STACL: Simultaneous Translation with Implicit Anticipation and Controllable Latency using Prefix-to-Prefix Framework *

Mingbo Ma[1,3] **Liang Huang**[1,3] **Hao Xiong**[2] **Renjie Zheng**[3] **Kaibo Liu**[1,3] **Baigong Zheng**[1]
Chuanqiang Zhang[2] **Zhongjun He**[2] **Hairong Liu**[1] **Xing Li**[1] **Hua Wu**[2] **Haifeng Wang**[2]
[1]Baidu Research, Sunnyvale, CA, USA
[2]Baidu, Inc., Beijing, China
[3]Oregon State University, Corvallis, OR, USA
{mingboma, lianghuang, xionghao05, hezhongjun}@baidu.com

Abstract

Simultaneous translation, which translates sentences before they are finished, is useful in many scenarios but is notoriously difficult due to word-order differences. While the conventional seq-to-seq framework is only suitable for full-sentence translation, we propose a novel prefix-to-prefix framework for simultaneous translation that implicitly learns to anticipate in a single translation model. Within this framework, we present a very simple yet surprisingly effective "wait-k" policy *trained* to generate the target sentence concurrently with the source sentence, but always k words behind. Experiments show our strategy achieves low latency and reasonable quality (compared to full-sentence translation) on 4 directions: zh↔en and de↔en.

1 Introduction

Simultaneous translation aims to automate simultaneous interpretation, which translates concurrently with the source-language speech, with a delay of only a few seconds. This *additive* latency is much more desirable than the *multiplicative* 2× slowdown in consecutive interpretation.

With this appealing property, simultaneous interpretation has been widely used in many scenarios including multilateral organizations (UN/EU), and international summits (APEC/G-20). However, due to the concurrent comprehension and production in two languages, it is extremely challenging and exhausting for humans: the number of qualified simultaneous interpreters worldwide is very limited, and each can only last for about 15-30 minutes in one turn, whose error rates grow exponentially after just minutes of interpreting (Moser-Mercer et al., 1998). Moreover, limited memory forces human interpreters to routinely omit source content (He et al., 2016). Therefore, there is a critical need to develop simultaneous machine translation techniques to reduce the burden of human interpreters and make it more accessible and affordable.

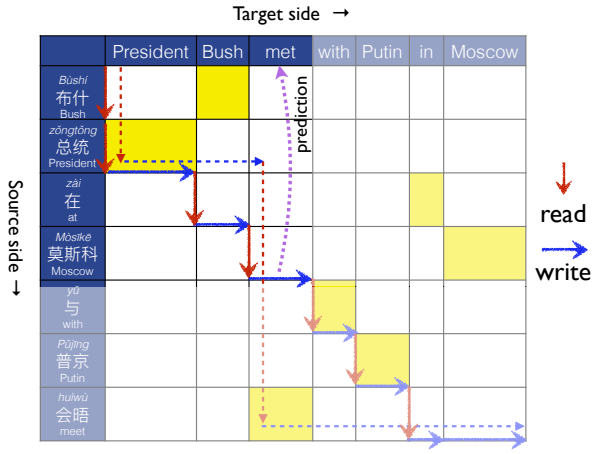

Figure 1: Our wait-k model emits target word y_t given source-side prefix $x_1 \dots x_{t+k-1}$, often before seeing the corresponding source word (here k=2, outputing y_3="met" before x_7="*huìwù*"). Without anticipation, a 5-word wait is needed (dashed arrows). See also Fig. 2.

Unfortunately, simultaneous translation is also notoriously difficult for machines, due in large part to the diverging word order between the source and target languages. For example, think about simultaneously translating an SOV language such as Japanese or German to an SVO language such as English or Chinese:[1] you have to wait until the source language verb. As a result, existing so-called "real-time" translation systems resort to conventional full-sentence translation, causing an undesirable latency of at least one sentence. Some researchers, on the other hand, have noticed the importance of verbs in SOV→SVO translation

* M.M. and L.H. contributed equally; L.H. conceived the main ideas (prefix-to-prefix and wait-k) and directed the project, while M.M. led the implementations on RNN and Transformer. See example videos, media reports, code, and data at https://simultrans-demo.github.io/.

[1] Technically, German is SOV+V2 in main clauses, and SOV in embedded clauses; Mandarin is a mix of SVO+SOV.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3025–3036
Florence, Italy, July 28 - August 2, 2019.

