ON-TRAC Consortium for End-to-End and Simultaneous Speech Translation Challenge Tasks at IWSLT 2020

Maha Elbayad, Ha Nguyen, Fethi Bougares, Natalia Tomashenko, Antoine Caubrière, Benjamin Lecouteux, Yannick Estève, Laurent Besacier

To cite this version:


HAL Id: hal-02895893
https://hal.archives-ouvertes.fr/hal-02895893
Submitted on 10 Jul 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
ON-TRAC Consortium for End-to-End and Simultaneous Speech Translation Challenge Tasks at IWSLT 2020

Maha Elbayad\textsuperscript{1,4}, Ha Nguyen\textsuperscript{1,2}, Fethi Bougares\textsuperscript{3}, Natalia Tomashenko\textsuperscript{2}, Antoine Caubrière\textsuperscript{3}, Benjamin Lecouteux\textsuperscript{1}, Yannick Estève\textsuperscript{2}, Laurent Besacier\textsuperscript{1}

\textsuperscript{1}LIG - Université Grenoble Alpes, France
\textsuperscript{2}LIA - Avignon Université, France
\textsuperscript{3}LIUM - Le Mans Université, France
\textsuperscript{4}Inria - Grenoble, France

Abstract

This paper describes the ON-TRAC Consortium translation systems developed for two challenge tracks featured in the Evaluation Campaign of IWSLT 2020, offline speech translation and simultaneous speech translation. ON-TRAC Consortium is composed of researchers from three French academic laboratories: LIA (Avignon Université), LIG (Université Grenoble Alpes), and LIUM (Le Mans Université). Attention-based encoder-decoder models, trained end-to-end, were used for our submissions to the offline speech translation track. Our contributions focused on data augmentation and ensembling of multiple models. In the simultaneous speech translation track, we build on Transformer-based wait-k models for the text-to-text subtask. For speech-to-text simultaneous translation, we attach a wait-k MT system to a hybrid ASR system. We propose an algorithm to control the latency of the ASR+MT cascade and achieve a good latency-quality trade-off on both subtasks.

1 Introduction

While cascaded speech-to-text translation (AST) systems (combining source language speech recognition (ASR) and source-to-target text translation (MT)) remain state-of-the-art, recent works have attempted to build end-to-end AST with very encouraging results (Bérard et al., 2016; Weiss et al., 2017; Bérard et al., 2018; Jia et al., 2019; Sperber et al., 2019). This year, IWSLT 2020 offline translation track attempts to evaluate if end-to-end AST will close the gap with cascaded AST for the English-to-German language pair.

Another increasingly popular topic is simultaneous (online) machine translation which consists in generating an output hypothesis before the entire input sequence is available. To deal with this low latency constraint, several strategies were proposed for neural machine translation with input text (Ma et al., 2019; Arivazhagan et al., 2019; Ma et al., 2020). Only a few works investigated low latency neural speech translation (Niehues et al., 2018). This year, IWSLT 2020 simultaneous translation track attempts to stimulate research on this challenging task. This paper describes the ON-TRAC consortium automatic speech translation (AST) systems for the IWSLT 2020 Shared Task (Ansari et al., 2020). ON-TRAC Consortium is composed of researchers from three French academic laboratories: LIA (Avignon Université), LIG (Université Grenoble Alpes), and LIUM (Le Mans Université).

We participated in:

- IWSLT 2020 offline translation track with end-to-end models for the English-German language pair,
- IWSLT 2020 simultaneous translation track with a cascade of an ASR system trained using Kaldi (Povey et al., 2011) and an online MT system with wait-k policies (Dalvi et al., 2018; Ma et al., 2019).

This paper goes as follows: we review the systems built for the offline speech translation track in §2. Then, we present our approaches to the simultaneous track for both text-to-text and speech-to-text subtasks in §3. We ultimately conclude this work in §4.

