Without Further Ado: Direct and Simultaneous Speech Translation by AppTek in 2021

Parnia Bahar[*], **Patrick Wilken**[*], **Mattia di Gangi, Evgeny Matusov**
Applications Technology (AppTek), Aachen, Germany
{pbahar,pwilken,mdigangi,ematusov}@apptek.com

Abstract

This paper describes the offline and simultaneous speech translation (ST) systems developed at AppTek for IWSLT 2021. Our offline ST submission includes the direct end-to-end system and the so-called posterior tight integrated model, which is akin to the cascade system but is trained in an end-to-end fashion, where all the cascaded modules are end-to-end models themselves. For simultaneous ST, we combine hybrid automatic speech recognition (ASR) with a machine translation (MT) approach whose translation policy decisions are learned from statistical word alignments. Compared to last year, we improve general quality and provide a wider range of quality/latency trade-offs, both due to a data augmentation method making the MT model robust to varying chunk sizes. Finally, we present a method for ASR output segmentation into sentences that introduces a minimal additional delay.

1 Introduction

In this paper, we describe the AppTek speech translation systems that participate in the offline and simultaneous tracks of the IWSLT 2021 evaluation campaign. This paper is organized as follows: In Section 2, we briefly address our data preparation. Section 3 describes our offline ST models followed by the experimental results in Section 3.6. For the offline end-to-end translation task, we train deep Transformer models that benefit from pretraining, data augmentation in the form of synthetic data and SpecAugment, as well as domain adaptation on TED talks. Motivated by Bahar et al. (2021), we also collapse the ASR and MT components into a *posterior model* which passes on the ASR posteriors as input to the MT model. This system is not considered a direct model since it is closer to the cascade system while being end-to-end trainable. Our simultaneous translation systems are covered in Section 4 with discussions on experimental results in Section 4.5. We resume the work on our streaming MT model developed for IWSLT 2020, which is based on splitting the stream of input words into chunks learned from statistical word alignment. Most notably, we can implement a flexible quality/latency trade-off by simulating different latencies at training time. We also meet this year's requirement to support unsegmented input by developing a neural sentence segmenter that splits the ASR output into suitable translation units, using a varying number of future words as context which minimizes the latency added by this component.

The experiments have been done using RASR (Wiesler et al., 2014), RETURNN (Zeyer et al., 2018a), and Sisyphus (Peter et al., 2018).

2 Data Preparation

2.1 Text Data

We participate in the constrained condition and divide the allowed bilingual training data into in-domain (the TED and MuST-C v2 corpora), clean (the NewsCommentary, Europarl, and WikiTitles corpora), and out-of-domain (the rest). The concatenation of MuST-C dev and IWSLT tst2014 is used as our dev set for all experiments. Our data preparation includes two main steps: data filtering and text conversion. We filter the out-of-domain data based on similarity to the in-domain data in the embedding space, reducing the size from 62.5M to 30.0M lines. For the details on data filtering, please refer to our last year's submission (Bahar et al., 2020).

For a tighter coupling between ASR and MT in the cascade system, we apply additional text normalization (TN) to the English side of the data. It lowercases the text, removes all punctuation

[*]equal contribution

Proceedings of the 18th International Conference on Spoken Language Translation, pages 52–63
Bangkok, Thailand (Online), August 5–6, 2021.

marks, expands abbreviations, and converts numbers, dates, and other digit-based entities into their spoken form. This year, our TN approach includes a language model to score multiple readings of digit-based entities and randomly samples one of the top-scoring readings. We refer to it as *ASR-like* preprocessing. The target text preserves the casing and punctuation such that the MT model is able to implicitly handle the mapping.

2.2 Speech Data

We use almost all allowed ASR data, including EuroParl, How2, MuST-C, TED-LIUM, LibriSpeech, Mozilla Common Voice, and IWSLT TED corpora in a total of approximately 2300 hours of speech. The MuST-C and IWSLT TED corpora are chosen to be the in-domain data. For the speech side of the data, 80-dimensional Mel-frequency cepstral coefficients (MFCC) features are extracted every 10ms. The English text is lower-cased, punctuation-free, and contains no transcriber tags.

3 Offline Speech Translation

3.1 Neural Machine Translation

Our MT model for the offline task is based on the *big* Transformer model (Vaswani et al., 2017). Both self-attentive encoder and decoder are composed of 6 stacked layers with 16 attention heads. The model size is 1024 with a ReLu layer equipped with 4096 nodes. The effective batch size has been increased by accumulating gradient with a factor of 8. Adam is used with an initial learning rate of 0.0003. The learning rate decays by a factor of 0.9 in case of 20 checkpoints of non-decreased dev set perplexity. Label smoothing (Pereyra et al., 2017) and dropout rates of 0.1 are used. SentencePiece (Kudo and Richardson, 2018) segmentation with a vocabulary size of 30K is applied to both the source and target sentences. We use a translation factor to predict the casing of the target words (Wilken and Matusov, 2019).

3.2 Automatic Speech Recognition

We have trained attention-based models (Bahdanau et al., 2015; Vaswani et al., 2017) for the offline task mainly following (Zeyer et al., 2019). To enable pre-training of the ST speech encoder with different architectures, we have trained two attention-based models. The first model is based on the 6-layer bidirectional long short-term memory (BiLSTM) (Hochreiter and Schmidhuber, 1997) in the encoder and 1-layer LSTM in the decoder with 1024 nodes each. Another model is based on the Transformer architecture with 12 layers of self-attentive encoder and decoder. The model size is chosen to be 512, while the feed-forward dimension is set to 2048. Both models employ layer-wise network construction (Zeyer et al., 2018b, 2019), SpecAugment (Park et al., 2019; Bahar et al., 2019) and the connectionist temporal classification (CTC) loss (Kim et al., 2017) during training. We further fine-tune the models on the in-domain data plus TED-LIUM. As shown in Table 1, the models obtain low word error rates without using an external language model (LM). These attention-based models also outperform the hybrid LSTM/HMM model used in our simultaneous speech translation task.

