From Simultaneous to Streaming Machine Translation by Leveraging Streaming History

Javier Iranzo-Sánchez and **Jorge Civera** and **Alfons Juan**
Machine Learning and Language Processing Group
Valencian Research Institute for Artificial Intelligence
Universitat Politècnica de València
Camí de Vera s/n, 46022 València, Spain
`{jairsan,jorcisai,ajuanci}@vrain.upv.es`

Abstract

Simultaneous Machine Translation is the task of incrementally translating an input sentence before it is fully available. Currently, simultaneous translation is carried out by translating each sentence independently of the previously translated text. More generally, Streaming MT can be understood as an extension of Simultaneous MT to the incremental translation of a continuous input text stream. In this work, a state-of-the-art simultaneous sentence-level MT system is extended to the streaming setup by leveraging the streaming history. Extensive empirical results are reported on IWSLT Translation Tasks, showing that leveraging the streaming history leads to significant quality gains. In particular, the proposed system proves to compare favorably to the best performing systems.

1 Introduction

Simultaneous Machine Translation (MT) is the task of incrementally translating an input sentence before it is fully available. Indeed, simultaneous MT can be naturally understood in the scenario of translating a text stream as a result of an upstream Automatic Speech Recognition (ASR) process. This setup defines a simultaneous Speech Translation (ST) scenario that is gaining momentum due to the vast number of industry applications that could be exploited based on this technology, from person-to-person communication to subtitling of audiovisual content, just to mention two main applications.

These real-world streaming applications motivate us to move from simultaneous to streaming MT, understanding streaming MT as the task of simultaneously translating a potentially unbounded and unsegmented text stream. Streaming MT poses two main additional challenges over simultaneous MT. First, the MT system must be able to leverage the streaming history beyond the sentence level both at training and inference time. Second, the system must work under latency constraints over the entire stream.

With regard to exploiting streaming history, or more generally sentence context, it is worth mentioning the significant amount of previous work in offline MT at sentence level (Tiedemann and Scherrer, 2017; Agrawal et al., 2018), document level (Scherrer et al., 2019; Ma et al., 2020a; Zheng et al., 2020b; Li et al., 2020; Maruf et al., 2021; Zhang et al., 2021), and in related areas such as language modelling (Dai et al., 2019) that has proved to lead to quality gains. Also, as reported in (Li et al., 2020), more robust ST systems can be trained by taking advantage of the context across sentence boundaries using a data augmentation strategy similar to the prefix training methods proposed in (Niehues et al., 2018; Ma et al., 2019). This data augmentation strategy was suspected to boost re-translation performance when compared to conventional simultaneous MT systems (Arivazhagan et al., 2020).

Nonetheless, with the notable exception of (Schneider and Waibel, 2020), sentences in simultaneous MT are still translated independently from each other ignoring the streaming history. (Schneider and Waibel, 2020) proposed an end-to-end streaming MT model with a Transformer architecture based on an Adaptive Computation Time method with a monotonic encoder-decoder attention. This model successfully uses the streaming history and a relative attention mechanism inspired by Transformer-XL (Dai et al., 2019). Indeed, this is an MT model that sequentially translates the input stream without the need for a segmentation model. However, it is hard to interpret the latency of their streaming MT model because the authors observe that the current sentence-level latency measures, Average Proportion (AP) (Cho and Esipova, 2016), Average Lagging (AL) (Ma et al., 2019) and Differentiable Average Lagging (DAL) (Cherry and Foster, 2019) do not perform

well on a streaming setup. This fact is closely related to the second challenge mentioned above, which is that the system must work under latency constraints over the entire stream. Indeed, current sentence-level latency measures do not allow us to appropriately gauge the latency of streaming MT systems. To this purpose, (Iranzo-Sánchez et al., 2021) recently proposed a stream-level adaptation of the sentence-level latency measures based on the conventional re-segmentation approach applied to the ST output in order to evaluate translation quality (Matusov et al., 2005).

In this work, the simultaneous MT model based on a unidirectional encoder-decoder and training along multiple wait-k paths proposed by (Elbayad et al., 2020a) is evolved into a streaming-ready simultaneous MT model. To achieve this, model training is performed following a sentence-boundary sliding-window strategy over the parallel stream that exploits the idea of prefix training, while inference is carried out in a single forward pass on the source stream that is segmented by a Direct Segmentation (DS) model (Iranzo-Sánchez et al., 2020). In addition, a refinement of the unidirectional encoder-decoder that takes advantage of longer context for encoding the initial positions of the streaming MT process is proposed. This streaming MT system is thoroughly assessed on IWSLT translation tasks to show how leveraging the streaming history provides systematic and significant BLEU improvements over the baseline, while reported stream-adapted latency measures are fully consistent and interpretable. Finally, our system favourably compares in terms of translation quality and latency to the latest state-of-the-art simultaneous MT systems (Ansari et al., 2020).

This paper is organized as follows. Next section provides a formal framework for streaming MT to accommodate streaming history in simultaneous MT. Section 3 presents the streaming experimental setup whose results are reported and discussed in Section 4. Finally, conclusions and future work are drawn in Section 5.

2 Streaming MT

In streaming MT, the source stream X to be translated into Y comes as an unsegmented and unbounded sequence of tokens. In this setup, the decoding process usually takes the greedy decision of which token appears next at the i-th position of

the translation being generated

$$\hat{Y}_i = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, p\Big(y \mid X_1^{G(i)}, Y_1^{i-1}\Big) \qquad (1)$$

where $G(i)$ is a global delay function that tells us the last position in the source stream that was available when the i-th target token was output, and $\mathcal{Y}$ is the target vocabulary. However, taking into account the entire source and target streams can be prohibitive from a computational viewpoint, so the generation of the next token can be conditioned to the last $H(i)$ tokens of the stream as

$$\hat{Y}_i = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, p\Big(y \mid X_{G(i)-H(i)+1}^{G(i)}, Y_{i-H(i)}^{i-1}\Big). \quad (2)$$

Nevertheless, for practical purposes, the concept of sentence segmentation is usually introduced to explicitly indicate a monotonic alignment between source and target sentences in streaming MT. Let us consider for this purpose the random variables **a** and **b** for the source and target segmentation of the stream, respectively. Variables **a** and **b** can be understood as two vectors of equal length denoting that the n-th source sentence starts at position a_n, while the n-th target sentence does so at position b_n.

In the next sections, we reformulate simultaneous MT in terms of the more general framework of streaming MT. This reformulation allows us to consider opportunities for improvement of previous simultaneous MT models.

2.1 Simultaneous MT with streaming history

In the conventional simultaneous MT setup, the aforementioned variables **a** and **b** are uncovered at training and inference time, while in streaming MT **a** and **b** are considered hidden variables at inference time that may be uncovered by a segmentation model. In fact, in conventional simultaneous MT the history is limited to the current sentence being translated, while in streaming MT we could exploit the fact that the history could potentially span over all the previous tokens before the current sentence.

