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 D3.1

Report 1 on Spoken Language Translation

Barry Haddow (UEDIN), Rico Sennrich (UEDIN), Phil Williams (UEDIN), Yuekun Yao (UEDIN), Biao Zhang (UEDIN), Felix Schneider (KIT), Thai-Son Nguyen (KIT), Sebastian Stüker (KIT), Peter Polák (CUNI), Dominik Macháček (CUNI), Sangeet Sagar (CUNI), Ebrahim Ansari (CUNI), Mohammad Mahmoudi (CUNI)

Dissemination Level: Public

Final (Version 1.0), 30th June, 2020

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, 36 months
Dissemination level	Public
Contractual date of delivery	30th June, 2020
Actual date of delivery	30th June, 2020
Deliverable number	D3.1
Deliverable title	Report 1 on Spoken Language Translation
Type	Report
Status and version	Final (Version 1.0)
Number of pages	69
Contributing partners	UEDIN, CUNI, KIT, PV
WP leader	UEDIN
Author(s)	Barry Haddow (UEDIN), Rico Sennrich (UEDIN), Phil Williams (UEDIN), Yuekun Yao (UEDIN), Biao Zhang (UEDIN), Felix Schneider (KIT), Thai-Son Nguyen (KIT), Sebastian Stüker (KIT), Peter Polák (CUNI), Dominik Macháček (CUNI), Sangeet Sagar (CUNI), Ebrahim Ansari (CUNI), Mohammad Mahmoudi (CUNI)
EC project officer	Alexandru Ceausu
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 **4**

2 Introduction **5**

3 Normalisation and Segmentation of ASR **5**
3.1 KIT Punctuator, Truecaser and Normalizer 5
3.2 CUNI Punctuator and Truecaser 6
3.3 Segmentation 6

4 Online SLT **7**
4.1 Evaluating Online SLT 8
4.1.1 SLTev: Open-source toolkit for Evaluation of Spoken Language Translation 8
4.2 Improving the Retranslation Approach 9
4.2.1 Prefix Training 9
4.2.2 Multi-Sentence Prefix Training 10
4.2.3 Dynamic Masking 11

5 ELITR's Submissions to IWSLT **13**

6 End-to-End SLT **13**
6.1 Spoken Language Translation via Phoneme-level Representation of the Source Language 14
6.2 Attention-passing and Deep Transformer Models for End-to-End Speech Translation 16
6.3 Adaptive Feature Selection for End-to-End Spoken Language Translation 16

7 Conclusion **16**

References **17**

Appendices **20**

Appendix A Dynamic Masking for Improved Stability on Spoken Language Translation **20**

Appendix B ELITR Non-Native Speech Translation at IWSLT 2020 **28**

Appendix C Non-Native Speech Translation Task Results (selected pages from Findings of the IWSLT 2020 Evaluation Campaign) **37**

Appendix D Adaptive Feature Selection for End-to-End Speech Translation **46**

Appendix E Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation **57**

1 Executive Summary

This deliverable describes the work of WP3 in the first 18 months of the ELITR project. In this work-package we address spoken language translation (SLT); applying different strategies to improve the interface between automatic speech recognition (ASR) and machine translation (MT).

Firstly we explain how we post-process ASR output to make it more similar to the type of clean written text used to train MT. For this we have developed normalisation components, which take raw ASR output, remove disfluencies, add punctuation, and apply other fixes to make it more like written text.

Secondly, we consider the problem of online SLT, which is important for running SLT in live scenarios such as subtitling. Standard MT systems are trained to translate sentence-by-sentence and generally run in batch mode, but in online SLT, the input to the MT system is being provided incrementally, a few words at a time. Additionally, a low-latency ASR system will output an initial hypothesis which it can then update as the speaker continues and it obtains more context. The downstream MT system must be able to handle these sentence fragments well, and should aim for low-latency translation, without introducing too much "flicker" into the output. We show new research addressing both of these problems.

We also investigate a different strategy for building SLT systems – end-to-end models which perform both MT and ASR in the one model. These have attracted a lot of attention recently, and have some potential advantages such as robustness to noise and simplicity, but often their overall performance is not as good as a cascade of ASR and MT. We develop several different architectural improvements which improve performance of E2E SLT.

Finally we explain how we put all our SLT components together to build systems for a recent SLT shared task.

2 Introduction

This is the interim report on the research of the spoken language translation (SLT) work package. From the Grant Agreemet, the aim of this work package is:

> To ensure high-quality MT for spoken language input, which lags behind the quality of text-to-text translation due to the noisy nature of ASR.

Viewing this from a system-building perspective, WP3 is about improving the connection between ASR (as developed in WP2) and MT (as developed in WP4).

In the project plan, the work of WP3 was split into three tasks reflecting different ways of improving the ASR/MT interface. For the first two tasks we assume a "pipeline" (or "cascade") approach to SLT, where the discrete output of the ASR system is fed into the MT system. In task T3.1 (ASR transcript normalization) we develop methods for post-processing the output of the ASR so that it can be more accurately translated by a text-to-text MT system; whilst in task T3.2. we aim to make the MT system more robust to translating the output from ASR, especially when it contains errors. For task T3.3, we take a different approach, investigating approaches that try to model ASR and MT jointly – in an end-to-end SLT system.

Below we give a brief explanation of how our work fits with the three tasks in this WP, and in the sections that follow we describe our progress in more detail. Where appropriate, we include published or drafted research papers, explaining how they are linked to the project.

T3.1: Normalization of ASR Normalisation of ASR output includes disfluency removal, punctuation, segmentation and truecasing. For these steps we have developed specialised components, as described in Section 3. We also addressed normalisation of ASR in our submssions to the IWSLT (International Workshop on Spoken Language Translation) shared tasks, described in Section 5.

T3.2: Robust Neural Machine Translation with Noisy Input Early in the project we realised that an important problem for the MT system was the fact that the transcriptions from ASR were delivered incrementally, and could be rewritten as the ASR component updated its hypotheses. Rather than having the MT system wait for ASR to stabilise before translating, we investigated ways to produce the translation as early as possible, i.e. Online SLT (Section 4). We also experimented with online SLT systems in our IWSLT systems (Section 5).

T3.3: End-to-End Speech-to-Text Translation We have been investigating these models as a potential way of improving on standard pipeline models for SLT (Section 6) although so far we have only used pipeline models in production.

3 Normalisation and Segmentation of ASR

We use two approaches for normalizing raw ASR output into cased punctuated text. The first is implemented by KIT (Section 3.1), the other by CUNI (Section 3.2). In creating the second normalizer, CUNI's goal was to make a more generalizable component that would work with several different ASR systems, whereas KIT's component is closely coupled to KIT's ASR.

The normalization consists of punctuation restoration, truecasing, and, in case of KIT, of removing disfluency phenomena. The normalized output is then segmented into individual sentences for MT (Section 3.3).

3.1 KIT Punctuator, Truecaser and Normalizer

The hypotheses from speech recognition contain no punctuation. In order to support a machine translation system trained on well-structured, written sentence-level texts, we use a separate

component to insert punctuation and sentence boundaries into the ASR output. This component also adds correct capitalization to the otherwise lower-cased hypotheses.

Essentially, the punctuation system is a monolingual translation system, which translates the lower-cased, unsegmented outputs from the ASR components into well-formed texts prior to the translation system (Cho et al., 2015). We can employ any kind of translation approach and only a small amount of monolingual data is required for training. In our current punctuation system, for each language, we train a neural sequence-to-sequence model on spoken texts, e.g. the transcripts of TED talks. Using our compact representation described by Cho et al. (2017), we are able to add punctuation and correct capitalization in one go. Furthermore, this compact representation helps to reduce the vocabulary size of our monolingual translation system, thus, reducing the model size and making both the training and inference faster.

The models are trained on the multilingual TED corpus, which is just under 200k sentences for English, and somewhat less for the other languages.

This component is also described in our paper (Franceschini et al., 2020), published in IWLTP (International Workshop on Language Technology Platforms).

3.2 CUNI Punctuator and Truecaser

The CUNI Punctuator system uses a bidirectional recurrent neural network with an attention-based mechanism (as described by Tilk and Alumäe (2016)) to restore punctuation in the raw stream of ASR output. The attention mechanism enhances its capacity for determining the contextual meaning in the ASR stream in order to restore punctuation effectively.

We used the CzEng 2.0 dataset to train the Czech punctuator model and CzEng 1.6 (Bojar et al., 2016) to train the English punctuator model. We chose CzEng because it is a mixture of domains, including both originally spoken, which is close to the target domain, and written, which has richer vocabulary. Furthermore it includes both texts that were originally in English, and texts originally in Czech which are also expected in our target application. The details of the numbers of sentences in the training data and the vocabulary size are shown in Table 1 and the performance of the punctuator is shown in Table 2.

	Dataset	**Train**	**Test**	**Vocabulary size**
CS Punctuator	CzEng 2.0	38.60M	200K	1.18M
EN Punctuator	CzEng 1.6	3.29M	200K	100K

Table 1: Dataset description of punctuator systems and number of sentences used to train model along with most frequent words in the training corpus.

The punctuator system was further integrated with an English tri-gram truecaser by Lita et al. (2003) that restores the casing of non-cased or badly-cased sentences. We used the Czech side of Europarl's Czech-English parallel corpus to train the Czech truecaser and the English truecaser was trained on 2M English sentences from CzEng.

3.3 Segmentation

The KIT and CUNI punctuating, truecasing and normalization components are deployed as workers in the Mediator (refer to Franceschini et al., 2020 for a description of our platform). The punctuated and normalized text is segmented by a language specific rule-based Moses Sentence Splitter (Koehn et al., 2007) into individual sentences for MT.

The segmentation is performed on the same host as MT, since it is not resource-intensive. The advantages of this are that there is no additional delay due to communication with Mediator, and it enables flexibility of the MT cascade for easier swapping between bilingual MT, multilingual MT and pivoting.

	Comma			Period			Question			Overall			
	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1	SER
CS Punctuator	82.1	78.6	80.3	81.1	85.2	83.1	78.5	71.9	75.1	81.2	78.5	79.8	26.3
EN Punctuator	77.4	71.0	74.0	81.2	86.0	83.6	84.5	77.4	80.8	80.3	77.2	78.7	28.4
Tilk and Alumäe (2016)	64.4	45.2	53.1	72.3	71.5	71.9	67.5	58.7	62.8	68.9	58.1	63.1	51.3

Table 2: Results on CzEng dataset in terms of precision, recall, F1-score, and overall slot error rate (SER).

4 Online SLT

Since ELITR aims to develop a system for live captioning of events, we decided that it would be important for SLT to be able to pass the translations to the presentation layer (for output to the user) as quickly as possible. This means that the person watching the captions would be able to view them with very little delay after they have been spoken. We refer to this mode of operation as "Online SLT", as opposed to "Offline SLT" where we do not translate until we have the whole passage.

If we assume a pipeline approach to SLT, where we have the output of ASR feeding in to an MT system, then ELITR ASR is already able to operate in an "Online" fashion. This means that, in theory, ASR sends a message to MT every time it updates its hypothesis[1], where these updates may be short extensions, or rewrites of the most recent words. ASR also considers the older parts of its hypothesis as "stable", meaning that they are not updated.

More formally, we can consider that the current ASR hypothesis can be expressed as the concatenation of two buffers, the "stable buffer" (S), and the "unstable buffer" (U). ASR can send two different types of message to MT:

- Updates to the unstable buffer. This replaces U with U' where U' may be either a simple extension of U (i.e. U is a prefix of U') or it may be a rewrite of U.
- Extension of the stable region. A prefix of U is removed and appended to the end of S.

If we also view segmentation of the transcription to be formally part of ASR, we should note that an update of the unstable buffer may include a change of sentence segmentation. An example of online ASR is shown in Table 3

Thank.
Thank you. Like
Thank you like to
Thank you like to invite you.
Thank you like to invite you to close
Thank you like to invite you to close your eyes.
Thank you like to invite you to close your eyes. Imagine
Thank you like to invite you to close your eyes. Imagine yourself
Thank you like to invite you to close your eyes. Imagine yourself, Stan,
Thank you like to invite you to close your eyes. Imagine yourself standing

Table 3: Sample output from the ASR system. Each line is a separate update, and all of these updates are "unstable" (i.e. may be revised by later updates).

In the example we can see one point where the sentence segmentation changes (between the second and third lines the full stop before "like" is removed) and in the last line the end of

[1]In the current production system we find that the MT pipeline is not fast enough to translate every update, so we have a wrapper which buffers the updates and can skip sending them to MT if it is overloaded

the hypothesis is rewritten ("Stan" → "standing"). Also we note that all of these are unstable updates – our online ASR can often have two or three sentences in the unstable buffer.

To develop good online SLT, we need an MT system that can deal with ASR output of the above form in a sensible way. There are two broad approaches to online SLT, usually known as the *streaming* approach and the *retranslation* approach. Both of these assume a pipeline system (as opposed to end-to-end). In the streaming approach (Cho and Esipova, 2016; Gu et al., 2017; Ma et al., 2019; Zheng et al., 2019b,a; Arivazhagan et al., 2019), the basic idea is that the MT system has an agent which, at each step, chooses whether to extend the translation, or read more input from the ASR. For this approach, a specialised MT inference algorithm is required, and usually a specialised MT training algorithm.

In the retranslation approach (Niehues et al., 2016, 2018; Arivazhagan et al., 2020a), we wrap a standard MT system to create an online system, making it much easier to benefit from improvements to the standard, batch-translated text-to-text MT technology. As the ASR output is passed to the MT system, we simply retranslate each sentence prefix and pass the output to the presentation layer. In the ELITR production system we use the retranslation approach because of its simplicity, and since we use the fast Marian toolkit (Junczys-Dowmunt et al., 2018) for inference, we are less concerned about the additional computational load of retranslating. Indeed, recent work (Arivazhagan et al., 2020b) has shown that the retranslation approach can have distinct advantages over streaming (in terms of latency).

The baseline approach to retranslation, however, will perform poorly. A standard MT system is not trained on sentence prefixes so will not perform optimally on them, and since linguistic differences can force reordering in MT, there can be a lot of annoying flicker in the updates. To address these problems we have investigated improvements to the baseline retranslation system (Section 4.2). First though, we needed to understand how to evaluate online SLT (Section 4.1), in order to measure the trade-offs between different aspects of its output.

4.1 Evaluating Online SLT

In the evaluation of online SLT, we consider three different factors (quality, latency and flicker) which must be traded off against each other. A successful approach to online SLT should be able to provide systems which occupy the *Pareto frontier* for these three measures.

quality is measured using one of the usual automatic metrics, and can refer to the quality of sentence translation, as well as the quality of the translation of the prefixes.

latency is the "lag" between words being received by MT from ASR and their translation being sent to the presentation layer.

flicker is the amount that translations change between successive updates.

If the MT system only translates stable ASR output, then its output will always lag behind the speaker, often for 2-3 sentences; in other words it will have high latency, but quality will be the same as offline MT and there will be no flicker. If, on the other hand, the MT system translates immediately when it receives an update from ASR, then latency will be low but we are faced with the choice of rewriting the output when there is a change (and potentially having high flicker) or be continue to extend a poor early hypothesis (and harming quality).

4.1.1 SLTev: Open-source toolkit for Evaluation of Spoken Language Translation

To provide a clear standard for SLT evaluation, we develop SLTev, an open-source tool for assessing the quality of spoken language translation in a comprehensive way. Based on time-stamped reference transcript and reference translation into a target language, SLTev reports the quality, delay and stability of a given SLT candidate output. To overtake the segmentation problem, we proposed a new time-based segmentation, in addition to the classical segmentation provided by mwerSegmenter toolkit (Matusov et al., 2005).

To calculate the translation quality, we rely on sacreBLEU[2] (Post, 2018). Similarly to sacreBLEU in MT evaluation, SLTev tags the scores with a fingerprint unambiguously identifying the evaluation settings.

Delay calculation relies on timing information provided by the user for individual segments. Each produced word is assumed to have appeared at the time that corresponds proportionally to its (character) position in the segment. The same strategy is used for the reference words. Note that the candidate segmentation does not need to match the reference one, but in both cases, we get an estimated time span for each word. SLTev calculates several delay metrics based on different strategies. For example, one of them uses GIZA++ (Och and Ney, 2003) alignment to deal with word reordering in the translated text.

Finally, we proposed two stability metrics. The first is based on counting the number of words after the first difference between two consecutive sub-outputs. In the second metric, which is inspired by Arivazhagan et al. (2020a), we report a normalized revision score calculated by dividing the total number of words produced by the true output length, i.e. by the number of words in the completed sentences. We report the average score across all documents in the test set.

We use the proposed evaluation framework to evaluate translation quality, latency, and stability of submitted works in the IWSLT 2020 Non-Native Speech Translation Shared Task.[3] The results are reported in Ansari et al. (2020), see Appendix C for the relevant pages of the paper.

4.2 Improving the Retranslation Approach

4.2.1 Prefix Training

At UEDIN, we experimented with a similar prefix-training setup to that used by Niehues et al. (2016). We train a transformer-regular system using the IWSLT17 en-de training set (Cettolo et al., 2017). We show results on the tst2010 test set.

We trained two different systems. The baseline system used only the parallel, full-sentence data. For the prefix system, we randomly selected a prefix length for every sentence in the parallel data, then truncated the source sentence to that length, and truncated its corresponding target sentence proportionately. We experimented with using alignments from fastalign (Dyer et al., 2013) to create a more informed correspondence between source and target truncations, but the performance of this was not different from simple truncation (Niehues et al. (2016) made similar observations).

We evaluate the two models on full sentences, and on prefixes. In order to create a prefix test set we apply the same truncation to the source-side of the test set, calculate BLEU precision as normal, and modify the BLEU length penalty to use the expected length of the hypothesis, given the source prefix length. The results are shown in Table 4. We notice that, as could be expected, the prefix system does worse on full sentences, but better on prefixes. The baseline system tends to produce translations that are too long, when translating prefixes, as it tries to complete to a full sentence.

System	Full	Prefix	Length ratio
baseline	29.0	25.4	1.176
prefix	27.4	27.6	0.989

Table 4: Comparison of BLEU scores between system trained on only full sentences (baseline) and a system with training data augmented with truncated sentences (prefix). The systems are tested both on full sentences, and on prefixes, and the length ratio is shown for the prefix translations. This is for English→German, IWSLT tst2010.

[2] We use the default settings, i.e. the signature `BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.4.6`.
[3] http://iwslt.org/doku.php?id=non_native_speech_translation

We use the system trained on prefixes in our production MT workers at UEDIN.

4.2.2 Multi-Sentence Prefix Training

In CUNI, we attempted to adapt our English-to-Czech translation system for online SLT by retranslation, using a multi-sentence prefix-training approach. Our motivation was to propose a model that achieves higher stability than the baseline single-sentence model, and higher quality in inter-sentential context. We used the method of Niehues et al. (2018), who proposed fine-tuning of a basic single-sentence translation model on a mixture of full sentences, and aligned source-target prefixes.

We used our strong Marian model from WMT19 (Popel et al., 2019) as a base for fine-tuning. We used synthetic news data, with original Czech target and English source back-translated by the WMT18 Czech-English model (Popel, 2018). We extracted semantically well-aligned source-target prefixes using a GIZA++ word alignment. Then we generated 100-subword windows of the prefixes across the sentence boundaries within the documents.

The raw training data consist of 60 million sentences. We generated all the 100-subword windows from a first and second ninth of training data, randomly sampled 60 million from each ninth, and shuffled them with raw data. The ratio of full-sentence and prefixes is 1:1, however, the model could be overtrained for the sentences in the first two ninths. We selected this setup for initial validation of model performance.

We integrated the model into the ELITR SLT framework for online SLT, with an initial version of MT Wrapper (see Macháček et al., 2020, Section 5.3), that resolves ASR updates in one thread and translates and caches them in second thread. It skips the hypotheses outdated during translation.

We decoded the first TED talk from IWSLT tst2015 with the model. We assesed it with a simple manual comparison with a baseline single-sentence WMT19 model, and observed following:

- The prefix model had higher erasure rate than the base single-sentence model, because it often modified up to 100 subwords from the end, which can be around 3 or 5 sentences from the end. The baseline model updates only 1 or 2 sentences from the end.
- The stability could be improved by disabling updates of sentences behind certain size or timing threshold, and considering them only for translation context.
- The bigger the decoding window size, the longer the translation time. Translation of 100 subwords by Transformer Big takes around 2 seconds, while a single sentence can have around 20 subwords and takes 0.2 seconds.
- The stability and latency of the prefix model approaches the baseline only if we restrict the decoding window size to around a sentence size.
- The prefix model is able to correctly determine the number of sentences that it should output.
- The baseline model cannot be used in prefix mode because it produces single sentences, even when the input has several sentences.
- The KIT punctuator sometimes changes the punctuation in text a long way from the end. Freezing older punctuation could improve stability.

By comparing the BLEU scores on the whole tst2015, we realized that the model had lower quality than the baseline both on ASR and gold transcripts, probably due to the suboptimal data sampling, and due to overtraining for the news domain.

To conclude, this work resulted in improvements to the MT Wrapper and observations that we will use for designing further experiments.

4.2.3 Dynamic Masking

In a recent paper (Arivazhagan et al., 2020a), two methods were proposed for improving the retranslation approach: masking and biased beam search. In masking, the translation system "masks" the last k words of its output every time it translates, except at the end of a sentence. The idea of biased beam search is that the beam search during translation is "biased" towards the translation of the previous prefix.

Using masking and biased beam search, it is possible to a achieve a better trade-off between latency, flicker and quality. Arivazhagan et al. (2020a) showed how the could use these two techniques to significantly reduce the amount of flicker, with only a small increase in latency, and a small decrease in quality.

We built on the work of this paper in two ways:

1. investigating more fully the trade-offs between the evaluation measures offered by masking and biased beam search
2. developing a method to dynamically control the mask based on the current source and its translation

The disadvantages of using biased beam search are that it can have a negative impact on the quality of the full sentence translation (since it biases the translation of a prefix towards the translation of the previous prefix, and away from the translation with the highest model score), and it does not work in the scenario where ASR can rewrite as well as extend. Also, using a fixed mask is a blunt instrument – in Arivazhagan et al. (2020a) they get good stability with a global fixed mask of 10, but in many cases a stable translation can be achieved with a much smaller mask.

Our initial approach to dynamic masking (i.e. using a variable length mask) was based on word confidence, quite similar to the approach in Kepler et al. (2019). The idea was that, when we translate, we check the last words at the end of the hypothesis, and predict either "mask" or "no-mask", using a model based on running an LSTM over source and hypothesis. We were able to generate training data by translating sentences and their possible prefixes with our translation system, and deriving the optimal mask size for each prefix by considering the translation of the full sentence. Unfortunately this model was completely unable to predict the correct mask, and we abandoned it in favour of a different formulation of the problem which we describe below.

The second, and more successful, approach to dynamic masking is one that we refer to as *source prediction*. The intuition behind this approach is that the system makes probes to determine what effect potential extensions of the source would have on the translation. If these probes show that there could be large changes in the translation in the next prefix (i.e. the translation is unstable), then it needs to apply large mask, and conversely if the translation so far is shown to be stable, then a mask is not required.

More formally, the source prediction approach works like this. The MT system receives a prefix S from the ASR component. It then makes an extension of the prefix to S', and translates both S and S' to produce T and T', respectively. The system then compares T and T', and determines the longest common prefix (LCP) of these two strings. In general it will output this LCP, masking the remaining part of T, however to deal with some less common cases we introduce an additional refinement. If the LCP is a prefix of the previous translation output (i.e. the output from the prefix before S) then we just use the previous output. This last refinement helps to smooth out temporary instabilities in the translation which can cause unnecessarily large masks.

We experimented with different methods for predicting the extension to S: using a language model, randomly sampling from the source vocabulary, and simply add UNK symbols. All methods were able to improve on the latency–flicker trade-off curve given by mask-k, but using a language model tended to offer reduced latency than the other methods, at the expense of slightly larger flicker. A further adjustment of the flicker–latency trade-off curve can be achieved by varying the length of the predicted extensions.

Further details of the source prediction method, and the experimental results can be found in our preprint (Yao and Haddow, 2020), which we include as Appendix A

We note that the results of this paper were obtained using simulated ASR. That is to say, we take the gold transcripts and feed them to the MT system one word at a time. The simulated ASR only extends, it does not rewrite, and it does not change sentence boundaries. To address this limitation, we now show results on actual online ASR output. In actual ASR output, sentence boundaries are inserted by a punctuation component as described in Section 3, but are unstable and may be revised as the ASR output is updated.

To test the dynamic mask, we used the ASR transcripts produced by our "primary online" ASR system in our IWSLT submission (Section 5). These transcripts show both the stable and unstable updates, as described at the start of Section 4. We use the English→German system as in Yao and Haddow (2020), and we provide evaluations of flicker and latency (as defined in the paper) for different types of masking. We do not measure translation quality since all variants produce the same full sentence output. We measure the flicker and latency on each document in the corpus, treating it as one long sentence for this evaluation. We use token counts for measuring latency, rather than actual time and use the same method as described in the paper. For the masking, we have zero mask at a sentence boundary (as in Yao and Haddow (2020)), despite the fact that sentence boundaries can be unstable.

We assess the following retranslation strategies:

baseline Translate as soon as the ASR output arrives

wait until complete Translate as soon as a complete sentence is available (it may be unstable).

wait until stable Translate a sentence only when the ASR system marks it as stable.

fixed mask Mask the last k words of the sentence, where $k = 5, 10, 15$.

lm-unk Use dynamic masking with the "unknown" strategy from Yao and Haddow (2020), with $k = 5, 10, 15, 20$.

lm-sample Use dynamic masking with the "sample" strategy, using 5 samples of length 5.

We show the comparison of these different strategies in Figure 1.

We notice from Figure 1 that the dynamic mask strategy occupies the Pareto frontier, outperforming the fixed mask (note that the horizontal axis is extended in this Figure because of the long latency of the "wait until stable" strategy). We also note that all the masking strategies have relatively high erasure; this is because of the instability of sentence boundaries and the fact that we do not mask at sentence boundaries. Because of the relatively short sentences, and instability of the sentence boundaries, the "wait until complete" strategy performs relatively well, showing improved flicker, but worse latency than the dynamic mask strategies. With better ASR (and so more stable sentence boundaries) the dynamic masking strategy should offer a greater advantage. An alternative would be to allow masking across sentence boundaries, which we have not yet investigated.

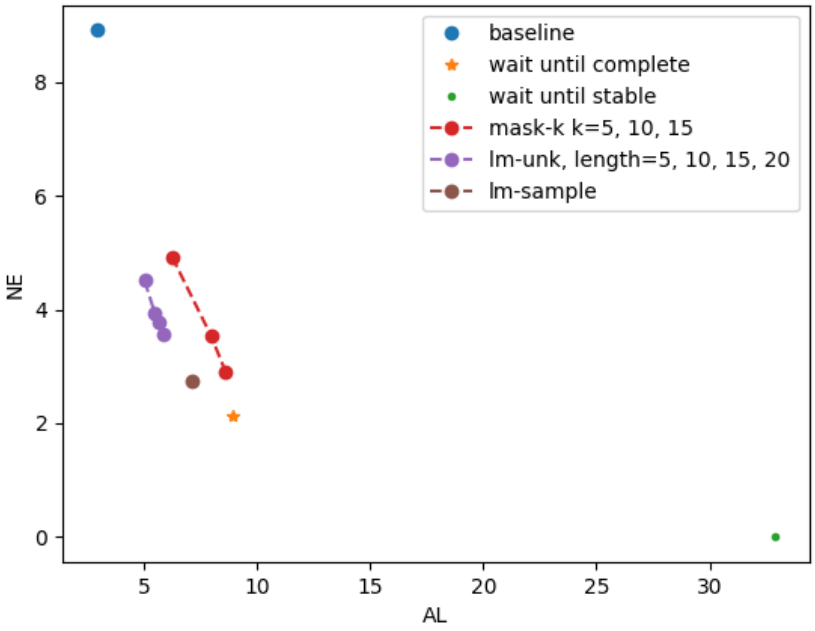


Figure 1: Comparison of dynamic and fixed mask, with retranslation strategies on ASR transcripts. We show the latency (average lag) and flicker (normalised erasure) tradeoff curve.