#	Model	TED tst2015	MuST-C tst-HE	MuST-C tst-COMMON
1	LSTM	6.9	7.5	9.7
2	Transformer	5.2	5.5	7.3

Table 1: ASR word error rate results in [%].

3.3 Speech Translation

The ST models are trained using all the speech translation English→German corpora i.e. IWSLT TED, MuST-C, EuroParl ST, and CoVoST. After removing the off-limits talks from the training data, we end up with 740k segments. 5k and 32k byte-pair-encoding (BPE) (Sennrich et al., 2016) is applied to the English and German texts, respectively. We have done the data processing as described in Section 2. We also fine-tune on the in-domain data, using a lower learning rate of 8×10^{-5}.

3.3.1 End-to-End Direct Model

Following our experiments from last year, the direct ST model uses a combination of an LSTM speech encoder and a big Transformer decoder. The speech LSTM encoder has 6 BiLSTM layers with 1024 nodes each. We refer to this model as *LSTM-enc Transformer-dec*. The model is initialized by the encoder of LSTM-based ASR (line 1 in Table 1) and the decoder of the MT Transformer model.

We also experiment with the pure Transformer model both in the encoder and decoder. The Transformer-based ST models follow the network configuration used for speech recognition in Section 3.2. In order to shrink the input speech sequence, we add 2 layers of BiLSTM interleaved with max-pooling on top of the feature vectors in the encoder with a total length reduction of 6.

Layer-wise construction is done including the de-

coder: we start with two layers in the encoder and decoder and double the number of layers after every 5 sub-epochs (approx. 7k batches). During this, we linearly increase the hidden dimensions from 256 to 512 nodes and disable dropout, afterwards it is set to 10%. Based on our initial observation, the layer-wise construction helps convergence, in particular for such deep architectures. The CTC loss is also applied on top of the speech encoder during training. The Transformer-based model uses 10 steps of warm-up with an initial learning rate of 8×10^{-4}. We set the minimum learning rate to be 50 times smaller than this initial value. We also apply SpecAugment without time warping to the input frame sequence to reduce overfitting.

3.3.2 Posterior Tight Integration

The *posterior* model is inspired by Bahar et al. (2021) where the cascade components, i.e. the end-to-end ASR and MT models, are collapsed into a single end-to-end trainable model. The idea is to benefit from all types of available data, i.e. the ASR, MT, and direct ST corpora, and optimize all parameters jointly. To this end, we concatenate the trained Transformer-based ASR and MT models, but instead of passing the one-hot vectors for the source words to the MT model, we pass on the word posteriors as a soft decision. We sharpen the source word distribution by an exponent γ and then renormalize the probabilities.

A value of $\gamma = 1$ produces the posterior distribution itself, while larger values produce a more peaked distribution (almost one-hot representation). To convey more uncertainty, we use $\gamma = 1.0$ in training and $\gamma = 1.5$ in decoding to pick the most plausible token. We further continue training of the end-to-end model using the direct ST parallel data as a fine-tuning step. The constraint is that the ASR output and the MT input must have the same vocabulary. Therefore, we need to train a new MT model with the appropriate English vocabulary with 5K subwords. The ASR model is trained with SpecAugment, the Adam optimizer with an initial learning rate of 1×10^{-4}, and gradient accumulation of 20 steps. We also apply 10 steps of learning warm-up. We employ beam search with a size of 12 to generate the best recognized word sequence and then pass it to MT with the corresponding word posterior vectors.

3.4 Synthetic Data

To provide more parallel audio-translation pairs, we translate the English side of the ASR data (Jia et al., 2019) with our MT model. From our initial observations, we exclude those corpora for which we have the ground-truth target reference and only add those with the missing German side. Therefore, combining the real ST data with the synthetic data generated from the How2, TED-LIUM, LibriSpeech corpora, and the English→French part of MuST-C (Gaido et al., 2020b), we obtain about 1.7M parallel utterances corresponding to 33M English and 37M German words, respectively.

3.5 Speech Segmentation

To comply with the offline evaluation conditions for a direct speech translation system with unsegmented input, we cannot rely on ASR source transcripts for sentence segmentation. Thus, we train a segmenter aiming to generate homogeneous utterances based on voice activity detection (VAD) and endpoint detection (EP). The segmenter is a frame-level acoustic model that applies a 5-layer feed-forward network and predicts 3530 class labels, including one silence and 3529 speech phonemes. It compares the average silence score of 10 successive frames with the average of the best phoneme score from each of those frames to classify silence segments. We wait for a minimum of 20 consecutive silence frames between two speech segments, whereas the minimal number of continuous speech frames to form a speech segment is 100.

Besides improving audio segmentation, following the idea by Gaido et al. (2020a), we fine-tune the direct model on automatically segmented data to increase its robustness against sub-optimal non-homogeneous utterances. To resegment the German reference translations, we first use the baseline direct model to generate the German MT output for the automatically determined English segments. Then, we align this MT output with the reference translations and resegment the latter using a variant of the edit distance algorithm implemented in the mwerSegmenter tool (Matusov et al., 2005).

