotheses and the corresponding context vectors. Rows 4-6 show the same trend as rows 1-3 with slightly improved performance; these improvements are mainly due to better performing ASR system: As described in Section 2.2, training ASR systems only on ASR-MT-Train-set (data from Mustc, IWSLT and Europarl with erroneous transcriptions) did not lead to convergence. How-

Table 4: Performances of Joint-ASR-MT systems under various ensemble combinations, the results are reported on MustC-Common test set.

	[ASR]⇒[MT]	A		B		C		D	
		no-pretraining +Norm-ASR/MT		pre-training +Norm-ASR/MT		pre-training +Punc-ASR/MT		pretraining +Punc-ASR/MT+ tightly-coupled	
		BLEU	WER	BLEU	WER	BLEU	WER	BLEU	WER
1.	[Ext-ASR]⇒[Ext-MT]	23.20	18.20	23.20	18.20	26.15	21.54	26.15	21.54
2.	[Ext-ASR]⇒[Joint-MT]	20.19	-	22.59	-	28.56	-	29.00	-
3.	[Ext-ASR]⇒[Joint-MT + Ext-MT]	24.02	-	24.13	-	29.07	-	29.44	-
4.	[Joint-ASR]⇒[Ext-MT]	23.86	16.14	23.86	13.01	29.70	15.71	30.24	15.63
5.	[Joint-ASR]⇒[Joint-MT]	20.75	-	23.97	-	31.23	-	32.68	-
6.	[Joint-ASR]⇒[Joint-MT + Ext-MT]	24.65	-	25.95	-	32.51	-	33.68	-
7.	[Ext-ASR + Joint-ASR]⇒[Ext-MT]	24.60	14.84	25.00	13.54	29.00	16.46	29.35	16.19
8.	[Ext-ASR + Joint-ASR]⇒[Joint-MT]	20.89	-	23.59	-	30.52	-	31.58	-
9.	[Ext-ASR + Joint-ASR]⇒[Joint-MT + Ext-MT]	25.11	-	25.65	-	31.86	-	32.67	-
10.	[Ext-ASR + Joint-ASR]⇒[Joint-MT + Ext-MT] +ens*	25.35	14.61	26.14	13.05	32.67	15.71	33.78	15.63

Table 5: Comparing the performance of Joint-ASR-MT systems while processing n-best hypotheses from the ASR.

[ASR]⇒[MT]	A		B		C		D	
	no-pretraining +Norm-ASR/MT		pre-training +Norm-ASR/MT		pre-training +Punc-ASR/MT		pretraining +Punc-ASR/MT+ tightly-coupled	
	BLEU	WER	BLEU	WER	BLEU	WER	BLEU	WER
[Ext-ASR + Joint-ASR]⇒[Joint-MT + Ext-MT] +ens*	25.35	14.61	26.14	13.05	32.67	15.71	33.78	15.63
+ASR-LM			26.90	12.80				
+MT-LM			27.16					
+2-best-input	-	-	27.24	-	32.69	-	33.82	-
+4-best-input	-	-	27.35	-	32.80	-	33.87	-
+6-best-input	-	-	27.35	-	32.85	-	33.86	-
+8-best-input	-	-	27.46	-	32.94	-	33.77	-
+10-best-input	-	-	27.51	-	32.87	-	33.79	-

ever, when the same data is used to train Joint-ASR-MT model for speech translation task, we observe that the ASR module in this model trained well. The reason for that is that the ASR-module is not directly trained on erroneous transcriptions, instead, it is trained to produce transcriptions that lead to good translations. This training can be seen as a form of light supervision which can mitigate the problem with the erroneous transcriptions.

At the end, this system trained only on ASR-MT-Train-set achieves better ASR performance (WER 16.14%) compared to Ext.ASR (WER 18.20%), Which is pre-trained on ASR-Train-set (Approx 2000hrs) and fine-tuned on erroneous ASR-MT-Train-set. Similar trend will be observed with the systems in columns B, C and D.