5 ELITR's Submissions to IWSLT

ELITR participated in the non-native speech translation task at IWSLT 2020. The challenge was online and offline SLT of non-native speech from English into Czech and German. Our short-term motivation for this work was to integrate all the components that consortium partners had available into a working cascade for online and offline SLT, and evaluate them end-to-end.

We compared and selected primary candidates from a pool of 3 online English ASRs candidates, 4 offlines, and 9 from MT systems: 3 into Czech, 2 into German and 4 multilingual. We validated the candidates on a very limited validation set, and selected KIT-hybrid ASR for online ASR, KIT-seq2seq for offline ASR, UEDIN IWSLT19 MT model into Czech, and UEDIN OPUS-B multilingual model for German. More details on primary systems are in Macháček et al. (2020) and in Appendix B.

The evaluation results are in Ansari et al. (2020) and we reproduce the pages relevant for the respective IWSLT shared task results here in Appendix C.

6 End-to-End SLT

In the third task of this work package, we have been investigating an alternative approach to SLT which does not use the traditional cascade. The idea is to train a single model that performs both ASR and MT, and so can directly convert speech in one language into text in another language. This is known as end-to-end SLT, or E2E SLT. The potential value of such models is that ASR and MT learn to cooperate well with each other at training time, becoming one system, rather than a heuristic pipeline of technologies forced into working together. An E2E system could also take advantage of non-verbal information in the speech signal.

There has been a lot of recent interest in E2E SLT in the literature (see Sperber and Paulik (2020) for a useful review) but the general consensus is that pipeline systems can still perform better, due to being better able to take advantage of large training data sets, and benefitting from many years of research into their components. Also, online SLT using E2E systems is still

difficult (see *D2.1* for the related problem of online ASR using end-to-end ASR systems).

One claim frequently made about E2E SLT is that it offers better resistance to ASR errors due to acoustic noise, than a pipeline system. There is little experimental evidence to back this up though, so this is one avenue we are exploring at UEDIN, by manipulating the input audio in order to investigate the effect on the output.

In the following subsections we describe three pieces of research (either published or under-review) contributed by ELITR researchers, and aimed at improving E2E SLT. Since the papers are included as appendices, we only include a brief summary of the work here.

6.1 Spoken Language Translation via Phoneme-level Representation of the Source Language

In recent years, there has been a lot of interest in end-to-end architectures for SLT. The advantage of end-to-end solutions is their technical simplicity and better preservation of nonverbal information contained in the speech. On the other hand, their obvious limitation is the requirement of "end-to-end" training datasets, i.e., source language voice recordings paired with the translated transcripts in the target language. There are only a small number of such datasets available, compared to the much larger range of datasets for MT and ASR.

Motivated by the findings of Salesky et al. (2019), we explore a re-purposing of the more traditional pipeline (i.e., a tandem of standalone ASR and MT) in the context of non-native English to Czech SLT and Czech to English SLT. As the intermediate representation, we choose phonemes instead of conventional graphemes. Specifically, we compare the traditional approach (with graphemes as intermediate step) with the proposed one (with the phonemes as intermediate step).

To evaluate the pipeline, we collect our own test set. The test set is based on newstest2015 and is read by a Czech woman. We report the results in terms of BLEU in Tables 5 and 6.

As we can observe in the tables, the SLT approach with phoneme-level intermediate step outperforms the baseline SLT approach in both directions. When tested on clean source (i.e., only MT), the baseline seems to be a better option.

For further details, refer to Peter Polák's MSc thesis (Polák, 2020).

	Model	Source	Source error	Target	Cased, interpuct. Beam Size 1	4	16	Uncased, no interpunct. Beam Size 1	4	16
SLT (ASR source)	P2G	ASR Phon	PWER 34.33 %	En Graph	19.47	**20.07**	19.75	19.56	**19.96**	19.43
	baseline	ASR Graph	WER 34.51 %		18.39	19.69	19.01	18.5	19.74	18.99
Translation (clean source)	P2G	Clean Phon	0 %	En Graph	30.14	30.82	30.35	30.44	30.82	30.56
	baseline	Clean† Graph			30.31	**31.07**	30.45	30.74	**31.55**	30.77

Table 5: Evaluation of the proposed Czech to English model (phonemes to graphemes — P2G) and the Czech to English baseline (graphemes to graphemes). We evaluate performance on SLT and Translation task. SLT task obtained source from ASR transcripts. Translation task is done on clean (original) source.
† ASR-like Graph is original lowercase source with stripped interpunction.

	Model	Souce	Souce error	Target	Cased, interpuct. Beam Size 1	4	16	Uncased, no interpunct. Beam Size 1	4	16
SLT (ASR source)	P2G	ASR Phon	PWER 18.61 %	Cs Graph	16.64	17.12	**17.35**	14.28	14.76	**14.9**
	baseline	ASR Graph	WER 18.80 %		16.51	16.83	16.68	14.29	14.53	14.42
Translation (clean source)	P2G	Clean Phon	0 %	Cs Graph	21.75	22.32	22.26	19.6	20.01	20.06
	baseline	Clean† Graph			21.85	**22.47**	22.21	19.74	**20.09**	20.07

Table 6: Evaluation of the proposed English to Czech model (phonemes to graphemes — P2G) and the English to Czech baseline (graphemes to graphemes). We evaluate performance on SLT and Translation task. SLT task obtained source from ASR transcripts. Translation task is done on clean (original) source.
† ASR-like Graph is original lowercase source with stripped interpunction.

6.2 Attention-passing and Deep Transformer Models for End-to-End Speech Translation

In Sperber et al. (2019) we proposed an end-to-end speech translation architecture that is able to make better use of the available datasets for speech recognition and machine translation. This is an important step, because the amount of data for direct end-to-end speech translation is very limited. Being able to train parts of a model on existing ASR or MT data can allow the model to use significantly more data and achieve better results.

The method is based on an attention-passing method, sharing an attention method between all combination of source text and audio encoding and source and target text decoding. By additionally sharing parameters between the source text encoder and decoder, this setup ensures that the model can get the maximum training effect out of the auxiliary ASR and MT data.

While multitask training is an important technique for end-to-end speech translation, it seems that utilising the Transformer model (Vaswani et al., 2017) is another key step to achieving comparable performance to cascaded models. The Transformer has already demonstrated competitive performance to hybrid models on the ASR task (Pham et al., 2019), applying the same deep models to SLT is the next promising step for this research.

We include the paper in Appendix E.

6.3 Adaptive Feature Selection for End-to-End Spoken Language Translation

Information in speech signals is not evenly distributed, making it an additional challenge for end-to-end (E2E) speech translation (ST) to learn to focus on informative features. We have investigated adaptive feature selection (AFS) for encoder-decoder based end-to-end speech translation. We first pre-train an ASR encoder and apply AFS to dynamically estimate the importance of each encoded speech feature to ASR. A speech translation encoder, stacked on top of the ASR encoder, then receives the filtered features from the (frozen) ASR encoder. We take L0Drop (Zhang et al., 2020b) as the backbone for AFS, and adapt it to sparsify speech features with respect to both temporal and feature dimensions.

Results on LibriSpeech En-Fr (Kocabiyikoglu et al., 2018) and MuST-C (Di Gangi et al., 2019) benchmarks show that AFS facilitates learning of speech translation by pruning out ∼84% temporal features, yielding an average translation gain of ∼1.3–1.6 BLEU and a decoding speedup of ∼1.4×. In particular, AFS reduces the performance gap compared to the cascade baseline, and outperforms it on LibriSpeech En-Fr with a BLEU score of 18.56 (without data augmentation).

More details are in Appendix D, which shows a preprint currently under review.

7 Conclusion

This deliverable has presented the work to date on SLT in ELITR. In this work-package, we have focused on building and improving the ELITR production system, evaluation of online SLT, and research into online SLT and end-to-end SLT. In the second half of the project, we expect to continue with these lines of research, as well as transferring the research to the production system wherever possible.

Papers

The following papers have resulted from the work of WP3. They are all available in the appendices.

- *Dynamic Masking for Improved Stability on Spoken Language Translation* Yao and Haddow (2020) (Published on Arxiv, and will be submitted to AMTA)
- *ELITR Non-Native Speech Translation at IWSLT 2020* Macháček et al. (2020) (To appear in IWSLT 2020)

- *Findings of the IWSLT 2020 Evaluation Campaign* Ansari et al. (2020) (organization of the IWSLT 2020 Shared Task on Non-Native Speech Translation. To appear in IWSLT 2020)
- *Adaptive Feature Selection for End-to-End Speech Translation* Zhang et al. (2020a) (Submitted to EMNLP)
- *Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation* Sperber et al. (2019) (Published in TACL)

References

Ebrahim Ansari, Amittai Axelrod, Nguyen Bach, Ondrej Bojar, Roldano Cattoni, Fahim Dalvi, Nadir Durrani, Marcello Federico, Christian Federmann, Jiatao Gu, Fei Huang, Kevin Knight, Xutai Ma, Ajay Nagesh, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Xing Shi, Sebastian Stüker, Marco Turchi, and Changhan Wang. Findings of the IWSLT 2020 Evaluation Campaign. In *Proceedings of the 17th International Conference on Spoken Language Translation (IWSLT 2020)*, Seattle, USA, 2020.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. Monotonic Infinite Lookback Attention for Simultaneous Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1126. URL https://www.aclweb.org/anthology/P19-1126.

Naveen Arivazhagan, Colin Cherry, Te I, Wolfgang Macherey, Pallavi Baljekar, and George Foster. Re-Translation Strategies For Long Form, Simultaneous, Spoken Language Translation. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020a.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George Foster. Re-translation versus streaming for simultaneous translation, 2020b.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In *TSD*, 2016.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Kumiko Sudoh, Kyotaro Yoshino, and Christian Federmann. Overview of the IWSLT 2017 Evaluation Campaign. In *Proceedings of IWSLT*, 2017.

Eunah Cho, Jan Niehues, Kevin Kilgour, and Alex Waibel. Punctuation Insertion for Real-time Spoken Language Translation. In *Proceedings of the Eleventh International Workshop on Spoken Language Translation (IWSLT 2015)*, Da Nang, Vietnam, 2015.

Eunah Cho, Jan Niehues, and Alex Waibel. NMT-Based Segmentation and Punctuation Insertion for Real-Time Spoken Language Translation. *Proc. Interspeech 2017*, pages 2645–2649, 2017.

Kyunghyun Cho and Masha Esipova. Can neural machine translation do simultaneous translation? *CoRR*, abs/1606.02012, 2016. URL http://arxiv.org/abs/1606.02012.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2012–2017, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1202. URL https://www.aclweb.org/anthology/N19-1202.

Chris Dyer, Victor Chahuneau, and Noah A Smith. A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of NAACL*, 2013.

Dario Franceschini, Chiara Canton, Ivan Simonini, Armin Schweinfurth, Adelheid Glott, Sebastian Stüker, Thai-Son Nguyen, Felix Schneider, Thanh-Le Ha, Alex Waibel, Barry Haddow, Philip Williams, Rico Sennrich, Ondřej Bojar, Sangeet Sagar, Dominik Macháček, and Otakar Smrž. Removing European language barriers with innovative machine translation technology. In *Proceedings of the 1st International Workshop on Language Technology Platforms*, pages 44–49, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-64-1. URL https://www.aclweb.org/anthology/2020.iwltp-1.7.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1099.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P18-4020.

Fabio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André F. T. Martins. OpenKiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 117–122, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3020. URL https://www.aclweb.org/anthology/P19-3020.

Ali Can Kocabiyikoglu, Laurent Besacier, and Olivier Kraif. Augmenting librispeech with French translations: A multimodal corpus for direct speech translation evaluation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L18-1001.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. TRuEcasIng. In *ACL*, 2003. doi: 10.3115/1075096.1075116. URL https://doi.org/10.3115/1075096.1075116.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1289. URL https://www.aclweb.org/anthology/P19-1289.

Dominik Macháček, Jonáš Kratochvíl, Sangeet Sagar, Matúš Žilinec, Ondřej Bojar, Thai-Son Nguyen, Felix Schneider, Philip Williams, and Yuekun Yao. ELITR Non-Native Speech Translation at IWSLT 2020. In *Proceedings of the 17th International Conference on Spoken Language Translation (IWSLT 2020)*, Seattle, USA, 2020.

E. Matusov, G. Leusch, O. Bender, , and H. Ney. Evaluating Machine Translation Output with Automatic Sentence Segmentation. In *Proceedings of the 2nd International Workshop on Spoken Language Translation (IWSLT)*, Pittsburgh, USA, 2005.

Jan Niehues, Thai Son Nguyen, Eunah Cho, Thanh-Le Ha, Kevin Kilgour, Markus Müller, Matthias Sperber, Sebastian Stüker, and Alex Waibel. Dynamic transcription for low-latency speech translation. In *Proceedings of Interspeech*, 2016.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. Low-latency neural speech translation. In *Proceedings of Interspeech*, 2018.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, and Alex Waibel. Very deep self-attention networks for end-to-end speech recognition. *CoRR*, abs/1904.13377, 2019. URL http://arxiv.org/abs/1904.13377.

Peter Polák. Spoken language translation via phoneme representation of the source language. Master's thesis, Charles University, 2020.

Martin Popel. CUNI transformer neural MT system for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 482–487, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6424. URL https://www.aclweb.org/anthology/W18-6424.

Martin Popel, Dominik Macháček, Michal Auersperger, Ondřej Bojar, and Pavel Pecina. English-czech systems in wmt19: Document-level transformer. In *WMT*, 2019. ISBN 978-1-950737-27-7.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

Elizabeth Salesky, Matthias Sperber, and Alan W Black. Exploring phoneme-level speech representations for end-to-end speech translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1835–1841, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1179. URL https://www.aclweb.org/anthology/P19-1179.

Matthias Sperber and Matthias Paulik. Speech translation and the end-to-end promise: Taking stock of where we are. In *Proceedings of ACL*, 2020.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. Attention-passing models for robust and data-efficient end-to-end speech translation. *Transactions of the Association for Computational Linguistics*, 7:313–325, 2019.

Ottokar Tilk and Tanel Alumäe. Bidirectional recurrent neural network with attention mechanism for punctuation restoration. 2016. doi: 10.21437/Interspeech.2016-1517.

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

Yuekun Yao and Barry Haddow. Dynamic Masking for Improved Stability in Spoken Language Translation. *arXiv e-prints*, art. arXiv:2006.00249, May 2020.

Biao Zhang, Ivan Titov, Barry Haddow, and Rico Sennrich. Adaptive feature selection for end-to-end speech translation. Submitted to EMNLP, 2020a.

Biao Zhang, Ivan Titov, and Rico Sennrich. On sparsifying encoder outputs in sequence-to-sequence models. *arXiv preprint arXiv:2004.11854*, 2020b.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1349–1354, Hong Kong, China, November 2019a. Association for Computational Linguistics. doi: 10.18653/v1/D19-1137. URL https://www.aclweb.org/anthology/D19-1137.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5816–5822, Florence, Italy, July 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1582. URL https://www.aclweb.org/anthology/P19-1582.

A Dynamic Masking for Improved Stability on Spoken Language Translation

Dynamic Masking for Improved Stability in Spoken Language Translation

Yuekun Yao and Barry Haddow
University of Edinburgh

Abstract

For spoken language translation (SLT) in live scenarios such as conferences, lectures and meetings it is desirable to show the translations to the user as quickly as possible, and certainly without waiting until the end of the sentence. To achieve this we can pipeline an ASR system that can deliver results incrementally, with a standard MT system which retranslates each sentence prefix it receives. Naively done, this will result in annoying "flicker" as the MT system updates its translation. A possible solution is to add a fixed delay, or "mask" to the the output of the MT system, but a fixed global mask introduces undesirable latency to the output. We show that this mask can be set dynamically, providing a more optimal balance between latency and flicker.

1 Introduction

A common approach to Spoken Language Translation (SLT) is to use a cascade (or pipeline) consisting of automatic speech recognition (ASR) and machine translation (MT). In a live translation setting, such as a lecture or conference, we would like the transcriptions or translations to appear as quickly as possible, so that they do not "lag" noticeably behind the speaker. In other words, we wish to minimise the latency of the system. Many popular ASR toolkits can produce partial, or incremental, transcriptions. However incremental MT is less well supported, and is complicated by the reordering which is often necessary in translation, and by the use of encoder-decoder models which assume sight of the whole source sentence.

A straightforward approach to delivering incremental MT is to use a standard MT system, and produce a new translation every time a partial sentence is received from the ASR system. This can be referred to as the *retranslation* approach and has the advantage that we can use any standard MT toolkit, for example Marian (Junczys-Dowmunt et al., 2018) which is highly optimised for translation speed. However, when using a completely unadapted MT system in a live translation setting, there are at least two problems we need to consider:

1. MT training data generally consists of full sentences, and systems may perform poorly on partial sentences
2. When MT systems are asked to translate progressively longer segments of the conversation, they may introduce radical changes in the translation as the prefixes are extended. If these updates are displayed to the user, they will introduce an annoying "flicker" in the output, making it hard to read.

We illustrate these points using the small example in Figure 1. When the MT system receives the first prefix ("Several") it attempts to make a longer translation, due to its bias towards producing sentences. When the prefix is extended (to "Several years ago"), the MT system completely revises its original translation hypothesis. This is caused by the differing word order between German and English.

Several	$\longrightarrow$	Mehrere Male
		Several times
Several years ago	$\longrightarrow$	Vor einigen Jahre
		Several years ago

Figure 1: Sample translation with standard en→de MT system. We show the translation output, and its back-translation into English.

The first problem above could be addressed by simply adding sentence prefixes to the training data of the MT system. In fact Niehues et al.

(2018) showed that this can be an effective strategy for translating sentence fragments, and that the training pairs can be produced simply by truncating parallel sentences, without taking account of the word alignment. In our experiments we found that using prefixes in training required careful mixing of data, and even then performance of the model trained on truncated sentences was often worse on full sentences.

A way to address both problems above is with an appropriate retranslation strategy. In other words, when the MT system receives a new prefix, it should decide whether to transmit its translation in full, partially, or wait for further input, and the system can take into account translations it previously produced. A good retranslation strategy will address the second problem above (too much flickering as translations are revised) and in so doing so address the first (over-eagerness to produce full sentences).

In this paper, we focus on the retranslation methods introduced by Arivazhagan et al. (2020a) – mask-k and biased beam search. The former is a delayed output strategy which does not affect overall quality, but can significantly increase the latency of the translation system. The latter alters the beam search to take into account the translation of the previous prefix, and is used to reduce flicker without influencing latency much, but can also damage translation quality. We will show that using a straightforward method to predict the value of k in the mask-k strategy, we obtain a more optimal trade-off of flicker and latency, with no modifications to the underlying MT system, and with no effect on translation quality.

2 Incremental MT and Retranslation

Early work on incremental MT used prosody (Bangalore et al., 2012) or lexical cues (Rangarajan Sridhar et al., 2013) to make the translate-or-wait decision. The frst work on incremental neural MT used confidence to decide whether to wait or translate (Cho and Esipova, 2016), whilst in (Gu et al., 2017) they learn the translation schedule with reinforcement learning. In all these systems, the translation of a prefix is only ever extended, which removes flicker, but overall quality is degraded.

In Ma et al. (2019), they address simultaneous translation using a transformer (Vaswani et al., 2017) model with a modified attention mechanism, which is trained on prefixes. They introduce the idea of wait-k, where the translation is always k words behind the input, reducing flicker. This work was extended by Zheng et al. (2019b,a), where a "delay" token is added to the target vocabulary so the model can learn when to wait, through being trained by imitation learning. The MILk attention (Arivazhagan et al., 2019) also provides a way of learning the translation schedule along with the MT model, and is able to directly optimise the latency metric.

In contrast with these recent approaches, retranslation strategies allow the use of a standard MT toolkit, with little modification, and so are able to leverage all the performance and quality optimisations in that toolkit. The mask-k strategy of (Arivazhagan et al., 2020a) is inspired by the wait-k strategy, but is very easy to implement.

In an even more recent paper, contemporaneous with our work (Arivazhagan et al., 2020b) they further combine their re-translation system with prefix training and make comparison with current best streaming models (e.g. MILk and wait-k models), showing such a retranslation system is a strong baseline for Simultaneous Translation.

The idea of mask-k is simply that the MT system does not transmit the last k tokens of its output – in other words it masks them. Once the system receives a full sentence, it transmits the translation in full, without masking. The value of k is set globally and can be tuned to reduce the amount of flicker, at the cost of increasing latency.

Arivazhagan et al. (2020a) also introduced the idea of biased beam search, which requires a small modification to the translation algorithm, to change the search objective. Biased beam search aims to reduce flicker by ensuring that the translation produced by the MT system stays closer to the translation of the previous (shorter) prefix. Suppose that S is a source prefix, S' is the extension of that source prefix provided by the ASR, and T is the translation of S produced by the system (after masking). Then to create the translation T' of S', biased beam search substitutes the model probability $p(t'_i|t'_{<i}, S)$ with the following expression:

$$p^B(t'_i|t'_{<i}, S') = (1-\beta)\cdot p(t'_i|t'_{<i}, S') + \beta\cdot\delta(t'_i, t_i)$$

where t'_i is the i^{th} token of the translation hypothesis T', and β is a weighting which we set to 0 when $t_{<i} \neq t'_{<i}$. In other words, they interpolate the translation model with a function that keeps it

close to the previous target, but stop applying the biasing once the new translation diverges from the previous one.

As we noted earlier, biased beam search can degrade the quality of the translation, and we show experiments to illustrate this in Section 5. We also note that biased beam search assumes that the ASR simply extends its output each time it updates, when in fact ASR systems may rewrite their output.

3 Dynamic Masking

In the previous section we showed that biased beam search could address flicker (as measured by reduced erasure) without increasing latency, at the expense of reduction in translation quality. The mask-k strategy is also able to reduce the flicker, without affecting translation quality, but will cause an increase in latency. In this section we present an improvement on the mask-k strategy, which dynamically sets the mask, and so is able to offer reduced flicker with only a limited effect on latency, and still without affecting quality.

The main idea is to *predict* what the next source word will be, and check what effect this would have on the translation. If this changes the translation, then we mask, if not we output the full translation.

More formally, we suppose that we have a source prefx $S = s_1 \dots s_p$, a source-to-target translation system, and a function $pred_k$, which can predict the next k tokens following S. We translate S using the translation system to give a translation hypothesis $T = t_1 \dots t_q$. We then use $pred_k$ to predict the tokens following s_p in the source sentence to give an extended source prefix $S' = s_1 \dots s_p s_{p+1} \dots s_{p+k}$, and translate this to give another translation hypothesis T'. Comparing T and T', we select the longest common prefix T^*, and output this as the translation, thus masking the final $|T| - |T^*|$ tokens of the translation. If S is a complete sentence, then we do not mask any of the output, as in the mask-k strategy. The overall procedure is illustrated in Figure 2.

In fact, after initial experiments, we found it was more effective to refine our strategy, and not mask at all if the translation after dynamic mask is a prefix of the last translation. In this case we directly output the last translation. In other words, we do not mask if $is\_prefix(T_i^*, T_{i-1}^*)$ but instead output T_{i-1}^* again, where T_i^* denotes the masked translation for the ith ASR input. We also notice that this refinement does not give benefit in mask-k strategy in our experiments.

To predict the source extensions (i.e. to define the $pred_k$ function), we experimented with 4 different strategies:

lm-sample We sample the next token from a language model (LSTM) trained on the source-side of the parallel training data. We can choose n possible extensions by choosing n distinct samples.

lm-greedy This also uses an LM, but chooses the most probably token at each step.

unknown We extend the source sentence using the UNK token from the vocabulary.

random We extend by sampling randomly from the vocabulary, under a uniform distribution. As with lm-sample, we can generalise this strategy by choosing n different extensions.

4 Evaluation of Retranslation Strategies

As we stated in Section 2, we consider that the performance of our translation system can be assessed according to three different aspects – quality, latency and flicker. The system will generally need to trade off these properties against each other. For example outputting translations as early as possible will reduce latency, but if these early outputs are incorrect then either they can be corrected (increasing flicker) or retained as part of the later translation (reducing quality). In this section we will define precisely how we measure these system aspects. We assume that the optimal trade-off between quality, latency and flicker is a question that can only be settled by user testing.

Latency The latency of the MT system should provide a measure of the time between the MT system receiving input from the ASR, and it producing output that can be potentially be sent to the user. A standard MT system would wait until it has received a full sentence before it produces any output, which exhibits high latency. Cho and Esipova (2016) define latency by counting how many source tokens were seen before outputting each target token, then normalising this by sentence length. Ma et al. (2019) refer to this earlier method as average proportion (AP) and criticise

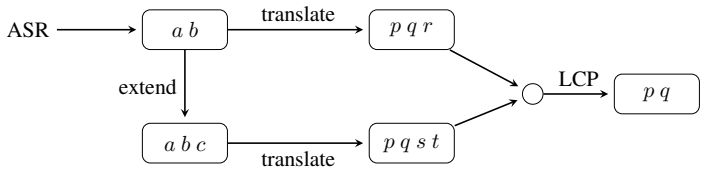


Figure 2: The source prediction process. The string $a\ b$ is provided by the ASR system. The MT system then produces translations of the string and its extension, compares them, and outputs the longest common prefix (LCP)

it because of the way it treats long and short sentences differently, and because it is hard to interpret. Instead they introduce a measure called average lag (AL), which measures the degree to which the output lags behind the input. This is done by averaging the difference between the number of words the system has output, and the number of words expected, given the length of the source prefix received, and the ratio between source and target length. Formally, AL for source and target sentences S and T is defined as:

$$\mathrm{AL}(S,T) = \frac{1}{\tau}\sum_{t=1}^{\tau} g(t) - \frac{(t-1)|S|}{|T|}$$

where τ is the number of target words generated by the time the whole source sentence is received, $g(t)$ is the number of source words processed when the t target word is produced.

Arivazhagan et al. (2020a) define latency as the mean time between a source word being received and the translation of that source word being finalised. However, this definition conflates latency and flicker, since outputting a translation and then updating is penalised for both aspects. The update is penalised for flicker since the translation is updated (see below) and it is penalied for latency, since the timestamp of the initial output is ignored in the latency calculation.

Given the shortcomings of AP, as pointed out by Ma et al. (2019), and that we prefer latency to be purely a measure of how quickly a translation is produced, relative to the input, we adopt AL (average lag) as our measure of latency. In our implementation, we calculate the AL at token (not subword) level with the standard tokenizer in sacreBLEU (Post, 2018), meaning that for Chinese output we calculate AL on characters.

Flicker The idea of flicker is to obtain a measure of the potentially distracting changes that are made to the MT output, as its ASR-supplied source sentence is extended. We assume that straightforward extensions of the MT output are fine, but changes which require re-writing of part of the MT output should result a higher (i.e. worse) flicker score. Following Arivazhagan et al. (2020a), we measure flicker using the normalised erasure (NE), which is defined as the minimum number of tokens that must be erased from each translation hypothesis when outputting the subsequent hypothesis, normalised across the sentence. As with AL, we also calculate the NE at token level.