	Bùshí	***zǒngtǒng***	***zài***	Mòsīkē	*yǔ*	*Pǔjīng*	*huìwù*
	布什	总统	在	莫斯科	与	普京	会晤
	Bush	**president**	**in**	**Moscow**	with/and	Putin	meet
(a) simultaneous: our wait-2	*...wait 2 words...*		pres.	bush	**met**	with	putin in moscow
(b) non-simultaneous baseline	*..... wait whole sentence .....*						pres. bush met with putin in moscow
(c) simultaneous: test-time wait-2	*...wait 2 words...*		pres.	bush	in	moscow and	pol- ite meeting
	布什	总统			在 莫斯科 与		普京 会晤
(d) simultaneous: non-predictive	*...wait 2 words...*		pres.	bush	*..... wait 5 words .....*		met with putin in moscow

Figure 2: Another view of Fig. 1, highlighting the prediction of English "met" corresponding to the sentence-final Chinese verb *huìwù*. (a) Our wait-k policy (here $k = 2$) translates concurrently with the source sentence, but always k words behind. It correclty predicts the English verb given just the first 4 Chinese words (in bold), lit. "Bush president in Moscow", because it is trained in a prefix-to-prefix fashion (Sec. 3), and the training data contains many prefix-pairs in the form of (X *zài* Y ..., X met ...). (c) The test-time wait-k decoding (Sec. 3.2) using the full-sentence model in (b) can not anticipate and produces nonsense translation. (d) A simultaneous translator without anticipation such as Gu et al. (2017) has to wait 5 words.

(Grissom II et al., 2016), and have attempted to reduce latency by explicitly predicting the sentence-final German (Grissom II et al., 2014) or English verbs (Matsubarayx et al., 2000), which is limited to this particular case, or unseen syntactic constituents (Oda et al., 2015; He et al., 2015), which requires incremental parsing on the source sentence. Some researchers propose to translate on an optimized sentence segment level to get better translation accuracy (Oda et al., 2014; Fujita et al., 2013; Bangalore et al., 2012). More recently, Gu et al. (2017) propose a two-stage model whose base model is a full-sentence model, On top of that, they use a READ/WRITE (R/W) model to decide, at every step, whether to wait for another source word (READ) or to emit a target word using the pretrained base model (WRITE), and this R/W model is trained by reinforcement learning to prefer (rather than enforce) a specific latency, without updating the base model. All these efforts have the following major limitations: (a) none of them can achieve any arbitrary given latency such as "3-word delay"; (b) their base translation model is still trained on full sentences; and (c) their systems are complicated, involving many components (such as pretrained model, prediction, and RL) and are difficult to train.

We instead present a very simple yet effective solution, designing a novel prefix-to-prefix framework that predicts target words using only prefixes of the source sentence. Within this framework, we study a special case, the "wait-k" policy, whose translation is always k words behind the input. Consider the Chinese-to-English example in Figs. 1–2, where the translation of the sentence-final Chinese verb *huìwù* ("meet") needs to be emitted earlier to avoid a long delay. Our wait-2 model correctly anticipates the English verb given only the first 4 Chinese words (which provide enough clue for this prediction given many similar prefixes in the training data). We make the following contributions:

- Our prefix-to-prefix framework is tailored to simultaneous translation and trained from scratch without using full-sentence models.
- It seamlessly integrates implicit anticipation and translation in a single model that directly predicts *target* words without explictly hallucinating *source* ones.
- As a special case, we present a "wait-k" policy that can satisfy any latency requirements.
- This strategy can be applied to most sequence-to-sequence models with relatively minor changes. Due to space constraints, we only present its performance the Transformer (Vaswani et al., 2017), though our initial experiments on RNNs (Bahdanau et al., 2014) showed equally strong results (see our November 2018 arXiv version `https://arxiv.org/abs/1810.08398v3`).
- Experiments show our strategy achieves low latency and reasonable BLEU scores (compared to full-sentence translation baselines) on 4 directions: zh↔en and de↔en.

2 Preliminaries: Full-Sentence NMT

We first briefly review standard (full-sentence) neural translation to set up the notations.

Regardless of the particular design of different seq-to-seq models, the encoder always takes

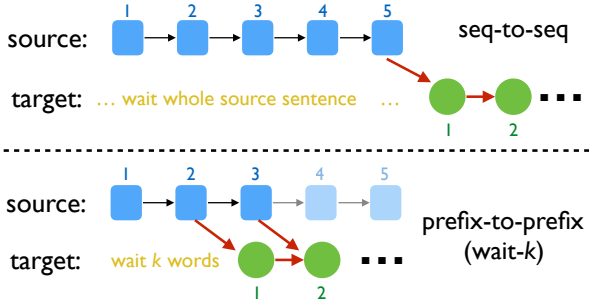

Figure 3: Seq-to-seq vs. our prefix-to-prefix frameworks (showing wait-2 as an example).

the input sequence $\mathbf{x} = (x_1, ..., x_n)$ where each $x_i \in \mathbb{R}^{d_x}$ is a word embedding of d_x dimensions, and produces a new sequence of hidden states $\mathbf{h} = f(\mathbf{x}) = (h_1, ..., h_n)$. The encoding function f can be implemented by RNN or Transformer.