The systems described in rows 7-9 are similar to those from rows 1-3, except that the ASR hy-

potheses are obtained by ensembling the Ext.ASR and ASR-module in Joint-ASR-MT model. The ensembling is performed in a similar way as described for the MT-system (row 2). All the ensemble systems in rows 3, 6, and 7-9 are ensembled giving equal weight to both the systems, except for row 10, where the ensemble weights are tuned on the development set. For all these systems, we can see that the ensembling consistently improves the performances.

The systems in column B are similar to the ones in Column A, but for the Joint-ASR-MT model, the weights of ASR and MT module are initialized from the Ext.ASR and Ext.MT. Only then, the Joint-ASR-MT model is fine-tuned using ASR-MT-Train-set. Comparing column-A and column-B, we can see that such pre-training has significantly improved the performance.

We also see that the MT system using continuous representations (Joint-MT) (row 5; BLEU 23.97) outperforms the system with the Ext.MT (row 4; BLEU-23.86) and similar trend can be seen in columns C and D. This is in contrast to the system in column A where we did not use enough data for training the Joint-ASR-MT model; now, with the pre-training, the joint-ASR-MT model is effectively trained on the same amount of data as the Ext.MT systems.

The systems in column C are similar to the ones in Column B, but the ASR and MT modules used here are Punc-ASR (ASR systems which can generate punctuated text) and Punc-MT (MT systems which can process punctuated text as input), respectively. We can see that the systems from column-C perform significantly and consistently better than the corresponding ones in column-B. This shows that it is more effective to train an ASR module to generate punctuated text rather than leaving the punctuation task to the MT module. Note that the ASR performances reported in columns C and D is computed including the punctuation symbols, which results in higher WERs.

Finally, the systems in column D are the same as the ones in column C except that we additionally use the ASR decoder adaptation scheme described in section 6. The consistent improvements observed in column D as compared to column C show the effectiveness of this adaptation scheme. They are able to make use of the large amount of text-only MT training data to train also the ASR decoder in order to tighten the coupling between

ASR-decoder and MT-module. Apart from improving MT-module, this adaptation has also improved the performance of ASR-decoder on its own. This can be observed by comparing WER's of row 4 in columns C and D.

The results of passing the n-best hypotheses from ASR to MT models are presented in Table 5. Passing the n-best hypothesis from ASR to MT module has better performance, but not significantly. This result is not in line with our previous studies (Vydana et al., 2021), where we have seen significant gains from switching from 1-best to n-best.

8 Conclusion

In this work, we have explored joint-training of ASR-MT models for speech translation. Initializing these models from pre-trained ASR and MT models has helped in better optimization. The joint training has improved the performance of the ASR module significantly as the additional MT module has provided better (light) supervision in the context of erroneous ASR transcripts. Adding the punctuation information into the input text improves the performance of the MT-model greatly. In line with this observation, use of ASR system generating punctuated text also improves the MT performance significantly in a cascade pipeline. Use of the MT text only data to adapt the ASR decoder to the MT module in the joint-ASR-MT model further improves the performances of these systems. The systems trained in this work are offline models and their performances needs to be studied from the perspective of online or streaming models.

9 Acknowledgements

The work was supported by Czech National Science Foundation(GACR) project "NEUREM3" No. 19-26934X. Part of high-performance computation run on IT4I supercomputer and was supported by the Ministry of Education, Youth and Sports of the Czech Republic through e-INFRA CZ (ID:90140).

References

- Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. *arXiv preprint arXiv:1802.06655*.