Quality We measure quality by comparing the full sentence output of our system against the full reference sentence. This removes the need for a heuristic to determine partial references, and furthermore we assume that if the partial sentences are of poor quality, that this will be reflected in the other two measures. We use BLEU to assess the difference between the system output and the reference, applying the sacreBLEU implementation (Post, 2018).

5 Experiments

5.1 Biased Beam Search and Mask-k

We first assess the effectiveness of biased beam search and mask-k (with a fixed k), providing a more complete experimental picture than in (Arivazhagan et al., 2020a). For these experiments we use data released for the IWSLT MT task (Cettolo et al., 2017), in both English→German and English→Chinese. We consider a simulated ASR system, which supplies the input to the MT system one token at a time.

For training we use the TED talk data, with dev2010 as heldout and tst2010 as test set. The raw data set sizes are 206112 sentences (en-de) and 231266 sentences (en-zh). We preprocess using the Moses (Koehn et al., 2007) tokenizer and

truecaser (for English and German) and jieba[1] for Chinese. We apply BPE (Sennrich et al., 2016) jointly with 90k merge operations. For our MT system, we use the transformer-base architecture (Vaswani et al., 2017) as implemented by Nematus (Sennrich et al., 2017). We use 256 sentence mini-batches, and a 4000 iteration warm-up in training.

As we mentioned in the introduction, we did experiment with prefix training (using both alignment-based and length-based truncation) and found that this improved the translation of prefixes, but generally degraded translation for full sentences. Since prefix translation can also be improved using the masking and biasing techniques, and the former does not degrade full sentence translation, we only include experimental results when training on full sentences.

In Figure 3 we show the effect of varying β and k on our three evaluation measures, for English→German.

Looking at Figure 3(a) we notice that biased beam search has a strong impact in reducing flicker (erasure) at all values of β. However the problem with this approach is clear in Figure 3(b), where we can see the reduction in BLEU caused by this biasing. This can be offset by increasing masking, also noted in (Arivazhagan et al., 2020a), but as we show in Figure 3(c) this comes at the cost of an increase in latency.

Our experiments with en→zh show a roughly similar pattern, as shown in Figure 4. We find that lower levels of masking are required to reduce the detrimental effect on BLEU of the biasing, but latency increases more rapidly with masking.

5.2 Dynamic Masking

We now turn our attention to the dynamic masking technique introduced in Section 3. We use the same data sets and MT systems as in the previous section. To train the LM, we use the source side of the parallel training data, and train an LSTM-based LM.

To assess the performance of dynamic masking, we measure latency and flicker as we vary the length of the source extension (k) and the number of source extensions (n) e consider the 4 different extension strategies described at the end of Section 3. The results for both en→de and en→zh are shown in Figure 5, where we compare to the strategy of using a fixed mask-k. The oracle data point is where we use the full-sentence translations to set the mask so as to completely avoid flicker.

We observe from Figure 5 that our dynamic mask mechanism improves over the fixed mask in all cases, by reducing both latency and flicker. Varying the source prediction strategy and parameters appears to preserve the same inverse relation between latency and flicker, although offering a different trade-off. Using several random source predictions (the green curve in both plots) offers the lowest flicker, at the expense of high latency, possibly because the prefix extension translations show a lot of variability. Using the LM for source prediction tends to have the opposite effect, favouring a reduction in latency. The pattern across the two language pairs is similar, although we observe a more dispersed picture for the en→zh results.

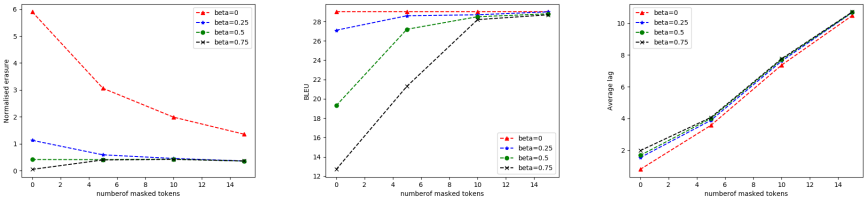
We now test our retranslation strategy on a better performing model, trained on a larger data set. Specifically, we train the model on the entire parallel training data for the WMT20 en-zh task[2], in addition to the TED corpus used above. For the larger model we prepare as before, except with 30k BPE merges separately on each language, and then we train a transformer-regular using Marian. The results are shown in Figure 6. We can verify that the pattern is unchanged in this setting.

To further explore how this dynamic mask strategy improves stability, we look at the English→German corpus and give several examples in Table 1. Here we do not compare with gold translations because we want to focus on how dynamic mask reduces the flicker caused by the MT system, rather than the overall quality of the translation (which is unaffected by dynamic mask). Note that in the second example, the longest common prefix between *MT (Extension)* and *MT* (e.g. empty string) is a prefix of *Previous Output*, thus we simply take the previous translation as the output for current source as described in Section 3.

We can see that in both examples, dynamic masks give more stable translations. Although fixed mask-k strategy can also avoid flicker, it would require a very large global k value to avoid flicker in the second example, and so result in redundant latency in the first example. Noticeably, translations for these two examples share similar length, which indicates that we cannot relax the

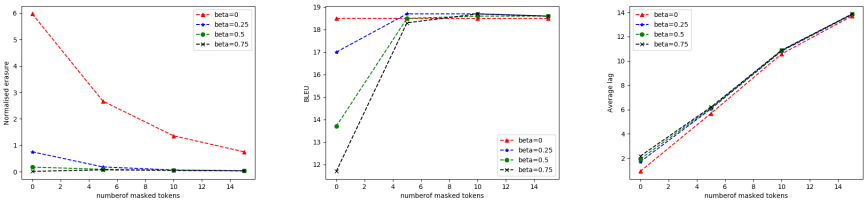
[1] https://github.com/fxsjy/jieba

[2] www.statmt.org/wmt20/translation-task.html



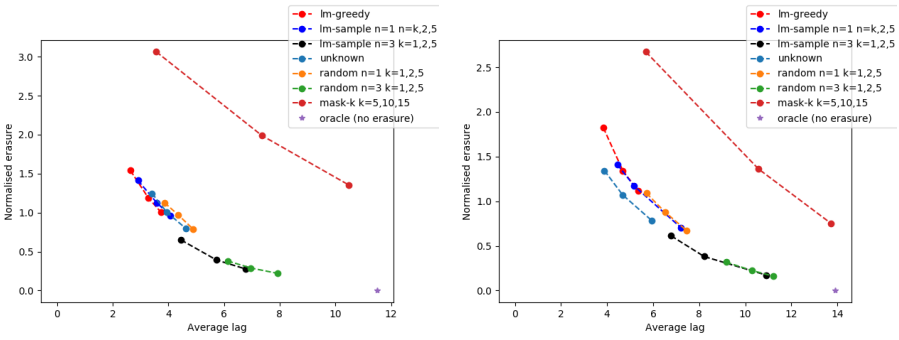
(a) Masking versus flicker (measued by erasure) (b) Masking versus quality (measured by BLEU). (c) Masking versus latency (measured by average lagging).

Figure 3: Effect of varying mask-k, at different values of the biased beam search interpolation parameter, on the three measures proposed in Section 4.



(a) Masking versus flicker (measued by erasure) (b) Masking versus quality (measured by BLEU). (c) Masking versus latency (measured by average lagging).

Figure 4: Effect of varying mask-k, at different values of the biased beam search interpolation parameter, on English-to-Chinese corpus



(a) Dynamic Masking for en→de MT system trained on full sentences (measured by AL and NE) (b) Dynamic Masking for en→zh MT system trained on full sentences (measured by AL and NE)

Figure 5: Effect of dynamic mask with different source prediction strategies. The strategies are explained in Section 3, and the oracle means that we use the full-sentence translation to set the mask.

global mask-k strategy with length ratios of sentences to handle both examples perfectly. Thus, our proposed dynamic mask strategy is more flexible and accurate than mask-k used in (Arivazhagan et al., 2020a).

6 Conclusion

We propose a dynamic mask strategy to improve the stability for re-translation method in Simultaneous Machine Translation. We have shown that combining biased beam search with mask-k works

Source	and I wonder what you'd choose , because I've been asking my friends
Extension (Pred)	and I wonder what you'd choose , because I've been asking my friends to find my own single
MT	Und ich frage mich, was Sie wählen würden, denn ich habe meine Freunde gefragt .
MT (Extension)	Und ich frage mich, was Sie wählen würden, denn ich habe meine Freunde gebeten, meine eigenen Einzelheiten zu finden.
MT (masked)	Und ich frage mich, was Sie wählen würden, denn ich habe meine Freunde
MT (stable)	Und ich frage mich, was Sie wählen würden, denn ich habe meine Freunde diese Frage oft gestellt und sie wollen alle zurück gehen.
Previous Output	Und ich frage mich, was Sie wählen würden, denn ich habe
Source	and , in fact , these kids don't , so they're going out and reading their school work
Extension (Pred)	and , in fact , these kids don't , so they're going out and reading their school work under them .
MT	Tatsächlich tun diese Kinder das nicht, also gehen sie raus und lesen ihre Schularbeit.
MT (Extension)	Und tatsächlich tun diese Kinder das nicht, also gehen sie raus und lesen ihre Schularbeit unter ihnen.
MT (masked)	Und tatsächlich tun diese Kinder das nicht, also gehen sie raus und lesen ihre
MT (stable)	Und tatsächlich tun diese Kinder das nicht, also gehen sie raus und lesen ihre Schularbeit unter den Straßenlampen.
Previous Output	Und tatsächlich tun diese Kinder das nicht, also gehen sie raus und lesen ihre

Table 1: Examples from the English→German test set. *Source* row denotes English prefix. *MT* row denotes German translation directly from our MT system. *Extension (Pred)* denotes the extended prefix predicted with our proposed strategies in Section 3. Here we use lm-sample strategy with $k = 5$. *MT (extension)* denotes the translation for this predicted source extension by our MT system. *MT (masked)* denotes the final output by our MT system after dynamic masks and *MT (stable)* denotes the translation by our MT system for the corresponding full sentence, which we regard as the stable translation. Finally, *Previous Output* is the masked output of the MT system from the previous prefix. We use blue color to denote source and red to denote tokens to be erased.

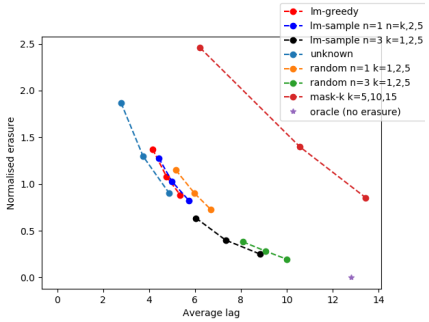


Figure 6: Effect of dynamic mask with different strategies

well in re-translation systems, but biased beam search hurts the quality and additional mask-k used to reduce this effect gives a high latency. Instead, dynamic mask strategy maintains the translation quality but gives a much better trade-off between latency and flicker than mask-k. Our experiments also show that the effect of this strategy depends on both the length and number of predicted extensions, but the quality of predicted extensions is less important.

For future research, we would like to combine biased beam search with dynamic mask to see if it can give a better trade-off between quality and latency than (Arivazhagan et al., 2020a) when the flicker is small enough.

Acknowledgments

The acknowledgments should go immediately before the references. Do not number the acknowledgments section. Do not include this section when submitting your paper for review.

References

Naveen Arivazhagan, Colin Cherry, Te I, Wolfgang Macherey, Pallavi Baljekar, and George Foster. 2020a. Re-Translation Strategies For Long Form, Simultaneous, Spoken Language Translation. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic Infinite Lookback Attention for Simultaneous Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George Foster. 2020b. Re-translation versus streaming for simultaneous translation.

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 437–445, Montréal, Canada. Association for Computational Linguistics.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Kumiko Sudoh, Kyotaro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 Evaluation Campaign. In *Proceedings of IWSLT*.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *CoRR*, abs/1606.02012.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-latency neural speech translation. In *Proceedings of Interspeech*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238, Atlanta, Georgia. Association for Computational Linguistics.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nădejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

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.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1349–1354, Hong Kong, China. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019b. Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5816–5822, Florence, Italy. Association for Computational Linguistics.

B ELITR Non-Native Speech Translation at IWSLT 2020

ELITR Non-Native Speech Translation at IWSLT 2020

Dominik Macháček[†] and **Jonáš Kratochvíl**[†] and **Sangeet Sagar**[†] and **Matúš Žilinec**[†] and **Ondřej Bojar**[†] and **Thai-Son Nguyen**[‡] and **Felix Schneider**[‡] and **Philip Williams**[*] and **Yuekun Yao**[*]
[†]Charles University, [‡]Karlsruhe Institute of Technology, [*]University of Edinburgh
[†]{surname}@ufal.mff.cuni.cz, except jkratochvil@ufal.mff.cuni.cz,
[‡]{firstname.lastname}@kit.edu,
[*]pwillia4@inf.ed.ac.uk,yyao2@exseed.ed.ac.uk

Abstract

This paper is an ELITR system submission for the non-native speech translation task at IWSLT 2020. We describe systems for offline ASR, real-time ASR, and our cascaded approach to offline SLT and real-time SLT. We select our primary candidates from a pool of pre-existing systems, develop a new end-to-end general ASR system, and a hybrid ASR trained on non-native speech. The provided small validation set prevents us from carrying out a complex validation, but we submit all the unselected candidates for contrastive evaluation on the test set.

1 Introduction

This paper describes the submission of the EU project ELITR (European Live Translator)[1] to the non-native speech translation task at IWSLT 2020 (Ansari et al., 2020). It is a result of a collaboration of project partners Charles University (CUNI), Karlsruhe Institute of Technology (KIT), and University of Edinburgh (UEDIN), relying on the infrastructure provided to the project by PerVoice company.

The non-native speech translation shared task at IWSLT 2020 complements other IWSLT tasks by new challenges. Source speech is non-native English. It is spontaneous, sometimes disfluent, and some of the recordings come from a particularly noisy environment. The speakers often have a significant non-native accent. In-domain training data are not available. They consist only of native out-domain speech and non-spoken parallel corpora. The validation data are limited to 6 manually transcribed documents, from which only 4 have reference translations. The target languages are Czech and German.

The task objectives are quality and simultaneity, unlike the previous tasks, which focused only on the quality. Despite the complexity, the resulting systems can be potentially appreciated by many users attending an event in a language they do not speak or having difficulties understanding due to unfamiliar non-native accents or unusual vocabulary.

We build on our experience from the past IWSLT and WMT tasks, see e.g. Pham et al. (2019); Nguyen et al. (2017); Pham et al. (2017); Wetesko et al. (2019); Bawden et al. (2019); Popel et al. (2019). Each of the participating institutions has offered independent ASR and MT systems trained for various purposes and previous shared tasks. We also create some new systems for this task and deployment for the purposes of the ELITR project. Our short-term motivation for this work is to connect the existing systems into a working cascade for SLT and evaluate it empirically, end-to-end. In the long-term, we want to advance state of the art in non-native speech translation.

2 Overview of Our Submissions

This paper is a joint report for two primary submissions, for online and offline sub-track of the non-native simultaneous speech translation task.

First, we collected all ASR systems that were available for us (Section 3.1) and evaluated them on the validation set (Section 3.2). We selected the best candidate for offline ASR to serve as the source for offline SLT. Then, from the ASR systems, which are usable in online mode, we selected the best candidate for online ASR and as a source for online SLT.

In the next step (Section 4), we punctuated and truecased the online ASR outputs of the validation set, segmented them to individual sentences, and translated them by all the MT systems we had available (Section 5.1). We integrated the online ASRs and MTs into our platform for online SLT

[1] http://elitr.eu

Proceedings of the 17th International Conference on Spoken Language Translation (IWSLT), pages 200–208

https://doi.org/10.18653/v1/P17

(Sections 5.2 and 5.3). We compared them using automatic MT quality measures and by simple human decision, to compensate for the very limited and thus unreliable validation set (Section 5.4). We selected the best candidate systems for each target language, for Czech and German.

Both best candidate MT systems are very fast (see Section 5.5). Therefore, we use them both for the online SLT, where the low translation time is critical, and for offline SLT.

In addition to the primary submissions, we included all the other candidate systems and some public services as contrastive submissions.

3 Automatic Speech Recognition

This section describes our automatic speech recognition systems and their selection.

3.1 ASR Systems

We use three groups of ASR systems. They are described in the following sections.

3.1.1 KIT ASR

KIT has provided three hybrid HMM/ANN ASR systems and an end-to-end sequence-to-sequence ASR system.

The hybrid systems, called KIT-h-large-lm1, KIT-h-large-lm2 and KIT-hybrid, were developed to run on the online low-latency condition, and differ in the use of the language models.

The KIT-h-large-lm adopted a 4-gram language model which was trained on a large text corpus (Nguyen et al., 2017), while the KIT-hybrid employed only the manual transcripts of the speech training data. We would refer the readers to the system paper by Nguyen et al. (2017) for more information on the training data and the studies by Nguyen et al. (2020); Niehues et al. (2018) for more information about the online setup.

The end-to-end ASR, so-called KIT-seq2seq, followed the architecture and the optimizations described by Nguyen et al. (2019). It was trained on a large speech corpus, which is the combination of Switchboard, Fisher, LibriSpeech, TED-LIUM, and Mozilla Common Voice datasets. It was used solely without an external language model.

All KIT ASR systems are unconstrained because they use more training data than allowed for the task.

3.1.2 Kaldi ASR Systems

We used three systems trained in the Kaldi ASR toolkit (Povey et al., 2011). These systems were trained on Mozilla Common Voice, TED-LIUM, and AMI datasets together with additional textual data for language modeling.

Kaldi-Mozilla For Kaldi-Mozilla, we used the Mozilla Common Voice baseline Kaldi recipe.[2] The training data consist of 260 hours of audio. The number of unique words in the lexicon is 7996, and the number of sentences used for the baseline language model is 6994, i.e., the corpus is very repetitive. We first train the GMM-HMM part of the model, where the final number of hidden states for the HMM is 2500, and the number of GMM components is 15000. We then train the chain model, which uses the Time delay neural network (TDNN) architecture (Peddinti et al., 2015) together with the Batch normalization regularization and ReLU activation. We use MFCC features to represent audio frames, and we concatenate them with the 100-dimensional I-vector features for the neural network training. We recompile the final chain model with CMU lexicon to increase the model capacity to 127384 words and 4-gram language model trained with SRILM (Stolcke, 2002) on 18M sentences taken from English news articles.

Kaldi-TedLium serves as another baseline, trained on 130 hours of TED-LIUM data (Rousseau et al., 2012) collected before the year 2012. The Kaldi-TedLium model was developed by the University of Edinburgh and was fully described by Klejch et al. (2019). This model was primarily developed for discriminative acoustic adaptation to domains distinct from the original training domain. It is achieved by reusing the decoded lattices from the first decoding pass and by finetuning for TED-LIUM development and test set. The setup follows the Kaldi 1f TED-LIUM recipe. The architecture is similar to Kaldi-Mozilla and uses a combination of TDNN layers with batch normalization and ReLU activation. The input features are MFCC and I-vectors.

Kaldi-AMI was trained on the 100 hours of the AMI data, which comprise of staged meeting recordings (Mccowan et al., 2005). These data

[2] https://github.com/kaldi-asr/kaldi/tree/master/egs/commonvoice/s5

domain	AMI				Antrecorp		Auditing
document	AMIa	AMIb	AMIc	AMId	Teddy	Autocentrum	Auditing
KIT-h-large-lm1	50.71	47.96	53.11	50.43	65.92	19.25	18.54
KIT-h-large-lm2	47.82	41.71	42.10	45.77	75.87	28.59	19.81
KIT-hybrid	40.72	38.45	41.09	43.28	58.99	21.04	21.44
KIT-seq2seq	33.73	28.54	34.45	42.24	42.57	9.91	10.45
Kaldi-TedLium	42.44	38.56	41.83	44.36	61.12	18.68	22.81
Kaldi-Mozilla	52.89	56.37	58.50	58.90	68.72	45.41	34.36
Kaldi-AMI	~~28.01~~	~~23.04~~	~~26.87~~	~~29.34~~	59.66	20.62	28.39
Microsoft	53.72	52.62	56.67	58.58	87.82	39.64	24.22
Google	51.52	49.47	53.11	56.88	61.01	14.12	17.47

Table 1: WER rates of individual documents in the development set. Kaldi-AMI scores on AMI domain are striked through because they are unreliable due to an overlap with the training data.

domain	document	sents.	tokens	duration	references
Antrecorp	Teddy	11	171	1:15	2
Antrecorp	Autocentrum	12	174	1:06	2
Auditing	Auditing	25	528	5:38	1
AMI	AMIa	220	1788	15:09	1
AMI	AMIb	614	4868	35:17	0
AMI	AMIc	401	3454	24:06	0
AMI	AMId	281	1614	13:01	0

Table 2: The size of the development set `iwslt2020-nonnative-minidevset-v2`. The duration is in minutes and seconds. As "references" we mean the number of independent referential translations into Czech and German.

	WER weighted average			avg
	AMI	Antrecorp	Auditing	domain
KIT-seq2seq[1]	**32.96**	**26.10**	**10.45**	**23.17**
Kaldi-TedLium	40.91	39.72	22.81	34.48
Kaldi-Mozilla	56.82	56.96	34.36	49.38
Kaldi-AMI	~~25.79~~	39.97	28.39	31.38
Microsoft	54.80	63.52	24.22	47.51
Google	51.88	37.36	17.47	35.57
KIT-h-large-lm1	50.24	42.38	**18.54**[2]	37.05
KIT-h-large-lm2	43.32	52.02	19.81	38.38
KIT-hybrid	**40.24**[1]	**39.85**[1]	21.44	**33.84**

Table 3: Weighted average WER for the domains in validation set, and their average. The top line-separated group are offline ASR systems, the bottom are online. Bold numbers are the lowest considerable WER in the group. Kaldi-AMI score on AMI is not considered due to overlap with training data. Bold names are the primary (marked with [1]) and secondary (marked with [2]) candidates.

were recorded mostly by non-native English speakers with a different microphone and acoustic environment conditions. The model setup used follows the Kaldi 1i ami recipe. Kaldi-AMI cannot be reliably assessed on the AMI part of the development due to the overlap of training and development data. We have decided not to exclude this overlap so that we do not limit the amount of available training data for our model.

3.1.3 Public ASR Services

As part of our baseline models, we have used Google Cloud Speech-to-Text API[3] and Microsoft Azure Speech to Text.[4] Both of these services provide an API for transcription of audio files in WAV format, and they use neural network acoustic models. We kept the default settings of these systems.

The Google Cloud system supports over 100 languages and several types of English dialects (such as Canada, Ireland, Ghana, or the United Kingdom). For decoding of the development and test set, we have used the United Kingdom English dialect option. The system can be run either in real-time or offline mode. We have used the offline option for this experiment.

The Microsoft Azure Bing Speech API supports fewer languages than Google Cloud ASR but adds more customization options of the final model. It can be also run both in real-time or offline mode. For the evaluation, we have used the offline mode and the United Kingdom English (en-GB) dialect.

3.2 Selection of ASR Candidates

We processed the validation set with all the ASR systems, evaluated WER, and summarized them in Table 1. The validation set (Table 2) contains three different domains with various document sizes, and the distribution does not fully correspond to the test set. The AMI domain is not present in the test set at all, but it is a part of Kaldi-AMI training data. Therefore, a simple selection by an average WER on the whole validation set could favor the systems which perform well on the AMI domain, but they

[3] https://cloud.google.com/speech-to-text

[4] https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/

could not be good candidates for the other domains.

In Table 3, we present the weighted average of WER in the validation domains. We weight it by the number of gold transcription words in each of the documents. We observe that Kaldi-AMI has a good performance on the AMI domain, but it is worse on the others. We assume it is overfitted for this domain, and therefore we do not use it as the primary system.

For offline ASR, we use KIT-seq2seq as the primary system because it showed the lowest error rate on the averaged domain.

The online ASR systems can exhibit somewhat lower performance than offline systems. We select KIT-h-large-lm1 as the primary online ASR candidate for Auditing, and KIT-hybrid as primary for the other domains.

Our second primary offline ASR is Kaldi-AMI.

4 Punctuation and Segmentation

All our ASR systems output unpunctuated, often all lowercased text. The MT systems are designed mostly for individual sentences with proper casing and punctuation. To overcome this, we first insert punctuation and casing to the ASR output. Then, we split it into individual sentences by the punctuation marks by a rule-based language-dependent Moses sentence splitter (Koehn et al., 2007).

Depending on the ASR system, we use one of two possible punctuators. Both of them are usable in online mode.

4.1 KIT Punctuator

The KIT ASR systems use an NMT-based model to insert punctuation and capitalization in an otherwise unsegmented lowercase input stream (Cho et al., 2012, 2015). The system is a monolingual translation system that translates from raw ASR output to well-formed text by converting words to upper case, inserting punctuation marks, and dropping words that belong to disfluency phenomena. It does not use the typical sequence-to-sequence approach of machine translation. However, it considers a sliding window of recent (uncased) words and classifying each one according to the punctuation that should be inserted and whether the word should be dropped for being a part of disfluency. This gives the system a constant input and output size, removing the need for a sequence-to-sequence model.

While inserting punctuation is strictly necessary for MT to function at all, inserting capitalization and removing disfluencies improves MT performance by making the test case more similar to the MT training conditions (Cho et al., 2017).

4.2 BiRNN Punctuator

For other systems, we use a bidirectional recurrent neural network with an attention-based mechanism by Tilk and Alumäe (2016) to restore punctuation in the raw stream of ASR output. The model was trained on 4M English sentences from CzEng 1.6 (Bojar et al., 2016) data and a vocabulary of 100K most frequently occurring words. We use CzEng because it is a mixture of domains, both originally spoken, which is close to the target domain, and written, which has richer vocabulary, and both original English texts and translations, which we also expect in the target domain. The punctuated transcript is then capitalized using an English tri-gram truecaser by Lita et al. (2003). The truecaser was trained on 2M English sentences from CzEng.

5 Machine Translation

This section describes the translation part of SLT.

5.1 MT Systems

See Table 4 for the summary of the MT systems. All except de-LSTM are Transformer-based neural models using Marian (Junczys-Dowmunt et al., 2018) or Tensor2Tensor (Vaswani et al., 2018) back-end. All of them, except de-T2T, are unconstrained because they are trained not only on the data sets allowed in the task description, but all the used data are publicly available.

5.1.1 WMT Models

WMT19 Marian and WMT18 T2T models are Marian and T2T single-sentence models from Popel et al. (2019) and Popel (2018). WMT18 T2T was originally trained for the English-Czech WMT18 news translation task, and reused in WMT19. WMT19 Marian is its reimplementation in Marian for WMT19. The T2T model has a slightly higher quality on the news text domain than the Marian model. The Marian model translates faster, as we show in Section 5.5.