3.6 Offline Speech Translation Results

The offline speech translation systems results in terms of BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) are presented in Table 2. The first group of results shows the text translation using the ASR-like processing. By comparing lines 1 and 3, we see an improvement in our MT develop-

#	System	TED tst2015 BLEU	TED tst2015 TER	MuST-C tst-HE BLEU	MuST-C tst-HE TER	MuST-C tst-COMMON BLEU	MuST-C tst-COMMON TER
Text MT (ASR-like source processing)							
1	AppTek 2020 submission	32.7	57.3	31.0	59.4	32.7	55.0
2	Transformer	32.4	57.8	30.8	60.0	33.1	54.5
3	+ fine-tuning	33.8	56.5	32.0	58.6	34.5	53.1
Cascaded ASR → MT							
4	AppTek 2020 submission (single)	30.9	61.0	29.3	61.7	30.0	58.0
5	AppTek 2020 submission (ensemble)	31.0	61.2	29.5	61.8	30.8	57.3
6	Transformer	31.4	59.3	30.1	60.7	31.4	56.9
7	**Posterior ASR → MT**	31.3	59.8	29.2	60.7	31.8	56.3
Direct ST							
8	AppTek 2020 submission (single)	26.4	64.7	24.7	66.9	29.4	58.6
9	LSTM-enc Transformer-dec	28.8	62.7	28.5	61.9	31.4	56.9
10	+ fine-tuning	28.3	64.8	27.8	62.8	33.1	55.6
11	+ resegmentation	28.0	63.3	27.3	62.8	31.1	57.1
12	Transformer	29.7	62.5	28.6	62.1	30.7	57.3
13	+ fine-tuning	29.5	62.7	28.6	62.4	31.0	57.1
Ensemble							
14	AppTek 2020 submission	28.0	63.2	27.4	63.3	30.4	57.8
15	lines 10(2x), 13(2x)	30.4	61.7	29.6	60.2	33.8	54.5

Table 2: Offline speech translation results measured in BLEU [%] and TER [%].

ment over time. As intended, fine-tuning using the in-domain data brings a significant gain. The MT model in line 3 and the Transformer-based ASR model from Table 1 make up the cascade system that outperforms our last year's submission, which ranked first on tst2020 using given segmentation. However, note that this year's cascade system is a single-shot try without careful model choice and fine-tuning. This result indicates fast progress of the speech translation task. As discussed in Section 3.3.2, passing ASR posteriors into the MT model, we further fine-tune the cascade model on the direct ST data. Therefore, the posterior model guarantees better or equal performance compared to the cascade system. Line 7 shows its competitiveness.

Regarding direct ST, we observe that the pure Transformer model (line 12) performs on par with the model with the LSTM-based encoder (line 9). Our main goal has been to employ different model choices to potentially capture different knowledge. These models already use synthetic data. The direct model with the LSTM encoder uses pretraining of components, while all pretraining experiments on the Transformer model degrade the translation quality. The reason might be partly attributed to the fact that we use a deep encoder (12 layers with size 512) and a large decoder (6 layers with model size 1024) with 3 to 6 layers of adaptors in between. The training deals with a more complex error propagation, causing a sub-optimal solution for the entire optimization problem. Again, fine-tuning helps both models in terms of the translation quality, in particular on tst-COMMON. Using the resegmeted MuST-C training data (line 11) leads to degradation; however, we have observed that this model generates less noise and fewer repeated phrases.

Finally, we ensemble 4 models (two checkpoints each from lines 10 and 13) constituting our primary submission for the 2021 IWSLT evaluation. In comparison to the 2020 submission, improvements of more than 2% in BLEU can be observed for both single and ensemble models.

4 Simultaneous Speech Translation

For the IWSLT 2021 simultaneous speech translation English→German tracks, we continue exploring our last year's alignment-based approach (Wilken et al., 2020), which uses a cascade of a streaming ASR system and an MT model.

4.1 Simultaneous MT Model

This section gives a short summary of (Wilken et al., 2020). Our simultaneous MT method is

based on the observation that latency in translation is mainly caused by word order differences between the source and target language. For example, an interpreter might have to wait for a verb at the end of a source sentence if it appears earlier in the target language. We therefore extract such word reordering information from statistical word alignments (generated using the Eflomal tool (Östling and Tiedemann, 2016)) by splitting sentence pairs into bilingual chunks such that word reordering happens only within chunk boundaries.

For the MT model, we use the LSTM-based attention model (Bahdanau et al., 2015). We make the following changes to support streaming decoding: **1.** We only use a forward encoder.[1] **2.** We add a binary softmax on top of the encoder trained to predict source chunk boundaries as extracted from the word alignment. Importantly, we add a delay D to the boundaries such that a detection at position j corresponds to a chunk boundary after position $j-D$. The future context available this way greatly increases the prediction accuracy. **3.** We add another softmax on top of the decoder to predict the target-side chunk boundaries. They are needed as a stopping criterion in beam search. **4.** We mask the attention energies such that when generating the k-th target chunk only the source words encoding in the chunks 1 to k can be accessed.

Inference happens by reading source words until a chunk boundary is predicted. Then the decoder is run using beam search until all hypotheses have predicted chunk end. During this, all source positions of the current sentence read so far are considered by the attention mechanism. Finally, the first best hypothesis is output and the process starts over.

4.2 Random Dropping of Chunk Boundaries

One evident limitation of our IWSLT 2020 systems (Bahar et al., 2020; Wilken et al., 2020) has been that we could not provide a range of different quality-latency trade-offs. This is because basing translation policy on hard word alignments leads to a fixed "operation point" whose average lagging is solely determined by the amount of differences in word order between the source and target language.