On the other hand, a (greedy) decoder predicts the next output word y_t given the source sequence (actually its representation $\mathbf{h}$) and previously generated words, denoted $\mathbf{y}_{<t} = (y_1, ..., y_{t-1})$. The decoder stops when it emits <eos>, and the final hypothesis $\mathbf{y} = (y_1, ..., \text{<eos>})$ has probability

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) \quad (1)$$

At training time, we maximize the conditional probability of each ground-truth target sentence $\mathbf{y}^\star$ given input $\mathbf{x}$ over the whole training data D, or equivalently minimizing the following loss:

$$\ell(D) = -\sum_{(\mathbf{x},\mathbf{y}^\star)\in D} \log p(\mathbf{y}^\star \mid \mathbf{x}) \quad (2)$$

3 Prefix-to-Prefix and Wait-k Policy

In full-sentence translation (Sec. 2), each y_i is predicted using the entire source sentence $\mathbf{x}$. But in simultaneous translation, we need to translate concurrently with the (growing) source sentence, so we design a new prefix-to-prefix architecture to (be trained to) predict using a source prefix.

3.1 Prefix-to-Prefix Architecture

Definition 1. Let $g(t)$ be a **monotonic non-decreasing** function of t that denotes the number of source words processed by the encoder when deciding the target word y_t.

For example, in Figs. 1–2, $g(3) = 4$, i.e., a 4-word Chinese prefix is used to predict y_3="met". We use the source prefix $(x_1, ..., x_{g(t)})$ rather than the whole $\mathbf{x}$ to predict y_t: $p(y_t \mid \mathbf{x}_{\leq g(t)}, \mathbf{y}_{<t})$. Therefore the decoding probability is:

$$p_g(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t \mid \mathbf{x}_{\leq g(t)}, \mathbf{y}_{<t}) \quad (3)$$

and given training D, the training objective is:

$$\ell_g(D) = -\sum_{(\mathbf{x},\mathbf{y}^\star)\in D} \log p_g(\mathbf{y}^\star \mid \mathbf{x}) \quad (4)$$

Generally speaking, $g(t)$ can be used to represent any arbitrary policy, and we give two special cases where $g(t)$ is constant: (a) $g(t) = |\mathbf{x}|$: baseline full-sentence translation; (b) $g(t) = 0$: an "oracle" that does not rely on any source information. Note that in any case, $0 \leq g(t) \leq |\mathbf{x}|$ for all t.

Definition 2. We define the **"cut-off" step**, $\tau_g(|\mathbf{x}|)$, to be the decoding step when source sentence finishes:

$$\tau_g(|\mathbf{x}|) = \min\{t \mid g(t) = |\mathbf{x}|\} \quad (5)$$

For example, in Figs. 1–2, the cut-off step is 6, i.e., the Chinese sentence finishes right before y_6="in".

Training vs. Test-Time Prefix-to-Prefix. While most previous work in simultaneous translation, in particular Bangalore et al. (2012) and Gu et al. (2017), might be seen as special cases in this framework, we note that only their *decoders* are prefix-to-prefix, while their training is still full-sentence-based. In other words, they use a full-sentence translation model to do simultaneous decoding, which is a mismatch between training and testing. The essence of our idea, however, is to *train* the model to predict using source prefixes. Most importantly, this new training implicitly learns anticipation as a by-product, overcoming word-order differences such as SOV→SVO. Using the example in Figs. 1–2, the anticipation of the English verb is possible because the training data contains many prefix-pairs in the form of (X *zài* Y ..., X met ...), thus although the prefix $\mathbf{x}_{\leq 4}$="*Bùshí zǒngtǒng zài Mòsikē*" (lit. "Bush president in Moscow") does not contain the verb, it still provides enough clue to predict "met".

3.2 Wait-k Policy

As a very simple example within the prefix-to-prefix framework, we present a wait-k policy, which first wait k source words, and then translates concurrently with the rest of source sentence, i.e., the output is always k words behind the input. This is inspired by human simultaneous interpreters who generally start translating a few seconds into the speakers' speech, and finishes a few seconds after the speaker finishes. For example, if $k = 2$, the first target word is predicted using the first 2 source words, and the second target word

using the first 3 source words, etc; see Fig. 3. More formally, its $g(t)$ is defined as follows:

$$g_{\text{wait-}k}(t) = \min\{k + t - 1,\ |\mathbf{x}|\} \quad (6)$$

For this policy, the cut-off point $\tau_{g_{\text{wait-}k}}(|\mathbf{x}|)$ is exactly $|\mathbf{x}| - k + 1$ (see Fig. 14). From this step on, $g_{\text{wait-}k}(t)$ is fixed to $|\mathbf{x}|$, which means the remaining target words (including this step) are generated using the full source sentence, similar to conventional MT. We call this part of output, $\mathbf{y}_{\geq|\mathbf{x}|-k}$, the "tail", and can perform beam search on it (which we call "tail beam search"), but all earlier words are generated greedily one by one (see Appendix).