5.1.2 IWSLT19 Model

The IWSLT19 system is an ensemble of two English-to-Czech Transformer Big models trained using the Marian toolkit. The models were originally trained on WMT19 data and then finetuned

system	back-end	source-target	constrained	reference
WMT19 Marian	Marian	en→cs	no	Popel et al. (2019), Section 5.1.1
WMT18 T2T	T2T	en→cs	no	Popel et al. (2019), Section 5.1.1
IWSLT19	Marian	en→cs	no	Wetesko et al. (2019), Section 5.1.2
OPUS-A	Marian	en↔{cs,de+5 l.}	no	Section 5.1.3
OPUS-B	Marian	en↔{cs,de+39 l.}	no	Section 5.1.3
T2T-multi	T2T	en↔{cs,de,en+39 l.}	no	Section 5.1.4
T2T-multi-big	T2T	en↔{cs,de,en+39 l.}	no	Section 5.1.4
de-LSTM	NMTGMinor	en→de	no	Dessloch et al. (2018), Section 5.1.6
de-T2T	T2T	en→de	yes	Section 5.1.5

Table 4: The summary of our MT systems.

on MuST-C TED data. The ensemble was a component of Edinburgh and Samsung's submission to the IWSLT19 Text Translation task. See Section 4 of Wetesko et al. (2019) for further details of the system.

5.1.3 OPUS Multi-Lingual Models

The OPUS multilingual systems are one-to-many systems developed within the ELITR project. Both were trained on data randomly sampled from the OPUS collection (Tiedemann, 2012), although they use distinct datasets. OPUS-A is a Transformer Base model trained on 1M sentence pairs each for 7 European target languages: Czech, Dutch, French, German, Hungarian, Polish, and Romanian. OPUS-B is a Transformer Big model trained on a total of 231M sentence pairs covering 41 target languages that are of particular interest to the project[5] After initial training, OPUS-B was finetuned on an augmented version of the dataset that includes partial sentence pairs, artificially generated by truncating the original sentence pairs (similar to Niehues et al., 2018). We produce up to 10 truncated sentence pairs for every one original pair.

5.1.4 T2T Multi-Lingual Models

T2T-multi and T2T-multi-big are respectively Transformer and Transformer Big models trained on a Cloud TPU based on the default T2T hyper-parameters, with the addition of target language tokens as in Johnson et al. (2017). The models were trained with a shared vocabulary on a dataset of English-to-many and many-to-English sentence pairs from OPUS-B containing 42 languages in total, making them suitable for pivoting. The models do not use finetuning.

[5]The 41 target languages include all EU languages (other than English) and 18 languages that are official languages of EUROSAI member countries. Specifically, these are Albanian, Arabic, Armenian, Azerbaijani, Belorussian, Bosnian, Georgian, Hebrew, Icelandic, Kazakh, Luxembourgish, Macedonian, Montenegrin, Norwegian, Russian, Serbian, Turkish, and Ukrainian.

5.1.5 de-T2T

de-T2T translation model is based on a Tensor2Tensor translation model model using training hyper-parameters similar to Popel and Bojar (2018). The model is trained using all the parallel corpora provided for the English-German WMT19 News Translation Task, without back-translation. We use the last training checkpoint during model inference. To reduce the decoding time, we apply greedy decoding instead of a beam search.

5.1.6 KIT Model

KIT's translation model is based on an LSTM encoder-decoder framework with attention (Pham et al., 2017). As it is developed for our lecture translation framework (Müller et al., 2016), it is finetuned for lecture content. In order to optimize for a low-latency translation task, the model is also trained on partial sentences in order to provide more stable translations (Niehues et al., 2016).

5.2 ELITR SLT Platform

We use a server called Mediator for the integration of independent ASR and MT systems into a cascade for online SLT. It is a part of the ELITR platform for simultaneous multilingual speech translation (Franceschini et al., 2020). The workers, which can generally be any audio-to-text or text-to-text processors, such as ASR and MT systems, run inside of their specific software and hardware environments located physically in their home labs around Europe. They connect to Mediator and offer a service. A client, often located in another lab, requests Mediator for a cascade of services, and Mediator connects them. This platform simplifies the cross-institutional collaboration when one institution offers ASR, the other MT, and the third tests them as a client. The platform enables using the SLT pipeline easily in real-time.

5.3 MT Wrapper

The simultaneous ASR incrementally produces the recognition hypotheses and gradually improves them. The machine translation system translates one batch of segments from the ASR output at a time. If the translation is not instant, then some ASR hypotheses may be outdated during the translation and can be skipped. We use a program called MT Wrapper for connecting the output of self-updating ASR with non-instant NMT systems.

MT Wrapper has two threads. The receiving thread segments the input for our MTs into individual sentences, saves the input into a buffer, and continuously updates it. The translating thread is a loop that retrieves the new content from the buffer. If a segment has been translated earlier in the current process, it is outputted immediately. Otherwise, the new segments are sent in one batch to the NMT system, stored to a cache and outputted.

For reproducibility, the translation cache is empty at the beginning of a process, but in theory it could be populated by a translation memory. The cache significantly reduces the latency because the punctuator often oscillates between two variants of casing or punctuation marks within a short time.

MT Wrapper has a parameter to control the stability and latency. It can mask the last k words of incomplete sentences from the ASR output, as in Ma et al. (2019) and Arivazhagan et al. (2019), considering only the currently completed sentences, or only the "stable" sentences, which are beyond the ASR and punctuator processing window and never change. We do not tune these parameters in the validation. We do not mask any words or segments in our primary submission, but we submit multiple non-primary systems differing in these parameters.

5.4 Quality Validation

For comparing the MT candidates for SLT, we processed the validation set by three online ASR systems, translated them by the candidates, aligned them with reference by mwerSegmenter (Matusov et al., 2005) and evaluated the BLEU score (Post, 2018; Papineni et al., 2002) of the individual documents. However, we were aware that the size of the validation set is extremely limited (see Table 2) and that the automatic metrics as the BLEU score estimate the human judgment of the MT quality reliably only if there is a sufficient number of sentences or references. It is not the case of this validation set.

Therefore, we examined them by a simple comparison with source and reference. We realized that the high BLEU score in the Autocentrum document is induced by the fact that one of the translated sentences matches exactly matches a reference because it is a single word "thanks". This sentence increases the average score of the whole document, although the rest is unusable due to mistranslated words. The ASR quality of the two Antrecorp documents is very low, and the documents are short. Therefore we decided to omit them in comparison of the MT candidates.

We examined the differences between the candidate translations on the Auditing document, and we have not seen significant differences, because this document is very short. The AMIa document is longer, but it contains long pauses and many isolated single-word sentences, which are challenging for ASR. The part with a coherent speech is very short.

Finally, we selected the MT candidate, which showed the highest average BLEU score on the three KIT online ASR systems both on Auditing and AMIa document because we believe that averaging the three ASR sources shows robustness against ASR imperfections. See Table 5 and Table 6 for the BLEU scores on Czech and German. The selected candidates are IWSLT19 for Czech and OPUS-B for German. However, we also submit all other candidates as non-primary systems to test them on a significantly larger test set. We use these candidates both for online and offline SLT.

5.5 Translation Time

We measured the average time, in which the MT systems process a batch of segments of the validation set (Table 7). If the ASR updates are distributed uniformly in time, than the average batch translation time is also the expected delay of machine translation. The shortest delay is almost zero; in cases when the translation is cached or for very short segments. The longest delay happens when an ASR update arrives while the machine is busy with processing the previous batch. The delay is time for translating two subsequent batches, waiting and translating.

We suppose that the translation time of our primary candidates is sufficient for real-time translation, as we verified in on online SLT test sessions.

We observe differences between the MT systems.

MT	document	gold	KIT-hybrid	KIT-h-large-lm1	KIT-h-large-lm2	avg KIT
OPUS-B	Teddy	42.8463	2.418	2.697	1.360	2.158
IWSLT19	Teddy	51.397	1.379	2.451	1.679	1.836
WMT19 Marian	Teddy	49.328	1.831	1.271	1.649	1.584
WMT18 T2T	Teddy	54.778	1.881	1.197	1.051	1.376
OPUS-A	Teddy	25.197	1.394	1.117	1.070	1.194
T2T-multi	Teddy	36.759	1.775	0.876	0.561	1.071
WMT18 T2T	Autocentrum	42.520	12.134	13.220	14.249	13.201
WMT19 Marian	Autocentrum	39.885	10.899	10.695	12.475	11.356
OPUS-B	Autocentrum	29.690	12.050	10.873	9.818	10.914
IWSLT19	Autocentrum	37.217	9.901	8.996	8.900	9.266
OPUS-A	Autocentrum	30.552	9.201	9.277	8.483	8.987
T2T-multi	Autocentrum	20.011	6.221	2.701	3.812	4.245
IWSLT19	AMIa	**22.878**	5.377	2.531	3.480	**3.796**
WMT18 T2T	AMIa	21.091	5.487	2.286	3.411	3.728
WMT19 Marian	AMIa	22.036	4.646	2.780	3.739	3.722
OPUS-B	AMIa	19.224	4.382	3.424	2.672	3.493
OPUS-A	AMIa	15.432	3.131	2.431	2.500	2.687
T2T-multi	AMIa	13.340	2.546	2.061	1.847	2.151
IWSLT19	Auditing	**9.231**	1.096	3.861	2.656	**2.538**
OPUS-B	Auditing	6.449	1.282	3.607	2.274	2.388
OPUS-A	Auditing	8.032	1.930	4.079	0.900	2.303
WMT19 Marian	Auditing	8.537	1.087	3.571	1.417	2.025
WMT18 T2T	Auditing	9.033	1.201	2.935	1.576	1.904
T2T-multi	Auditing	3.923	1.039	1.318	1.110	1.156

Table 5: Validation BLEU scores in percents (range 0-100) for SLT into Czech from ASR sources. The column "gold" is translation from the gold transript. It shows the differences between MT systems, but was not used in validation.

The size and the model type of WMT19 Marian and WMT18 T2T are the same (see Popel et al., 2019), but they differ in implementation.

WMT19 Marian is slightly faster than IWSLT19 model because the latter is an ensemble of two models. OPUS-B is slower than OPUS-A because the former is bigger. Both are slower than WMT19 Marian due to multi-targeting and different preprocessing. WMT19 Marian uses embedded SentencePiece (Kudo and Richardson, 2018), while the multi-target models use an external Python process for BPE (Sennrich et al., 2016). The timing may be affected also by different hardware.

At the validation time, T2T-multi and T2T-multi-big used suboptimal setup.

6 Conclusion

We presented ELITR submission for non-native SLT at IWSLT 2020. We observe a significant qualitative difference between the end-to-end offline ASR methods and hybrid online methods. The component that constrains the offline SLT from real-time processing is the ASR, not the MT.

We selected the best candidates from a pool of pre-existing and newly developed components, and submitted our primary submissions, although the size of the development set limits us from a reliable validation. Therefore, we submitted all our unselected candidates for contrastive evaluation on the test set. For the results, we refer to Ansari et al. (2020).

Acknowledgments

The research was partially supported by the grants 19-26934X (NEUREM3) of the Czech Science Foundation, H2020-ICT-2018-2-825460 (ELITR) of the EU, 398120 of the Grant Agency of Charles University, and by SVV project number 260 575.

References

Ebrahim Ansari, Amittai Axelroad, Nguyen Bach, Ondrej Bojar, Roldano Cattoni, Fahim Dalvi, Nadir Durrani, Marcello Federico, Christian Federmann, Jiatao Gu, Fei Huang, Kevin Knight, Xutai Ma, Ajay Nagesh, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Xing Shi, Sebastian Stüker, Marco Turchi, and Changhan Wang. 2020. Findings of the IWSLT 2020 Evaluation Campaign. In *Proceedings of the 17th International Conference on Spoken Language Translation (IWSLT 2020)*, Seattle, USA.

Naveen Arivazhagan, Colin Cherry, Isabelle Te, Wolfgang Macherey, Pallavi Baljekar, and George Foster. 2019. Re-translation strategies for long form, simultaneous, spoken language translation. *ArXiv*, abs/1912.03393.

MT	document	gold	KIT-hybrid	KIT-h-large-lm1	KIT-h-large-lm2	avg KIT
de-T2T	Teddy	45.578	2.847	3.181	1.411	2.480
OPUS-A	Teddy	29.868	1.873	1.664	1.139	1.559
de-LSTM	Teddy	3.133	2.368	2.089	1.254	1.904
OPUS-B	Teddy	41.547	2.352	1.878	1.454	1.895
T2T-multi	Teddy	31.939	1.792	3.497	1.661	2.317
de-T2T	Autocentrum	36.564	9.031	6.229	3.167	6.142
OPUS-A	Autocentrum	26.647	8.898	13.004	2.324	8.075
de-LSTM	Autocentrum	19.573	10.395	13.026	2.322	8.581
OPUS-B	Autocentrum	28.841	10.153	12.134	9.060	10.449
T2T-multi	Autocentrum	22.631	8.327	8.708	6.651	7.895
de-T2T	AMIa	34.958	8.048	5.654	7.467	7.056
OPUS-A	AMIa	30.203	7.653	5.705	5.899	6.419
de-LSTM	AMIa	31.762	7.635	6.642	1.843	5.373
OPUS-B	AMIa	38.315	8.960	7.613	6.837	7.803
T2T-multi	AMIa	28.279	6.202	3.382	3.869	4.484
de-T2T	Auditing	38.973	11.589	17.377	18.841	15.936
OPUS-A	Auditing	38.866	10.355	19.414	18.540	16.103
de-LSTM	Auditing	21.780	10.590	12.633	11.098	11.440
OPUS-B	Auditing	38.173	10.523	18.237	17.644	15.468
T2T-multi	Auditing	22.442	7.896	8.664	11.269	9.276

Table 6: Validation BLEU scores in percents (range 0-100) for MT translations into German from ASR outputs and from the gold transcript.

MT	**avg ± std dev**
T2T-multi	2876.52 ±1804.63
T2T-multi-big	5531.30 ±3256.81
IWSLT19	275.51 ± 119.44
WMT19 Marian	184.08 ± 89.17
WMT18 T2T	421.11 ± 201.64
OPUS-B	287.52 ± 141.28
OPUS-A	263.31 ± 124.75

Table 7: Average and standard deviation time for translating one batch in validation set, in milliseconds. Bold are the candidate systems for online SLT.

Rachel Bawden, Nikolay Bogoychev, Ulrich Germann, Roman Grundkiewicz, Faheem Kirefu, Antonio Valerio Miceli Barone, and Alexandra Birch. 2019. The university of edinburgh's submissions to the wmt19 news translation task. In *WMT*.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. 2016. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In *TSD*.

Eunah Cho, Jan Niehues, Kevin Kilgour, and Alex Waibel. 2015. Punctuation insertion for real-time spoken language translation. In *IWSLT*.

Eunah Cho, Jan Niehues, and Alex Waibel. 2012. Segmentation and punctuation prediction in speech language translation using a monolingual translation system. In *IWSLT*.

Eunah Cho, Jan Niehues, and Alex Waibel. 2017. Nmt-based segmentation and punctuation insertion for real-time spoken language translation. In *INTERSPEECH*.

Florian Dessloch, Thanh-Le Ha, Markus Müller, Jan Niehues, Thai-Son Nguyen, Ngoc-Quan Pham, Elizabeth Salesky, Matthias Sperber, Sebastian Stüker, Thomas Zenkel, and Alexander Waibel. 2018. KIT lecture translator: Multilingual speech translation with one-shot learning. In *COLING: System Demonstrations*.

Dario Franceschini et al. 2020. Removing european language barriers with innovative machine translation technology. In *LREC IWLTP*. In print.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Marcin Junczys-Dowmunt et al. 2018. Marian: Fast neural machine translation in C++. In *ACL System Demonstrations*.

Ondrej Klejch, Joachim Fainberg, Peter Bell, and Steve Renals. 2019. Lattice-based unsupervised test-time adaptation of neural networkacoustic models. *CoRR*, abs/1906.11521.

Philipp Koehn et al. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL Interactive Poster and Demonstration Sessions*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP: System Demonstrations*.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. TRuEcasIng. In *ACL*.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *ACL*.

Evgeny Matusov, Gregor Leusch, Oliver Bender, and Hermann Ney. 2005. Evaluating machine translation output with automatic sentence segmentation. In *International Workshop on Spoken Language Translation*, pages 148–154, Pittsburgh, PA, USA.

Iain Mccowan, J Carletta, Wessel Kraaij, Simone Ashby, S Bourban, M Flynn, M Guillemot, Thomas Hain, J Kadlec, Vasilis Karaiskos, M Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska Masson, Wilfried Post, Dennis Reidsma, and P Wellner. 2005. The ami meeting corpus. *Int'l. Conf. on Methods and Techniques in Behavioral Research*.

Markus Müller, Thai Son Nguyen, Jan Niehues, Eunah Cho, Bastian Krüger, Thanh-Le Ha, Kevin Kilgour, Matthias Sperber, Mohammed Mediani, Sebastian Stüker, et al. 2016. Lecture translator-speech translation framework for simultaneous lecture translation. In *NAACL: Demonstrations*.

Thai-Son Nguyen, Markus Müller, Sebastian Sperber, Thomas Zenkel, Sebastian Stüker, and Alex Waibel. 2017. The 2017 KIT IWSLT Speech-to-Text Systems for English and German. In *IWSLT*.

Thai Son Nguyen, Jan Niehues, Eunah Cho, Thanh-Le Ha, Kevin Kilgour, Markus Muller, Matthias Sperber, Sebastian Stueker, and Alex Waibel. 2020. Low latency asr for simultaneous speech translation.

Thai-Son Nguyen, Sebastian Stueker, Jan Niehues, and Alex Waibel. 2019. Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation. *arXiv preprint arXiv:1910.13296*.

Jan Niehues, Thai Son Nguyen, Eunah Cho, Thanh-Le Ha, Kevin Kilgour, Markus Müller, Matthias Sperber, Sebastian Stüker, and Alex Waibel. 2016. Dynamic transcription for low-latency speech translation. In *Interspeech*.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-latency neural speech translation. In *Proc. Interspeech 2018*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH*.

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

Ngoc-Quan Pham, Matthias Sperber, Elizabeth Salesky, Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2017. Kit's multilingual neural machine translation systems for iwslt 2017. In *The International Workshop on Spoken Language Translation (IWSLT)*, Tokyo, Japan.

Martin Popel. 2018. CUNI transformer neural MT system for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 482–487, Belgium, Brussels. Association for Computational Linguistics.

Martin Popel and Ondrej Bojar. 2018. Training Tips for the Transformer Model. *PBML*, 110.

Martin Popel, Dominik Macháček, Michal Auersperger, Ondřej Bojar, and Pavel Pecina. 2019. English-czech systems in wmt19: Document-level transformer. In *WMT*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *WMT*.

Daniel Povey et al. 2011. The kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

Anthony Rousseau, Paul Deléglise, and Yannick Esteve. 2012. Ted-lium: an automatic speech recognition dedicated corpus. In *LREC*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *ArXiv*, abs/1508.07909.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *ICLSP*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *LREC*.

Ottokar Tilk and Tanel Alumäe. 2016. Bidirectional recurrent neural network with attention mechanism for punctuation restoration.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416.

Joanna Wetesko, Marcin Chochowski, Pawel Przybysz, Philip Williams, Roman Grundkiewicz, Rico Sennrich, Barry Haddow, Antonio Valerio Miceli Barone, and Alexandra Birch. 2019. Samsung and University of Edinburgh's System for the IWSLT 2019. In *IWSLT*.

C Non-Native Speech Translation Task Results (selected pages from Findings of the IWSLT 2020 Evaluation Campaign)

FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN

Ebrahim Ansari
Charles U./IASBS

Amittai Axelrod
DiDi Labs

Nguyen Bach
Alibaba

Ondřej Bojar
Charles U.

Roldano Cattoni
FBK

Fahim Dalvi
QCRI

Nadir Durrani
QCRI

Marcello Federico
Amazon AI

Christian Federmann
Microsoft Research

Jiatao Gu
Facebook AI

Fei Huang
Alibaba

Kevin Knight
DiDi Labs

Xutai Ma
JHU/Facebook AI

Ajay Nagesh
DiDi Labs

Matteo Negri
FBK

Jan Niehues
Maastricht U.

Juan Pino
Facebook AI

Elizabeth Salesky
JHU

Xing Shi
DiDi Labs

Sebastian Stüker
KIT

Marco Turchi
FBK

Alex Waibel
CMU/KIT

Changhan Wang
Facebook AI

Abstract

The evaluation campaign of the International Conference on Spoken Language Translation (IWSLT 2020) featured this year six challenge tracks: (i) Simultaneous speech translation, (ii) Video speech translation, (iii) Offline speech translation, (iv) Conversational speech translation, (v) Open domain translation, and (vi) Non-native speech translation. A total of 30 teams participated in at least one of the tracks. This paper introduces each track's goal, data and evaluation metrics, and reports the results of the received submissions.

1 Introduction [Marcello]

The International Conference on Spoken Language Translation (IWSLT) is an annual scientific conference (Akiba et al., 2004; Eck and Hori, 2005; Paul, 2006; Fordyce, 2007; Paul, 2008, 2009; Paul et al., 2010; Federico et al., 2011, 2012; Cettolo et al., 2013, 2014, 2015, 2016, 2017; Niehues et al., 2018, 2019) for the study, development and evaluation of spoken language translation technology, including: speech-to-text, speech-to-speech translation, simultaneous and consecutive translation, speech dubbing, cross-lingual communication including all multimodal, emotional, para-linguistic, and stylistic aspects and their applications in the field. The goal of the conference is to organize evaluations and sessions around challenge areas, and to present scientific work and system descriptions. This paper reports on the evaluation campaign organized by IWSLT 2020, which features six challenge tracks:

- **Simultaneous speech translation**, addressing low latency translation of talks, from English to German, either from a speech file into text, or from a ground-truth transcript into text;
- **Video speech translation**, targeting multimodal speech translation of video clips into text, either from Chinese into English or from English into Russian
- **Offline speech translation**, proposing speech translation of talks from English into German, using either cascade architectures or end-to-end models, able to directly translate source speech into target text;
- **Conversational speech translation**, targeting the translation of highly disfluent conver-

Proceedings of the 17th International Conference on Spoken Language Translation (IWSLT), pages 1–34

https://doi.org/10.18653/v1/P17

language model fusion techniques.

CASIA (Wang et al., 2020b) ensembled many models into their submission. They used the unfiltered data for backtranslation, used a domain classifier based on segment provenance, and also performed knowledge-distillation. They also used 13k parallel sentences from external data; see the "External data" note in Section 6.6.

6.6 Results and Discussion

Appendix A.5 contains the results of the Japanese-to-Chinese and Chinese-to-Japanese open-domain translation tasks. Some comments follow below.

Data filtering was unsurprisingly helpful. We released 4 corpora as part of the shared task. All participants used `existing_parallel` and `webcrawled_parallel_filtered`. Overall, participants filtered out 15%-90% of the data, and system performance increased by around 2-5 BLEU points. The `webcrawled_parallel_unfiltered` corpus was also used successfully, but required even more aggressive filtering. The `webcrawled_unaligned` data was even harder to use, and we were pleased to see some teams rise to the challenge. *Data augmentation* via backtranslation also consistently helped. However, there was interesting variation in how participants selected the data to be translated. *Provenance* information is not common in MT evaluations; we were curious how it would be used. Hagiwara (2020) tried filtering `web_crawled_parallel_filtered` using a provenance indicator, but found it was too aggressive. Wang et al. (2020b) instead trained a domain classifier, and used it at decoding time to reweight the domain-specific translation models in the ensemble.

External data was explicitly allowed, potentially allowing the sharing of external resources that were unknown to us. Hagiwara (2020) improved on their submitted system, in a separate experiment, by gathering 80k external parallel question-answer pairs from HiNative and incorporating them into the training set. Wang et al. (2020b) also improved their system by adding 13k external sentence pairs from hujiangjp. However, this inadvertently included data from one of the websites from which the task's blind test set was drawn, resulting in 383/875 and 421/875 exact matching segments on the Chinese side and Japanese side respectively.

Overall, we are heartened by the participation in this first edition of the open-domain Chinese-Japanese shared task, and encourage participation in the next one.

7 Non-Native Speech Translation

The non-native speech translation task has been added to IWSLT this year. The task focuses on the very frequent setting of non-native spontaneous speech in somewhat noisy conditions, one of the test files even contained speech transmitted through a remote conferencing platform. We were interested in submissions of both types: the standard two-stage pipeline (ASR+MT, denoted "Cascaded") as well as end-to-end ("E2E") solutions.

This first year, we had English as the only source language and Czech and German as the target languages. Participants were allowed to submit just one of the target languages.

The training data sets permitted for "constrained" submissions were agreed upon the training data with the Offline Translation Task (Section 4) so that task participants could reuse their systems in both tasks. Participants were however also allowed to use any other training data, rendering their submissions "unconstrained".

7.1 Challenge

The main evaluation measure is translation quality but we invited participants to report time-stamped outputs if possible, so that we could assess their systems also using metrics related to *simultaneous* speech translation.

In practice, the translation quality is severely limited by the speech recognition quality. Indeed, the nature of our test set recordings is extremely challenging, see below. For that reason, we also asked the participants with cascaded submissions to provide their intermediate ASR outputs (again with exact timing information, if possible) and score it against our golden transcripts.

A further critical complication is the lack of input sound segmentation to sentence-like units. The Offline Speech Translation Task (Section 4) this year allowed the participants to come up either with their own segmentation, or to rely upon the provided sound segments. In the Non-Native task, no sound segmentation was available. In some cases, this could have caused even a computational challenge, because our longest test document is

25:55 long, well beyond the common length of segments in the training corpora. The reference translations in our test set do come in segments and we acknowledge the risk of automatic scores being affected by the (mis-)match of candidate and reference segmentation, see below.

7.1.1 SLT Evaluation Measures

The SLT evaluation measures were calculated by SLTev,[20] a comprehensive tool for evaluation of (on-line) spoken language translation.

SLT Quality (BLEU$_1$ and BLEU$_{mw}$) As said, we primarily focus on *translation quality* and we approximate it with BLEU (Papineni et al., 2002a) for simplicity, despite all the known shortcomings of the metric, e.g. Bojar et al. (2010).

BLEU was designed for text translation with a clear correspondence between source and target segments (sentences) of the text. We have explored multiple ways of aligning the segments produced by the participating SLT systems with the reference segments. For systems reporting timestamps of individual source-language words, the segment-level alignment can be based on the exact timing. Unfortunately, only one system provided this detailed information, so we decided to report only two simpler variants of BLEU-based metrics:

BLEU$_1$ The whole text is concatenated and treated as *one* segment for BLEU. Note that this is rather inappropriate for longer recordings where many n-grams could be matched far from their correct location.

BLEU$_{mw}$ (mwerSegmenter + standard BLEU). For this, first we concatenate the whole document and segment it using the mwerSegmenter tool (Matusov et al., 2005). Then we calculate the BLEU score for each document in the test set and report the average.

Since the BLEU implementations differ in many details, we rely on a stable one, namely sacreBLEU (Post, 2018).[21]

SLT Simultaneity In online speech translation, one can trade translation quality for delay and vice versa. Waiting for more input generally allows the system to produce a better translation. A compromise is sought by systems that quickly produce first candidate outputs and *update* them later, at the cost of potentially increasing cognitive load for the user by showing output that will become irrelevant.

The key properties of this trade-off are captured by observing some form of *delay*, i.e. how long the user has to wait for the translation of the various pieces of the message compared to directly following the source, and *flicker*, i.e. how much "the output changes". We considered several possible definitions of delay and flicker, including or ignoring information on timing, segmentation, word reordering etc., and calculated each of them for each submission. For simplicity, report only the following ones:

Flicker is inspired by Arivazhagan et al. (2019a). We report a normalized revision score calculated by dividing the total number of words produced by the true output length, i.e. by the number of words in the completed sentences. We report the average score across all documents in the test set.