To overcome this, we make the observation that two subsequent chunks can be merged without violating the monotonicity constraint. This corresponds to skipping a chunk boundary at inference time and waiting for further context, at the cost of higher latency. The number of skipped chunk boundaries can be controlled by adjusting the threshold probability t_b which is used to make the source chunk boundary decision. In (Wilken et al., 2020), we have found that a threshold t_b different than 0.5 hurts MT performance because the decoder strongly adapts to the chunks seen in training, such that longer merged chunks are not translated well.

To solve this issue, we simulate higher detection thresholds t_b at training time by dropping each chunk boundary in the data randomly with a probability of p_{drop}. In fact, we create several duplicates of the training data applying different values of p_{drop} and shuffle them. This way the model learns to translate (merged) chunks with a wide variety of lengths, in the extreme case of $p_{\text{drop}} = 1$ even full sentences. This goes in the direction of general data augmentation by extracting prefix-pairs as done by Dalvi et al. (2018); Niehues et al. (2018). Importantly, we still train the source chunk prediction softmax on *all* boundaries to not distort the estimated probabilities.

4.3 Streaming ASR

As the ASR component, we use the same hybrid LSTM/HMM model (Bourlard and Wellekens, 1989) as in last year's submission (Bahar et al., 2020). The acoustic model consists of four BiLSTM layers with 512 units and is trained with the cross-entropy loss on triphone states. A count-based n-gram look-ahead language model is used. The streaming recognizer implements a version of chunked processing (Chen and Huo, 2016; Zeyer et al., 2016), where the acoustic model processes the input audio in fixed-length overlapping windows. The initial state of the backward LSTM is initialized for each window, while – as opposed to last year's system – the forward LSTM state is propagated among different windows. This state carry-over improves general recognition quality and allows us to use smaller window sizes W_{ASR} to achieve lower latencies.

4.4 Sentence Segmentation

This year's simultaneous MT track also requires supporting unsegmented input. To split the unsegmented source word stream into suitable translation units, we employ two different methods for the text and speech input condition.

[1] Although we experiment with a BiLSTM encoder in streaming, we are unable to achieve an improved performance.

4.4.1 Text Input

For the text-to-text translation task, the input contains punctuation marks that can be used for reliable sentence segmentation. We heuristically insert sentence ends whenever the following conditions are fulfilled:

1. the current token ends in sentence final punctuation (`. ? ! ;`), or punctuation plus quote (`." ?" !" ;"`), yet is not contained in a closed list of abbreviations (`Mrs. Dr. etc.`, ...);

2. the first character of the next word is not lowercased.

Those heuristics are sufficient to recover the original sentence boundaries of the MuST-C dev set with a precision of 96% and a recall of 82%, where most of the remaining differences can be attributed to lines with multiple sentences in the original segmentation. The described method uses one future word as context and therefore does not introduce additional delay into the system compared to awaiting a sentence end token. We enable this kind of sentence splitting also in the case of segmented input as we find that splitting lines with multiple sentences slightly increases translation performance.

4.4.2 Speech Input

For the speech-to-text translation task, sentence segmentation is a much harder problem. Our streaming ASR system does not require segmentation of the input; however, its output is lower-cased and punctuation-free text.

In the literature, the problem of segmenting ASR output into sentences has been approached using count-based language models (Stolcke and Shriberg, 1996), conditional random fields (Liu et al., 2005), and other classical models. Recently, recurrent neural networks have been applied, either in the form of language models (Wang et al., 2016) or sequence labeling (Iranzo-Sánchez et al., 2020). These methods either are meant for offline segmentation or require a fixed context of future words, thus increasing the overall latency of the system.

Wang et al. (2019) predict sentence boundaries with a various number of future words as context within the same model, allowing for dynamic segmentation decisions at inference time depending on the necessary context. We adopt the proposed model, which is a 3-layer LSTM with a hidden size of 512, generating softmax distributions over the labels $y^{(k)}$, $k \in \{0, \ldots, m\}$, where m is the maximum context length. For each timestep t, $y_t^{(k)}$ represents a sentence boundary at position $t - k$, i.e. k words in the past. $y^{(0)}$ represents the case of no boundary. To generate training examples, each sentence is extended with the first m words of the next sentence, and those words are labelled with $y^{(1)}$ to $y^{(m)}$.

However, we make a crucial change on how the model is applied: instead of outputting words only after a sentence end decision[2], we output words as soon as the model is confident that they still belong to the current sentence. For this purpose, we reinterpret the threshold vector $\theta^{(k)}$ such that $p(y_t^{(k)}) > \theta^{(k)}$ detects a *possible* instead of a definite sentence boundary at position $t - k$. The idea is that as long as no incoming word is considered a possible sentence end, all words can be passed on to MT *without any delay*. Only if $p(y^{(1)}) > \theta^{(1)}$, the current word is buffered, and we wait for the second word of context to make a more informed decision. If for $k = 2$ the boundary is still possible, a third word is read, and so on. A final sentence end decision is only made at the maximum context length ($k = m$). In this case, a sentence end token is emitted and the inference is restarted using the buffered words. If during the process $p(y^{(k)}) < \theta^{(k)}$ for any k, the word buffer is flushed, except for words still needed for pending decisions at later positions. Note that false negative decisions are not corrected later using more context because the corresponding words in the output stream have already been read and possibly translated by the MT system.