To this purpose, the global delay function $G(i)$ introduced above would replace the sentence-level delay function $g(i)$ commonly used in simultaneous MT. However, it should be noticed that we could express $g(i)$ as $G(i) - a_n$ with $b_n \leq i < b_{n+1}$. Delay functions are defined as a result of the policy being applied. This policy decides what action to take at each timestep, whether to read

a token from the input or to write a target token. Policies can be either fixed (Ma et al., 2019; Dalvi et al., 2018) depending only on the current timestep, or adaptive (Arivazhagan et al., 2019; Ma et al., 2020b; Zheng et al., 2020a) being also conditioned on the available input source words. Among those fixed policies, the sentence-level wait-k policy proposed by (Ma et al., 2019) is widely used in simultaneous MT with the simple local delay function

$$g(i) = k + i - 1. \qquad (3)$$

This policy initially reads k source tokens without writing a target token, and then outputs a target token every time a source token is read. This is true in the case that the ratio between the source and target sentence lengths is one. However, in the general case, a catch-up factor γ computed as the inverse of the source-target length ratio defines how many target tokens are written for every read token, that generalises Eq. 3 as

$$g(i) = \left\lfloor k + \frac{i-1}{\gamma} \right\rfloor. \qquad (4)$$

The wait-k policy can be reformulated in streaming MT so that the wait-k behaviour is carried out for each sentence as

$$G(i) = \left\lfloor k + \frac{i - b_n}{\gamma} \right\rfloor + a_n - 1 \qquad (5)$$

where $b_n \leq i < b_{n+1}$.

In streaming MT, we could take advantage of the streaming history by learning the probability distribution stated in Eq. 2, whenever streaming samples would be available. However, training such a model with arbitrarily long streaming samples poses a series of challenges that need to be addressed. Firstly, it would be necessary to carefully define $G(i)$ and $H(i)$ functions so that, at each timestep, the available source and target streams are perfectly aligned. Given that the source-target length ratio may vary over the stream, if one uses a wait-k policy with a fixed γ, there is a significant chance that source and target are misaligned at some points over the stream. Secondly, every target token can potentially have a different $G(i)$ and $H(i)$, so the encoder-decoder representation and contribution to the loss would need to be recomputed for each target token at a significant computational expense. Lastly, current MT architectures and training procedures have evolved conditioned by the availability of sentence-level

parallel corpora for training, so they need to be adapted to learn from parallel streams.

To tackle the aforementioned challenges in streaming MT, a compromise practical solution is to uncover the source and target sentence segmentations. At training time, parallel samples are extracted by a sentence-boundary sliding window spanning over several sentences of the stream that shifts to the right one sentence at a time. In other words, each sentence pair is concatenated with its corresponding streaming history that includes previous sentence pairs simulating long-span prefix training. Doing so, we ensure that source and target streams are properly aligned at all times, and training can be efficiently carried out by considering a limited history. The inference process is performed in a purely streaming fashion in a single forward pass as defined in Eq. 2 with $H(i)$ being consistently defined in line with training, so that the streaming history spans over previous sentences already translated.

2.2 Partial Bidirectional Encoder

In simultaneous MT, the conventional Transformer-based bidirectional encoder representation (of the l-th layer) of a source token at any position j is constrained to the current n-th sentence

$$e_j^{(l)} = \text{Enc}\left(e_{a_n:G(i)}^{(l-1)}\right) \qquad (6)$$

where $a_n \leq j \leq G(i)$, while the decoder can only attend to previous target words and the encoding of those source words that are available at each timestep

$$s_i^{(l)} = \text{Dec}\left(s_{b_n:i-1}^{(l-1)}, e_{a_n:G(i)}^{(l-1)}\right). \qquad (7)$$

As a result, the encoder and decoder representations for positions j and i, respectively, could be computed taking advantage of subsequent positions to position j up to position $G(i)$ at inference time. However, at training time, this means that this bidirectional encoding-decoding of the source sentence has to be computed for every timestep, taking up to $|y|$ times longer than the conventional Transformer model.

To alleviate this problem, (Elbayad et al., 2020a) proposes a wait-k simultaneous MT model based on a modification of the Transformer architecture that uses unidirectional encoders and multiple values of k at training time. In this way, the

model is consistent with the limited-input restriction of simultaneous MT at inference time. The proposed unidirectional encoder can be stated as

$$e_j^{(l)} = \text{Enc}\left(e_{a_n:j}^{(l-1)}\right), \qquad (8)$$

that is more restrictive than that in Eq. 6, and it consequently conditions the decoder representation, since $G(i)$ in Eq. 7 depends on the specific k value employed at each training step.

As mentioned above, the unidirectional encoder just requires a single forward pass of the encoder at training time, and therefore there is no additional computational cost compared with a conventional Transformer. However, it does not take into account all possible input tokens for different values of k. Indeed, the encoding of the j-th input token will not consider those tokens beyond the j-th position, even if including them into the encoding process does not prevent us from performing a single forward pass.

A trade-off between the unidirectional and bidirectional encoders is what we have dubbed Partial Bidirectional Encoder (PBE), which modifies the unidirectional encoder to allow the first $k-1$ source positions to have access to succeeding tokens according to

$$e_j^{(l)} = \text{Enc}\left(e_{a_n:\max(a_n+k-1,j)}^{(l-1)}\right). \qquad (9)$$

PBE allows for a longer context when encoding the initial positions and is consistent with Eq. 7. At training time a single forward pass of the encoder-decoder is still possible as in the unidirectional encoder, and therefore no additional training cost is incurred. At inference time, we fall back to the bidirectional encoder.

Figure 1 shows a graphical comparison of the attention mechanism in $j = 3$ across the bidirectional (left), unidirectional (center) and PBE (right) encoders with $k = 4$ for two consecutive timesteps $i = 1$ with $G(1) = 4$ (top) and $i = 2$ with $G(2) = 5$ (bottom). As observed, PBE can take advantage of additional positions from $j + 1$ up to k with respect to the unidirectional encoder.

In a streaming setup, the bidirectional encoder-decoder of Eqs. 6 and 7 are not necessarily constrained to the current sentence and could exploit a streaming history of $H(i)$ tokens

$$e_j^{(l)} = \text{Enc}\left(e_{G(i)-H(i)+1:G(i)}^{(l-1)}\right) \qquad (10)$$

$$s_i^{(l)} = \text{Dec}\left(s_{i-H(i):i-1}^{(l-1)}, e_{G(i)-H(i)+1:G(i)}^{(l-1)}\right). \qquad (11)$$

Likewise, the proposed PBE with streaming history states as follows

$$e_j^{(l)} = \text{Enc}\left(e_{G(i)-H(i)+1:\max(G(i)-H(i)+k,j)}^{(l-1)}\right). \qquad (12)$$

3 Experimental setup

Table 1: Basic statistics of the training data from the IWSLT 2020 Evaluation Campaign (M = Millions).