Delay$_{ts}$ relies on timing information provided by the participants for individual *segments*. Each produced word is assumed to have appeared at the time that corresponds proportionally to its (character) position in the segment. The same strategy is used for the reference words. Note that the candidate segmentation does not need to match the reference one, but in both cases, we get an estimated time span for each word.

Delay$_{mw}$ uses mwerSegmenter to first find correspondences between candidate and reference segments based on the actual words. Then the same strategy of estimating the timing of each word is used.

The Delay is summed over all words and divided by the total number of words considered in the calculation to show the average delay per word.

Note that we use a simple exact match of the candidate and reference word; a better strategy would be to use some form of monolingual word alignment which could handle e.g. synonyms. In our case, non-matched words are ignored and do not contribute to the calculation of the delay at all,

[20] https://github.com/ELITR/SLTev

[21] We use the default settings, i.e. the signature `BLEU+case.mixed+numrefs.1+smooth.exp++tok.13a+version.1.4.6`.

Domain	Files	Overall Duration	Segments	EN Words	CS Words	DE Words
Antrecorp	28	0h38m	427	5040	4071	4660
Khan Academy	5	0h18m	346	2886	2272	2660
SAO	6	1h39m	654	11928	9395	10613
Total	39	2h35m	1427	19854	15738	17933

Table 3: Non-Native Speech Translation Task test data composition. Words are estimated simply by splitting at whitespace without tokenization.

reducing the reliability of the estimate. To provide an indication of how reliable the reported Delays are, we list also the percentage of reference words matched, i.e. successfully found in the candidate translation. This percentage ranges from 20% to up to 90% across various submissions.

Note that only one team provided us with timing details. In order to examine the empirical relations between these conflicting measures, we focus on the several contrastive runs submitted by this them in Section 7.4.1.

7.1.2 ASR Evaluation Measures

The ASR-related scores were also calculated by SLTev, using the script ASRev which assumes that the "translation" is just an identity operation.

We decided to calculate WER using two different strategies:

WER$_1$ concatenating all segments into one long sequence of tokens, and

WER$_{mw}$ first concatenating all segments provided by task participants and then using mwerSegmenter to reconstruct the segmentation that best matches the reference.

In both cases, we pre-process both the candidate and reference by lower casing and removing punctuation.

7.2 Data

7.2.1 Training Data for Constrained Submissions

The training data was aligned with the Offline Speech Translation Task (Section 4) to allow cross-submission in English-to-German SLT. English-to-Czech was unique to the Non-Native Task.

The permitted data for constrained submissions were:

For English ASR:

- LibriSpeech ASR corpus (Panayotov et al., 2015),
- Mozilla Common Voice,[22]
- Speech-Translation TED corpus.[23]

For English→Czech Translation:

- MuST-C (Di Gangi et al., 2019a), release 1.1 contains English-Czech pair,
- CzEng 1.7 (Bojar et al., 2016).[24] Note that CzEng overlaps with English-German test data of the Offline Speech Translation Task so it was not allowed to use this English-Czech corpus to train English-German (multi-lingual) systems.

For English→Czech Translation:

- All the data for English-German track by WMT 2019[25] News Translation Task, i.e.:
  - English-German parallel data,
  - German monolingual data,
- MuST-C (Di Gangi et al., 2019a), release 1.0 contains English-German pair,
- Speech-Translation TED corpus,[26] the English-German texts,
- WIT^3 (Cettolo et al., 2012).

[22] https://voice.mozilla.org/en/datasets – English version en_1488h_2019-12-10
[23] http://i13pc106.ira.uka.de/~mmueller/iwslt-corpus.zip
[24] https://ufal.mff.cuni.cz/czeng/czeng17
[25] http://www.statmt.org/wmt19/
[26] http://i13pc106.ira.uka.de/~mmueller/iwslt-corpus.zip

7.2.2 Test Data

The test set was prepared by the EU project ELITR[27] which aims at automatic simultaneous translation of speech into subtitles in the particular domain of conference speeches on auditing.

The overall size of the test set is in Table 3. The details about the preparation of test set components are in Appendix A.6.

7.3 Submissions

Five teams from three institutions took part in the task. Each team provided one "primary" submission and some teams provided several further "contrastive" submissions. The primary submissions are briefly described in Table 4. Note that two teams (AppTek/RWTH and BUT) took the opportunity to reuse their systems from Offline Translation Task (Section 4) also in our task.

For the purposes of comparison, we also included freely available ASR services and MT services by two companies and denote the cascaded run for each of them as PUBLIC-A and PUBLIC-B. The ASR was run at the task submission deadline, the MT was added only later, on May 25, 2020.

7.4 Results

Appendix A.6 presents the results of the Non-Native Speech Translation Task for English→German and English→Czech, resp.

Note that the primary choice of most teams does not agree with which of their runs received the best scores in our evaluation. This can be easily explained by the partial domain mismatch between the development set and the test set.

The scores in both German and Czech results indicate considerable differences among the systems both in ASR quality as well as in BLEU scores. Before drawing strong conclusions from these scores, one has to consider that the results are heavily affected by the lack of reliable segmentation. If MT systems receive sequences of words not well matching sentence boundaries, they tend to reconstruct the sentence structure, causing serious translation errors.

The lack of golden sound segmentation also affects the evaluation: mwerSegmenter used in preprocessing of WER_{mw} and BLEU_{mw} optimizes WER score but it operates on a slightly different tokenization and casing. While the instability will be small in WER evaluation, it could cause more problems in BLEU_{mw}. Our BLEU calculation comes from sacreBLEU it its default setting. Furthermore, it needs to be considered that this is the first instance of the Non-Native shared task and not all peculiarities of the used evaluation measures and tools are quite known.[28] A manual evaluation would be desirable but even that would be inevitably biased depending on the exact way of presenting system outputs to the annotators. A procedure for a reliable manual evaluation of spoken language translation without pre-defined segmentation is yet to be sought.

The ASR quality scores[29] WER_1 and WER_{mw} are consistent with each other (Pearson .99), ranging from 14 (best submission by AppTek/RWTH) to 33 WER_1. WER_{mw} is always 1–3.5 points absolute higher.

Translation quality scores BLEU_1 and BLEU_{mw} show a similarly high correlation (Pearson .987) and reach up to 16. For English-to-German, the best translation was achieved by the secondary submissions of AppTek/RWTH, followed by the primary ELITR-offline and one of the secondary submissions of CUNI-nn. The public services seem to score worse, PUBLIC-B follows very closely and PUBLIC-A seems to seriously underperform, but it is quite possible that our cascaded application of their APIs was suboptimal. The only on-line set of submissions (ELITR) score between the two public systems.

The situation for English-to-Czech is similar, except that AppTek/RWTH did not take part in this, so ELITR-offline provided the best ASR as well as translations (one of their secondary submissions).

Often, there is a big variance of BLEU scores across all the submissions of one team. This indicates that the test set was hard to prepare for and that for a practical deployment, testing on the real input data is critical.

As expected, the ASR quality limits the trans-

[27] http://elitr.eu/

[28] In our analysis, we also used BLEU as implemented in NLTK (Bird et al., 2009), observing substantial score differences. For instance, BUT1 received NLTK-BLEU of 12.68 instead of 0.63 reported in Appendix A.6 BLEU_{mw}. For other submissions, NLTK-BLEU dropped to zero without a clear reason, possibly some unexpected character in the output. The explanation of why NLTK can inflate scores is still pending but it should be performed to be sure that sacreBLEU does not unduly penalize BUT submissions.

[29] Note that the same ASR system was often used as the basis for translation into both Czech and German so the same ASR scores appear on multiple lines in Tables in Appendix A.6.

Team	Paper	Training Data	Off/On-Line	Cascaded
APPTEK/RWTH	Bahar et al. (2020a)†	Unconstrained	Off-Line	Cascaded
BUT	(unpublished draft)	Unconstrained	Off-Line	Ensemble E2E+Cascaded
CUNI	Polák et al. (2020)	Unconstrained	Off-Line	Cascaded
ELITR	Macháček et al. (2020)	Unconstrained	On-Line	Cascaded
ELITR-OFFLINE	Macháček et al. (2020)	Unconstrained	Off-Line	Cascaded
PUBLIC-A	– (public service)	Unconstrained	Off-Line	Cascaded
PUBLIC-B	– (public service)	Unconstrained	Off-Line	Cascaded

† The paper describes the basis of the systems but does not explicitly refer to non-native translation task.

Table 4: Primary submissions to Non-Native Speech Translation Task. The public web-based services were added by task organizers for comparison, no details are known about the underlying systems.

lation quality. WER_1 and $BLEU_1$ correlate negatively (Pearson -.82 for translation to German and -.66 for translation to Czech). Same correlations were observed for WER_{mw} and $BLEU_{mw}$.

The test set as well as the system outputs will be made available at the task web page[30] for future deep inspection.

7.4.1 Trade-Offs in Simultaneous SLT

The trade-offs in simultaneity of the translation can be studied only on submissions of ELITR, see Appendix A.6. We see that the Delay ranges between 1 and up to 2.5 seconds, with $Delay_{mw}$ giving sligthly lower scores on average, correlated reasonably well with $Delay_{ts}$ (Pearson .989). Delay into German seems higher for this particular set of MT systems.

The best score observed for Flicker is 5.18 and the worst is 7.51. At the same time, Flicker is not really negatively correlated with the Delays, e.g. $Delay_{ts}$ vs. Flicker have the Pearson correlation of -.20.

Unfortunately, our current scoring does not allow to study the relationship between the translation quality and simultaneity, because our BLEU scores are calculated only on the final segments. Any intermediate changes to the translation text are not reflected in the scores.

Note that the timing information on when each output was produced was provided by the participants themselves. A fully reliable evaluation would require participants installing their systems on our hardware to avoid effects of network traffic, which is clearly beyond the goals of this task.

8 Conclusions

The evaluation campaign of the IWSLT 2020 conference offered six challenge tracks which attracted a total of 30 teams, both from academy and industry. The increasing number of participants witnesses the growing interest towards research on spoken language translation by the NLP community, which we believe has been partly driven by the availability of suitable training resources as well as the versatility of neural network models, which now permit to directly tackle complex tasks, such as speech-to-text translation, which formerly required building very complex system. We hope that this trend will continue and invite researchers interested in proposing new challenges for the next edition to get in touch with us. Finally, results of the human evaluation, which was still ongoing at the time of writing the overview paper, will be reported at the conference and will be included in an updated version of this paper.

[30] http://iwslt.org/doku.php?id=non_native_speech_translation

9 Acknowledgements

The offline Speech Translation task has been partially supported by the "End-to-end Spoken Language Translation in Rich Data Conditions" Amazon AWS ML Grant. The Non-Native Speech Translation Task was supported by the grants 19-26934X (NEUREM3) of the Czech Science Foundation, and H2020-ICT-2018-2-825460 (ELITR) of the EU. We are also grateful to Mohammad Mahmoudi for the assistance in the task evaluation and to Jonáš Kratochvíl for processing the input with public web ASR and MT services by two well-known companies.
The Open Domain Translation Task acknowledges the contributions of Yiqi Huang, Boliang Zhang and Arkady Arkhangorodsky, colleagues at DiDi Labs, for their help with the organization and sincerely thank Anqi Huang, a bilingual speaker, for validating the quality of the collected evaluation dataset.

A.6. Non-Native Speech Translation

English→German

· Complete result for English-German SLT systems followed by public systems PUBLIC-A and PUBLIC-B for comparison.
· Primary submissions are indicated by gray background. Best results in bold.

	SLT					ASR	
	Quality		Simultaneity			Quality	
System	$BLEU_1$	$BLEU_{mw}$	Flicker	$Delay_{ts}$[Match%]	$Delay_{mw}$[Match%]	WER_1	WER_{mw}
APPTEK/RWTH1	14.70	13.28	-	-	-	**14.27**	**16.26**
APPTEK/RWTH2	**16.14**	**15.00**	-	-	-	**14.27**	**16.26**
APPTEK/RWTH3	15.92	14.50	-	-	-	**14.27**	**16.26**
BUT1	2.25	0.63	-	-	-	32.33	34.09
BUT2	2.25	0.67	-	-	-	32.91	34.46
BUT3	1.93	0.59	-	-	-	32.91	34.46
BUT4	2.29	0.72		-	-	32.91	34.46
CUNI-NN11	6.37	5.86	-	-	-	28.68	32.10
CUNI-NN12	14.08	12.38	-	-	-	17.39	20.46
CUNI-NN13	14.32	12.73	-	-	-	17.02	19.98
CUNI-NN14	6.65	6.20	-	-	-	28.75	32.23
CUNI-NN15	12.51	10.88	-	-	-	16.54	18.19
CUNI-NN16	13.15	11.50	-	-	-	16.33	17.95
ELITR31	9.72	7.22	6.71	**1.901 [50.91%]**	1.926 [30.01%]	23.77	25.15
ELITR32	9.18	7.32	7.48	1.926 [30.01%]	1.944 [30.42%]	22.91	24.26
ELITR33	9.18	7.32	7.48	1.972 [52.61%]	1.945 [30.43%]	22.91	24.26
ELITR34	9.18	7.32	7.43	1.951 [52.53%]	1.923 [30.41%]	22.91	24.26
ELITR35	9.18	7.32	6.48	2.038 [52.84%]	2.024 [30.76%]	22.91	24.26
ELITR36	9.18	7.32	5.97	2.034 [52.66%]	2.029 [30.79%]	22.91	24.26
ELITR37	9.39	7.05	6.33	2.471 [34.14%]	**1.828 [31.81%]**	23.81	25.25
ELITR38	9.40	7.06	6.35	2.461 [34.24%]	1.846 [31.85%]	23.81	25.25
ELITR39	9.40	7.06	6.33	2.380 [33.37%]	**1.810 [31.63%]**	23.81	25.25
ELITR40	9.39	7.05	5.66	2.544 [34.28%]	1.964 [32.28%]	23.81	25.25
ELITR41	9.39	7.06	**5.30**	2.391 [34.09%]	1.957 [32.28%]	23.81	25.25
ELITR-OFFLINE21	14.83	12.67	-	-	-	15.29	17.67
ELITR-OFFLINE22	13.31	11.35	-	-	-	15.29	17.67
ELITR-OFFLINE23	14.08	12.33	-	-	-	15.29	17.67
ELITR-OFFLINE24	13.03	10.76	-	-	-	15.29	17.67
ELITR-OFFLINE25	12.88	10.83	-	-	-	15.29	17.67
ELITR-OFFLINE26	10.45	8.32	-	-	-	15.29	17.67
ELITR-OFFLINE27	11.58	9.87	-	-	-	16.33	17.95
ELITR-OFFLINE28	11.76	9.83	-	-	-	16.33	17.95
ELITR-OFFLINE29	12.51	10.88	-	-	-	16.33	17.95
ELITR-OFFLINE30	11.34	9.42	-	-	-	16.33	17.95
ELITR-OFFLINE31	12.51	10.53	-	-	-	16.33	17.95
ELITR-OFFLINE32	7.89	5.72	-	-	-	16.33	17.95
CUNI-KALDI01	-	-	-	-	-	22.88	24.53
CUNI-KALDI02	-	-	-	-	-	30.42	31.17
CUNI-KALDI03	-	-	-	-	-	21.25	23.40
PUBLIC-A	4.29	3.02	-	-	-	30.10	31.09
PUBLIC-B	13.75	12.35	-	-	-	21.54	23.59

English→Czech

· Complete result for English-Czech SLT systems followed by public systems PUBLIC-A and PUBLIC-B for comparison.
· Primary submissions are indicated by gray background. Best results in bold.

	SLT					ASR	
	Quality		Simultaneity			Quality	
System	$BLEU_1$	$BLEU_{mw}$	Flicker	$Delay_{ts}$[Match%]	$Delay_{mw}$[Match%]	WER_1	WER_{mw}
CUNI-NN01	10.57	10.34	-	-	-	28.68	32.10
CUNI-NN02	10.89	11.50	-	-	-	17.39	20.46
CUNI-NN03	12.74	11.38	-	-	-	17.02	19.98
CUNI-NN04	10.24	10.21	-	-	-	28.75	32.23
CUNI-NN05	11.85	10.57	-	-	-	16.54	18.19
CUNI-NN06	12.27	11.00	-	-	-	16.33	17.95
ELITR01	7.87	6.22	7.00	1.530 [42.45%]	1.575 [23.93%]	23.77	25.15
ELITR02	7.56	5.95	6.46	1.696 [22.01%]	1.561 [25.25%]	23.81	25.25
ELITR03	7.56	5.95	6.38	1.744 [22.26%]	1.618 [25.34%]	23.81	25.25
ELITR04	7.54	5.93	6.38	1.725 [22.09%]	1.603 [25.32%]	23.81	25.25
ELITR05	8.93	7.67	7.51	1.605 [44.80%]	1.623 [92.49%]	23.81	25.25
ELITR06	8.79	7.54	7.00	**1.198 [52.55%]**	**1.082 [32.18%]**	23.81	25.25
ELITR07	8.93	7.67	6.97	1.596 [44.79%]	1.630 [24.86%]	23.81	25.25
ELITR08	8.93	7.67	6.54	1.586 [44.64%]	1.629 [24.91%]	23.81	25.25
ELITR09	8.93	7.65	7.38	1.520 [42.80%]	1.503 [23.23%]	23.81	25.25
ELITR10	8.93	7.67	7.41	1.630 [44.77%]	1.667 [24.96%]	23.81	25.25
ELITR11	6.50	4.94	6.00	1.677 [20.99%]	1.595 [24.58%]	23.81	25.25
ELITR12	6.50	4.94	6.26	1.610 [20.87%]	1.504 [24.35%]	23.81	25.25
ELITR13	6.50	4.94	6.26	1.495 [19.47%]	1.399 [23.30%]	23.81	25.25
ELITR14	6.52	4.95	5.69	1.650 [20.88%]	1.597 [24.63%]	23.81	25.25
ELITR15	6.50	4.94	**5.18**	1.541 [20.71%]	1.594 [24.59%]	23.81	25.25
ELITR16	7.40	5.74	6.64	1.633 [21.89%]	1.468 [24.43%]	23.81	25.25
ELITR17	8.45	7.32	6.56	1.597 [44.85%]	1.728 [25.35%]	22.91	24.26
ELITR18	8.36	7.17	6.00	1.514 [45.58%]	1.629 [26.54%]	22.91	24.26
ELITR19	8.56	7.45	5.31	1.600 [46.81%]	1.713 [27.94%]	22.91	24.26
ELITR20	8.55	7.41	6.31	1.557 [45.78%]	1.704 [26.51%]	22.91	24.26
ELITR-OFFLINE01	13.33	11.75	-	-	-	**15.29**	**17.67**
ELITR-OFFLINE02	13.44	11.64	-	-	-	**15.29**	**17.67**
ELITR-OFFLINE03	13.56	11.79	-	-	-	**15.29**	**17.67**
ELITR-OFFLINE04	**14.08**	**12.57**	-	-	-	**15.29**	**17.67**
ELITR-OFFLINE05	10.07	8.23	-	-	-	**15.29**	**17.67**
ELITR-OFFLINE06	8.42	6.99	-	-	-	**15.29**	**17.67**
ELITR-OFFLINE07	9.62	8.16	-	-	-	**15.29**	**17.67**
ELITR-OFFLINE08	11.88	10.26	-	-	-	16.33	17.95
ELITR-OFFLINE09	11.52	9.83	-	-	-	16.33	17.95
ELITR-OFFLINE10	11.43	9.99	-	-	-	16.33	17.95
ELITR-OFFLINE11	11.85	10.57	-	-	-	16.33	17.95
ELITR-OFFLINE12	9.29	7.76	-	-	-	16.33	17.95
ELITR-OFFLINE13	7.76	6.35	-	-	-	16.33	17.95
ELITR-OFFLINE14	7.37	6.54	-	-	-	16.33	17.95
PUBLIC-A	3.30	2.47	-	-	-	30.10	31.09
PUBLIC-B	10.79	9.85	-	-	-	21.54	23.59

Test Set Provenance

Only a limited amount of resources could have been invested in the preparations of the test set and the test set thus build upon some existing datasets. The components of the test sets are:

Antrecorp[36] (Macháček et al., 2019), a test set of up to 90-second mock business presentations given by high school students in very noisy conditions. None of the speakers is a native speaker of English (see the paper for the composition of nationalities) and their English contains many lexical, grammatical and pronunciation errors as well as disfluencies due to the spontaneous nature of the speech.

For the purposes of this task, we equipped Antrecorp with manual translations into Czech and German. No MT system was used to pre-translate the text to avoid bias in automatic evaluation.

[36] http://hdl.handle.net/11234/1-3023

Because the presentations are very informal and their translation can vary considerably, we created two independent translations into Czech. In the end, only the first one of them was used as the reference, to keep BLEU scores across test set parts somewhat comparable.

Khan Academy[37] is a large collection of educational videos. The speaker is not a native speaker of English but his accent is generally rather good. The difficulty in this part of the test lies in the domain and also the generally missing natural segmentation into sentences.

SAO is a test set created by ELITR particularly for this shared task, to satisfy the need of the Supreme Audit Office of the Czech Republic. The test sets consists of 6 presentations given in English by officers of several supreme audit institutions (SAI) in Europe and by the Europan Court of Auditors. The speakers nationality (Austrian, Belgian, Dutch, Polish, Romanian and Spanish) affects their accent. The Dutch file is a recording of a remote conference call with further distorted sound quality.

The development set contained 2 other files from Antrecorp, one other file from the SAO domain and it also included 4 files from the AMI corpus (Mccowan et al., 2005) to illustrate non-native accents. We did not include data from AMI corpus in the test set because we found out that some participants trained their (non-constrained) submissions on it.

For SAO and Antrecorp, our test set was created in the most straightforward way: starting with the original sound, manual transcription was obtained (with the help of ASR) as a line-oriented plaintext. The transcribers were instructed to preserve all words uttered[38] and break the sequence of words into sentences in as natural way as possible. Correct punctuation and casing was introduced at this stage, too. Finally, the documents were translated in Czech and German, preserving the segmentation into "sentences".

For the evaluation of SLT simultaneity, we force-aligned words from the transcript to the sound using a model trained with Jasper (Li et al., 2019) and resorted to fully manual identification of word boundaries in the few files where forced alignment failed.

Despite a careful curation of the dataset, we are aware of the following limitations. None of them are too frequent or too serious but they still deserve to be mentioned:

- Khan Academy subtitles never had proper segmentation into sentences and manual correction of punctuation and casing. The subtitles were supposedly manually refined but the focus was on their presentation in the running video lecture, not on style and typesetting.
- Khan Academy contains many numbers (written mostly as numbers). For small numbers, both digits and words are often equally suitable but automatic metrics treat this difference as a mistranslation and no straightforward reliable normalization is possible either, so we did not apply any.
- Minor translation errors into German were seen in Khan Academy videos and in the "Belgian" SAO file.

[37] http://www.khanacademy.org/

[38] This decision is possibly less common in the ASR community but it is motivated by the subsequent translation step which has the capacity to recover from disfluences as needed.

D Adaptive Feature Selection for End-to-End Speech Translation

Adaptive Feature Selection for End-to-End Speech Translation

Biao Zhang[1] **Ivan Titov**[1,2] **Barry Haddow**[1] **Rico Sennrich**[3,1]

[1]School of Informatics, University of Edinburgh
[2]ILLC, University of Amsterdam
[3]Department of Computational Linguistics, University of Zurich

Abstract

Information in speech signals is not evenly distributed, making it an additional challenge for end-to-end (E2E) speech translation (ST) to learn to focus on informative features. In this paper, we propose adaptive feature selection (AFS) for encoder-decoder based E2E ST. We first pre-train an ASR encoder and apply AFS to dynamically estimate the importance of each encoded speech feature to ASR. A ST encoder, stacked on top of the ASR encoder, then receives the filtered features from the (frozen) ASR encoder. We take $\mathcal{L}_0$DROP (Zhang et al., 2020) as the backbone for AFS, and adapt it to sparsify speech features with respect to both temporal and feature dimensions. Results on LibriSpeech En-Fr and MuST-C benchmarks show that AFS facilitates learning of ST by pruning out ∼84% temporal features, yielding an average translation gain of ∼1.3–1.6 BLEU and a decoding speedup of ∼1.4×. In particular, AFS reduces the performance gap compared to the cascade baseline, and outperforms it on LibriSpeech En-Fr with a BLEU score of 18.56 (without data augmentation).[1]

1 Introduction

End-to-end (E2E) speech translation (ST), a paradigm that directly maps an audio to a foreign text, has been gaining popularity recently (Duong et al., 2016; Bérard et al., 2016; Bansal et al., 2018; Di Gangi et al., 2019; Wang et al., 2019). Based on the attentional encoder-decoder framework (Bahdanau et al., 2015), it optimizes model parameters under direct translation supervision. This end-to-end paradigm avoids the problem of error propagation that is inherent in cascade models where an automatic speech recognition (ASR) model and a machine translation (MT) model are chained together. Nonetheless, previous work still reports that E2E ST delivers inferior performance compared to cascade methods (Niehues et al., 2019).

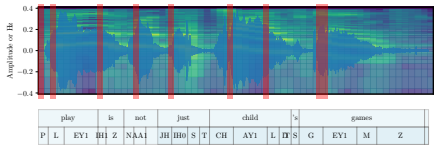


Figure 1: An example for motivation illustration. We plot the amplitude and frequency spectrum of an audio (top), paired with its time-aligned words and phonemes (bottom). Information inside an audio is not uniformly distributed. We propose to dynamically capture speech features corresponding to informative signals (red rectangles) to improve ST.

We study one reason for the difficulty of training E2E ST models, namely the uneven spread of information in the speech signal, as visualized in Figure 1, and the consequent difficulty of extracting informative features. Features corresponding to uninformative signals, such as pauses or noise, increase the input length and bring in unmanageable noises for ST. This increases the difficulty of learning (Zhang et al., 2019b; Na et al., 2019) and reduces translation performance.

In this paper, we propose adaptive feature selection (AFS) for ST to explicitly eliminate uninformative features. Figure 2 shows the overall architecture. We employ a pretrained ASR encoder to induce contextual speech features, followed by a ST encoder bridging the gap between speech and translation modalities. AFS is inserted in-between them to select a subset of features for ST encoding (see red rectangles in Figure 1). To ensure that the selected features are well-aligned to transcriptions, we pretrain AFS on ASR. AFS estimates the informativeness of each feature through a parameterized gate, and encourages the dropping of features (pushing the gate to 0) that contribute little to ASR. Intuitively, features irrelevant for ASR are also unimportant for ST.

[1]We will release our source code at XXX.

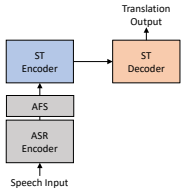


Figure 2: Overview of our E2E ST model. AFS is inserted between the ST encoder (blue) and a pretrained ASR encoder (gray) to filter speech features for translation. We pretrain AFS jointly with ASR and freeze it during ST training.

We base AFS on $\mathcal{L}_0$DROP (Zhang et al., 2020), a sparsity-inducing method for encoder-decoder models, and extend it to sparsify speech features. The acoustic input of speech signals involves two dimensions: *temporal* and *feature*, where the latter one describes the spectrum extracted from time frames. Accordingly, we adapt $\mathcal{L}_0$DROP to sparsify encoder states along temporal and feature dimensions but using different gating networks. In contrast to (Zhang et al., 2020), who focus on efficiency and report a trade-off between sparsity and quality for MT and summarization, we find that sparsity also improves translation quality for ST.