2 Offline Speech Translation Track

In this work, we developed several end-to-end speech translation systems, using a similar architecture as last year (Nguyen et al., 2019) and adapting it for translating English speech into German text (En-De). All the systems were developed using the
Table 1: Statistics of training and evaluation data. The statistics of tst2019 and tst2020 are measured on the segmented version provided by IWSLT2020 organizers.

<table>
<thead>
<tr>
<th>Name</th>
<th>#segments</th>
<th>Total length (in hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MuST-C train</td>
<td>229.703</td>
<td>400</td>
</tr>
<tr>
<td>MuST-C dev</td>
<td>1.423</td>
<td>2.5</td>
</tr>
<tr>
<td>MuST-C tst-COMMON</td>
<td>2.641</td>
<td>4.1</td>
</tr>
<tr>
<td>MuST-C tst-HE</td>
<td>600</td>
<td>1.2</td>
</tr>
<tr>
<td>Europarl train</td>
<td>32.628</td>
<td>77</td>
</tr>
<tr>
<td>Europarl dev</td>
<td>1.320</td>
<td>3.1</td>
</tr>
<tr>
<td>How2 synthetic</td>
<td>176.564</td>
<td>285.5</td>
</tr>
<tr>
<td>tst2019</td>
<td>2.813</td>
<td>5.1</td>
</tr>
<tr>
<td>tst2020</td>
<td>2.263</td>
<td>4.1</td>
</tr>
</tbody>
</table>

Figure 1: Architecture of the speech encoder: a stack of two VGG blocks followed by 5 BLSTM layers.

2.1 Data and pre-processing

Data. We relied on MuST-C (Di Gangi et al., 2019) English-to-German (hereafter called MuST-C original), and Europarl (Iranzo-Sánchez et al., 2020) English-to-German as our main corpora. Besides, we automatically translated (into German) the English transcription of MuST-C and How2 (Sanabria et al., 2018) in order to augment training data. This resulted in two synthetic corpora, which are called MuST-C synthetic and How2 synthetic respectively. The statistics of these corpora, along with the provided evaluation data, can be found in Table 1. We experimented with different ways of combining these corpora. The details of these experiments are presented later in this section.

Speech features and data augmentation. 80-dimensional Mel filter-bank features, concatenated with 3-dimensional pitch features\(^1\) are extracted from windows of 25ms with a frame shift of 10ms. We computed mean and variance normalization on these raw features of the training set, then applied it on all the data. Beside speed perturbation with factors of 0.9, 1.0, and 1.1, SpecAugment (Park et al., 2019) is applied to the training data (Ko et al., 2015). All three SpecAugment methods were used, including time warping \((W = 5)\), frequency masking \((F = 30)\), and time masking \((T = 40)\).

Text preprocessing. The same as last year, we normalize punctuation, and tokenize all the German text using Moses.\(^2\) Texts are case-sensitive and contain punctuation. Moreover, the texts of the MuST-C corpus contain multiple non speech events (i.e. ‘Laughter’, ‘Applause’ etc.). All these marks are removed from the texts before training our models. This results in a vocabulary of 201 characters. We find that some of these characters should not appear in the German text, for example, 音, 首, 道, etc. Therefore, we manually exclude them from the vocabulary. In the end, we settle with an output vocabulary of 182 characters.

2.2 Architecture

We reuse our last year attention-based encoder-decoder architecture. As illustrated in Figure 1, the encoder has two VGG-like (Simonyan and Zisserman, 2015) CNN blocks followed by five stacked 1024-dimensional BLSTM layers. Each VGG block is a stack of two 2D-convolution layers followed by a 2D-maxpooling layer aiming to reduce both time \((T)\) and frequency \((D)\) dimensions of the input speech features by a factor of 2. After these two VGG blocks, input speech features’ shape is transformed from \((T \times D)\) to \((T/4 \times D/4)\). We used Bahdanau’s attention mechanism (Bahdanau et al., 2015) in all our experiments. The decoder is a stack of two LSTM layers 1024-dimensional memory cells. We would like to men-