4.5 Simultaneous MT Experiments

4.5.1 MT Model Training

We use the data described in Section 2.1 to train the simultaneous MT models. For the text input condition, no ASR-like preprocessing is applied as the input is natural text. SentencePiece vocabularies of size 30K are used for source and target. We create copies of the training data with dropped chunk boundaries (Section 4.2) with probabilities of $p_{\text{drop}} = 0.0, 0.2, 0.5$ and 1.0. 6 encoder and 2 decoder layers with a hidden size of 1000 are used, the word embedding size is 620. The chunk boundary delay is set to $D = 2$. Dropout and label smoothing is used as for the offline MT model. Adam optimizer is used with an initial learning rate of 0.001, decreased by factor 0.9 after 10 subepochs of non-decreasing dev set perplexity. Train-

[2] This is only appropriate in their scenario of an offline MT system as the next step in the pipeline.

ing takes 150 and 138 sub-epochs of 1M lines each for text and speech input, respectively.

4.5.2 Latency/Quality Trade-Off Parameters

As described in Section 4.2, we can vary the boundary prediction threshold probability t_b to set different latency/quality trade-offs at inference time. In our experiments, we observe that the longer a chunk gets the less confident the model is in predicting its boundary, leading in some cases to very large chunks and thus high latency. To counteract this effect, we introduce another meta-variable Δt_b which defines a decrement of the threshold per source subword in the chunk, making the current threshold t'_b at a given chunk length l: $t'_b = t_b - \Delta t_b \cdot (l-1)$. This usually leads to chunks of reasonable length, while also setting a theoretical limit of $l \leq \lceil t_b / \Delta t_b \rceil + 1$.

For the speech input condition, we vary the ASR window size W_{ASR} of the acoustic model in the ASR system between 250ms, 500ms and 1000ms.

Finally, we apply length normalization by dividing the model scores by I^α, I being the chunk translation length in subwords, and tune α to values ≤ 1 for low latency trade-offs as we notice the MT model tends to overtranslate in this range.

4.5.3 Fine-tuning

We fine-tune all simultaneous MT models on in-domain data described in Section 2. We also add a copy of MuST-C where the transcriptions produced by our hybrid ASR system are used as source to make MT somewhat robust against ASR errors.

Furthermore, we create **low latency** systems by fine-tuning as above, but changing the chunk boundary prediction delay D from 2 to 1. This way the latency of the MT component is pushed to a minimum; however, at the cost of reduced translation quality caused by unreliable chunking decisions with a context of only one future word.

4.5.4 Sentence Segmenter

We train the sentence segmenter for unsegmented audio input (Section 4.4.2) on the English source side of the MT training data to which we apply ASR-like preprocessing and subword splitting. Note that the sentence splitting of the MT data itself is not perfect, and a better data selection might have improved results.

We set the maximum length of the future context to $m = 3$ as the baseline results in Wang et al. (2019) indicate no major improvement for longer contexts. Adam is used with a learning rate of 0.001, reduced by factor 0.8 after 3 epochs of non-improved dev set perplexity. Training takes 27 sub-epochs of 690K sentences each. For inference, we set the threshold vector to $\theta = (0.05, 0.1, 0.5)$ by analysing the amount of false negatives depending on $\theta^{(k)}$ for $k = 1, 2$ and by determining a good recall/precision trade-off for $k = 3$. The resulting segmenter has a recall of 61.4% and a precision of 64.1% on the original tst-COMMON sentence boundaries. Words are buffered for only 0.4 positions on average.

W_{ASR} (ms)	**dev**	**tst-HE**	**tst-COMMON**
250	11.7	11.1	12.4
500	10.7	10.3	10.8
1000	10.4	9.7	10.4

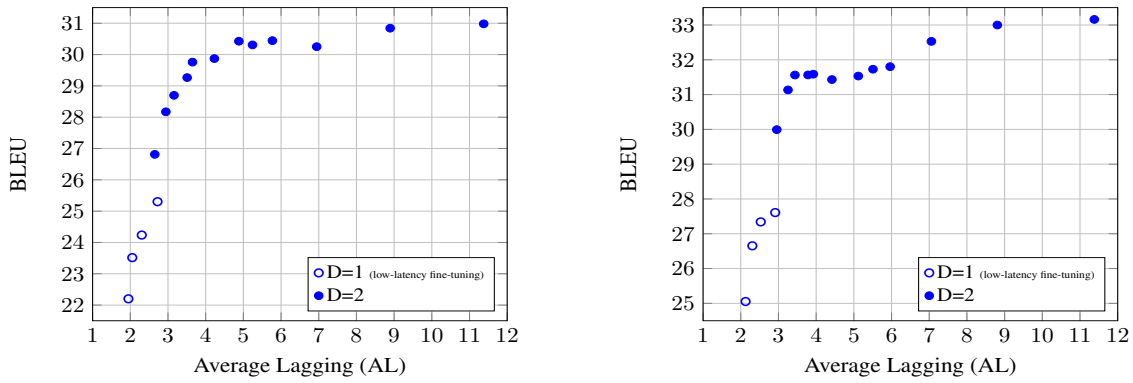
Table 3: WER [%] of streaming hybrid ASR on MuST-C test sets for various window sizes W_{ASR}

4.5.5 Simultaneous MT Results

The simultaneous MT systems are evaluated with the SimulEval tool (Ma et al., 2020). The BLEU and Average Lagging (AL) (Ma et al., 2019) metrics are used to score the different latency/quality trade-offs. Beam size 12 is used in all cases.