We conduct extensive experiments with Transformer (Vaswani et al., 2017) on LibriSpeech En-Fr and MuST-C speech translation tasks, covering 8 different language pairs. Results show that AFS only retains about 16% of temporal speech features, revealing heavy redundancy in speech encodings and yielding a decoding speedup of $\sim$1.4$\times$. AFS eases model convergence, and improves the translation quality by $\sim$1.3–1.6 BLEU, surpassing several strong baselines. Specially, without data augmentation, AFS narrows the performance gap against the cascade approach, and outperforms it on LibriSpeech En-Fr by 0.29 BLEU, reaching 18.56. We compare against fixed-rate feature selection confirming that our adaptive feature selection offers better translation quality.

2 Related Work

Speech Translation Pioneering studies on ST used a cascade of separately trained ASR and MT systems (Ney, 1999). Despite its simplicity, this approach inevitably suffers from mistakes made by ASR models, and is error prone. Research in this direction often focuses on strategies capable of mitigating the mismatch between ASR output and MT input, such as representing ASR outputs with lattices (Saleem et al., 2004; Mathias and Byrne, 2006; Zhang et al., 2019a; Beck et al., 2019), injecting synthetic ASR errors for robust MT (Tsvetkov et al., 2014; Cheng et al., 2018) and differentiable cascade modeling (Kano et al., 2017; Anastasopoulos and Chiang, 2018; Sperber et al., 2019).

In contrast to cascading, another option is to perform direct speech-to-text translation. Duong et al. (2016) and Bérard et al. (2016) employ the attentional encoder-decoder model (Bahdanau et al., 2015) for E2E ST without accessing any intermediate transcriptions. E2E ST opens the way to bridging the modality gap directly, but it is data-hungry, sample-inefficient and often underperforms cascade models especially in low-resource settings (Bansal et al., 2018). This led researchers to explore solutions ranging from efficient neural architecture design (Karita et al., 2019; Di Gangi et al., 2019; Sung et al., 2019) to extra training signal incorporation, including multi-task learning (Weiss et al., 2017; Liu et al., 2019b), submodule pretraining (Bansal et al., 2019; Stoian et al., 2020; Wang et al., 2020), knowledge distillation (Liu et al., 2019a), meta-learning (Indurthi et al., 2019) and data augmentation (Kocabiyikoglu et al., 2018; Jia et al., 2019; Pino et al., 2019). Our work focuses on E2E ST, but we investigate feature selection which has rarely been studied before.

Speech Feature Selection Encoding speech signals is challenging as acoustic input is lengthy, noisy and redundant. To ease model learning, previous work often selected features via downsampling techniques, such as convolutional modeling (Di Gangi et al., 2019) and fixed-rate subsampling (Lu et al., 2015). Recently, Zhang et al. (2019b) and Na et al. (2019) proposed dynamic subsampling for ASR which learns to skip uninformative features during recurrent encoding. Unfortunately, their methods are deeply embedded into recurrent networks, hard to adapt to other architectures like Transformer (Vaswani et al., 2017). Recently, Salesky et al. (2020) have explored phoneme-level representations for E2E ST, but this requires nontrivial phoneme recognition and alignment.

Instead, we employ $\mathcal{L}_0$DROP (Zhang et al., 2020) for AFS to dynamically retain informative speech features, which is fully differentiable and independent of concrete encoder/decoder architectures. We extend $\mathcal{L}_0$DROP by handling both temporal and feature dimensions with different gating networks, and apply it to E2E ST.

3 Background: $\mathcal{L}_0$DROP

$\mathcal{L}_0$DROP provides a selective mechanism for encoder-decoder models which encourages removing uninformative encoder outputs via a sparsity-inducing objective (Zhang et al., 2020). Given a source sequence $X = \{x_1, x_2, \ldots, x_n\}$, $\mathcal{L}_0$DROP assigns each encoded source state $\mathbf{x}_i \in \mathbb{R}^d$ with a scalar gate $g_i \in [0, 1]$ as follows:

$$\mathcal{L}_0\text{DROP}(\mathbf{x}_i) = g_i \mathbf{x}_i, \quad (1)$$

$$\text{with } g_i \sim \text{HardConcrete}(\alpha_i, \beta, \epsilon), \quad (2)$$

where $\alpha_i, \beta, \epsilon$ are hyperparameters of the hard concrete distribution (HardConcrete) (Louizos et al., 2018; Bastings et al., 2019).

Note that the hyperparameter α_i is crucial with HardConcrete, it directly governs its shape. By associating α_i with $\mathbf{x}_i$ through a gating network:

$$\log \alpha_i = \mathbf{x}_i^T \cdot \mathbf{w}, \quad (3)$$

$\mathcal{L}_0$DROP could schedule HardConcrete via α_i to put more probability mass at either 0 (i.e $g_i \to 0$) or 1 (i.e. $g_i \to 1$). $\mathbf{w} \in \mathbb{R}^d$ is a trainable parameter. Intuitively, $\mathcal{L}_0$DROP controls the openness of gate g_i via α_i so as to determine whether removing ($g_i = 0$) or retaining ($g_i = 1$) the state $\mathbf{x}_i$.

$\mathcal{L}_0$DROP enforces sparsity by pushing the probability mass of HardConcrete towards 0, according to the following penalty term:

$$\mathcal{L}_0(X) = \sum_{i=1}^{n} 1 - p(g_i = 0 | \alpha_i, \beta, \epsilon). \quad (4)$$

By sampling g_i with reparameterization (Kingma and Welling, 2013), $\mathcal{L}_0$DROP is fully differentiable and optimized with an upper bound on the objective: $\mathcal{L}_{\text{MLE}} + \lambda \mathcal{L}_0(X)$, where λ is a hyperparameter affecting the degree of sparsity and $\mathcal{L}_{\text{MLE}}$ denotes the maximum likelihood loss. The expected value of g_i is used during inference. $\mathcal{L}_0$DROP successfully prunes out 40–70% encoder outputs on MT and summarization tasks, without compromising generation quality (Zhang et al., 2020). We adapt it to E2E ST.

4 Adaptive Feature Selection

One difficulty with applying encoder-decoder models to E2E ST is deciding how to encode speech signals. In contrast to text where word boundaries can be easily identified, the spectrum features of speech are continuous, varying remarkably across different speakers for the same transcript. In addition, redundant information, like pauses in-between neighbouring words, can be of arbitrary duration at any position as shown in Figure 1, while contributing little to translation. This increases the burden and occupies the capacity of ST encoder, leading to inferior performance (Duong et al., 2016; Bérard et al., 2016). Rather than developing complex encoder architectures, we resort to feature selection to explicitly clear out those uninformative speech features.

Figure 2 gives an overview of our model. We use a pretrained and frozen ASR encoder to extract contextual speech features, and collect the informative ones from them via AFS before transmitting to the ST encoder. AFS drops pauses, noise and other uninformative features and retains features that are relevant for ASR. We speculate that these retained features are also the most relevant for ST, and that the sparser representation simplifies the learning problem for ST, for example the learning of attention strength between encoder states and target language (sub)words. Given a training tuple (audio, source transcription, translation), denoted as (X, Y, Z) respectively,[2] we outline the overall framework below, including three steps:

E2E ST with AFS

1. Train ASR model with the following objective and model architecture until convergence:

$$\mathcal{L}^{\text{ASR}} = \eta \mathcal{L}_{\text{MLE}}(Y|X) + \gamma \mathcal{L}_{\text{CTC}}(Y|X), \quad (5)$$

$$\mathcal{M}^{\text{ASR}} = D^{\text{ASR}}\left(Y, E^{\text{ASR}}(X)\right). \quad (6)$$

2. Finetune ASR model with AFS for m steps:

$$\mathcal{L}^{\text{AFS}} = \mathcal{L}_{\text{MLE}}(Y|X) + \lambda \mathcal{L}_0(X), \quad (7)$$

$$\mathcal{M}^{\text{AFS}} = D^{\text{ASR}}\left(Y, F\left(E^{\text{ASR}}(X)\right)\right). \quad (8)$$

3. Train ST model with pretrained and frozen ASR and AFS submodules until convergence:

$$\mathcal{L}^{\text{ST}} = \mathcal{L}_{\text{MLE}}(Z|X), \quad (9)$$

$$\mathcal{M}^{\text{ST}} = D^{\text{ST}}\left(Z, E^{\text{ST}}\left(\overline{FE}^{\text{ASR}}(X)\right)\right). \quad (10)$$

We handle both ASR and ST as sequence-to-sequence problem with encoder-decoder models. We use $E^*(\cdot)$ and $D^*(\cdot, \cdot)$ to denote the corresponding encoder and decoder respectively. $F(\cdot)$ denotes the AFS approach, and $\overline{FE}$ means freezing the ASR encoder and the AFS module during train-

[2] Note that our model only requires pair-wise training corpora, (X, Y) for ASR, and (X, Z) for ST.

ing. Note that our framework puts no constraint on the architecture of the encoder and decoder in any task, although we adopt the multi-head dot-product attention network (Vaswani et al., 2017) for our experiments.

ASR Pretraining The ASR model $\mathcal{M}^{\text{ASR}}$ (Eq. 6) directly maps an audio input to its transcription. To improve speech encoding, we apply logarithmic penalty on attention to enforce short-range dependency (Di Gangi et al., 2019) and use trainable positional embedding with a maximum length of 2048. Apart from $\mathcal{L}_{\text{MLE}}$, we augment the training objective with the connectionist temporal classification (Graves et al., 2006, CTC) loss $\mathcal{L}_{\text{CTC}}$ as in Eq. 5. Note $\eta = 1 - \gamma$. The CTC loss is applied to the encoder outputs, guiding them to align with their corresponding transcription (sub)words and improving the encoder's robustness (Karita et al., 2019). Following previous work (Karita et al., 2019; Wang et al., 2020), we set γ to 0.3.

AFS Finetuning This stage aims at using AFS to dynamically pick out a subset of ASR encoder's outputs that are most relevant for ASR performance (see red rectangles in Figure 1). We follow Zhang et al. (2020) and place AFS in-between ASR's encoder and decoder during finetuning (see $F(\cdot)$ in $\mathcal{M}^{\text{AFS}}$, Eq. 8). We exclude the CTC loss in the training objective (Eq. 7) to relax the alignment constraint and increase the flexibility of feature adaptation. We use $\mathcal{L}_0$DROP for AFS in two ways.

AFS^t The direct application of $\mathcal{L}_0$DROP on ASR encoder results in AFS^t, sparsifying encodings along the temporal dimension $\{\mathbf{x}_i\}_{i=1}^n$:

$$\begin{aligned} F^t(\mathbf{x}_i) &= \text{AFS}^t(\mathbf{x}_i) = g_i^t \mathbf{x}_i, \\ \text{with} \quad \log \alpha_i^t &= \mathbf{x}_i^T \cdot \mathbf{w}^t, \\ g_i^t &\sim \text{HardConcrete}(\alpha_i^t, \beta, \epsilon), \end{aligned} \tag{11}$$

where α_i^t is a positive scalar powered by a simple linear gating layer, and $\mathbf{w}^t \in \mathbb{R}^d$ is a trainable parameter of dimension d. $\mathbf{g}^t$ is the temporal gate. The sparsity penalty of AFS^t follows Eq. 4:

$$\mathcal{L}_0^t(X) = \sum_{i=1}^{n} 1 - p(g_i^t = 0 | \alpha_i^t, \beta, \epsilon). \tag{12}$$

AFS^{t,f} In contrast to text processing, speech processing often extracts spectrum from overlapping time frames to form the acoustic input, similar to the word embedding. As each encoded speech feature contains temporal information, it is reasonable to extend AFS^t to AFS^{t,f}, including sparsification along the feature dimension $\{\mathbf{x}_{i,j}\}_{j=1}^d$:

$$\begin{aligned} F^{t,f}(\mathbf{x}_i) &= \text{AFS}^{t,f}(\mathbf{x}_i) = g_i^t \mathbf{x}_i \odot \mathbf{g}^f, \\ \text{with} \quad \log \boldsymbol{\alpha}^f &= \mathbf{w}^f, \\ \mathbf{g}_j^f &\sim \text{HardConcrete}(\boldsymbol{\alpha}_j^f, \beta, \epsilon), \end{aligned} \tag{13}$$

where $\boldsymbol{\alpha}^f \in \mathbb{R}^d$ estimates the weights of each feature, dominated by an input-independent gating model with trainable parameter $\mathbf{w}^f \in \mathbb{R}^d$.[3] $\mathbf{g}^f$ is the feature gate. Note that $\boldsymbol{\alpha}^f$ is shared for all speech features. $\odot$ denotes element-wise multiplication. AFS^{t,f} reuses g_i^t-relevant submodules in Eq. 11, and extends the sparsity penalty $\mathcal{L}_0^t$ in Eq. 12 as follows:

$$\mathcal{L}_0^{t,f}(X) = \mathcal{L}_0^t + \sum_{j=1}^{d} 1 - p(\mathbf{g}_j^f = 0 | \boldsymbol{\alpha}_j^f, \beta, \epsilon). \tag{14}$$

We perform the finetuning by replacing $(F, \mathcal{L}_0)$ in Eq. (8-7) with either AFS^t $(F^t, \mathcal{L}_0^t)$ or AFS^{t,f} $(F^{t,f}, \mathcal{L}_0^{t,f})$ for extra m steps. We compare these two variants in our experiments.

E2E ST Training We treat the pretrained ASR and AFS model as a speech feature extractor, and freeze them during ST training. We gather the speech features emitted by the ASR encoder that correspond to $g_i^t > 0$, and pass them similarly as done with word embeddings to the ST encoder. We employ sinusoidal positional encoding to distinguish features at different positions. Except for the input to the ST encoder, our E2E ST follows the standard encoder-decoder translation model ($\mathcal{M}^{\text{ST}}$ in Eq. 10) and is optimized with $\mathcal{L}_{\text{MLE}}$ alone as in Eq. 9. Intuitively, AFS bridges the gap between ASR output and MT input by selecting transcript-aligned speech features.

5 Experiments

Datasets and Preprocessing We experiment with two benchmarks: the Augmented LibriSpeech dataset (LibriSpeech En-Fr) (Kocabiyikoglu et al., 2018) and the multilingual MuST-C dataset (MuST-C) (Di Gangi et al., 2019). LibriSpeech En-Fr is collected by aligning e-books in French with English utterances of LibriSpeech, further augmented with French translations offered by Google Translate service. We use the 100 hours clean training

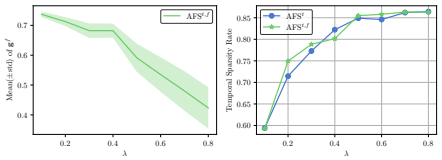
[3] Other candidate gating models, like linear mapping upon mean-pooled encoder outputs, delivered worse performance in our preliminary experiments.

set for training, including 47K utterances to train ASR models and double size for ST models after concatenated with the Google translations. We report results on the test set (2048 utterances) using models selected on the dev set (1071 utterances). MuST-C is built from English TED talks, covering English→German (De), Spanish (Es), French (Fr), Italian (It), Dutch (Nl), Portuguese (Pt), Romanian (Ro) and Russian (Ru) translation directions (8 in total). We train ASR and ST models on the given training set, containing ∼452 hours with ∼252K utterances on average for each translation pair. We adopt the given dev set for model selection and report results on the common test set, whose size ranges from 2502 (Es) to 2641 (De) utterances.

For all datasets, we extract 40-dimensional log-Mel filterbanks with a step size of 10ms and window size of 25ms as the acoustic features. We expand these features with their first and second-order derivatives, and stabilize them using mean subtraction and variance normalization. We stack the features corresponding to three consecutive frames without overlapping to the left, resulting in the final 360-dimensional acoustic input. For transcriptions and translations, we tokenize and truecase all the text using the Moses scripts.[4] We train subword models (Sennrich et al., 2016) on each dataset with a joint vocabulary size of 16K to handle rare words, and share the model for ASR, MT and ST. We train all models without removing punctuation.

Model Settings and Baselines We adopt the Transformer architecture (Vaswani et al., 2017) for all tasks, including $\mathcal{M}^{\text{ASR}}$ (Eq. 6), $\mathcal{M}^{\text{AFS}}$ (Eq. 8) and $\mathcal{M}^{\text{ST}}$ (Eq. 10). The encoder and decoder consist of 6 identical layers, each including a self-attention sublayer, a cross-attention sublayer (decoder alone) and a feedforward sublayer. We employ the base setting for experiments: hidden size $d = 512$, attention head 8 and feedforward size 2048. We schedule learning rate via Adam ($\beta_1 = 0.9, \beta_2 = 0.98$) (Kingma and Ba, 2015), paired with a warmup step of 4K. We apply dropout to attention weights and residual connections with a rate of 0.1 and 0.2 respectively, and also add label smoothing of 0.1 to handle overfitting. We train all models with a maximum step size of 30K and a minibatch size of around 25K target subwords. We average the last 5 checkpoints for evaluation. We use beam search for decoding, and set the beam size and length penalty to 4 and 0.6, respectively.

[4] https://www.statmt.org/moses/



(a) Feature Gate Value (b) Temporal Sparsity Rate

Figure 3: Feature gate value and temporal sparsity rate as a function of λ on MuST-C En-De dev set. Larger λ decreases the gate value of $\mathbf{g}^f$ but without dropping any neurons, i.e. feature sparsity rate 0%. By contrast, speech features are of high redundancy along temporal dimension, easily inducing high sparsity rate of ∼85%.

We set $\epsilon = -0.1$, and $\beta = 2/3$ for AFS following Louizos et al. (2018), and finetune AFS for an additional $m = 5$K steps. We evaluate translation quality with tokenized case-sensitive BLEU (Papineni et al., 2002), and report WER for ASR performance without punctuation.

We compare our models with four baselines:

ST: A vanilla Transformer-based E2E ST model of 6 encoder and decoder layers. Logarithmic attention penalty (Di Gangi et al., 2019) is used to improve the encoder.

ST + ASR-PT: We perform the ASR pretraining (ASR-PT) for E2E ST. This is the same model as ours (Figure 2) but without AFS finetuning.

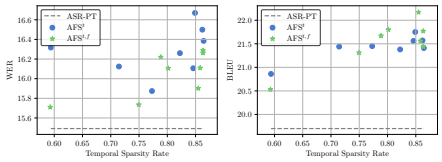
Cascade: We first transcribe the speech input using an ASR model, and then passes the results on to an MT model. We also use the logarithmic attention penalty (Di Gangi et al., 2019) for the ASR encoder.

ST + Fixed Rate: Instead of dynamically selecting features, we replace AFS with subsampling at a fixed rate: we extract the speech encodings after every k positions.

5.1 Results on MuST-C En-De

We perform a thorough study on MuST-C En-De. With AFS, the first question is its feasibility. We start by analyzing the degree of sparsity in speech features (i.e. sparsity rate) yielded by AFS, focusing on the temporal sparsity rate $\#\{g_i^t{=}0\}/n$ and the feature sparsity rate $\#\{g_j^f{=}0\}/d$. To obtain different rates, we vary the hyperparameter λ in Eq. 7 in a range of $[0.1, 0.8]$ with a step size 0.1.

Results in Figure 3 show that large amount of encoded speech features ($> 59\%$) can be easily pruned out, revealing heavy inner-speech redundancy. Both AFS^t and AFS^{t,f} drop ∼60% tempo-



(a) ASR (b) ST

Figure 4: ASR (WER↓) and ST (BLEU↑) performance as a function of *temporal* sparsity rate on MuST-C En-De dev set. Pruning out ∼85% temporal speech features largely improves translation quality and retains ∼95% ASR accuracy.

Model	BLEU↑	Speedup↑
MT	29.69	-
Cascade	22.52	1.06×
ST	17.44	0.87×
ST + ASR-PT	20.67	1.00×
ST + Fixed Rate ($k = 6$)	21.14 (83.3%)	1.42×
ST + Fixed Rate ($k = 7$)	20.87 (85.7%)	1.43×
ST + AFS^t	21.57 (84.4%)	1.38×
ST + AFS^{t,f}	22.38 (85.1%)	1.37×

Table 1: BLEU↑ and speedup↑ on MuST-C En-De test set. $\lambda = 0.5$. We evaluate the speedup on GeForce GTX 1080 Ti with a decoding batch size of 16, and report average results over 3 runs. Numbers in parentheses are the sparsity rate.

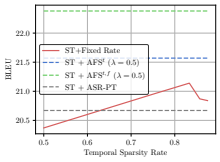


Figure 5: Impact of k in fixed-rate subsampling on ST performance on MuST-C En-De test set. Sparsity rate: $^{k-1}/_{k}$. This subsampling underperforms AFS, and degenerates the ST performance at suboptimal rates.

ral features with λ of 0.1, and enlarge this number to $> 85\%$ when $\lambda \geq 0.5$ (Figure 3b), remarkably surpassing the sparsity rate reported by Zhang et al. (2020) on text summarization (71.5%). In contrast to rich temporal sparsification, we get a feature sparsity rate of 0, regardless of λ's value, although increasing λ decreases $\mathbf{g}^f$ (Figure 3a). This suggests that selecting neurons from the feature dimension is harder. Rather than filtering neurons, the feature gate $\mathbf{g}^f$ acts more like a weighting mechanism on them. In the rest of the paper, we use *sparsity rate* for the temporal sparsity rate.

We continue to explore the impact of varied sparsity rates on the ASR and ST performance. Figure 4 shows their correlation. We observe that AFS slightly degenerates ASR accuracy (Figure 4a), but still retains ∼95% accuracy on average; AFS^{t,f} often performs better than AFS^t with similar sparsity rate. The fact that only 15% speech features successfully support 95% ASR accuracy proves the informativeness of these selected features. These findings echo with (Zhang et al., 2020), where they observe a trade-off between sparsity and quality.

However, when AFS is applied to ST, we find consistent improvements to translation quality by > 0.8 BLEU, shown in Figure 4b. Translation quality on the development set peaks at 22.17 BLEU achieved by AFS^{t,f} with a sparsity rate of 85.5%. We set $\lambda = 0.5$ (corresponding to sparsity rate of ∼85%) for all other experiments, since AFS^t and AFS^{t,f} reach their optimal result at this point.

We summarize the test results in Table 1, where we set $k = 6$ or $k = 7$ for *ST+Fixed Rate* with a sparsity rate of around 85% inspired by our above analysis. Our vanilla ST model yields a BLEU score of 17.44; pretraining on ASR further enhances the performance to 20.67, significantly outperforming the results of Di Gangi et al. (2019) by 3.37 BLEU. This also suggests the importance of speech encoder pretraining (Di Gangi et al., 2019; Stoian et al., 2020; Wang et al., 2020). We treat ST w/ ASR-PT as our real baseline. We observe improved translation quality with fixed-rate subsampling, +0.47 BLEU at $k = 6$. Subsampling offers a chance to bypass noisy speech signals and reducing the number of source states makes learning translation alignment easier, but deciding the optimal sampling rate is tough. Results in Figure 5 reveal that fixed-rate subsampling deteriorates ST performance with suboptimal rates. By contrast, the proposed AFS is data-driven, shifting the decision burden to the data and model themselves. As a result, AFS^t and AFS^{t,f} surpass ASR-PT by 0.9 BLEU and 1.71 BLEU, respectively, substantially narrowing the performance gap compared to the cascade baseline (-0.14 BLEU).

We also observe improved decoding speed: AFS runs ∼1.37× faster than ASR-PT. Compared to the fixed-rate subsampling, AFS is slightly slower which we ascribe to the overhead introduced by the gating module. Surprisingly, Table 1 shows that the vanilla ST runs slower than ASR-PT (0.87×) while the cascade model is slightly faster (1.06×). By digging into the beam search algorithm, we discover that ASR pretraining shortens the number of steps in beam-decoding: 94 ASR-PT vs. 112

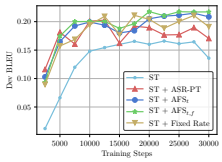


Figure 6: ST training curves (MuST-C En-De dev set). ASR pretraining significantly accelerates model convergence, and feature selection further stabilizes and improves training. $\lambda = 0.5, k = 6$.

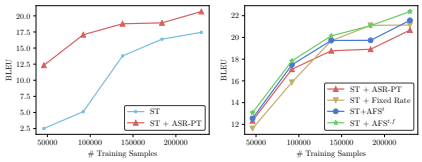


Figure 7: BLEU as a function of training data size on MuST-C En-De. We split the original training data into non-overlapped five subsets, and train different models with accumulated subsets. Results are reported on the test set. Note that we perform ASR pretraining on the original dataset. $\lambda = 0.5, k = 6$.

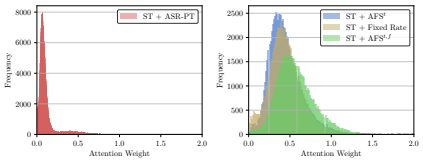
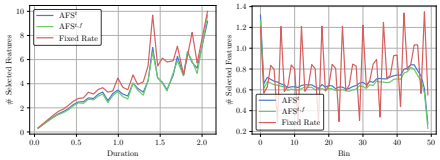


Figure 8: Histogram of the cross-attention weights received per ST encoder output on MuST-C En-De test set. For each instance, we collect attention weights averaged over different heads and decoder layers following Zhang et al. (2020). Larger weight indicates stronger impact of the encoder output on translation. Feature selection biases the distribution towards larger weights. $\lambda = 0.5, k = 6$.



(a) Duration Analysis (b) Position Analysis

Figure 9: The number of selected features vs. word duration (left) and position (right) on MuST-C En-De test set. For word duration, we align the audio and its transcription by Montreal Forced Aligner (McAuliffe et al., 2017), and collect each words' duration and its corresponding retained feature number. For position, we uniformly split each input into 50 pieces, and count the average number of retained features in each piece. $\lambda = 0.5, k = 6$.

vanilla ST (on average). The speedup brought by cascading is due to the smaller English vocabulary size compared to the German vocabulary when processing audio inputs.

5.2 Why (Adaptive) Feature Selection?

Apart from the benefits in translation quality, we go deeper to study other potential impacts of (adaptive) feature selection. We begin with inspecting training curves. Figure 6 shows that ASR pretraining improves model convergence; feature selection makes training more stable. Compared to other models, the curve of ST w/ AFS is much smoother, suggesting its better regularization effect.

We then investigate the effect of training data size, and show the results in Figure 7. Overall, we do not observe higher data efficiency by feature selection on low-resource settings. But instead, our results suggest that feature selection delivers larger performance improvement when more training data is available. With respect to data efficiency, ASR pretraining seems to be more promising (Figure 7, left) (Bansal et al., 2019; Stoian et al., 2020). Compared to AFS, the fixed-rate subsampling suffers more from small-scale training: it yields worse performance than ASR-PT when data size ≤ 100K, highlighting better generalization of AFS.

In addition to model performance, we also look into the ST model itself, and focus on the cross-attention weights. Figure 8 visualize the attention value distribution, where ST models with feature selection noticeably shift the distribution towards larger weights. This suggests that each ST encoder output exerts greater influence on the translation. By removing redundant and noisy speech features, feature selection eases the learning of the ST encoder, and also enhances its connection strength with the ST decoder. This helps bridge the modality gap between speech and text translation. Although fixed-rate subsampling also delivers distribution shift similar to AFS, its inferior ST performance compared to AFS corroborates the better quality of adaptively selected features.