Test-Time Wait-k. As an example of test-time prefix-to-prefix in the above subsection, we present a very simple "test-time wait-k" method, i.e., using a full-sentence model but decoding it with a wait-k policy (see also Fig. 2(c)). Our experiments show that this method, without the anticipation capability, performs much worse than our genuine wait-k when k is small, but gradually catches up, and eventually both methods approach the full-sentence baseline ($k = \infty$).

4 New Latency Metric: Average Lagging

Beside translation quality, latency is another crucial aspect for evaluating simultaneous translation. We first review existing latency metrics, highlighting their limitations, aand then propose our new latency metric that address these limitations.

4.1 Existing Metrics: CW and AP

Consecutive Wait (CW) (Gu et al., 2017) is the number of source words waited between two target words. Using our notation, for a policy $g(\cdot)$, the per-step CW at step t is $\text{CW}_g(t) = g(t) - g(t-1)$. The CW of a sentence-pair $(\mathbf{x}, \mathbf{y})$ is the average CW over all consecutive wait segments:

$$\text{CW}_g(\mathbf{x}, \mathbf{y}) = \frac{\sum_{t=1}^{|\mathbf{y}|} \text{CW}_g(t)}{\sum_{t=1}^{|\mathbf{y}|} \mathbb{1}_{\text{CW}_g(t)>0}} = \frac{|\mathbf{x}|}{\sum_{t=1}^{|\mathbf{y}|} \mathbb{1}_{\text{CW}_g(t)>0}}$$

In other words, CW measures the average source segment length (the best case is 1 for word-by-word translation or our wait-1 and the worst case is $|\mathbf{x}|$ for full-sentence MT). The drawback of CW is that CW is local latency measurement which is insensitive to the actual lagging behind.

Another latency measurement, Average Proportion (AP) (Cho and Esipova, 2016) measures the proportion of the area above a policy path in Fig. 1:

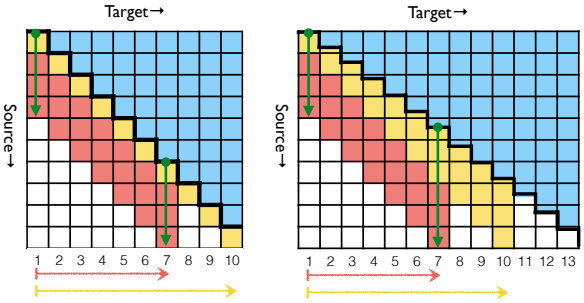

Figure 4: Illustration of our proposed Average Lagging latency metric. The left figure shows a simple case when $|\mathbf{x}| = |\mathbf{y}|$ while the right figure shows a more general case when $|\mathbf{x}| \neq |\mathbf{y}|$. The red policy is wait-4, the yellow is wait-1, and the thick black is a policy whose AL is 0.

$$\text{AP}_g(\mathbf{x}, \mathbf{y}) = \frac{1}{|\mathbf{x}|\ |\mathbf{y}|} \sum_{t=1}^{|\mathbf{y}|} g(t) \quad (7)$$

AP has two major flaws: First, it is sensitive to input length. For example, consider our wait-1 policy. When $|\mathbf{x}| = |\mathbf{y}| = 1$, AP is 1, and when $|\mathbf{x}| = |\mathbf{y}| = 2$, AP is 0.75, and eventually AP approaches 0.5 when $|\mathbf{x}| = |\mathbf{y}| \to \infty$. However, in all these cases, there is a one word delay, so AP is not fair between long and short sentences. Second, being a percentage, it is not obvious to the user the actual delays in number of words.