Corpus	Doc	Sents(M)	Tokens(M)	
			German	English
News-Comm.	✓	0.3	7.4	7.2
Wikititles		1.3	2.7	3.1
Europarl	✓	1.8	42.5	45.5
Rapid	✓	1.5	26.0	26.9
MuST-C	✓	0.2	3.9	4.2
TED	✓	0.2	3.3	3.6
LibriVox		0.1	0.9	1.1
Paracrawl		31.4	465.2	502.9

A series of comparative experiments in terms of translation quality and latency have been carried out using data from the IWSLT 2020 Evaluation Campaign (Ansari et al., 2020), for both German→English and English→German. For the streaming condition, our system is tuned on the 2010 dev set, and evaluated on the 2010 test set for comparison with (Schneider and Waibel, 2020). Under this setting, words were lowercased and punctuation was removed in order to simulate a basic upstream ASR system. Also, a second non-streaming setting is used for the English→German direction to compare our system with top-of-the-line sentence-based simultaneous MT systems participating in the IWSLT 2020 Simultaneous Translation Task.

Table 1 summarizes the basic statistics of the IWSLT corpora used for training the streaming MT systems. Corpora for which document information is readily available are processed for training using the sliding window technique mentioned in Section 2.1. Specifically, for each training sentence, we prepend previous sentences, which are added one by one until a threshold h of history tokens is reached. Sentence boundaries are defined on the presence of special tokens (`<DOC>`, `<CONT>`, `<BRK>`, `<SEP>`) as in (Junczys-Dowmunt, 2019). Byte Pair Encoding (Sennrich et al., 2016) with 40K merge operations is applied to the data after preprocessing.

Our streaming MT system is evaluated in terms of latency and translation quality with BLEU (Papineni et al., 2002). Traditionally, latency evaluation in simultaneous MT has been carried out using
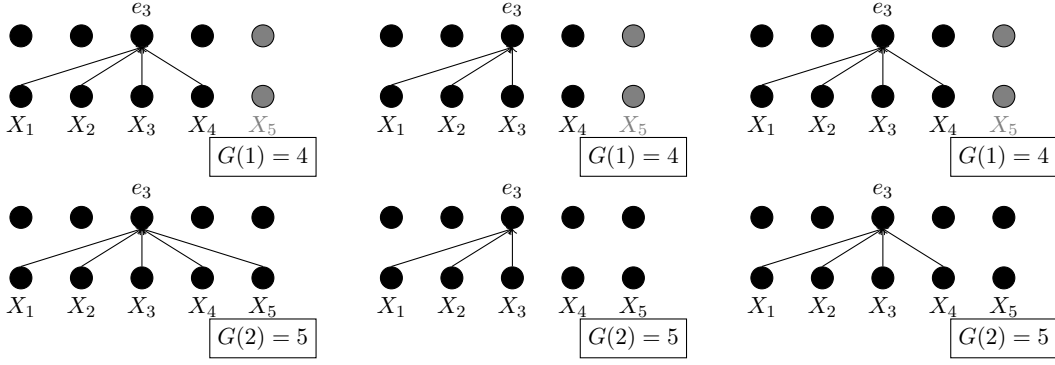
Figure 1: Comparison of attention positions in $j = 3$ for bidirectional (left), unidirectional (center) and PBE (right) encoders with $k = 4$ in two consecutive timesteps $i = 1$ with $G(1) = 4$ (top) and $i = 2$ with $G(2) = 5$ (bottom).

AP, AL and DAL. However, these measures have been devised for sentence-level evaluation, where the latency of every sentence is computed independently from each other and as mentioned before, they do not perform well on a streaming setup. Thus, we revert to the stream-based adaptation of these measures proposed in (Iranzo-Sánchez et al., 2021) unless stated otherwise.

Latency measures for a sentence pair $(\boldsymbol{x}, \boldsymbol{y})$ are based on a cost function $C_i(\boldsymbol{x}, \boldsymbol{y})$ and a normalization term $Z(\boldsymbol{x}, \boldsymbol{y})$

$$L(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{y})} \sum_i C_i(\boldsymbol{x}, \boldsymbol{y}) \qquad (13)$$

where

$$C_i(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} g(i) & \text{AP} \\ g(i) - \frac{i-1}{\gamma} & \text{AL} \\ g'(i) - \frac{i-1}{\gamma} & \text{DAL} \end{cases} \qquad (14)$$

and

$$Z(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} |\boldsymbol{x}| \cdot |\boldsymbol{y}| & \text{AP} \\ \underset{i:g(i)=|\boldsymbol{x}|}{\text{argmin}} \ i & \text{AL} \\ |\boldsymbol{y}| & \text{DAL} \end{cases} \qquad (15)$$

Latency measures can be computed in a streaming manner by considering a global delay function $G(i)$, that is mapped into a relative delay so that it can be compared with the sentence-level oracle delay. For the i-th target position of the n-th sentence, the associated relative delay can be obtained from the global delay function as $g_n(i) = G(i+b_n) - a_n$. So, the stream-adapted cost function of the latency

measures is defined as

$$C_i(\boldsymbol{x}_n, \boldsymbol{y}_n) = \begin{cases} g_n(i) & \text{AP} \\ g_n(i) - \frac{i-1}{\gamma_n} & \text{AL} \\ g'_n(i) - \frac{i-1}{\gamma_n} & \text{DAL} \end{cases} \qquad (16)$$

with $g'_n(i)$ defined as

$$\max \begin{cases} g_n(i) \\ g'_{n-1}(|\boldsymbol{x}_{n-1}|) + \frac{1}{\gamma_{n-1}} & i = 1 \\ g'_n(i-1) + \frac{1}{\gamma_n} & i > 1 \end{cases} \qquad (17)$$

This definition assumes that the source and target sentence segmentation of the stream are uncovered, but this is not always the case (Schneider and Waibel, 2020) or they may not match that of the reference translations. However, sentence boundaries can be obtained by re-segmenting the system hypothesis following exactly the same procedure applied to compute translation quality in ST evaluation. To this purpose, we use the MWER segmenter (Matusov et al., 2005) to compute sentence boundaries according to the reference translations.

Our streaming MT models have been trained following the conventional Transformer BASE (German↔English streaming MT) and BIG (English→German simultaneous MT) configurations (Vaswani et al., 2017). As in (Schneider and Waibel, 2020), after training is finished, the models are finetuned on the training set of MuST-C (Di Gangi et al., 2019).

The proposed model in Section 2 assumes that at inference time the source stream has been segmented into sentences. To this purpose, we opt for the text-based DS model (Iranzo-Sánchez et al., 2020), a sliding-window segmenter that moves over the source stream taking a split decision at each

token based on a local-context window that extends to both past and future tokens. This segmenter is streaming-ready and obtains superior translation quality when compared with other segmenters (Stolcke, 2002; Cho et al., 2017). As the future window length of the DS segmenter conditions the latency of the streaming MT system, this length was adjusted to find a tradeoff between latency and translation quality. The DS segmenter was trained on the TED corpus (Cettolo et al., 2012).