AFS vs. Fixed Rate We compare these two approaches by analyzing the number of retained features with respect to word duration and temporal position. Results in Figure 9a show that the underlying pattern behind these two methods is similar: words with longer duration correspond to more speech features. However, when it comes to temporal position, Figure 9b illustrates their difference:

	Model	De	Es	Fr	It	Nl	Pt	Ro	Ru
BLEU↑	(Di Gangi et al., 2019)	17.30	20.80	26.90	16.80	18.80	20.10	16.50	10.50
	Transformer + ASR-PT*	21.77	26.41	31.56	21.46	25.22	26.84	20.53	14.31
	ST	17.44	23.85	28.43	19.54	21.23	22.55	17.66	12.10
	ST + ASR-PT	20.67	25.96	32.24	20.84	23.27	24.83	19.94	13.96
	Cascade	22.52	27.92	34.53	24.02	26.74	27.57	22.61	16.13
	ST + AFS^{t}	21.57	26.78	33.34	23.08	24.68	26.13	21.73	15.10
	ST + AFS^{t,f}	22.38	27.04	33.43	23.35	25.05	26.55	21.87	14.92
Temporal Sparsity Rate	ST + AFS^{t}	84.4%	84.5%	83.2%	84.9%	84.4%	84.4%	84.7%	84.2%
	ST + AFS^{t,f}	85.1%	84.5%	84.7%	84.9%	83.5%	85.1%	84.8%	84.7%
Speedup ↑	ST + AFS^{t}	1.38×	1.35×	1.50×	1.34×	1.54×	1.43×	1.59×	1.31×
	ST + AFS^{t,f}	1.37×	1.34×	1.50×	1.39×	1.42×	1.26×	1.46×	1.37×

Table 2: Performance over 8 languages on MuST-C dataset. *: results reported by the ESPNet toolkit (Watanabe et al., 2018), where the hyperparameters of beam search are tuned for each dataset.

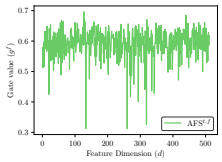


Figure 10: Illustration of feature gate $\mathbf{g}^f$ with $\lambda = 0.5$.

fixed-rate subsampling is context-independent, periodically picking up features; while AFS decides feature selection based on context information. The curve of AFS is more smooth, indicating that features kept by AFS are more uniformly distributed across different positions, ensuring the features' informativeness.

AFS^{t} vs. AFS^{t,f} Their only difference lies at the feature gate $\mathbf{g}^f$. We visualize this gate in Figure 10. Although this gate induces no sparsification, it offers AFS^{t,f} the capability of adjusting the weight of each neuron. In other words, AFS^{t,f} has more freedom in manipulating speech features.

5.3 Results on MuST-C and LibriSpeech

Table 2 and Table 3 list the results on MuST-C and LibriSpeech En-Fr, respectively.[5] Over all tasks, AFS^{t}/AFS^{t,f} substantially outperforms ASR-PT by 1.34/1.60 average BLEU, pruning out 84.5% temporal speech features on average and yielding an average decoding speedup of 1.45×. Our model narrows the gap against the cascade model to -0.8 average BLEU, where AFS surpasses Cascade on LibriSpeech En-Fr, without using KD (Liu et al., 2019a) and data augmentation (Wang et al., 2020).

[5]Unfortunately, comparing BLEU scores across papers is not recommended due to differences in tokenization and cases.

Model		BLEU↑
LSTM + PT + MTL (Bérard et al., 2018)		13.40
LSTM + PT (Watanabe et al., 2018)		16.68
Transformer + PT + KD (Liu et al., 2019a)		17.02
TCEN-LSTM + PT (Wang et al., 2019)		17.05
Transformer + ASR-PT (Wang et al., 2020)		17.66
ST		14.32
ST + ASR-PT		17.05
Cascade		18.27
ST + AFS^{t}	BLEU↑	18.33
ST + AFS^{t,f}		18.56
ST + AFS^{t}	Temporal	84.7%
ST + AFS^{t,f}	Sparsity Rate	83.5%
ST + AFS^{t}	Speedup↑	1.84×
ST + AFS^{t,f}		1.78×

Table 3: Performance on LibriSpeech En-Fr. *KD*: knowledge distillation. *MTL*: multi-task learning.

6 Conclusion and Future Work

In this paper, we propose adaptive feature selection for E2E ST to handle redundant and noisy speech signals. We insert AFS in-between the ST encoder and a pretrained, frozen ASR encoder to filter out uninformative features contributing little to ASR. We base AFS on $\mathcal{L}_0$DROP (Zhang et al., 2020), and extend it to modeling both temporal and feature dimensions. Results show that AFS stabilizes training, improves translation quality and accelerates decoding by ∼1.4× with an average temporal sparsity rate of ∼84%. AFS successfully narrow or even close the performance gap compared to cascading models.

In the future, we will work on adapting AFS to simultaneous speech translation.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 82–91, New Orleans, Louisiana. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Low-resource speech-to-text translation. In *Proc. Interspeech 2018*, pages 1298–1302.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 58–68, Minneapolis, Minnesota. Association for Computational Linguistics.

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Daniel Beck, Trevor Cohn, and Gholamreza Haffari. 2019. Neural speech translation using lattice transformations and graph networks. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 26–31, Hong Kong. Association for Computational Linguistics.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6224–6228. IEEE.

Alexandre Bérard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. 2016. Listen and translate: A proof of concept for end-to-end speech-to-text translation. In *NIPS Workshop on End-to-end Learning for Speech and Audio Processing*, Barcelona, Spain.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1766, Melbourne, Australia. Association for Computational Linguistics.

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 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2012–2017, Minneapolis, Minnesota. Association for Computational Linguistics.

Mattia A. Di Gangi, Matteo Negri, and Marco Turchi. 2019. Adapting Transformer to End-to-End Spoken Language Translation. In *Proc. Interspeech 2019*, pages 1133–1137.

Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. 2016. An attentional model for speech translation without transcription. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 949–959, San Diego, California. Association for Computational Linguistics.

Alex Graves, Santiago Fernández, and Faustino Gomez. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *In Proceedings of the International Conference on Machine Learning, ICML 2006*, pages 369–376.

Sathish Indurthi, Houjeung Han, Nikhil Kumar Lakumarapu, Beomseok Lee, Insoo Chung, Sangha Kim, and Chanwoo Kim. 2019. Data efficient direct speech-to-text translation with modality agnostic meta-learning. *arXiv preprint arXiv:1911.04283*.

Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu. 2019. Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7180–7184. IEEE.

Takatomo Kano, Sakriani Sakti, and Satoshi Nakamura. 2017. Structured-based curriculum learning for end-to-end english-japanese speech translation. In *Proc. Interspeech 2017*, pages 2630–2634.

S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, S. Watanabe, T. Yoshimura, and W. Zhang. 2019. A comparative study on transformer vs rnn in speech applications. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 449–456.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Ali Can Kocabiyikoglu, Laurent Besacier, and Olivier Kraif. 2018. Augmenting librispeech with French translations: A multimodal corpus for direct speech translation evaluation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Yuchen Liu, Hao Xiong, Jiajun Zhang, Zhongjun He, Hua Wu, Haifeng Wang, and Chengqing Zong. 2019a. End-to-End Speech Translation with Knowledge Distillation. In *Proc. Interspeech 2019*, pages 1128–1132.

Yuchen Liu, Jiajun Zhang, Hao Xiong, Long Zhou, Zhongjun He, Hua Wu, Haifeng Wang, and Chengqing Zong. 2019b. Synchronous speech recognition and speech-to-text translation with interactive decoding. *arXiv preprint arXiv:1912.07240*.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

Liang Lu, Xingxing Zhang, Kyunghyun Cho, and Steve Renals. 2015. A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Lambert Mathias and William Byrne. 2006. Statistical phrase-based speech translation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE.

Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. 2017. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Proc. Interspeech 2017*, pages 498–502.

R. Na, J. Hou, W. Guo, Y. Song, and L. Dai. 2019. Learning adaptive downsampling encoding for online end-to-end speech recognition. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 850–854.

Hermann Ney. 1999. Speech translation: Coupling of recognition and translation. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 1, pages 517–520. IEEE.

J. Niehues, R. Cattoni, S. Stüker, M. Negri, M. Turchi, E. Salesky, R. Sanabria, L. Barrault, L. Specia, and M. Federico. 2019. The iwslt 2019 evaluation campaign. In *Proceedings of the 16th International Workshop on Spoken Language Translation (IWSLT 2019)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Juan Pino, Liezl Puzon, Jiatao Gu, Xutai Ma, Arya D McCarthy, and Deepak Gopinath. 2019. Harnessing indirect training data for end-to-end automatic speech translation: Tricks of the trade. In *Proceedings of the 16th International Workshop on Spoken Language Translation (IWSLT)*.

Shirin Saleem, Szu-Chen (Stan) Jou, Stephan Vogel, and Tanja Schultz. 2004. Using word lattice information for a tighter coupling in speech translation systems. In *International Conference of Spoken Language Processing*.

Elizabeth Salesky, Matthias Sperber, and Alan W. Black. 2020. Exploring phoneme-level speech representations for end-to-end speech translation. In *Proceedings of the 2020 Annual Conference of the Association for Computational Linguistics*. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

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.

Mihaela C Stoian, Sameer Bansal, and Sharon Goldwater. 2020. Analyzing asr pretraining for low-resource speech-to-text translation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7909–7913. IEEE.

Tzu-Wei Sung, Jun-You Liu, Hung-yi Lee, and Lin-shan Lee. 2019. Towards end-to-end speech-to-text translation with two-pass decoding. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7175–7179. IEEE.

Yulia Tsvetkov, Florian Metze, and Chris Dyer. 2014. Augmenting translation models with simulated acoustic confusions for improved spoken language translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 616–625, Gothenburg, Sweden. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Chengyi Wang, Yu Wu, Shujie Liu, Zhenglu Yang, and Ming Zhou. 2019. Bridging the gap between pretraining and fine-tuning for end-to-end speech translation. *arXiv preprint arXiv:1909.07575*.

Chengyi Wang, Yu Wu, Shujie Liu, Ming Zhou, and Zhenglu Yang. 2020. Curriculum pre-training for end-to-end speech translation. *arXiv preprint arXiv:2004.10093*.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. 2018. Espnet: End-to-end speech processing toolkit. In *Proc. Interspeech 2018*, pages 2207–2211.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly translate foreign speech. In *Proc. Interspeech 2017*, pages 2625–2629.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2020. On sparsifying encoder outputs in sequence-to-sequence models. *arXiv preprint arXiv:2004.11854*.

Pei Zhang, Niyu Ge, Boxing Chen, and Kai Fan. 2019a. Lattice transformer for speech translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6475–6484, Florence, Italy. Association for Computational Linguistics.

Shucong Zhang, Erfan Loweimi, Yumo Xu, Peter Bell, and Steve Renals. 2019b. Trainable Dynamic Subsampling for End-to-End Speech Recognition. In *Proc. Interspeech 2019*, pages 1413–1417.

E Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation

Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation

Matthias Sperber[1], **Graham Neubig**[2], **Jan Niehues**[1], **Alex Waibel**[1,2]

[1]Karlsruhe Institute of Technology, Germany
[2]Carnegie Mellon University, USA
{first}.{last}@kit.edu, gneubig@cs.cmu.edu

Abstract

Speech translation has traditionally been approached through cascaded models consisting of a speech recognizer trained on a corpus of transcribed speech, and a machine translation system trained on parallel texts. Several recent works have shown the feasibility of collapsing the cascade into a single, direct model that can be trained in an end-to-end fashion on a corpus of translated speech. However, experiments are inconclusive on whether the cascade or the direct model is stronger, and have only been conducted under the unrealistic assumption that both are trained on equal amounts of data, ignoring other available speech recognition and machine translation corpora.

In this paper, we demonstrate that direct speech translation models require more data to perform well than cascaded models, and although they allow including auxiliary data through multi-task training, they are poor at exploiting such data, putting them at a severe disadvantage. As a remedy, we propose the use of end-to-end trainable models with two attention mechanisms, the first establishing source speech to source text alignments, the second modeling source to target text alignment. We show that such models naturally decompose into multi-task–trainable recognition and translation tasks and propose an *attention-passing* technique that alleviates error propagation issues in a previous formulation of a model with two attention stages. Our proposed model outperforms all examined baselines and is able to exploit auxiliary training data much more effectively than direct attentional models.

1 Introduction

Speech translation takes audio signals of speech as input and produces text translations as output. Although traditionally realized by cascading an automatic speech recognition (ASR) and a machine translation (MT) component, recent work has shown that it is feasible to use a single sequence-to-sequence model instead (Duong et al., 2016; Weiss et al., 2017; Bérard et al., 2018; Anastasopoulos and Chiang, 2018). An appealing property of such direct models is that we no longer suffer from propagation of errors, where the speech recognizer passes an erroneous source text to the machine translation component, potentially leading to compounding follow-up errors. Another advantage is the ability to train all model parameters jointly.

Despite these obvious advantages, two problems persist: (1) Reports on whether direct models outperform cascaded models (Fig. 1a,d) are inconclusive, with some work in favor of direct models (Weiss et al., 2017), some work in favor of cascaded models (Kano et al., 2017; Bérard et al., 2018), and one work in favor of direct models for two out of the three examined language pairs (Anastasopoulos and Chiang, 2018). (2) Cascaded and direct models have been compared under identical data situations, but this is an unrealistic assumption: In practice, cascaded models can be trained on much more abundant independent ASR and MT corpora, whereas end-to-end models require hard-to-acquire end-to-end corpora of speech utterances paired with textual translations.

Our first contribution is a closer investigation of these two issues. Regarding the question of whether direct models or cascaded models are generally stronger, we hypothesize that direct models require more data to work well, due to the more complex mapping between inputs (source speech) and outputs (target text). This would imply that direct models outperform cascades when enough data are available, but underperform in low-data scenarios. We conduct experiments and present empirical evidence in favor of this

Transactions of the Association for Computational Linguistics, vol. 7, pp. 313–325, 2019. Action Editor: Adam Lopez.
Submission batch: 12/2018; Revision batch: 2/2019; Published 6/2019.

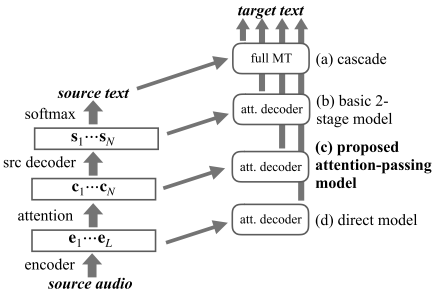


Figure 1: Conceptual diagrams for various speech translation approaches. *Cascade* (a) uses separate machine translation and speech recognition models. The *direct* model (d) is a standard attentional encoder-decoder model. The basic 2-stage model (b) uses two attention stages and passes source-text decoder states to the translation component. Our proposed *attention-passing* model (c) applies two attention stages, but passes context vectors to the translation component for improved robustness.

hypothesis. Next, for a more realistic comparison with regard to data conditions, we train a direct speech translation model using more auxiliary ASR and MT training data than end-to-end data. This can be implemented through multi-task training (Weiss et al., 2017; Bérard et al., 2018). Our results show that the auxiliary data are beneficial only to a limited extent, and that direct multi-task models are still heavily dependent on the end-to-end data.

As our second contribution, we apply a *two-stage* model (Tu et al., 2017; Kano et al., 2017) as an alternative solution to our problem, hoping that such models may overcome the data efficiency shortcoming of the direct model. Two-stage models consist of a first-stage attentional sequence-to-sequence model that performs speech recognition and then passes the decoder states as input to a second attentional model that performs translation (Fig. 1b). This architecture is closer to cascaded translation while maintaining end-to-end trainability. Introducing supervision from the source-side transcripts midway through the model creates inductive bias that guides the complex transformation between source speech and target text through a reasonable intermediate representation closely tied to the source text. The architecture has been proposed by Tu et al. (2017) to realize a reconstruction objective, and a similar model was also applied to speech translation (Kano et al., 2017) to ease trainability, although no experiments under varying data conditions have been conducted. We hypothesize that such a model may help to address the identified data efficiency issue: Unlike multi-task training for the direct model that trains auxiliary models on additional data but then discards many of the additionally learned parameters, the two-stage model uses all parameters of sub-models in the final end-to-end model. Empirical results confirm that the two-stage model is indeed successful at improving data efficiency, but suffers from some degradation in translation accuracy under high data conditions compared with the direct model. One reason for this degradation is that this model re-introduces the problem of error propagation, because the second stage of the model depends on the decoder states of the first model stage which often contain errors.

Our third contribution, therefore, is an *attention-passing* variant of the two-stage model that, rather than passing on possibly erroneous decoder states from the first to the second stage, passes on only the computed attention context vectors (Fig. 1c). We can view this approach as replacing the early decision on a source-side transcript by an early decision only on the *attention scores* needed to compute the same transcript, where the attention scores are expectedly more robust to errors in source text decoding. We explore several variants of this model and show that it outperforms both the direct model and the vanilla two-stage model, while maintaining the improved data efficiency of the latter. Through an analysis, we further observe a trade-off between sensitivity to error propagation and data efficiency.

2 Baseline Models

This section introduces two types of end-to-end trainable models for speech translation, along with a cascaded approach, which will serve as our baselines. All models are based on the attentional encoder-decoder architecture of Bahdanau et al. (2015) with character-level outputs, and use the architecture described in §2.1 as audio encoders. The end-to-end trainable models include a direct model and a two-stage model. Both are limited[1] by the fact that they can *only* be trained on end-to-end

[1]Prior work noted that in severe low-resource situations it may actually be easier to collect speech paired with translations than transcriptions (Duong et al., 2016). However, we focus on well-resourced languages for which ASR

data, which is much harder to obtain than ASR or MT data used to train traditional cascades.[2] §3 will introduce multi-task training as a way to overcome this limitation.

2.1 Audio Encoder

Sequence-to-sequence models can be adopted for audio inputs by directly feeding speech features (here, Mel filterbank features) instead of word embeddings as encoder inputs (Chorowski et al., 2015; Chan et al., 2016). Such an encoder transforms M feature vectors $\mathbf{x}_{1:M}$ into L encoded vectors $\mathbf{e}_{1:L}$, performing downsampling such that $L<M$. We use an encoder architecture that follows one of the variants described by Zhang et al. (2017): We stack two blocks, each consisting of a bidirectional long short-term memory (LSTM), a network-in-network (NiN) projection that downsamples by factor two, and batch normalization. After the second block, we add a final bidirectional LSTM layer. NiN denotes a simple linear projection applied at every time step, performing downsampling by concatenating pairs of adjacent projection inputs. Because of space constraints, we do not present detailed equations, but refer interested readers to Zhang et al. (2017) as well as to our provided code for details.

2.2 Direct Model

The sequence-to-sequence model with audio inputs outlined above can be trained as a direct speech translation model by using speech data as input and the corresponding translations as outputs. Such a model does not rely on intermediate ASR output and is therefore not subject to error propagation. However, the transformation from source speech inputs to target text outputs is much more complex than that of an ASR or MT system taken individually, which may cause the model to require more data to perform well.

To make matters precise, given L audio encoder states $\mathbf{e}_{1:L}$ computed by the audio encoder as described in §2.1, the direct model is computed as

$$\mathbf{s}_i = \text{LSTM}\left(\left[W_e y_{i-1}; \mathbf{c}_{i-1}\right], \mathbf{s}_{i-1}; \theta_{\text{lstm}}\right) \quad (1)$$

$$\mathbf{c}_i = \text{Attention}\left(\mathbf{s}_i, \mathbf{e}_{1:L}; \theta_{\text{att}}\right) \quad (2)$$

$$\tilde{\mathbf{s}}_i = \tanh\left(W_{\text{s}}\left[\mathbf{s}_i; \mathbf{c}_i\right] + \mathbf{b}_{\text{s}}\right) \quad (3)$$

$$p\left(y_i \mid y_{<i}, \mathbf{e}_{1:L}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_i; \theta_{\text{out}}\right). \quad (4)$$

Here, W_*, $\mathbf{b}_*$, and θ_* are the trainable parameters, y_i are output characters, and SoftmaxOut denotes an affine projection followed by a softmax operation. $\mathbf{s}_i$ are decoder states with $\mathbf{s}_0$ initialized to the last encoder state, and $\mathbf{c}_i$ are attentional context vectors with $\mathbf{c}_0$=$\mathbf{0}$. In Equation 2, we compute Attention$(\cdot)$=$\sum_{j=1}^{L}\alpha_{ij}\mathbf{e}_j$ with weights α_{ij} conditioned on $\mathbf{e}_j$ and $\mathbf{s}_i$, parameterized by θ_{att}, and normalized via a softmax operation.

2.3 Two-Stage Model

As an alternative to the direct model, the two-stage model uses a cascaded information flow while maintaining end-to-end trainability. Our main motivation for using this model is the potentially improved data efficiency when adding auxiliary ASR and MT training data (§3). This model is similar to the architecture first described by Tu et al. (2017). It combines two encoder-decoder models in a cascade-like fashion, with the decoder of the first stage and the encoder of the second stage being shared (Fig. 2). In other words, while a cascade would use the source-text outputs of the first stage as inputs into the second stage, in this model the second stage directly computes attentional context vectors over the decoder states of the first stage. The inputs of the two-stage model are speech frames, the outputs of the first stage are transcribed characters in the source language, and the outputs of the second stage are translated characters in the target language.

Again assuming L audio encoder states $\mathbf{e}_{1:L}$, the first stage outputs of length N are computed identically to equations 1–4, except that input feeding (conditioning the decoding step on the previous context vector) is not used in the first

and MT corpora exist and for which it is more realistic to obtain good speech translation accuracy.

[2] As a case in point, the largest available speech translation corpora (Post et al., 2013; Kocabiyikoglu et al., 2018) are an order of magnitude smaller than the largest speech recognition corpora (Cieri et al., 2004; Panayotov et al., 2015) ($\sim$ 200 hours vs 2000 hours) and several orders of magnitude smaller than the largest machine translation corpora, e.g., those provided by the Conference on Machine Translation (WMT).

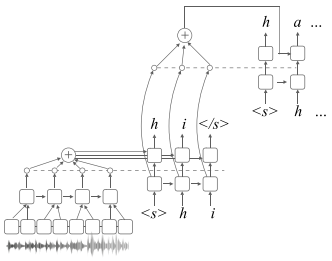


Figure 2: Basic two-stage model. Decoder states of the first stage double as encoder states for the second stage.

stage decoder to keep components compatible for multi-task training (§3.2):

$$\mathbf{s}_i^{\text{src}} = \text{LSTM}\left(W_{\text{e}}^{\text{src}} y_{i-1}^{\text{src}}, \mathbf{s}_{i-1}^{\text{src}}; \theta_{\text{lstm}}^{\text{src}}\right) \quad (5)$$

$$\mathbf{c}_i^{\text{src}} = \text{Attention}\left(\mathbf{s}_i^{\text{src}}, \mathbf{e}_{1:L}; \theta_{\text{att}}^{\text{src}}\right) \quad (6)$$

$$\tilde{\mathbf{s}}_i^{\text{src}} = \tanh\left(W_{\text{s}}^{\text{src}} \left[\mathbf{s}_i^{\text{src}}; \mathbf{c}_i^{\text{src}}\right] + \mathbf{b}_{\text{s}}^{\text{src}}\right) \quad (7)$$

$$p\left(y_i^{\text{src}} \mid y_{<i}, \mathbf{e}_{1:L}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_i^{\text{src}}; \theta_{\text{out}}^{\text{src}}\right) \quad (8)$$

Next, the second stage proceeds similarly but uses the stage 1 decoder states as input:

$$\mathbf{s}_j^{\text{trg}} = \text{LSTM}\left(\left[W_{\text{e}}^{\text{trg}} y_{i-1}^{\text{trg}}; \mathbf{c}_{j-1}^{\text{trg}}\right], \mathbf{s}_{j-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{trg}}\right) \quad (9)$$

$$\mathbf{c}_j^{\text{trg}} = \text{Attention}\left(\mathbf{s}_j^{\text{trg}}, \mathbf{s}_{1:N}^{\text{src}}; \theta_{\text{att}}^{\text{trg}}\right) \quad (10)$$

$$\tilde{\mathbf{s}}_j^{\text{trg}} = \tanh\left(W_{\text{s}}^{\text{trg}} \left[\mathbf{s}_j^{\text{trg}}; \mathbf{c}_j^{\text{trg}}\right] + \mathbf{b}_{\text{s}}^{\text{trg}}\right) \quad (11)$$

$$p\left(y_j^{\text{trg}} \mid y_{<j}, \mathbf{s}_{1:N}^{\text{src}}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_j^{\text{trg}}; \theta_{\text{out}}^{\text{trg}}\right) \quad (12)$$

2.4 Cascaded Model

We finally utilize a traditional cascaded model as a baseline, whose architecture is kept as similar to the above models as possible in order to facilitate meaningful comparisons. The cascade consists of an ASR component and an MT component, which are both attentional sequence-to-sequence models according to equations 1–4, trained on the appropriate data. The ASR component uses the acoustic encoder of §2.1, and the MT model uses a bidirectional LSTM with 2 layers as encoder.

3 Incorporating Auxiliary Data

The models described in §2.2 and §2.3 are trained only on speech utterances paired with translations (and transcripts in the case of §2.3), which is a severe limitation. To incorporate auxiliary ASR and MT data into the training, we make use of a multi-task training strategy. Such a strategy trains auxiliary ASR and MT models that share certain parameters with the main speech translation model. We implement multi-task training by drawing several minibatches, one minibatch for each task, and performing an update based on the accumulated gradients across tasks. Note that this results in a balanced contribution of each task.[3]

3.1 Multi-Task Training for the Direct Model

Multi-task training for direct speech translation models has previously been used by Weiss et al. (2017) and Bérard et al. (2018), although not for the purpose of adding additional training utterances that are not shown to the main speech translation task.[4] We distinguish five model components: a source speech encoder, a source text encoder (a two-layer bidirectional LSTM working on character level), a source text decoder, a target text decoder, and an attention mechanism that we opt to share across all tasks. There are four ways in which these components can be combined into a complete sequence-to-sequence model (see Figure 3), corresponding to the following four tasks:

ASR: Combines source speech encoder, general-purpose attention, source text decoder. This is similar to the auxiliary ASR task used by Weiss et al. (2017) and can be trained on common ASR data.

MT: Combines source text encoder, general-purpose-attention, target text decoder. The addition of an MT task has been mentioned by Bérard et al. (2018) and allows training on common MT data.

ST: Combines source speech encoder, general-purpose-attention, target text decoder. This is our main task and requires end-to-end data for training.

[3] We also experimented with a final fine-tuning phase on only the main task (Niehues and Cho, 2017), but discarded this strategy for lack of consistent gains.

[4] Note that Bansal et al. (2019) do experiment with additional speech recognition data, although, differently from our work, for purposes of cross-lingual transfer learning.

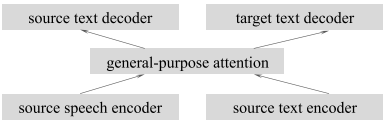


Figure 3: Direct multi-task model.

Auto-encoder (AE): Combines source text encoder, general-purpose attention, source text decoder. The AE task can be trained on monolingual corpora in the source language and may serve to tighten the coupling between components and potentially improves the parameters of the general-purpose attention model. We have observed slight improvements by adding the AE task in preliminary experiments and will therefore use it throughout this paper.

3.2 Multi-Task Training for the Two-Stage Model

Including an auxiliary ASR task is straightforward with the two-stage model by simply computing the cross-entropy loss with respect to the softmax output of the first stage, and dropping the second stage.