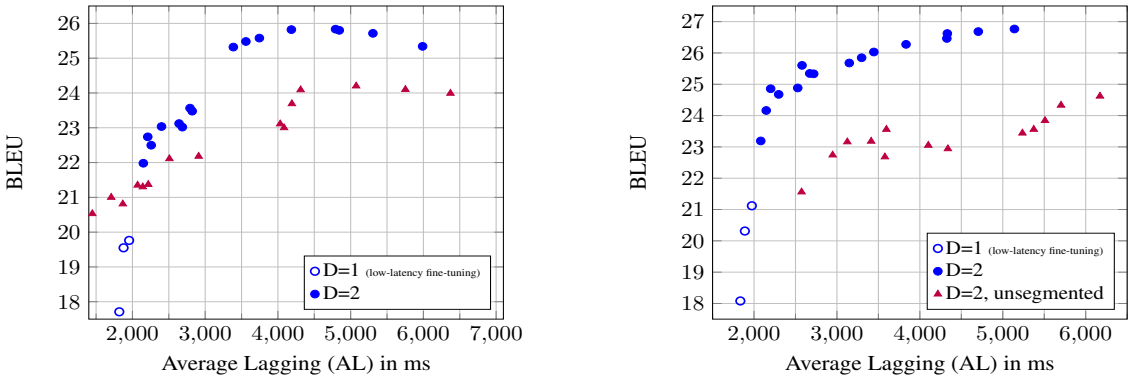
Figure 1 shows the results for the text input condition for MuST-C tst-HE and tst-COMMON. The filled data points correspond to the main text-input MT model. The points without fill show the results after low-latency fine-tuning with $D = 1$. The different trade-offs are achieved by varying the boundary threshold t_b from 0.3 to 0.9 using various decrements Δt_b. The full list of trade-off parameters is given in the appendix, Table 6. With the low-latency system an AL value of 2 words is achieved; however, at the cost of low BLEU scores of 22.2 and 25.1 on tst-HE and tst-COMMON, respectively. A reasonable operation point could for example be at an AL of 4, where BLEU scores of around 29.8 and 31.6 are achieved. For higher latency values, translation quality increases less rapidly, peaking at 31.0 and 33.1 BLEU for the two test sets. On tst-COMMON, a bump in the graph can be observed between 4 and 6 AL. This correlates with a problem of too short translations of up to 3% less words than the reference in this range. Below 4 AL, we are able to tune the hypothesis lengths via the length normalization exponent α. But above 4 AL, the optimal α is already 1, and setting $\alpha > 1$ does not yield improvements.

Figure 2 shows the results for the speech input condition. The trade-offs are achieved using sim-



(a) tst-HE (b) tst-COMMON

Figure 1: Results for English→German text-to-text simultaneous translation



(a) tst-HE (b) tst-COMMON

Figure 2: Results for English→German speech-to-text simultaneous translation

ilar parameters as for the text input (Table 7 in the appendix shows the full list). Additionally, we vary the ASR window size: for the 7 data points with lowest latency $W_{\text{ASR}} = 250\text{ms}$ is used, for the highest 3 $W_{\text{ASR}} = 1000\text{ms}$. The remaining points use a value of 500ms. The word error rates for different W_{ASR} are shown in Table 3. On tst-COMMON, the general shape of the curve is similar to text input. The lowest obtained AL is 1.8s. For high latencies, BLEU saturates at 26.8. On tst-HE, quality improves less rapidly with increased latency and even decreases slightly for AL values $> 5\text{s}$. This indicates that the trade-off parameters, which have been tuned on dev, do not translate perfectly to other test sets in all cases. When comparing text and speech input results for high latency values, we conclude that recognition errors in the ASR system lead to a drop in translation quality by about 5-6% absolute in terms of BLEU.

Figure 2 also shows results for unsegmented input[3]. Since no official scoring conditions have been defined, we therefore create partly unsegmented test sets ourselves by concatenating every 10 subsequent sentences of the test sets. The AL scores are taken as-is from SimulEval, the BLEU scores were computed using the mwerSegmenter tool. (Scoring the segmented results with mwerSegmenter leads to unaltered scores.) In general, the missing segmentation seems to lead to a drop of 2-3% BLEU. For tst-HE, unsegmented input leads to better results in the low latency range which is unrealistic and indicates that the AL values computed for single and multiple sentences are not comparable. In future work, we will analyze the scoring of the unsegmented case further and use trade-off parameters which are tuned for this case.

5 Final Results

In comparison to last year's submission (Bahar et al., 2020), the result of offline speech translation models have improved. The official results on the tst2020 and tst2021 test sets are shown in Table 4, as evaluated by the IWSLT 2021 organizers. This year, there are two references along with the BLEU score using both of them together. *Ref1* is the original one from the TED website, while *Ref2* has been created to simulate shorter translations as used in subtitles.

[3]For tst-COMMON we skip the 3 points with highest latency for better visibility of the other points.

Our end-to-end direct (an ensemble of 4 models), cascade (a single model) and posterior (a single model) systems correspond to the lines 15, 6 and 7 of Table 2, respectively. We observe that the provided reference segmentation negatively affects the ST quality regardless of the systems themselves. In contrast, the segmentation obtained by our segmentation model provides segments which apparently are more sentence-like including less noise and thus can be better translated. We note that our end-to-end direct primary and contrastive systems have the identical model parameters with an ensemble of 4 models while they utilize different speech segmentations. In the direct contrastive system, we apply our last year's segmentation which seems to be slightly better than that of this year. Similar to the MuST-C tst-COMMON set in Table 2, the direct model outperforms the cascaded-wise systems on tst2020 whereas it is behind on tst2021 with automatic segmentation. On the condition with reference segmentation, the difference between our cascade and direct models is lower where both systems almost preform the same. More results can be found in (Anastasopoulos et al., 2021).