4.2 New Metric: Average Lagging

Inspired by the idea of "lagging behind the ideal policy", we propose a new metric called "average lagging" (AL), shown in Fig. 4. The goal of AL is to quantify the degree the user is out of sync with the speaker, in terms of the number of source words. The left figure shows a special case when $|\mathbf{x}| = |\mathbf{y}|$ for simplicity reasons. The thick black line indicates the "wait-0" policy where the decoder is alway one word *ahead* of the encoder and we define this policy to have an AL of 0. The diagonal yellow policy is our "wait-1" which is always one word behind the wait-0 policy. In this case, we define its AL to be 1. The red policy is our wait-4, and it is always 4 words behind the wait-0 policy, so its AL is 4. Note that in both cases, we only count up to (but including) the cut-off point (indicated by the horizontal yellow/red arrows, or 10 and 7, resp.) because the tail can be generated instantly without further delay. More formally, for the ideal case where $|\mathbf{x} = |\mathbf{y}|$, we can define:

$$\text{AL}_g(\mathbf{x}, \mathbf{y}) = \frac{1}{\tau_g(|\mathbf{x}|)} \sum_{t=1}^{\tau_g(|\mathbf{x}|)} g(t) - (t-1) \quad (8)$$

We can infer that the AL for wait-k is exactly k.

When we have more realistic cases like the right side of Fig. 4 when $|\mathbf{x}| < |\mathbf{y}|$, there are more and more delays accumulated when target sentence grows.For example, for the yellow wait-1 policy has a delay of more than 3 words at decoding its cut-off step 10, and the red wait-4 policy has a delay of almost 6 words at its cut-off step 7. This difference is mainly caused by the tgt/src ratio. For the right example, there are 1.3 target words per source word. More generally, we need to offset the "wait-0" policy and redefine:

$$\mathrm{AL}_g(\mathbf{x}, \mathbf{y}) = \frac{1}{\tau_g(|\mathbf{x}|)} \sum_{t=1}^{\tau_g(|\mathbf{x}|)} g(t) - \frac{t-1}{r} \quad (9)$$

where $\tau_g(|\mathbf{x}|)$ denotes the cut-off step, and $r = |y|/|x|$ is the target-to-source length ratio. We observe that wait-k with catchup has an AL $\simeq k$.

5 Implementation Details

While RNN-based implementation of our wait-k model is straightforward and our initial experiments showed equally strong results, due to space constraints we will only present Transformer-based results. Here we describe the implementation details for training a prefix-to-prefix Transformer, which is a bit more involved than RNN.

5.1 Background: Full-Sentence Transformer

We first briefly review the Transformer architecture step by step to highlight the difference between the conventional and simultaneous Transformer. The encoder of Transformer works in a self-attention fashion and takes an input sequence $\mathbf{x}$, and produces a new sequence of hidden states $\mathbf{z} = (z_1, ..., z_n)$ where $z_i \in \mathbb{R}^{d_z}$ is as follows:

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \; P_{W_\mathrm{V}}(x_j) \quad (10)$$

Here $P_{W_\mathrm{V}}(\cdot)$ is a projection function from the input space to the value space, and α_{ij} denotes the attention weights:

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{l=1}^{n} \exp e_{il}}, \; e_{ij} = \frac{P_{W_\mathrm{Q}}(x_i) P_{W_\mathrm{V}}(x_j)^T}{\sqrt{d_x}} \quad (11)$$

where e_{ij} measures similarity between inputs. Here $P_{W_\mathrm{Q}}(x_i)$ and $P_{W_\mathrm{K}}(x_j)$ project x_i and x_j to query and key spaces, resp. We use 6 layers of self-attention and use $\mathbf{h}$ to denote the top layer output sequence (i.e., the source context).

On the decoder side, during training time, the gold output sequence $\mathbf{y}^* = (y_1^*, ..., y_m^*)$ goes through the same self-attention to generate hidden self-attended state sequence $\mathbf{c} = (c_1, ..., c_m)$. Note that because decoding is incremental, we let $\alpha_{ij} = 0$ if $j > i$ in Eq. 11 to restrict self-attention to previously generated words.

In each layer, after we gather all the hidden representations for each target word through self-attention, we perform target-to-source attention:

$$c'_i = \sum_{j=1}^{n} \beta_{ij} \; P_{W_{\mathrm{V}'}}(h_j)$$

similar to self-attention, β_{ij} measures the similarity between h_j and c_i as in Eq. 11.

5.2 Training Simultaneous Transformer

Simultaneous translation requires feeding the source words incrementally to the encoder, but a naive implementation of such incremental encoder/decoder is inefficient. Below we describe a faster implementation.