The auxiliary MT task computes only the second stage, replacing the inputs $\mathbf{s}^{\text{src}}_{1:N}$ by states $\mathbf{e}^{\text{asr}}_{1:N}$ computed as:

$$\mathbf{e}^{\text{asr}}_i = \text{LSTM}\left(W^{\text{src}}_{\text{e}} y^{\text{transcr}}_i, \mathbf{e}^{\text{src}}_{i-1}; \theta^{\text{src}}_{\text{lstm}}\right). \quad (13)$$

That is, instead of computing the second-stage inputs using the first stage, we compute these inputs through a conventional encoder that encodes the reference transcript $y^{\text{transcr}}_{1:N}$ and uses the same embeddings matrix and unidirectional LSTM as the first stage decoder. Note that there is no equivalent to the auxiliary auto-encoder task of the direct multi-task model here.

Why might this architecture help to make better use of auxiliary ASR and MT data? Note that in the direct model only roughly half of the model parameters are shared between the main task and the ASR task, and likewise for main and MT tasks (§3.1). Additional data would therefore only have a rather indirect impact on the main task. In contrast, in the two-stage model all parameters of the auxiliary tasks are shared with the main task and therefore have a more direct impact, potentially leading to better data efficiency.

Note that somewhat related to our multi-task strategy, Kano et al. (2017) have decomposed their two-stage model in a similar way to perform pretraining for the individual stages, although not with the goal of incorporating additional auxiliary data.

4 Attention-Passing Model

We have so far described a direct model that has the appealing property of avoiding error propagation in a principled way but that may not be particularly data-efficient, and have described a two-stage model that addresses the latter disadvantage. Unfortunately, the two-stage model re-introduces the error propagation problem into end-to-end modeling, because the second stage heavily depends on the potentially erroneous decoder states of the first stage. We therefore propose an improved *attention-passing* model in this section that is less impacted by error propagation issues.

4.1 Preventing Error Propagation

The main idea behind the attention-passing model is to not feed the erroneous first-stage decoder states to the second stage, but instead to pass on only the *context vectors* that summarize the relevant encoded audio at each decoding step. The first stage decoder is unfolded as usual by using discrete source-text representations, but the only information exposed to the translation stage are the per-timestep context vectors created as a by-product of the decoding. Figure 4 illustrates this idea. Intuitively, we expect this to help because we no longer make an early decision on the identity of the source-language text, but only on the corresponding attentions. This is motivated by our observation that speech recognition attentions are sufficiently robust against decoding errors (§5.7).

Formally, the first stage remains unchanged from equations 5–8. The context vectors $\mathbf{c}^{\text{src}}_i$ then form the input to the second stage:

$$\mathbf{x}^{\text{trg}}_i = \text{LSTM}\left(\mathbf{c}^{\text{src}}_i, \mathbf{x}^{\text{trg}}_{i-1}; \theta^{\text{src}}_{\text{lstm}}\right) \quad (14)$$

$$\mathbf{s}^{\text{trg}}_j = \text{LSTM}\left(\left[W^{\text{trg}}_{\text{e}} y^{\text{trg}}_{i-1}; \mathbf{c}^{\text{trg}}_{j-1}\right], \mathbf{s}^{\text{trg}}_{j-1}; \theta^{\text{trg}}_{\text{lstm}}\right) \quad (15)$$

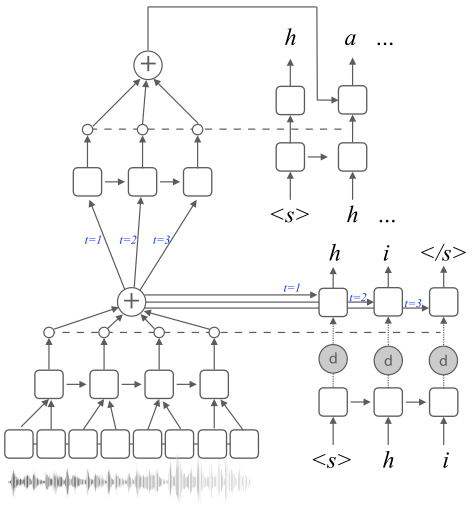


Figure 4: Attention-passing model.

$$\mathbf{c}_j^{\text{trg}} = \text{Attention}\left(\mathbf{s}_j^{\text{trg}}, \mathbf{x}_{1:N}^{\text{trg}}; \theta_{\text{att}}^{\text{trg}}\right) \quad (16)$$

$$\tilde{\mathbf{s}}_j^{\text{trg}} = \tanh\left(W_{\text{s}}^{\text{trg}}\left[\mathbf{s}_j^{\text{trg}}; \mathbf{c}_j^{\text{src}}\right] + \mathbf{b}_{\text{s}}^{\text{trg}}\right) \quad (17)$$

$$p\left(y_j^{\text{trg}} \mid y_{<j}, \mathbf{s}_{1:N}^{\text{src}}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_j^{\text{trg}}; \theta_{\text{out}}^{\text{trg}}\right) \quad (18)$$

4.2 Decoder State Drop-Out

Along with the modifications described in §4.1, we introduce an additional block drop-out operation (Ammar et al., 2016) on the decoder states, replacing equation 7 by

$$\tilde{\mathbf{s}}_i^{\text{src}} = \tanh\left(W_{\text{s}}^{\text{src}}\left[\text{BDrop}\left\{\mathbf{s}_i^{\text{src}}\right\}; \mathbf{c}_i^{\text{src}}\right] + \mathbf{b}_{\text{s}}^{\text{src}}\right).$$

The block drop-out operation, denoted as BDrop, replaces the whole vector by zero with a certain probability (here: 0.5). This results in the context vectors $\mathbf{c}_i^{\text{src}}$ becoming the only information available to the output layer whenever the decoder states are dropped out. The motivation for this is to force the model to maximize the informativeness of the context vectors, which are later relied upon as sole inputs to the second stage.

4.3 Multi-Task Training

Similar to the basic two-stage model, the attention-passing model as a whole is trained on speech-transcript-translation triplets, but can be decomposed into two sub-models that correspond to ASR and MT tasks. In fact, the ASR task is unchanged with the exception of the new block dropout operation. The MT task is obtained by replacing equation 14 by $\mathbf{x}_i^{\text{trg}} = \text{LSTM}\left(W_e y_i^{\text{src}}, \mathbf{x}_{i-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{src}}\right)$—that is, by using the transcript character embeddings as inputs instead of the context vectors used when training the main task. Note that the LSTMs in equations 5 and 14 are shared in order to have a match between stage 1 decoder and stage 2 encoder as with the basic model.

4.4 Cross Connections

As a further extension to the attention-passing model of §4.1, we can introduce cross connections that concatenate the dropped-out first stage hidden decoder states to the second-stage inputs encoder. This causes equation 14 to be replaced by

$$\mathbf{x}_i^{\text{trg}} = \quad (19)$$
$$\text{LSTM}\left(\text{Affine}\left[\mathbf{c}_i^{\text{src}}; \text{BDrop}\left\{\mathbf{s}_i^{\text{src}}\right\}\right], \mathbf{x}_{i-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{src}}\right)$$

This extension moves the model closer to the basic two-stage model, and the inclusion of the context vectors and the block drop-out operation on the hidden decoder states ensures that the second stage decoder does not rely too strongly on the first stage outputs.

4.5 Additional Loss

We further experiment with introducing an additional loss aimed at making the LSTM inputs between first stage decoder and second stage encoder RNN more similarly. Recall that in our attention-passing model, both RNNs share parameters (equations 5 and 14), so that similar inputs at both times is desirable. The loss is defined as follows:

$$\mathcal{L}_{\text{add}} = ||\mathbf{c}_i^{\text{src}} - W_e y_i^{\text{src}}||_2.$$

If combined with the cross connections (§4.4), the formula is adjusted to $\mathcal{L}_{\text{add}} = ||\text{Affine}\left[\mathbf{c}_i^{\text{src}}; \text{BDrop}\left\{\mathbf{s}_i^{\text{src}}\right\}\right] - W_e y_i^{\text{src}}||_2$. We did not find it beneficial to apply a scaling factor when adding this loss to the main cross-entropy loss in our experiments.

5 Experiments

We conduct experiments on the Fisher and Callhome Spanish–English Speech Translation

Corpus (Post et al., 2013), a corpus of Spanish telephone conversations that includes audio, transcriptions, and translations into English. We use the Fisher portion that consists of telephone conversations between strangers. The training data size is 138,819 sentences, corresponding to 162 hours of speech. ASR word error rates on this dataset are usually relatively high because of the spontaneous speaking style and challenging acoustics. From a translation viewpoint, the data can be considered as relatively easy with regard to both the topical domain and particular language pair.

Our implementation is based on the `xnmt` toolkit.[5] We use the speech recognition recipe as a starting point, which has previously been shown to achieve competitive ASR results (Neubig et al., 2018). [6]

The vocabulary consists of the common characters appearing in English and Spanish, apostrophe, whitespace, and special start-of-sequence and unknown-character tokens. The same vocabulary is used on both encoder (for the MT auxiliary task) and decoder sides. We set the batch size dynamically depending on the input sequence size such that the average batch size is 24 sentences. We use Adam (Kingma and Ba, 2014) with initial learning rate of 0.0005, decayed by 0.5 when the validation BLEU score did not improve over 10 check points initially and 5 check points after the first decay. We initialize attention-passing models using weights from a basic two-stage model trained on the same data.

Following Weiss et al. (2017), we lowercase texts and remove punctuation. As speech features, we use 40-dimensional Mel filter bank features with per-speaker mean and variance normalization. We exclude a small number of utterances longer than 1500 frames from training to avoid running out of memory. The encoder-decoder attention is MLP-based, and the decoder uses a single LSTM layer.[7] Source text encoders for the multi-task direct model and the cascaded models use two LSTM layers. The number of hidden units is 128 for the encoder-decoder attention MLP, 64 for target character embeddings, 256 for the encoder LSTMs in each direction, and 512 elsewhere. The model uses variational recurrent dropout with probability 0.3 and target character dropout with probability 0.1 (Gal and Ghahramani, 2016). We apply label smoothing (Szegedy et al., 2016) and fix the target embedding norm to 1 (Nguyen and Chiang, 2018). We use beam search with beam size 15 and polynomial length normalization with exponent 1.5.[8]

All BLEU scores are computed on Fisher/Test against 4 references.

Training sents.	Cascade	Direct model
139k	32.45	**35.30**
69k	**26.52**	24.68
35k	**16.84**	14.91
14k	**6.59**	6.08

Table 1: BLEU scores (4 references) on the Fisher/ Test for various amounts of training data. The direct (multi-task) model performs best in the full data condition, but the cascaded model is best in all reduced conditions.

5.1 Cascaded vs. Direct Models

We first wish to shed light on the question of whether cascaded or direct models can be expected to perform better. This question has been investigated previously (Weiss et al., 2017; Kano et al., 2017; Bérard et al., 2018; Anastasopoulos and Chiang, 2018), but with contradictory findings. We hypothesize that the increased complexity of the direct mapping from speech to translation increases the data requirements of such models. Table 1 compares the direct multi-task model (§3.1) against a cascaded model with identical architecture to the respective ASR and MT submodels of the multi-task model. The direct model is trained with multi-task training on the auxiliary ASR, MT, and AE tasks on the same data that outperformed single-task training considerably in preliminary experiments. As can be seen, the direct model outperforms the traditional cascaded setup only when both are trained on the full data, but not when using only parts of the training data. This

[5] `https://github.com/neulab/xnmt`.

[6] Code and configuration files can be found at `http://www.msperber.com/research/tacl-attention-passing/`.

[7] Weiss et al. (2017) report improvements from deeper decoders, but we encountered stability issues and therefore restricted the decoder to a single layer.

[8] For two-stage and attention-passing models, we apply beam search only for the second stage decoder. We do not use the two-phase beam search of Tu et al. (2017) because of its prohibitive memory requirements.

Model	BLEU
Cascade	32.45
Direct	35.30
Basic two-stage	34.36
APM	35.31
APM + cross connections	36.51
APM + cross conn. + additional loss	**36.70**
Best APM w/o block dropout	36.04

Table 2: Results for cascaded and multi-task models under full training data conditions.

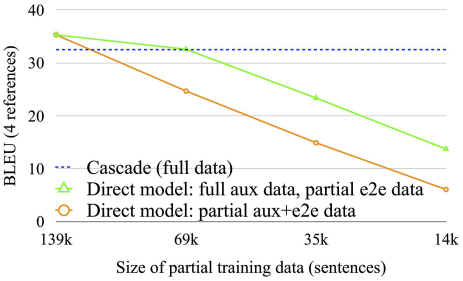


Figure 5: Data efficiency for the direct (multi-task) model, compared against cascade on full auxiliary data.

provides evidence in favor of our hypothesis and indicates that direct end-to-end models should be expected to perform strongly only in a case where enough training data is available.

5.2 Two-Stage Models

Next, we investigate the performance of the two-stage models, for both the basic variant (§3.2) and our proposed attention-passing model (§4). Again, all models are trained in a multi-task fashion by including auxiliary ASR and MT tasks based on the same data. Table 2 shows the results. The basic two-stage model performs in between the direct and cascaded models from §5.1. APM, the attention-passing model of §4.1 which is designed to circumvent the negative effects of error propagation, outperforms the basic variant and performs similarly to the direct model. The APM extensions (§4.4, §4.5) further improved the results, with the best model outperforming the direct model by 1.40 BLEU points and the basic two-stage model by 2.34 BLEU points absolute. The last row in the table confirms that the block dropout operation contributed to the gains: Removing it led to a drop by 0.66 BLEU points.

5.3 Data Efficiency: Direct Model

Having established results in favor of our proposed model on the full data, we now examine the data efficiency of the different models. Our experimental strategy is to compare model performance (1) when trained on the full data, (2) when trained on partial data, and (3) when trained on partial speech-to-translation data but full auxiliary (ASR+MT) data.[9]

Figure 5 shows the results, comparing the cascaded model against the direct model trained under conditions (1), (2), and (3).[10] Unsurprisingly, the performance of the direct model trained on partial data declines sharply as the amount of data is reduced. Adding auxiliary data through multi-task training improves performance in all cases. For instance, in the case of 69k speech-to-translation instances, adding the full auxiliary data helps to reach the accuracy of the cascaded model. However, this is already somewhat disappointing because the end-to-end data, which is not available to the cascaded model, no longer yields an advantage. Moreover, reducing the end-to-end data further reveals that multi-task training is not able to close the gap to the cascade. In the scenario with 35k end-to-end instances and full auxiliary data, the direct model underperforms the cascade by 9.14 BLEU points (32.50 vs. 23.36), despite being trained on *more* data. The unsatisfactory data efficiency in this controlled ablation study strongly indicates that the direct model will also fall behind a cascade that is trained on large amounts of external data. This claim is verified in §5.5.

5.4 Data Efficiency: Two-Stage Models

We showed that the direct model is poor at integrating auxiliary data and heavily depends on sufficient amounts of end-to-end training data.

[9] An alternative experimental strategy is to train on the full data and then add auxiliary data from other domains to the training. We pursue this strategy in §5.5 as a more realistic scenario, but point out several problems that lead us to not use this as our main approach: Adding external auxiliary data (1) leads to side-effects through domain mismatch and (2) severely limits the number of experiments that we can conduct because of the considerably increased training time.

[10] Note that the above hyper-parameters were selected for best full-data performance and are not re-tuned here.

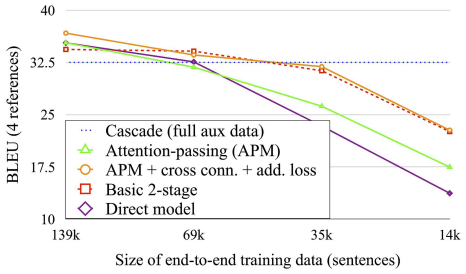


Figure 6: Data efficiency across model types. All models use full auxiliary data through multi-task training.

How do two-stage models behave with regard to this data efficiency issue? Figure 6 shows that both the basic two-stage model and the best APM perform reasonably well even when having seen much less end-to-end data. We can explain this by noticing that these models can be naturally decomposed into an ASR sub-model and an MT sub-model, while the direct model needs to add auxiliary sub-models to support multi-task training. Interestingly, the attention-passing model without cross-connections does better than the direct model with regard to data efficiency, but falls behind the basic and best proposed two-stage models. This indicates that access to ASR labels in some form contributes to favorable data efficiency of speech translation models.

5.5 Adding External Data

Our approach for evaluating data efficiency so far has been to assume that end-to-end data are available for only a subset of the available auxiliary data. In practice, we can often train ASR and MT tasks on abundant external data. We therefore run experiments in which we use the full Fisher training data for all tasks as before, and add OpenSubtitle[11] data for the auxiliary MT task. We clean and normalize the Spanish–English OpenSubtitle 2018 data (Lison and Tiedemann, 2016) to be consistent with the employed Fisher training data by lowercasing and removing punctuation. We apply a basic length filter and obtain 61 million sentences. During training, we include the same number of sentences from in-domain and out-of-domain MT tasks in each minibatch in order to prevent degradation due to domain mismatch.

[11] http://www.opensubtitles.org/.

Model	Fisher	Fisher+OpenSub
Cascade	32.45	34.58 (+6.2% rel.)
Direct model	35.30	36.45 (+3.2% rel.)
Basic two-stage	34.36	36.91 (+6.9% rel.)
Best APM	36.70	38.81 (+5.4% rel.)

Table 3: Adding auxiliary OpenSubtitles MT data to the training. The two-stage models benefit much more strongly than the direct model, with our proposed model yielding the strongest overall results.

Our models converged before a full pass over the OpenSubtitle data, but needed between two and three times more steps than the in-domain model to converge.

Table 3 shows that all models were able to benefit from the added data. However, when examining the relative gains we can see that both the cascaded model and the models with two attention stages benefitted about twice as much from the external data as the direct model. In fact, the basic two-stage model now slightly surpasses the direct model, and the best APM is ahead of the basic two-stage model by almost the same absolute difference as before (2.36 BLEU points). The superior relative gains show that our findings from §5.3 and §5.4, namely, that two-stage models are much more efficient at exploiting auxiliary training data, generalizes to the setting in which large amounts of out-of-domain data are added to the MT task. Out-of-domain data are often much easier to obtain, and we can therefore conclude that the proposed approach is preferable in many practically relevant situations. Because these experiments are very expensive to conduct, we leave experiments with external ASR data for future work.

5.6 Error Propagation

To better understand the impact of error propagation, we analyze how improved or degraded ASR labels impact the translation results. This experiment is applicable to APM, the two-stage model and the cascade, but not to the direct model which does not compute intermediate ASR outputs. We analyze three different settings: using the standard decoded ASR labels, replacing these labels with the gold labels, or artificially degrading the decoded labels by randomly introducing 10% of substitution, insertion, and deletion noise (Sperber

Labels	Gold	Decod.	Perturbed
Cascade	58.15 (+44%)	32.45	25.67 (-26%)
B2S	56.60 (+39%)	34.36	28.81 (-19%)
APM	40.70 (+13%)	35.31	31.96 (-10%)
+ cross	58.29 (+37%)	36.70	30.48 (-20%)

Table 4: Effect of altering the ASR labels for different models as a measure for robustness against error propagation. We compare results for the cascade, the basic two-stage model (B2S), and APM without and with cross connections. Percentages are relative to the results for unaltered (decoded) ASR labels.

et al., 2017). Intuitively, models that suffer from error propagation issues are expected to rely most heavily on these intermediate labels and would therefore be most impacted by both degraded and improved labels.

Table 4 shows the results. Unsurprisingly, the cascade responds most strongly to improved or degraded noise, confirming that it is severely impacted by error propagation. The APM, which does not directly expose the labels to the translation sub-model, is much less impacted. However, the impact is still more significant than perhaps expected, suggesting that improved attention models that are more robust to decoding errors (Chorowski et al., 2015; Tjandra et al., 2017) may serve to further improve our model in the future. Note that the APM benefits poorly from gold ASR labels, which is expected because gold labels only improve the ASR alignments and by extension the passed context vectors, but these are quite robust against decoding errors in the first place.

The basic two-stage model is impacted significantly, although less strongly than the cascade, in line with our claim that such models are subject to error propagation despite being end-to-end trainable. Note that it falls behind the cascade for gold labels, despite both models being seemingly identical under this condition. This can be explained by the cascaded model's use of beam search and greater number of encoder layers.

Somewhat contrary to our expectations, APM with cross connections appears equally subject to error propagation despite the block dropout on these connections, displaying the same accuracy gains across the three different settings. This suggests future explorations toward model variants with an even better trade-off between overall accuracy, data efficiency, and amount of degradation due to error propagation.

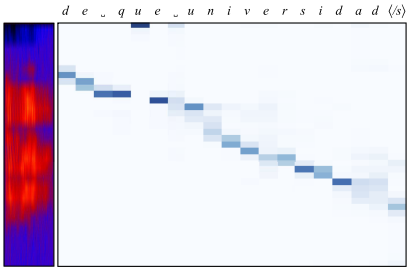


Figure 7: ASR attentions when force-decoding the oracle transcripts.

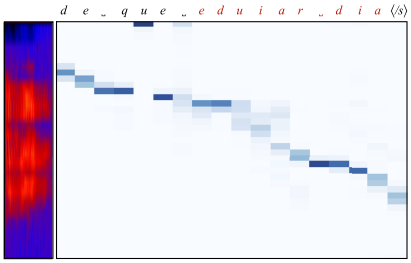


Figure 8: ASR attentions after regular decoding.

5.7 Robustness of ASR Attentions

The attention-passing model was motivated by the assumption that attention scores are relatively robust against recognition errors. We perform a qualitative analysis to validate this assumption. Figure 7 shows the first-stage attention matrix when force-decoding the reference transcript, and Figure 8 shows the same for regular decoding, which for this utterance produced significant errors. Despite the errors, we can see that the attention matrices are very similar. We manually inspected the first 100 test attention matrices and confirm that this behavior occurs very consistently. Further quantitative evidence is given in §5.6, which showed that the attention-passing model is more resistent to error propagation than the other models.

6 Prior Work

Model architectures similar to what we have referred to as the basic two-stage model have first been used by Tu et al. (2017) for a reconstruction

task, where the first stage performs translation and the second stage attempts to reconstruct the original inputs based on the outputs of the first stage. A second variant of a similar architecture are Xia et al. (2017)'s deliberation networks, where the second stage refines or polishes the outputs of the first stage. For our purposes, the first stage performs speech recognition, a natural intermediate representation for the speech translation task, corresponding to the second stage output. Toshniwal et al. (2017) explore a different way of lower-level supervision during training of an attentional speech recognizer by jointly training an auxiliary phoneme recognizer based on a lower layer in the acoustic encoder. Similarly to the discussed multi-task direct model, this approach discards many of the learned parameters when used on the main task and consequently may also suffer from data efficiency issues.

Direct end-to-end speech translation models were first used by Duong et al. (2016), although the authors did not actually evaluate translation performance. Weiss et al. (2017) extended this model into a multi-task model and report excellent translation results. Our baselines do not match their results, despite considerable efforts. We note that other research groups have encountered similar replicability issues (Bansal et al., 2018), explanations include the lack of a large GPU cluster to perform ASGD training, as well as to explore an ideal number of training schedules and other hyper-parameter settings. Bérard et al. (2018) explored the translation of audio books with direct models and report reasonable results, but do not outperform a cascaded baseline. Kano et al. (2017) have first used a basic two-stage model for speech translation. They use a pretraining strategy for the individual sub-models, related to our multi-task approach, but do not attempt to integrate auxiliary data. Moreover, the authors only evaluated the translation of synthesized speech, which greatly simplifies training and may not lead to generalizable conclusions, as indicated by the fact that they were actually able to outperform a translation model that used the gold transcripts as input. Anastasopoulos and Chiang (2018) conducted experiments on low-resource speech translation and used a triangle model that can be seen as a combination of a direct model and a two-stage model, but is not easily trainable in a multi-task fashion. It is therefore not a suitable choice for exploiting auxiliary data in order to compete with cascaded models under well-resourced data conditions. Finally, contemporaneous work explores transferring knowledge from high-resource to low-resource languages (Bansal et al., 2019).

7 Conclusion

This work explored *direct* and *two-stage* models for speech translation with the aim of obtaining models that are strong, not only in favorable data conditions, but are also efficient at exploiting auxiliary data. We started by demonstrating that direct models do outperform cascaded models, but only when enough data is available, shedding light on inconclusive results from prior work. We further showed that these models are poor at exploiting auxiliary data, making them a poor choice in realistic situations. We were motivated to use two-stage models by their ability to overcome this shortcoming of the direct models, and found that two-stage models are in fact more data-efficient, but suffer from error propagation issues. We addressed this by introducing a novel attention-passing model that alleviates error propagation issues, as well as several model variants. The best proposed model outperforms all other tested models and is much more data efficient than the direct model, allowing this model to compete with cascaded models even under realistic assumptions with auxiliary data available. Analysis showed that there seems to be a trade-off between data efficiency and error propagation. Avenues for future work include testing better ASR attention models; adding other types of external data such as ASR data, unlabeled speech, or monolingual texts; and exploring further model variants.

Acknowledgments

We thank Adam Lopez, Stefan Constantin, and the anonymous reviewers for their helpful comments. The work leading to these results has received funding from the European Union under grant agreement no. 825460.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith.

2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics (TACL)*, 4:431–444.

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Representation Learning (ICLR)*. San Diego, CA.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Low-resource speech-to-text translation. In *Annual Conference of the International Speech Communication Association (InterSpeech)*.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP)*.

Jan K. Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 577–585.

Christopher Cieri, David Miller, and Kevin Walker. 2004. The Fisher corpus: A resource for the next generations of speech-to-text. In *Language Resources and Evaluation (LREC)*, pages 69–71.

Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. 2016. An attentional model for speech translation without transcription. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 949–959, San Diego, CA.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Neural Information Processing Systems Conference (NIPS)*, pages 1019–1027, Barcelona.

Takatomo Kano, Sakriani Sakti, and Satoshi Nakamura. 2017. Structured-based curriculum Learning for end-to-end English-Japanese Speech translation. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 2630–2634.

Diederik P. Kingma and Jimmy L. Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, Banff.

Ali Can Kocabiyikoglu, Laurent Besacier, and Olivier Kraif. 2018. Augmenting Librispeech with French translations: A multimodal corpus for direct speech translation evaluation. In *Language Resources and Evaluation (LREC)*, Miyazaki, Japan.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Conference on Language Resources and Evaluation (LREC)*, pages 923–929. Portorož.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, John Hewitt, Rachid Riad, and Liming Wang. 2018. XNMT: The eXtensible Neural Machine Translation toolkit. In *Conference of the Association for Machine Translation in the Americas (AMTA) Open Source Software Showcase*, Boston, MA.

Toan Q. Nguyen and David Chiang. 2018. Improving lexical choice in neural machine translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, LA.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Conference on Machine Translation (WMT)*, pages 80–89. Copenhagen.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. Brisbane.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. Improved speech-to-text translation with the Fisher and Callhome Spanish–English speech translation corpus. In *International Workshop on Spoken Language Translation (IWSLT)*, Heidelberg.

Matthias Sperber, Jan Niehues, and Alex Waibel. 2017. Toward robust neural machine translation for noisy input sequences. In *International Workshop on Spoken Language Translation (IWSLT)*, Tokyo.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, NV.

Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2017. Local monotonic attention mechanism for end-to-end speech recognition. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 431–440.

Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. 2017. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, Stockholm.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Conference on Artificial Intelligence (AAAI)*.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly transcribe foreign speech. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, Stockholm.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: sequence generation. In *Neural Information Processing Systems Conference (NIPS)*, Long Beach, CA.

Yu Zhang, William Chan, and Navdeep Jaitly. 2017. Very deep convolutional Networks for end-to-end speech recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.