---

\(^1\)Pitch-features are computed using the Kaldi toolkit (Povey et al., 2011) and consist of the following values (Ghahremani et al., 2014): (1) probability of voicing (POV-feature), (2) pitch-feature and (3) delta-pitch feature. For details, see http://kaldi-asr.org/doc/process-kaldi-pitch-feats_8cc.html

\(^2\)http://www.statmt.org/moses/
<table>
<thead>
<tr>
<th>No.</th>
<th>Experiment</th>
<th>MuST-C tst-COMMON</th>
<th>MuST-C tst-HE</th>
<th>tst2015 (iwslt seg)</th>
<th>tst2015 (ASR seg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MuST-C original + EuroParl</td>
<td>20.18</td>
<td>19.82</td>
<td>12.59</td>
<td>14.85</td>
</tr>
<tr>
<td>2</td>
<td>MuST-C original + Europarl + How2 synthetic</td>
<td>20.51</td>
<td>20.10</td>
<td>12.10</td>
<td>13.66</td>
</tr>
<tr>
<td>3*</td>
<td>MuST-C original + Europarl + How2 synthetic</td>
<td>23.55</td>
<td>22.35</td>
<td>13.00</td>
<td>15.30</td>
</tr>
<tr>
<td>4*</td>
<td>MuST-C original + Europarl + How2 synthetic + MuST-C synthetic</td>
<td>22.75</td>
<td>21.31</td>
<td>14.00</td>
<td>16.45</td>
</tr>
<tr>
<td>5*</td>
<td>Finetune 3* on MuST-C original</td>
<td>23.60</td>
<td>22.26</td>
<td>13.71</td>
<td>15.30</td>
</tr>
<tr>
<td>6*</td>
<td>Finetune 3* on MuST-C original+ MuST-C synthetic</td>
<td>23.64</td>
<td>22.23</td>
<td>13.67</td>
<td>15.29</td>
</tr>
<tr>
<td>7</td>
<td>Ensemble (1 to 6)</td>
<td>25.22</td>
<td>23.80</td>
<td>15.20</td>
<td>16.53</td>
</tr>
</tbody>
</table>

Table 2: Detokenized case-sensitive BLEU scores for different experiments - * represents experiments that apply SpecAugment.

<table>
<thead>
<tr>
<th>Model</th>
<th>iwslt seg</th>
<th>ASR seg</th>
</tr>
</thead>
<tbody>
<tr>
<td>3*</td>
<td>constrastive5</td>
<td>constrastive3</td>
</tr>
<tr>
<td>4*</td>
<td>constrastive4</td>
<td>constrastive2</td>
</tr>
<tr>
<td>Ensemble</td>
<td>constrastive1</td>
<td>primary</td>
</tr>
</tbody>
</table>

Table 3: The ranking of out submitted systems. Model 3* and 4* are respectively corresponding to No.3* and No.4* of Table 2.

Hyperparameters’ details. All of our models are trained in maximum 20 epochs, with early stopping after 3 epochs if the accuracy on the development set does not improve. Dropout is set to 0.3 on the encoder part, and Adadelta is chosen as our optimizer. During decoding time, the beam size is set to 10. We prevent the models from generating too long sentences by setting a \( \text{maxlenratio} = 1.0 \). All our end-to-end models are similar in terms of architecture. They are different mainly in the following aspects: (1) training corpus; (2) type of tokenization units; (3) fine-tuning and pretraining strategies. Description of different models and evaluation results are given in Section 2.4.

2.3 Speech segmentation

Two types of segmentation of evaluation and development data were used for experiments and submitted systems: segmentation provided by the IWSLT organizers and automatic segmentation based on the output of an ASR system.