For the encoder, during training time, we still feed the entire sentence at once to the encoder. But different from the self-attention layer in conventional Transformer (Eq. 11), we constrain each source word to attend to its predecessors only (similar to decoder-side self-attention), effectively simulating an incremental encoder:

$$\alpha_{ij}^{(t)} = \begin{cases} \frac{\exp e_{ij}^{(t)}}{\sum_{l=1}^{g(t)} \exp e_{il}^{(t)}} & \text{if } \; i, j \le g(t) \\ 0 & \text{otherwise} \end{cases}$$

$$e_{ij}^{(t)} = \begin{cases} \frac{P_{W_\mathrm{Q}}(x_i) \; P_{W_\mathrm{K}}(x_j)^T}{\sqrt{d_x}} & \text{if } \; i, j \le g(t) \\ -\infty & \text{otherwise} \end{cases}$$

Then we have a newly defined hidden state sequence $\mathbf{z}^{(t)} = (z_1^{(t)}, ..., z_n^{(t)})$ at decoding step t:

$$z_i^{(t)} = \sum_{j=1}^{n} \alpha_{ij}^{(t)} \; P_{W_\mathrm{V}}(x_j) \quad (12)$$

When a new source word is received, all previous source words need to adjust their representations.

6 Experiments

6.1 Datasets and Systems Settings

We evaluate our work on four simultaneous translation directions: German↔English and Chinese↔English. For the training data, we use the parallel corpora available from WMT15[2]

[2] http://www.statmt.org/wmt15/translation-task.html

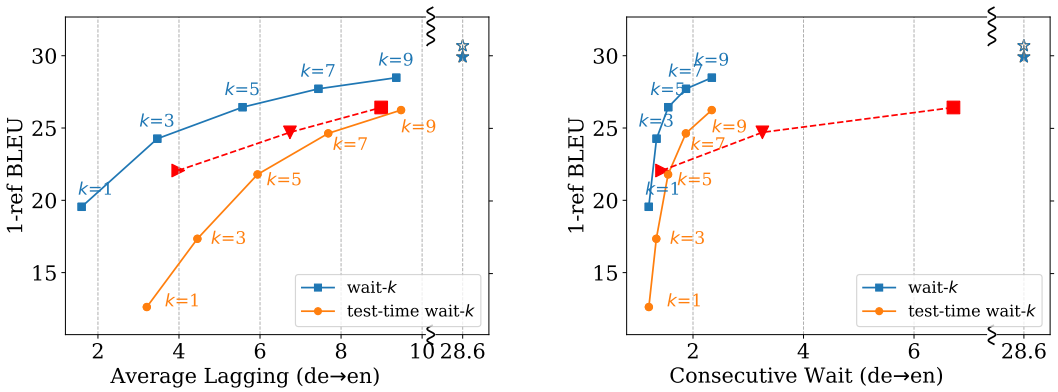

Figure 5: Translation quality against latency metrics (AL and CW) on German-to-English simultaneous translation, showing wait-k and test-time wait-k results, full-sentence baselines, and our adaptation of Gu et al. (2017) (▶:CW=2; ▼:CW=5; ■:CW=8), all based on the same Transformer. ★☆:full-sentence (greedy and beam-search).

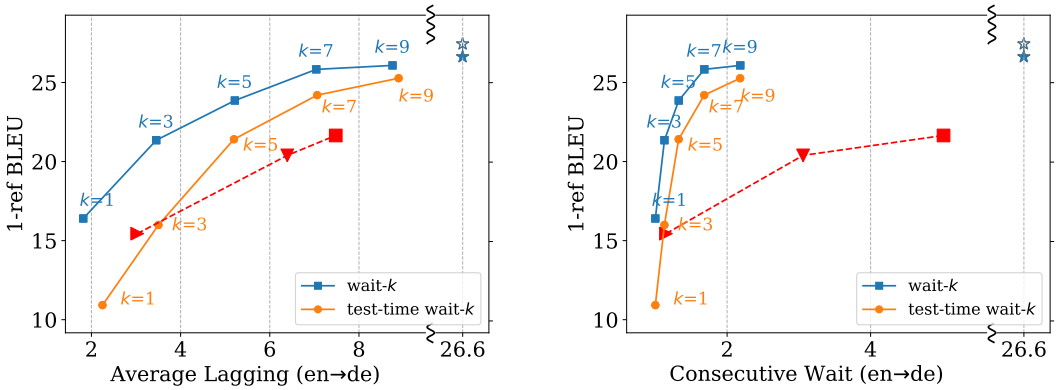

Figure 6: Translation quality against latency metrics on English-to-German simultaneous translation.