- Ebrahim Ansari, Nguyen Bach, Ondřej Bojar, Roldano Cattoni, Fahim Dalvi, Nadir Durrani, Marcello Federico, Christian Federmann, Jiatao Gu, Fei Huang, et al. 2020. Findings of the iwslt 2020 evaluation campaign. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 1–34.
- Parnia Bahar, Patrick Wilken, Tamer Alkhouli, Andreas Guta, Pavel Golik, Evgeny Matusov, and Christian Herold. 2020. [Start-before-end and end-to-end: Neural speech translation by AppTek and RWTH Aachen University](#). In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 44–54, Online. Association for Computational Linguistics.
- Parnia Bahar, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. On using specaugment for end-to-end speech translation. *arXiv preprint arXiv:1911.08876*.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Mattia A Di Gangi, Matteo Negri, and Marco Turchi. 2019. Adapting transformer to end-to-end spoken language translation. *Proc. INTERSPEECH*, pages 1133–1137.
- Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pages 5884–5888. IEEE.
- Qianqian Dong, Rong Ye, Mingxuan Wang, Hao Zhou, Shuang Xu, Bo Xu, and Lei Li. 2020. "listen, understand and translate": Triple supervision decouples end-to-end speech-to-text translation. *arXiv preprint arXiv:2009.09704*.
- Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Enrique Yalta Soplín, Tomoki Hayashi, and Shinji Watanabe. 2020. [Espnet-st: All-in-one speech translation toolkit](#).
- Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. Language modeling with deep transformers. *arXiv preprint arXiv:1905.04226*.
- Niehues Jan, Roldano Cattoni, Stuker Sebastian, Matteo Negri, Marco Turchi, Salesky Elizabeth, Sanabria Ramon, Barrault Loic, Specia Lucia, and Marcello Federico. 2019. The iwslt 2019 evaluation campaign. In *16th International Workshop on Spoken Language Translation 2019*.
- Jacob Kahn, Ann Lee, and Awni Hannun. 2019. Self-training for end-to-end speech recognition. *arXiv preprint arXiv:1909.09116*.
- Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyang Jiang, Masao Someki, Nelson Enrique Yalta Soplín, Ryuichi Yamamoto, Xiaofei Wang, et al. 2019. A comparative study on transformer vs rnn in speech applications. *arXiv preprint arXiv:1909.06317*.
- Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.
- Thanh-Le Ha, Juan Hussain Felix Schneider, Jan Niehues, Sebastian Stüker, Alexander Waibel, Ngoc-Quan Pham, Thai-Son Nguyen. 2019. The iwslt 2019 kit speech translation system. In *International Workshop on Spoken Language Translation (IWSLT)*. Hongkong.
- Ngoc-Quan Pham, Thanh-Le Ha, Tuan-Nam Nguyen, Thai-Son Nguyen, Elizabeth Salesky, Sebastian Stueker, Jan Niehues, and Alexander Waibel. 2020a. Relative positional encoding for speech recognition and direct translation. *arXiv preprint arXiv:2005.09940*.
- Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Muller, and Alex Waibel. 2019. 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 Alex Waibel. 2020b. Kit’s iwslt 2020 slt translation system. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 55–61.
- Tomasz Potapczyk, Paweł Przybyś, Marcin Chochowski, and Artur Szumaczk. 2019. Samsung’s system for the iwslt 2019 end-to-end speech translation task. In *Proc. of 16th International Workshop on Spoken Language Translation (IWSLT), Hong Kong*.
- Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. CONF. IEEE Signal Processing Society.
- Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. 2016. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Proc. INTERSPEECH*, pages 2751–2755.
- Hiroshi Seki, Takaaki Hori, Shinji Watanabe, Niko Moritz, and Jonathan Le Roux. 2019. Vectorized beam search for ctc-attention-based speech recognition. In *Proc. INTERSPEECH*, pages 3825–3829.
- Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2019. Attention-passing models for

robust and data-efficient end-to-end speech translation. *Transactions of the Association for Computational Linguistics*, 7:313–325.

Matthias Sperber and Matthias Paulik. 2020. Speech translation and the end-to-end promise: Taking stock of where we are. *arXiv preprint arXiv:2004.06358*.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

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 *Proc. Advances in Neural Information Processing Systems*, pages 5998–6008.

Hari Krishna Vydana, Martin Karafi’at, Katerina Zmolikova, Luk’as Burget, and Honza Cernocky. 2021. Jointly trained transformers models for spoken language translation. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*.

Pei Zhang, Boxing Chen, Niyu Ge, and Kai Fan. 2019. Lattice transformer for speech translation. *arXiv preprint arXiv:1906.05551*.

Chengqi Zhao, Mingxuan Wang, and Lei Li. 2020. NeurSt: Neural speech translation toolkit. *arXiv preprint arXiv:2012.10018*.