The ASR system, used to obtain automatic segmentation, was trained with the Kaldi speech recognition toolkit (Povey et al., 2011). An acoustic model was trained using the TED-LIUM 3 corpus (Hernandez et al., 2018). This ASR system produces recognized words with timecodes (start time and duration for each word). Then we form the speech segments based on this output following the rules: (1) if silence duration between two words is longer than a given threshold \( \Theta = 0.65 \) seconds, we split the audio file; (2) if the number of words in the current speech segment exceeds 40, then \( \Theta \) is reduced to 0.15 seconds in order to avoid too long segments. These thresholds have been optimised to get segment duration distribution.

\[ \text{maxlenratio} = \frac{\text{maximum_output_length}}{\text{encoder_hidden_state_length}} \]

\(^{5}\)All systems use 182 output caracter tokens except system 1 which has 201

\(^{6}\)The off-limit TED talks from IWSLT-2019 were excluded from the training subset
tion in the development and evaluation data that is similar to the one observed in the training data. It will be shown in next subsection that this ASR segmentation improves results over the provided segmentation when the latter is noisy (see experimental results on iwslt/tst2015).

2.4 Experiments and results

After witnessing the benefit of merging different corpora from our submission last year (Nguyen et al., 2019), we continue exploring different combinations of corpora in this submission. As shown in the first two rows of Table 2, merging How2 synthetic with the baseline (MuST-C original + Europarl) does not bring significant improvement. It is noticeable that this pool is worse than the baseline on both tst2015 (iwslt seg) and tst2015 (ASR seg). However, we find that applying data augmentation (SpecAugment) on this same combination helps outperform the baseline on every investigated testset, most significantly on MuST-C tst-COMMON, and MuST-C tst-HE. Therefore, SpecAugment is consistently applied to all the experiments that follow. Adding MuST-C synthetic to this pool surprisingly decreases BLEU scores on both MuST-C testsets, while significantly increases the scores on both tst2015 (iwslt seg) and tst2015 (ASR seg). Not being able to investigate further on this matter due to time constraint, instead of fine-tuning 4*, we decided to fine-tune 3*, which performs reasonably well among all the testsets, on MuST-C original and MuST-C original+synthetic. We witness that the impact of fine tuning is very limited. One can also see once again that adding MuST-C synthetic does not make much difference. Finally, the last row of the table shows the results of ensembling all six models at decoding time. It is clear from the table that ensembling yields the best BLEU scores across all the testsets.

2.5 Overview of systems submitted

Two conclusions that can be drawn from Table 2 are (1) ensembling all six models is the most promising among all presented models, (2) our own segmentation (tst2015 ASR segmentation) is better than the default one. Therefore, we choose as our primary submission the translations of the ASR segmentations generated by the ensemble of all six models. Model 3* and 4* (Table 2) are also used to translate our contrastive submission runs, whose ranks are shown in Table 3. The official results for all our submitted systems can be found in Table 4. They confirm that our segmentation approach proposed is beneficial.

3 Simultaneous Speech Translation Track

In this section, we describe our submission to the Simultaneous Speech Translation (SST) track. Our pipeline consists of an automatic speech recognition (ASR) system followed by an online machine translation (MT) system. We first define our online ASR and MT models in §3.1 and §3.2 respectively. Then, we outline in §3.3 how we arrange the two systems for the speech-to-text subtask. We detail our experimental setup and report our results on the text-to-text subtask in §3.4 and on the speech-to-text in §3.5.
3.1 Online ASR

Our ASR system is a hybrid HMM/DNN system trained with lattice-free MMI (Povey et al., 2016), using the Kaldi speech recognition toolkit (Povey et al., 2011). The acoustic model (AM) topology consists of a Time Delay Neural Network (TDNN) followed by a stack of 16 factorized TDNNs (Povey et al., 2018). The acoustic feature vector is a concatenation of 40-dimensional MFCCs without cepstral truncation (MFCC-40) and 100-dimensional i-vectors for speaker adaptation (Dehak et al., 2010). Audio samples were randomly perturbed in speed and amplitude during the training process. This approach is commonly called audio augmentation and is known to be beneficial for speech recognition (Ko et al., 2015).