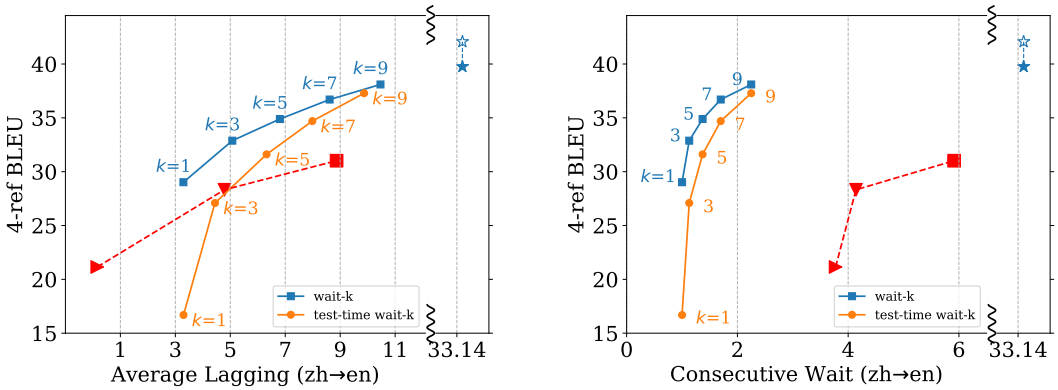

Figure 7: Translation quality against latency on Chinese-to-English simultaneous translation.

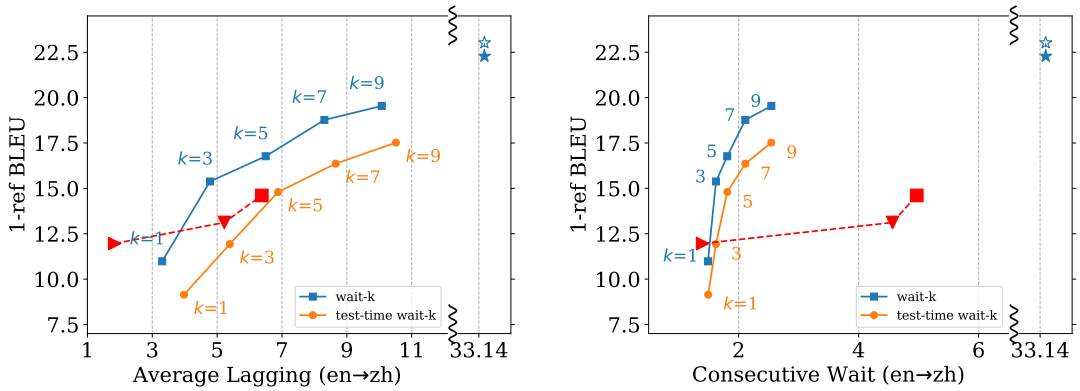

Figure 8: Translation quality against latency on English-to-Chinese, with encoder catchup (see Appendix A).

Train \ Test	k=1	k=3	k=5	k=7	k=9	k=∞
k'=1	*34.1*	33.3	31.8	31.2	30.0	15.4
k'=3	**34.7**	36.7	*37.1*	36.7	36.7	18.3
k'=5	30.7	36.7	37.8	38.4	*38.6*	22.4
k'=7	31.0	**37.0**	**39.4**	*40.0*	39.8	23.7
k'=9	26.4	35.6	39.1	**40.1**	***41.0***	28.6
k'=∞	21.8	30.2	36.0	38.9	39.9	***43.2***

Table 1: wait-k policy in training and test (4-ref BLEU, zh→en dev set). The bottom row is "test-time wait-k". Bold: best in a column; italic: best in a row.

	k=3	k=5	k=7	k=3	k=5	k=7
	zh→en			en→zh		
sent-level %	33	21	9	52	27	17
word-level %	2.5	1.5	0.6	5.8	3.4	1.4
accuracy	55.4	56.3	66.7	18.6	20.9	22.2
	de→en			en→de		
sent-level %	44	27	8	28	2	0
word-level %	4.5	1.5	0.6	1.4	0.1	0.0
accuracy	26.0	56.0	60.0	10.7	50.0	n/a

Table 2: Human evaluation for all four directions (100 examples each from dev sets). We report sentence- and word-level anticipation rates, and the word-level anticipation accuracy (among anticipated words).

for German↔English (4.5M sentence pairs) and NIST corpus for Chinese↔English (2M sentence pairs). We first apply BPE (Sennrich et al., 2015) on all texts in order to reduce the vocabulary sizes. For German↔English evaluation, we use newstest-2013 (dev) as our dev set and newstest-2015 (test) as our test set, with 3,000 and 2,169 sentence pairs, respectively. For Chinese↔English evaluation, we use NIST 2006 and NIST 2008 as our dev and test sets. They contain 616 and 691 Chinese sentences, each with 4 English references. When translating from Chinese to English, we report 4-reference BLEU scores, and in the reverse direction, we use the second among the four English references as the source text, and report 1-reference BLEU scores.

Our implementation is adapted from PyTorch-based OpenNMT (Klein et al., 2017). Our Transformer is essentially the same as the base model from the original paper (Vaswani et al., 2017).