System	TED tst2020	TED tst2021		
		Ref1	Ref2	both
reference segmentation				
direct (submission 2020)	20.5	-	-	-
direct	22.2	20.2	17.1	28.7
cascade	21.4	20.7	17.1	28.6
posterior	20.6	20.1	16.8	28.3
automatic segmentation				
direct (submission 2020)	23.5	-	-	-
direct primary	24.5	22.6	18.3	31.0
direct contrastive	25.1	22.8	18.9	32.0
cascade	24.0	23.3	19.2	32.1
posterior†	23.1	21.9	18.1	30.4

Table 4: AppTek IWSLT 2021 submission for offline speech translation measured by BLEU [%]. †: our cascade primary system at the time of submission.

Table 5 shows the official results for our simultaneous speech translation submission. The classification into different latency regimes is done by the organizers based on results on tst-COMMON. Due to dropping chunk boundaries in training, this year we are able to provide systems in all latency regimes, except for the speech track where a low-latency system ($\mathrm{AL} < 1\mathrm{s}$) is not possible to achieve with our cascade approach where the individual components already have a relatively high minimal latency.

latency regime	BLEU [%]	AL
text-to-text		
low	22.8	3.1
mid	25.7	6.2
high	26.6	12.0
speech-to-text		
mid	16.6	2.0s
high	21.0	4.0s

Table 5: AppTek IWSLT 2021 official simultaneous speech translation results on the **blind** text and speech input test sets.

6 Conclusion

This work summarizes the results of AppTek's participation in the IWSLT 2021 evaluation campaign for the offline and simultaneous speech translation tasks. Compared to AppTek's systems at IWSLT 2020, the cascade and direct systems present an improvement of 0.9% and 2.6% in BLEU and TER, respectively, averaging over 3 test sets. This shows that we further decreased the gap in MT quality between the cascade and direct models. We have also explored the posterior model, which enables generating translations along with transcripts. This is particularly important for applications when both sequences have to be displayed to users.

For the simultaneous translation systems, this year we are able to provide configurations in a wide latency range, starting at AL values of 2 words and 1.8s for text and speech input, respectively. For speech input, a maximal translation quality of 25.8 BLEU is achieved on tst-HE, 3% BLEU improvement compared to the previous system at a similar latency. By using future context of variable length we are able to do reliable sentence segmentation of ASR output designed to introduce minimal additional delay to the system.

References

Antonios Anastasopoulos, Ondřej Bojar, Jacob Bremerman, Roldano Cattoni, Maha Elbayad, Marcello Federico, Xutai Ma, Satoshi Nakamura, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Alex Waibel, Changhan Wang, and Matthew Wiesner. 2021. Findings of the IWSLT 2021 Evaluation Campaign. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, Online.

Parnia Bahar, Tobias Bieschke, Ralf Schlüter, and Hermann Ney. 2021. Tight integrated end-to-end training for cascaded speech translation. In *IEEE Spoken Language Technology Workshop*, pages 950–957, Shenzhen, China.

Parnia Bahar, Patrick Wilken, Tamer Alkhouli, Andreas Guta, Pavel Golik, Evgeny Matusov, and Christian Herold. 2020. Start-before-end and end-to-end: Neural speech translation by apptek and rwth aachen university. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 44–54.

Parnia Bahar, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. On using specaugment for end-to-end speech translation. In *International Workshop on Spoken Language Translation (IWSLT)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hervé Bourlard and Christian J. Wellekens. 1989. Links between Markov models and multilayer perceptrons. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 502–510. Morgan Kaufmann, San Mateo, CA, USA.

Kai Chen and Qiang Huo. 2016. Traininpg deep bidirectional LSTM acoustic model for LVCSR by a context-sensitive-chunk BPTT approach. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(7):1185–1193.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. *arXiv preprint arXiv:1806.03661*.

Marco Gaido, Mattia Antonino Di Gangi, Matteo Negri, Mauro Cettolo, and Marco Turchi. 2020a. Contextualized translation of automatically segmented speech. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 1471–1475. ISCA.

Marco Gaido, Mattia Antonino Di Gangi, Matteo Negri, and Marco Turchi. 2020b. End-to-end speech-translation with knowledge distillation: Fbk@iwslt2020. In *Proceedings of the 17th International Conference on Spoken Language Translation, IWSLT 2020, Online, July 9 - 10, 2020*, pages 80–88. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Javier Iranzo-Sánchez, Adrià Giménez Pastor, Joan Albert Silvestre-Cerdà, Pau Baquero-Arnal, Jorge Civera Saiz, and Alfons Juan. 2020. Direct segmentation models for streaming speech translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2599–2611.

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 *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 7180–7184. IEEE.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. 2017. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 4835–4839, New Orleans, LA, USA.

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

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 451–458.

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.

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020. Simuleval: An evaluation toolkit for simultaneous translation. *arXiv preprint arXiv:2007.16193*.

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.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-latency neural speech translation. *arXiv preprint arXiv:1808.00491*.

Robert Östling and Jörg Tiedemann. 2016. Efficient word alignment with Markov Chain Monte Carlo. *Prague Bulletin of Mathematical Linguistics*, 106:125–146.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

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

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548.

Jan-Thorsten Peter, Eugen Beck, and Hermann Ney. 2018. Sisyphus, a workflow manager designed for machine translation and automatic speech recognition. In *Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

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, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA.

Andreas Stolcke and Elizabeth Shriberg. 1996. Automatic linguistic segmentation of conversational speech. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 2, pages 1005–1008. IEEE.

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.

Xiaolin Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2016. An efficient and effective online sentence segmenter for simultaneous interpretation. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 139–148.