4 Evaluation

Figure 2 reports the evolution of BLEU scores on the German-English IWSLT 2010 dev set as a function of the k value in the wait-k policy for a range of streaming history lengths ($h = \{0, 20, 40, 60, 80\}$). We show results for the 3 encoders introduced previously. History lengths were selected taking into account that the average sentence length is 20 tokens. A history length of zero ($h = 0$) refers to the conventional sentence-level simultaneous MT model. The BLEU scores for the offline MT systems with a bidirectional encoder are also reported using horizontal lines, in order to serve as reference values. We report offline results for $h = 0$ and the best performing history configuration, $h = 60$. All systems used the reference segmentation during decoding.

As observed, BLEU scores of the simultaneous MT systems leveraging on the streaming history ($h > 0$) are systematically and notably higher than those of conventional sentence-based simultaneous MT system ($h = 0$) over the range of wait-k values. Indeed, as the streaming history increases, BLEU scores also do reaching what it seems the optimal history length at $h = 60$ and slightly degrading at $h = 80$. As expected, when replacing the unidirectional encoder by the PBE, BLEU scores improve as the wait-k value increases, since PBE has additional access to those tokens from $j + 1$ up to k. For instance, for $k = 32$ and $h = 60$, PBE is 0.7 BLEU points above the unidirectional encoder. On the other hand, it can be observed how using an encoder which is not fully bidirectional during training, creates a performance gap with respect to the offline bidirectional model when carrying out inference in an offline manner ($k \geq 32$). It can be also observed how the PBE model is better prepared for this scenario and shows a smaller gap. It is important to keep in mind that although both offline and PBE models behave the same way dur-
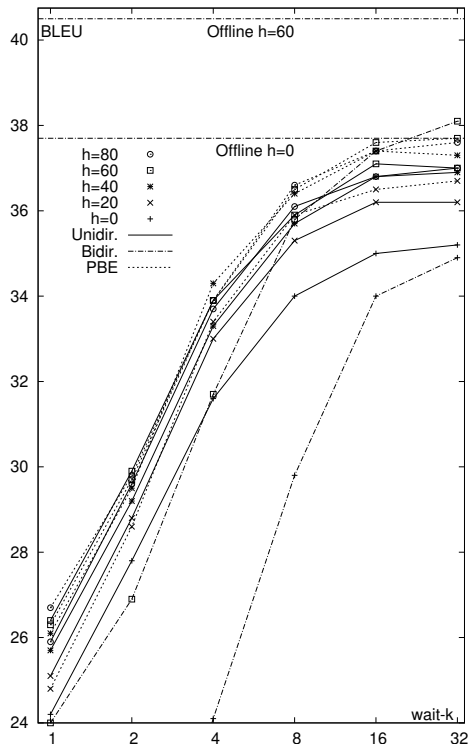

Figure 2: BLEU scores on the German-English IWSLT 2010 dev set as a function of the k value in the wait-k policy for a range of streaming history (h) lengths and encoder type (See Appendix A for a close-up).

ing inference for a large enough k, during training time the PBE model, trained using the multi-k with k randomly sampled for each batch, has been optimized jointly for low, medium and high latencies.

In general, the bidirectional encoder shows poor performance for simultaneous MT. This can be explained by the fact that there exists a mismatch between the training condition (whole source available) and the inference condition (only a prefix of the source is available for $k < 32$). These results are consistent with (Elbayad et al., 2020a). Keep in mind that this bidirectional model is different from the offline one because it has been subject to the constraints of Eq. 7 during training. As a result of the BLEU scores reported in Figure 2, the streaming MT system with $h = 60$ and PBE was used in the rest of the German-English experiments.

Following (Schneider and Waibel, 2020)'s setup, the test set is lowercased and concatenated into a single stream. In order to measure the latency of the pipeline defined by the segmenter followed by MT system, it is necessary to take into account not only the latency of the MT system but also that of the segmenter. Thankfully this is straightforward to do in our pipeline, as a segmenter with a
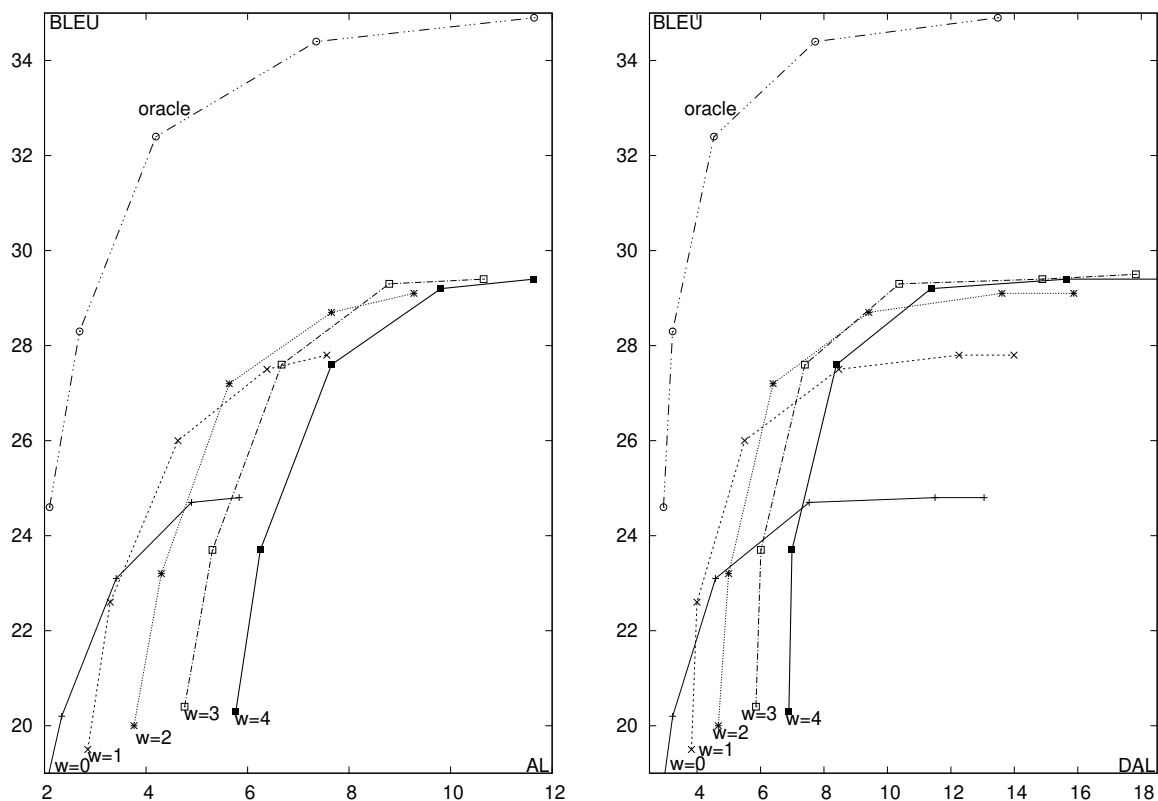
Figure 3: BLEU scores versus stream-adapted AL and DAL (scale s=0.85) with segmenters of future window length $w = \{0, 1, 2, 3, 4\}$ on the IWSLT 2010 test set. Points over each curve correspond to $k = \{1, 2, 4, 8, 16\}$ values of the wait-k policy used at inference time.