Online decoding with Kaldi. The online ASR system decodes under a set of rules to decide when to stop decoding and output a transcription. An endpoint is detected if either of the following conditions is satisfied:

(a) After \( t \) seconds of silence even if nothing was decoded.

(b) After \( t \) seconds of silence after decoding something, if the final-state was reached with \( \text{cost}_{\text{relative}} < c \).

(c) After \( t \) seconds of silence after decoding something, even if no final-state was reached.

(d) After the utterance is \( t \) seconds long regardless of anything else.

Each rule has an independent characteristic time \( t \) and condition (b) can be duplicated with different times and thresholds \((t, c)\). The value of \( \text{cost}_{\text{relative}} \) reflects the quality of the output, it is \textit{null} if a final-state of the decoding graph had the best cost at the final frame, and \textit{infinite} if no final-state was active.

3.2 Online MT

Our MT systems are Transformer-based (Vaswani et al., 2017) wait-\( k \) decoders with unidirectional encoders. Wait-\( k \) decoding starts by reading \( k \) source tokens, then alternates between reading and writing a single token at a time, until the source is depleted, or the target generation is terminated. With a source-target pair \((x, y)\), the number of source tokens read when decoding \( y_t \) following a \textit{wait-}\( k \) policy is \( z^k_t = \min(k + t - 1, |x|) \). To stop leaking signal from future source tokens, the energies of the encoder-decoder multihead-attention are masked to only include the \( z_t \) tokens read so far.

Unlike Transformer wait-\( k \) models introduced in Ma et al. (2019) where the source is processed with a bidirectional encoder, we opt for a unidirectional encoding of the source. In fact, this change alleviates the cost of re-encoding the source sequence after each read operation. Contrary to offline task, where bidirectional encoders are superior, unidirectional encoder achieve better quality-lagging trade-offs in online MT.

Ma et al. (2019) optimize their models with maximum likelihood estimation w.r.t. a single wait-\( k \) decoding path \( z^k \):

\[
\log p(y | x, z^k) = \sum_{t=1}^{|y|} \log p_0(y_t | y_{<t}, x_{\leq z^k_t}). \tag{1}
\]

Instead of optimizing a single decoding path, we jointly optimize across multiple wait-\( k \) paths. The additional loss terms provide a richer training signal, and potentially yield models that could perform well under different lagging constraints. Formally, we consider an exhaustive set of wait-\( k \) paths and in each training epoch we encode the source sequence then uniformly sample a path to decode with. As such, we optimize:

\[
Z = \{z^k | k \in \{1, \ldots, |x|\}\}, \tag{2}
\]

\[
\mathbb{E}_{z \sim Z} [\log p(y | x, z)] \approx \sum_{z \sim Z} \log p_0(y | x, z). \tag{3}
\]

We will refer to this training with \textit{multi-path}.

3.3 Cascaded ASR+MT

For speech-to-text online translation we pair an ASR system with our online MT system and decode following the algorithm described in Algorithm 1. In this setup, the lagging is controlled by the endpointing of the ASR system. The online MT system follows the lead of the ASR and translates prefix-to-prefix. Since the MT system is not trained to detect end of segments and can only halt the translation by emitting \(</s>\), we constrain it to decode \( \alpha | x^\text{ast} | + \beta \) tokens, where \( x^\text{ast} \) is the partial transcription and \((\alpha, \beta)\) two hyper-parameters.

Along with the hyper-parameters of the ASR’s endpointing rules, we tune \((\alpha, \beta)\) on a development set to achieve good latency-quality trade-offs.