Xiaolin Wang, Masao Utiyama, and Eiichiro Sumita. 2019. Online sentence segmentation for simultaneous interpretation using multi-shifted recurrent neural network. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 1–11.

Simon Wiesler, Alexander Richard, Pavel Golik, Ralf Schlüter, and Hermann Ney. 2014. RASR/NN: The RWTH neural network toolkit for speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3313–3317, Florence, Italy.

Patrick Wilken, Tamer Alkhouli, Evgeny Matusov, and Pavel Golik. 2020. Neural simultaneous speech translation using alignment-based chunking. In *International Workshop on Spoken Language Translation*.

Patrick Wilken and Evgeny Matusov. 2019. Novel applications of factored neural machine translation. *arXiv preprint arXiv:1910.03912*.

Albert Zeyer, Tamer Alkhouli, and Hermann Ney. 2018a. RETURNN as a generic flexible neural toolkit with application to translation and speech recognition. In *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations*, pages 128–133.

Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. 2019. A comparison of transformer and lstm encoder decoder models for asr. In *IEEE Automatic Speech Recognition and Understanding Workshop*, pages 8–15, Sentosa, Singapore.

Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney. 2018b. Improved training of end-to-end attention models for speech recognition. In *19th Annual Conf. Interspeech, Hyderabad, India, 2-6 Sep.*, pages 7–11.

Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2016. Towards online-recognition with deep bidirectional LSTM acoustic models. In *Interspeech*, pages 3424–3428, San Francisco, CA, USA.

A Appendix

trade-off id	D	t_b	Δt_b	α
1'	1	0.3	0.006	0.3
2'	1	0.4	0.008	0.6
3'	1	0.5	0.012	0.8
4'	1	0.6	0.012	1.0
1	2	0.3	0.006	0.3
2	2	0.4	0.008	0.4
3	2	0.5	0.012	0.6
4	2	0.6	0.012	0.8
5	2	0.6	0.008	0.8
6	2	0.7	0.012	1.0
7	2	0.9	0.032	1.0
8	2	0.9	0.027	1.0
9	2	0.9	0.023	1.0
10	2	0.9	0.017	1.0
11	2	0.9	0.012	1.0
12	2	0.9	0.008	1.0

Table 6: Trade-off parameters for submitted **text input** simultaneous MT systems, sorted from low to high latency. $D = 1$ refers to low latency fine-tuning described in Section 4.5.3. Other parameters are explained in Section 4.5.2.

trade-off id	D	W_{ASR} (ms)	t_b	Δt_b	α
1'	1	250	0.3	0.006	0.3
2'	1	250	0.4	0.008	0.6
3'	1	250	0.5	0.012	0.8
1	2	250	0.3	0.006	0.3
2	2	250	0.4	0.008	0.6
3	2	250	0.5	0.012	0.8
4	2	250	0.6	0.012	1.0
5	2	500	0.4	0.008	0.6
6	2	500	0.5	0.012	0.8
7	2	500	0.6	0.012	1.0
8	2	500	0.6	0.008	1.0
9	2	500	0.9	0.032	1.0
10	2	500	0.9	0.027	1.0
11	2	500	0.9	0.023	1.0
12	2	500	0.9	0.017	1.0
13	2	500	0.9	0.012	1.0
14	2	1000	0.9	0.017	1.0
15	2	1000	0.9	0.012	1.0
16	2	1000	0.9	0.008	1.0

Table 7: Trade-off parameters for submitted **speech input** simultaneous MT systems, sorted from low to high latency. $D = 1$ refers to low latency fine-tuning described in Section 4.5.3. Other parameters are explained in Section 4.5.2.

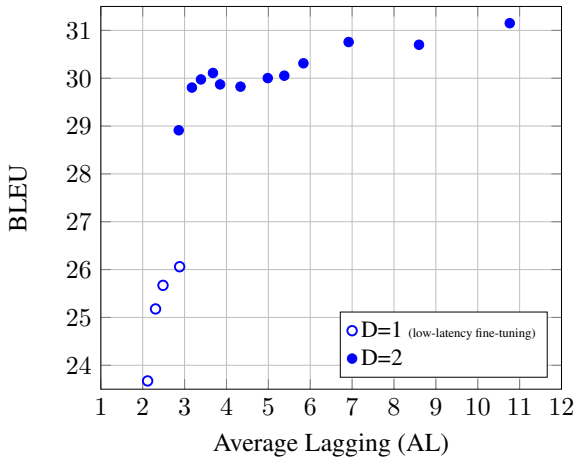


Figure 3: Results for English→German text-to-text simultaneous translation on MuST-C dev

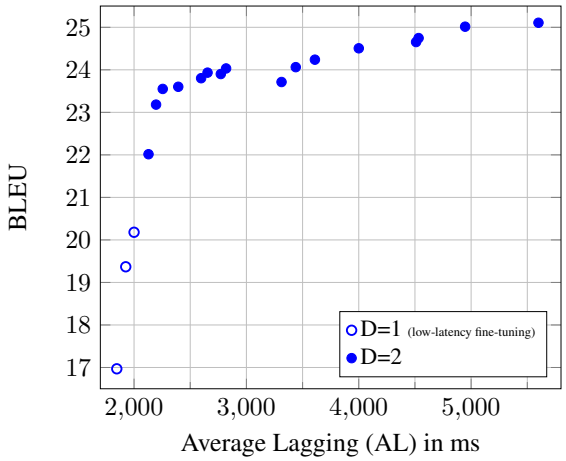


Figure 4: Results for English→German speech-to-text simultaneous translation on MuST-C dev