future window of length w modifies the pipeline policy so that, at the start of the stream, w READ actions are carried out to fill up the future window. Then, every time the MT system carries out a READ action, it receives one token from the segmenter. Thus, the integration of the segmenter into the pipeline is transparent from a latency viewpoint. Figure 3 shows BLEU scores versus stream-adapted AL and DAL (s scale = 0.85) figures reported with segmenters of future window length $w = \{0, 1, 2, 3, 4\}$ for a streaming evaluation on the IWSLT 2010 test set. Points over each curve correspond to $k = \{1, 2, 4, 8, 16\}$ values of the wait-k policy used at inference time. Results for a $w = 0$ oracle are also shown as an upper-bound.

As shown, stream-adapted AL and DAL figures achieved by our streaming MT system are reasonable, lagging 2-10 tokens behind the speaker for nearly maximum BLEU scores with a best BLEU score of 29.5 points. The same happens with AP figures ranging from 0.6 for $w = 0$ to 1.3 for $w = 4$. These figures highlight the advantages of tying together our translation policy with the sentence segmentation provided by the DS model. Every time

the DS model emits an end-of-sentence event, the MT model is forced to catch-up and translate the entire input. In this way, the MT model never strays too far from the speaker, even if the source-target length ratio differs from the γ defined at inference time. See Appendix A for streaming translation results in the reverse direction (English → German).

Next, we compare our proposed streaming MT (STR-MT) model with the $\lambda = 0.3$ ACT system (Schneider and Waibel, 2020) in terms of BLEU score and stream-adapted latency measures on Table 2. Stream-level AL and DAL indicate that the ACT models lags around 100 tokens behind the speaker. Although both MT systems achieve similar translation quality levels, they do so at significantly different latencies, since the ACT model

Table 2: Latency and quality comparison of ACT (Schneider and Waibel, 2020) and the proposed STR-MT on the IWSLT 2010 De-En test set.

Model	BLEU	AP	AL	DAL
ACT	30.3	10.3	100.1	101.8
STR-MT	29.5	1.2	11.2	17.8

lacks a catch-up mechanism to synchronize and keep the pace of the speaker.

The STR-MT model is now compared on the English-German IWSLT 2020 simultaneous text-to-text track (Ansari et al., 2020) with other participants: RWTH (Bahar et al., 2020), KIT (Pham et al., 2020) and ON-TRAC (Elbayad et al., 2020b). This comparison is carried out in order to assess whether the proposed streaming MT system is competitive with highly optimized systems for a simultaneous MT task. Given that the test set of this track remains blind, we use the results reported on the MuST-C corpus as a reference. In order to evaluate all systems under the same conditions, the reference segmentation of the MuST-C corpus is used instead of the DS model. Additionally, given that all other participants translate each sentence independently, the conventional sentence-level AL latency measure is reported. Figure 4 shows the comparison of BLEU scores versus AL measured in terms of detokenized tokens. As defined in the IWSLT text-to-text track, three AL regimes, low (AL ≤ 3), medium ($3 < $ AL ≤ 6) and high ($6 < $ AL ≤ 15) were considered.

ON-TRAC and our streaming MT system exhibit a similar progression, which is to be expected given that they are both based on the multi-k approach. However, our system consistently outperforms the ON-TRAC system by 1-2 BLEU. This confirms the importance of utilizing streaming history in order to significantly improve results, and how the proposed PBE model can take better advantage of the history.

RWTH and KIT systems are closer in translation quality to our proposal than ON-TRAC, for AL between 5 and 7. However, these systems do not show a flexible latency policy and are not comparable to our system at other regimes. Indeed, for that to be possible, these systems need to be re-trained, in contrast to our system in which latency is adjusted at inference time.

5 Conclusions

In this work, a formalization of streaming MT as a generalization of simultaneous MT has been proposed in order to define a theoretical framework in which our two contributions have been made. On the one hand, we successfully leverage streaming history across sentence boundaries for a simultaneous MT system based on multiple wait-k paths that allows our system to greatly improve the results of
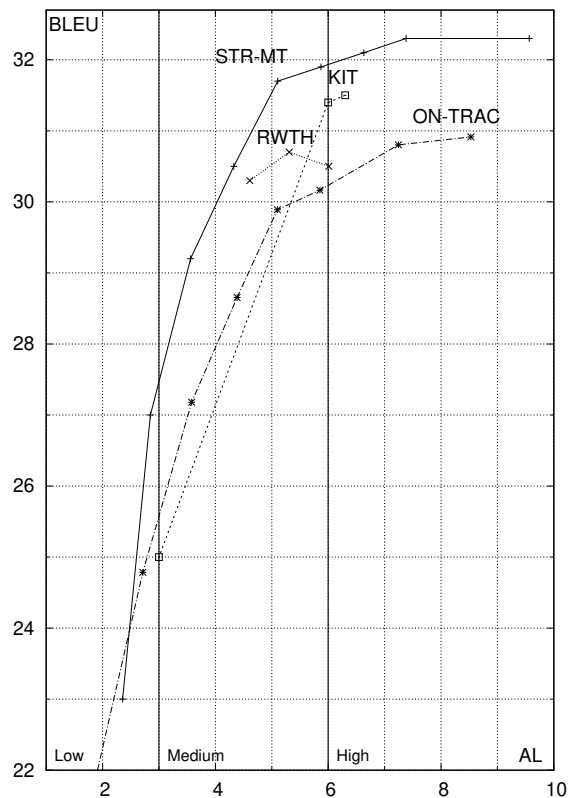

Figure 4: Comparative BLEU scores versus AL at three regimes, low, medium, and high latency, for IWSLT 2020 simultaneous text-to-text track participants, RWTH, ON-TRAC, KIT and our streaming MT (STR-MT) system on the MuST-C corpus.

the sentence-level baseline. On the other hand, our PBE is able to take into account longer context information than its unidirectional counterpart, while keeping the same training efficiency.

Our proposed MT system has been evaluated under a realistic streaming setting being able to reach similar translation quality than a state-of-the-art segmentation-free streaming MT system at a fraction of its latency. Additionally, our system has been shown to be competitive when compared with state-of-the-art simultaneous MT systems optimized for sentence-level translation, obtaining excellent results using a single model across a wide range of latency levels, thanks to its flexible inference policy.