3.4 Text-to-text translation subtask

Training MT. We train our online MT systems on English-to-German MuST-C (Di Gangi et al., 2019)
Algorithm 1 ASR+MT decoding algorithm

**Input:** source audio blocks $x$.

**Output:** translation hypothesis $y$.

**Initialization:** action = READ, $z = 0$, $t = 1$, $x_{\text{asr}} = ()$, $y = (<s>)$

**Hyper-parameters** $sz, \alpha, \beta$.

while $y_t \neq <s>$ do

while action = READ $\land z < |x|$ do

Read $sz$ elements from $x$. $z += sz$

Feed the new audio blocks to the ASR system.

if Endpoint detected $\lor z = |x|$ then

Output transcription and append it to $x_{\text{asr}}$.

action = WRITE

end if

end while

if $|y| < \alpha |x_{\text{asr}}| + \beta$ then

Given $y$ and $x_{\text{asr}}$, predict the next token $y_{t+1}$

$t += 1$

else

action = READ

end if

end while

---

Table 5: Parallel training data for the MT systems.

<table>
<thead>
<tr>
<th>Pairs</th>
<th>English words</th>
<th>German words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europarl</td>
<td>1,730K</td>
<td>43.7M</td>
</tr>
<tr>
<td>Common Crawl</td>
<td>1,543K</td>
<td>31.0M</td>
</tr>
<tr>
<td>News Commentary</td>
<td>320K</td>
<td>7.0M</td>
</tr>
<tr>
<td>MuST-C</td>
<td>214K</td>
<td>3.9M</td>
</tr>
</tbody>
</table>

---

and WMT’19 data, namely, Europarl (Koehn, 2005), News Commentary (Tiedemann, 2012) and Common Crawl (Smith et al., 2013). We remove pairs with a length-ratio exceeding 1.3 from Common Crawl and pairs exceeding a length-ratio of 1.5 from the rest. We develop on MuST-C dev and report results on MuST-C tst-COMMON. For open-vocabulary translation, we use SentencePiece (Kudo and Richardson, 2018) to segment the bi-texts with byte pair encoding (Sennrich et al., 2016). This results in a joint vocabulary of 32K types. Details of the training data are provided in Table 5.

We train Transformer big architectures and tie the embeddings of the encoder with the decoder’s input and output embeddings. We optimize our models with label-smoothed maximum likelihood (Szegedy et al., 2016) with a smoothing rate $\epsilon = 0.1$. The parameters are updated using Adam (Kingma and Ba, 2015) ($\beta_1, \beta_2 = 0.9, 0.98$) with a learning rate that follows an inverse square-root schedule. We train for a total of 50K updates and evaluate with the check-pointed weights corresponding to the lowest (best) loss on the development set. Our models are implemented with Fairseq (Ott et al., 2019). We generate translation hypotheses with greedy decoding and evaluate the latency-quality trade-off by measuring case-sensitive detokenized BLEU (Papineni et al., 2002) and word-level Average Lagging (AL) (Ma et al., 2019).

Results. We show in Figure 2 the performance of our systems on the test set (MuST tst-COMMON) measured with the provided evaluation server. We denote with $k_{\text{train}}=\infty$ a unidirectional model trained for wait-until-end decoding i.e. reading the full source before writing the target. We evaluate four wait-$k$ systems each trained with a value of $k_{\text{train}}$ in $\{5, 7, 9, \infty\}$ and decoded with $k_{\text{eval}}$ ranging from 2 to 11. We then ensemble the aforementioned wait-$k$ models and evaluate a multi-path model that jointly optimizes a large set of wait-$k$ paths. The results demonstrate that multi-path is competitive with wait-$k$ without the need to select which path to optimize (some values of $k$, e.g. 5, underperform in comparison). Ensembling the wait-$k$ models gives a boost of 1.43 BLEU points on average.