In terms of future work, additional training and inference procedures that take advantage of the streaming history in streaming MT are still open for research. One important avenue of improvement is to devise more robust training methods, so that simultaneous models can perform as well as their offline counterparts when carrying out inference at

higher latencies. The segmentation model, though proved useful in a streaming setup, adds complexity and can greatly affect translation quality. Thus, the development of segmentation-free streaming MT models is another interesting research topic.

Acknowledgements

References

Ruchit Agrawal, M. Turchi, and M. Negri. 2018. Contextual handling in neural machine translation: Look behind, ahead and on both sides. In *Proc. of EAMT*, pages 11–20.

Ebrahim Ansari, Amittai Axelrod, Nguyen Bach, Ondřej 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, Alexander Waibel, and Changhan Wang. 2020. FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN. In *Proc. of IWSLT*, pages 1–34, Online. ACL.

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 *Proc. of ACL*, pages 1313–1323. ACL.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George F. Foster. 2020. Retranslation versus streaming for simultaneous translation. In *Proc. of IWSLT*, pages 220–227. ACL.

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 *Proc. of IWSLT*, pages 44–54. ACL.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT[3]: Web Inventory of Transcribed and Translated Talks. In *Proc. of EAMT*, pages 261–268.

Colin Cherry and George Foster. 2019. Thinking slow about latency evaluation for simultaneous machine translation. *arXiv preprint arXiv:1906.00048*.

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

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proc. of ACL*, pages 2978–2988.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proc. of NAACL-HLT*, pages 493–499. ACL.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. MuST-C: a Multilingual Speech Translation Corpus. In *Proc. of NAACL-HLT*, pages 2012–2017. ACM.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2020a. Efficient Wait-k Models for Simultaneous Machine Translation. In *Proc. of Interspeech*, pages 1461–1465. ISCA.

Maha Elbayad, Ha Nguyen, Fethi Bougares, Natalia Tomashenko, Antoine Caubrière, Benjamin Lecouteux, Yannick Estève, and Laurent Besacier. 2020b. ON-TRAC consortium for end-to-end and simultaneous speech translation challenge tasks at IWSLT 2020. In *Proc. of IWSLT*, pages 35–43. ACL.

Javier Iranzo-Sánchez, Jorge Civera Saiz, and Alfons Juan. 2021. Stream-level latency evaluation for simultaneous machine translation. In *Findings of ACL: EMNLP*, pages 664–670. ACL.

Javier Iranzo-Sánchez, Adrià Giménez, Joan Albert Silvestre-Cerdà, Pau Baquero, Jorge Civera, and Alfons Juan. 2020. Direct Segmentation Models for Streaming Speech Translation. In *Proc. of EMNLP*, pages 2599–2611. ACL.

Marcin Junczys-Dowmunt. 2019. Microsoft Translator at WMT 2019: Towards Large-Scale Document-Level Neural Machine Translation. In *Proc. of WMT*, pages 225–233.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proc. of EMNLP: System Demonstrations*, pages 66–71. ACL.

Daniel Li, Te I, Naveen Arivazhagan, Colin Cherry, and Dirk Padfield. 2020. Sentence boundary augmentation for neural machine translation robustness. *arXiv preprint arXiv:2010.11132*.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proc. of ACL: System Demonstrations*, pages 25–30. 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 *Proc. of ACL*, pages 3025–3036. ACL.

Shuming Ma, Dongdong Zhang, and Ming Zhou. 2020a. A simple and effective unified encoder for document-level machine translation. In *Proc. of ACL*, pages 3505–3511. ACL.

Xutai Ma, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020b. Monotonic Multihead Attention. In *Proc. ICLR 2020*. OpenReview.net.

Sameen Maruf, Fahimeh Saleh, and Gholamreza Haffari. 2021. A survey on document-level machine translation: Methods and evaluation. *arXiv preprint arXiv:1912.08494*.

Evgeny Matusov, Gregor Leusch, Oliver Bender, and Hermann Ney. 2005. Evaluating machine translation output with automatic sentence segmentation. In *Proc. of IWSLT*. ISCA.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-latency neural speech translation. In *Proc. of Interspeech*, pages 1293–1297. ISCA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL*, pages 311–318.

Ngoc-Quan Pham, Felix Schneider, Tuan-Nam Nguyen, Thanh-Le Ha, Thai Son Nguyen, Maximilian Awiszus, Sebastian Stüker, and Alexander Waibel. 2020. KIT's IWSLT 2020 SLT translation system. In *Proc. of IWSLT*, pages 55–61. ACL.

Yves Scherrer, Jörg Tiedemann, and Sharid Loáiciga. 2019. Analysing concatenation approaches to document-level NMT in two different domains. In *Proc. of DiscoMT@EMNLP*, pages 51–61. ACL.

Felix Schneider and Alexander Waibel. 2020. Towards stream translation: Adaptive computation time for simultaneous machine translation. In *Proc. of IWSLT*, pages 228–236. ACL.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proc. of ACL*, pages 1715–1725. ACL.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proc. of Interspeech*, pages 901–904. ISCA.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proc. of DiscoMT@EMNLP*, pages 82–92. ACL.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NIPS*, pages 5998–6008.

Biao Zhang, Ivan Titov, Barry Haddow, and Rico Sennrich. 2021. Beyond sentence-level end-to-end speech translation: Context helps. In *Proc. of ACL*, pages 2566–2578. ACL.

Baigong Zheng et al. 2020a. Simultaneous translation policies: From fixed to adaptive. In *Proc. of ACL*, pages 2847–2853. ACL.

Zaixiang Zheng, Xiang Yue, Shujian Huang, Jiajun Chen, and Alexandra Birch. 2020b. Towards making the most of context in neural machine translation. In *Proc. of IJCAI*, pages 3983–3989.

A Extended Streaming Translation Results

Figure 5 shows a close-up of Figure 2, which contains results for the German-English IWSLT 2010 dev set. We can observe how the PBE models obtain consistent quality improvements over their unidirectional counterparts.

Apart from the previously reported German → English streaming MT results, we have also conducted experiments in the reverse direction, English → German. These are shown in Figure 6. The results show a similar trend to previous experiments, with the addition of streaming history allowing our systems to obtain significant improvements over the sentence-based baseline. Unlike the previous case, the optimum history size in this case is $h = 40$ instead of $h = 60$.

In order to enable streaming translation, the best performing $h = 40$ systems has been combined with a German DS system. Similarly to previous experiments, we have conducted tests using different values of w and k in order to balance the
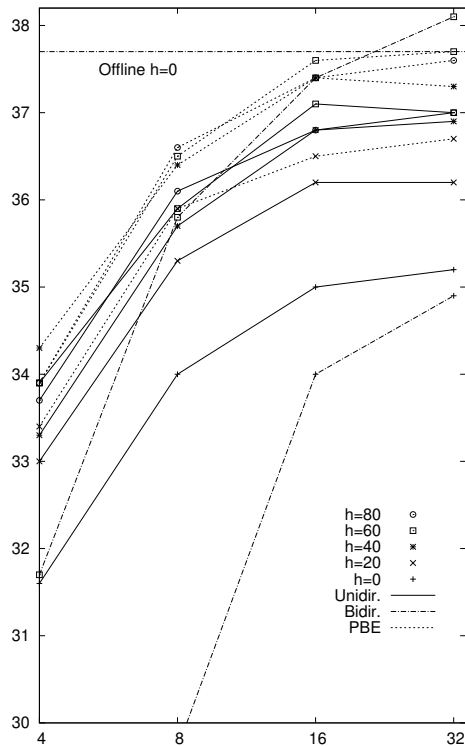
Figure 5: BLEU scores on the German-English IWSLT 2010 dev set as a function of the k value in the wait-k policy for a range of streaming history (h) lengths with a unidirectional encoder (solid lines), PBE (dashed line) or bidirectional (dashed line with points). This is a close-up of Figure 2.

Figure 6: BLEU scores on the English-German IWSLT 2010 dev set as a function of the k value in the wait-k policy for a range of streaming history (h) lengths using a PBE encoder.

latency-quality trade-off, shown in Figure 7. Under the streaming condition, the wait-k policy and DS model allow the model to follow closely the speaker while achieving good quality, with a latency that can be easily adjusted between 4 and 15 tokens depending on the requirements of the task. There are diminishing returns when increasing the latency above 6-7 tokens, as only marginal gains in quality are obtained.

B Efficiency of the proposed models

During training of the unidirectional and PBE encoders, the constraints imposed by Eqs. 8 and 9 are efficiently implemented by full self-attention, as in the bidirectional encoder, followed by an attention mask, for each token to only attend those tokens fulfilling the constraints. The attention mask sets the weights of the other tokens to $-\infty$ before application of the self-attention softmax. This is exactly the same mechanism used in the standard Transformer decoder to prevent the auto-regressive decoder from accessing future information.

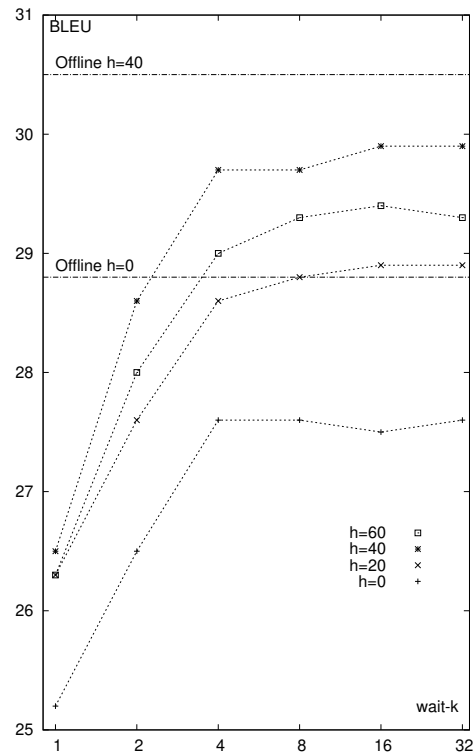This means that the three encoder types have an identical computational behavior. We are not aware of alternative GPU-based acceleration techniques to speed up the training of the unidirectional encoder. If so, this could be also applicable to the training of the standard Transformer decoder.

During inference time, however, the unidirectional encoder has some advantages. Given that the unidirectional encoder is incremental, meaning that the encodings of old tokens do not change when a new token becomes available, the process can be sped up by only computing the encoding of the newly available token. Although encoder self-attention still needs to be computed, a single vector is used as the query instead of the full matrix. Table 3 shows inference statistics for the different components of the En $\rightarrow$ De Transformer Big with $h=60$. Two setups have been tested: CPU-only inference, and GPU inference. Results were obtained on an Intel i9-7920X machine with an NVIDIA GTX 2080Ti.

The unidirectional encoder is four times faster than the bidirectional encoder when run on a CPU. However, both encoders perform the same when run on a GPU. For the streaming MT scenario considered in this work, no latency reduction is gained
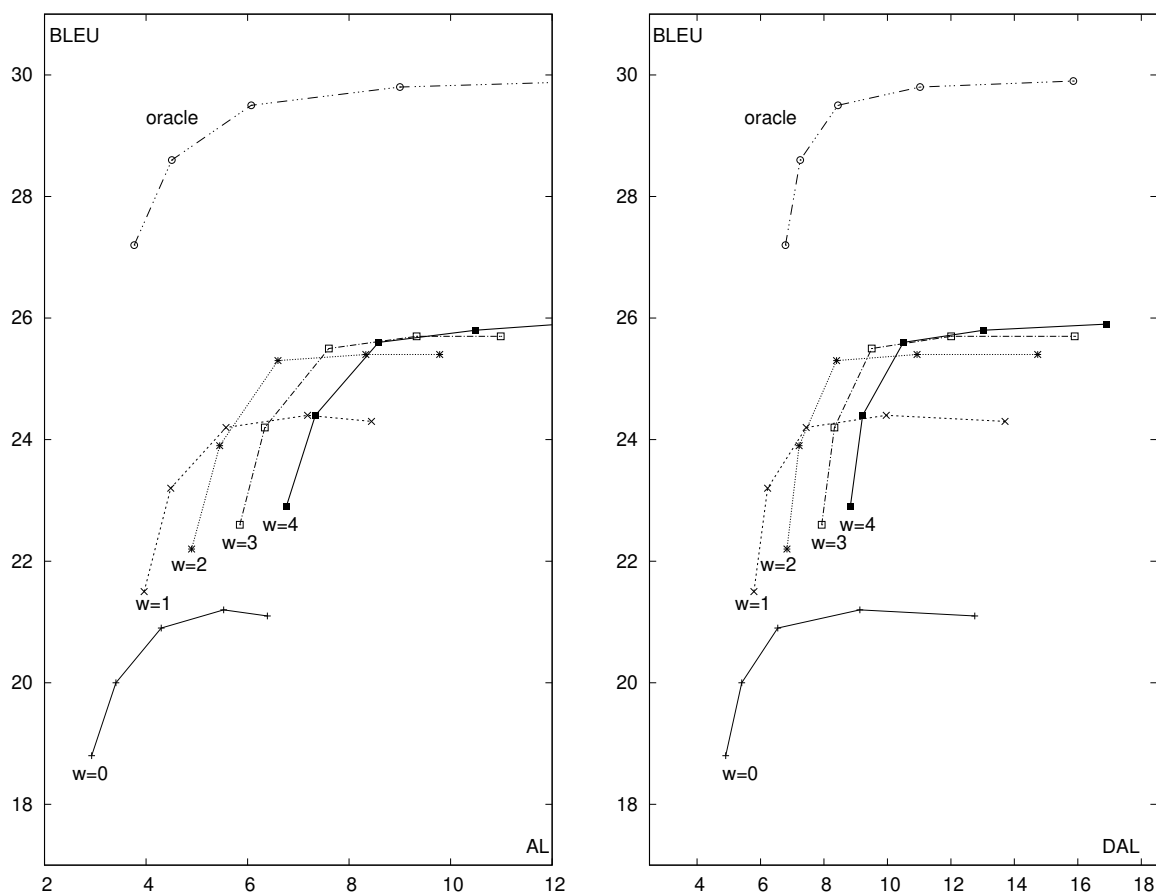
Figure 7: BLEU scores versus stream-adapted AL and DAL (scale s=0.85) with segmenters of future window length $w = \{0, 1, 2, 3, 4\}$ on the English-German IWSLT 2010 test set. Points over each curve correspond to $k = \{1, 2, 4, 8, 16\}$ values of the wait-k policy used at inference time.