---

Figure 2: [Text-to-Text] Latency-quality trade-offs evaluated on MuST-C tst-COMMON with greedy decoding. Offline systems have an AL of 18.55 words. The red vertical bars correspond to the AL evaluation thresholds.
Table 6: Corpora used for the acoustic model.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>#hours</th>
<th>#words</th>
<th>#speakers</th>
</tr>
</thead>
<tbody>
<tr>
<td>TED-LIUM 3</td>
<td>452</td>
<td>5.05M</td>
<td>2,028</td>
</tr>
<tr>
<td>How2</td>
<td>365</td>
<td>3.31M</td>
<td>13,147</td>
</tr>
<tr>
<td>Europarl</td>
<td>94</td>
<td>0.75M</td>
<td>171</td>
</tr>
</tbody>
</table>

Table 7: WERs for the ASR system with offline and online decoding (AL=5s for online)

<table>
<thead>
<tr>
<th>Decoding</th>
<th>Corpus</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline</td>
<td>TED-LIUM 3 dev</td>
<td>7.65</td>
</tr>
<tr>
<td>Offline</td>
<td>TED-LIUM 3 test</td>
<td>7.84</td>
</tr>
<tr>
<td>Offline</td>
<td>MuST-C tst-COMMON</td>
<td>14.2</td>
</tr>
<tr>
<td>Online</td>
<td>MuST-C tst-COMMON</td>
<td>16.3</td>
</tr>
</tbody>
</table>

3.5 Speech-to-text translation subtask

Training ASR. We train our system following the *tedlium* recipe while adapting it for the IWSLT task. The TDNN layers have a hidden dimension of 1536 with a linear bottleneck dimension of 160 in the factorized layers. The i-vector extractor is trained on all acoustic data (speech perturbed + speech) using a 10s window. The acoustic training data includes TED-LIUM 3, How2 and Europarl. These corpora are detailed in Table 6 and represent about 900 hours of audio.

As a language model, we use the 4-grams small model provided with TED-LIUM 3. The vocabulary size is 152K, with 1.2 million of 2-grams, 622K 3-grams and 70K 4-grams.

The final system is tuned on TED-LIUM 3 dev and tested with TED-LIUM 3 test and MuST-C tst-COMMON. Results are shown in Table 7.

Training MT. To train the MT system for the ASR+MT cascade we process source-side data (English) to match transcriptions of the ASR. This consists of lower-casing, removing punctuation and converting numbers into letters. For this task we use two distinct English and German vocabularies of 32K BPE tokens each. We train Transformer big architectures with tied input-output decoder embeddings following the setup described in §3.4.

Results. Similar to the text-to-text subtask, we show our results in a plot of BLEU-to-AL in Figure 3. The systems are evaluated on the test via the provided evaluation server where MuST-

![Figure 3: [Speech-to-Text] Latency-quality trade-offs evaluated on MuST-C tst-COMMON with greedy decoding. Offline systems have an AL of 5806 ms. The red vertical bars correspond to the AL evaluation thresholds.](https://example.com/figure3.png)

C’s sentence-level aligned segments are streamed and decoded online and the lagging is measured in milliseconds. Note that in this task we use a single ASR model and only ensemble the MT wait-$k$ models. The cascade of an online ASR with wait-$k$ MT follows the same trends as the text-to-text models. In particular, multi-path is competitive with specialized wait-$k$ models and ensembling boosts the BLEU scores by 0.67 points on average.

4 Conclusion

This paper described the ON-TRAC consortium submission to the IWSLT 2020 shared task. In the continuity of our 2019 participation, we have submitted several end-to-end systems to the offline speech translation track. A significant part of our efforts was also dedicated to the new simultaneous translation track: we improved wait-$k$ models with unidirectional encoders and multi-path training and cascaded them with a strong ASR system. Future work will be dedicated to simultaneous speech translation using end-to-end models.

5 Acknowledgements

This work was funded by the French Research Agency (ANR) through the ON-TRAC project under contract number ANR-18-CE23-0021.
References


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


Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2019. Attention-passing models for robust and data-efficient end-to-end speech translation. TACL.


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


Ron J Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly transcribe foreign speech. In Proc. of INTERSPEECH.