Table 3: Latency of translating a token (in seconds) for the proposed En-De h=60 Transformer Big model.

Component	CPU	GPU
Unidir. Encoder	0.034s	0.002s
Bidir. Encoder	0.138s	0.002s
Decoder	0.242s	0.004s

by not re-encoding previous tokens due to the GPU paralellization capability. When run on a GPU, the proposed model works seamlessly under real-time constraints.

C MT System configuration

The multi-k systems have been trained with the official implementation (https://github.com/elbayadm/attn2d). Models are trained for 0.5M steps on a machine with 4 2080Ti GPUs. Total training time was 40h for BASE models, and 60h for BIG models. The following command was used to train them:

```
cri=label_smoothed_cross_entropy;
ex=simultaneous_translation
fairseq-train $CORPUS_FOLDER \
-s $SOURCE_LANG_SUFFIX \
-t $TARGET_LANG_SUFFIX \
--user-dir $FAIRSEQ/examples/$ex \
--arch $ARCH waitk_transformer_base \
--share-decoder-input-output-embed \
--left-pad-source False \
--multi-waitk \
--optimizer adam \
--adam-betas '(0.9, 0.98)' \
--clip-norm 0.0 \
--lr-scheduler inverse_sqrt \
--warmup-init-lr 1e-07 \
--warmup-updates 4000 \
--lr 0.0005 \
--min-lr 1e-09 \
--dropout 0.1 \
--weight-decay 0.0 \
--criterion $cri \
--label-smoothing 0.1 \
--max-tokens $TOK \
--update-freq 2 \
--save-dir $MODEL_OUTPUT_FOLDER \
--no-progress-bar \
--log-interval 100 \
--max-update 500000 \
--save-interval-updates 10000 \
```

```
--keep-interval-updates 20 \
--ddp-backend=no_c10d \
--fp16
```

with

```
ARCH=waitk_transformer_base;
TOK=4000
```

for the BASE configuration, and

```
ARCH=waitk_transformer_big;
TOK=2000
```

for the BIG one.

For finetuning, we change to the following:

```
--lr-scheduler fixed \
--lr 4.47169e-05 \
```

For the streaming translation scenario, the data is lowercased and all punctuation signs are removed. For the simultaneous scenario (IWSLT 2020 simultaneous text- to-text), it is truecased and tokenized using Moses. We apply language identification to the training data using langid (Lui and Baldwin, 2012) and discard those sentences that have been tagged with the wrong language. SentencePiece (Kudo and Richardson, 2018) is used to learn the BPE units, and we use whitespace as a suffix in order to know when an entire target word has been written during decoding.

In order to obtain samples that can be used for training streaming MT models, a sliding window that moves over whole sentences is used to extract consistent source-target samples. Figure 8 shows an example of corpus construction using $h = 5$. The generated streaming data is upsampled to keep a 1-to-3 ratio with the regular sentence-level data.

D   Segmenter System configuration

The Direct Segmentation system has been trained with the official implementation (https://github.com/jairsan/Speech_Translation_Segmenter). The following command was used to train the segmenter system:

```
python3 train_text_model.py \
 --train_corpus train.$len_$window.txt \
 --dev_corpus  dev.$len_$window.txt \
 --output_folder $out_f \
 --vocabulary $corpus_f/train.vocab.txt \
 --checkpoint_interval 1 \
 --epochs 15 \
 --rnn_layer_size 256 \
 --embedding_size 256 \
 --n_classes 2 \
 --batch_size 256 \
 --min_split_samples_batch_ratio 0.3 \
 --optimizer adam \
```

```
--lr 0.0001 \
--lr_schedule reduce_on_plateau \
--lr_reduce_patience 5 \
--dropout 0.3 \
--model_architecture ff-text \
--feedforward_layers 2 \
--feedforward_size 128 \
--sample_max_len $len \
--sample_window_size $window
```

with the following configurations:

```
(len=11; window=0)
(len=12; window=1)
(len=13; window=2)
(len=14, window=3)
(len=15, window=4)
```

Sentece pair	Source	Target
1	$x_{1,1}\ x_{1,2}$	$y_{1,1}\ y_{1,2}$
2	$x_{2,1}\ x_{2,2}\ x_{2,3}$	$y_{2,1}\ y_{2,2}$
3	$x_{3,1}\ x_{3,2}\ x_{3,3}$	$y_{3,1}\ y_{3,2}\ y_{3,3}$
4	$x_{4,1}\ x_{4,2}$	$y_{4,1}\ y_{4,2}$

Sentence pair	Source
1	<DOC> $x_{1,1}\ x_{1,2}$ <BRK>
2	<DOC> $x_{1,1}\ x_{1,2}$ <SEP> $x_{2,1}\ x_{2,2}\ x_{2,3}$ <BRK>
3	<DOC> $x_{1,1}\ x_{1,2}$ <SEP> $x_{2,1}\ x_{2,2}\ x_{2,3}$ <SEP> $x_{3,1}\ x_{3,2}\ x_{3,3}$ <BRK>
4	<CONT> $x_{3,1}\ x_{3,2}\ x_{3,3}$ <SEP> $x_{4,1}x_{4,2}$ <END>

Sentence pair	Target
1	<DOC> $y_{1,1}\ y_{1,2}$ <BRK>
2	<DOC> $y_{1,1}\ y_{1,2}$ <SEP> $y_{2,1}\ y_{2,2}$ <BRK>
3	<DOC> $y_{1,1}\ y_{1,2}$ <SEP> $y_{2,1}\ y_{2,2}$ <SEP> $y_{3,1}\ y_{3,2}\ y_{3,3}$ <BRK>
4	<CONT> $y_{3,1}\ y_{3,2}\ y_{3,3}$ <SEP> $y_{4,1}y_{4,2}$ <END>

Figure 8: Illustrated example of sample construction with history. Starting from a corpus of ordered sentence pairs (top), streaming samples are constructed (bottom) using $h = 5$. Past history is shown in light gray. Sentence boundary and document tokens (Junczys-Dowmunt, 2019) are not counted for the history size limit. Notice how, for the last sample, the pair $(\boldsymbol{x}_2, \boldsymbol{y}_2)$ is not included in the sample, as the history size limit would have otherwise been exceeded on the source side.