6.2 Quality and Latency of Wait-k Model

Tab. 1 shows the results of a model trained with wait-k' but decoded with wait-k (where ∞ means full-sentence). Our wait-k is the diagonal, and the last row is the "test-time wait-k" decoding. Also, the best results of wait-k decoding is often from a model trained with a slightly larger k'.

Figs. 5–8 plot translation quality (in BLEU) against latency (in AL and CW) for full-sentence baselines, our wait-k, test-time wait-k (using full-sentence models), and our adaptation of Gu et al. (2017) from RNN to Transformer[3] on the same Transformer baseline. In all these figures, we observe that, as k increases, (a) wait-k improves in BLEU score and worsens in latency, and (b) the gap between test-time wait-k and wait-k shrinks. Eventually, both wait-k and test-time wait-k approaches the full-sentence baseline as $k \to \infty$. These results are consistent with our intuitions.

We next compare our results with our adaptation of Gu et al. (2017)'s two-staged full-sentence model + reinforcement learning on Transformer. We can see that while on BLEU-vs-AL plots, their models perform similarly to our test-time wait-k for de→en and zh→en, and slightly better than our test-time wait-k for en→zh, which is reasonable as both use a full-sentence model at the very core. However, on BLEU-vs-CW plots, their models have much worse CWs, which is also consistent with results in their paper (Gu, p.c.). This is because their R/W model prefers consecutive segments of READs and WRITEs (e.g., their model often produces R R R R R W W W W R R R W W W W R ...) while our wait-k translates concurrently with the input (the initial segment has length k, and all others have length 1, thus a much lower CW). We also found their training to be extremely brittle due to the use of RL whereas our work is very robust.

6.3 Human Evaluation on Anticipation

Tab. 2 shows human evaluations on anticipation rates and accuracy on all four directions, using 100 examples in each language pair from the dev sets. As expected, we can see that, with increasing k, the anticipation rates decrease (at both sentence and word levels), and the anticipation accuracy improves. Moreover, the anticipation rates are very different among the four directions, with

$$\text{en}\to\text{zh} > \text{de}\to\text{en} > \text{zh}\to\text{en} > \text{en}\to\text{de}$$

Interestingly, this order is exactly the same with the order of the BLEU-score gaps between our wait-9 and full-sentence models:

$$\text{en}\to\text{zh: } 2.7 > \text{de}\to\text{en: } 1.1 > \text{zh}\to\text{en: } 1.6^{\dagger} > \text{en}\to\text{de: } 0.3$$

[3] However, it is worth noting that, despite our best efforts, we failed to reproduce their work on their original RNN, regardless of using their code or our own. That being said, our successful implementation of their work on Transformer is also a notable contribution of this work. By contrast, it is very easy to make wait-k work on either RNN or Transformer.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	doch	während	man	sich	im	kongre-	ss	nicht	auf	ein	vorgehen	einigen	kann	,	warten	mehrere	bs.	nicht	länger
	but	while	they	-self	in	congress		not	on	one	action	agree	can	,	wait	several	states	no	longer
k=3				but	,	while	congress	has	not	agreed	on	a	course	of	action	,	several	states	no	longer	wait

Figure 9: German-to-English example in the dev set with anticipation. The main verb in the embedded clause, "einigen" (agree), is correctly predicted 3 words ahead of time (with "sich" providing a strong hint), while the aux. verb "kann" (can) is predicted as "has". The baseline translation is "but , while congressional action can not be agreed , several states are no longer waiting". bs.: bunndesstaaten.

	1	2	3	4	5	6	7	8	9	10	11	12
	tā	*hái*	*shuō*	*xiànzài*	*zhèngzài*	*wèi*	*zhè*	*yī*	*fǎngwèn*	*zuò*	*chū*	*ānpái*
	他	还	说	现在	正在	为	这	一	访问	作	出	安排
	he	also	said	now	(*prog.*)◇	for	this	one	visit	make	out	arrangement
k=1	he	also	said	that	he	is	now	making	preparations	for	this	visit
k=3			he	also	said	that	he	is	making	preparations	for	this visit
k=∞												he also said that arrangements are being made for this visit

Figure 10: Chinese-to-English example in the dev set with anticipation. Both wait-1 and wait-3 policies yield perfect translations, with "making preparations" predicted well ahead of time. ◇: progressive aspect marker.

	1	2	3	4	5	6	7	8	9	10	11
	jiāng	*zémín*	*duì*	*bùshí*	*zǒngtǒng*	*lái*	*huá*	*fǎngwèn*	*biǎoshì*	*rèliè*	*huānyíng*
	江	泽民	对	布什	总统	来	华	访问	表示	热烈	欢迎
	jiang	zeming	to	bush	president	come-to	china	visit	express	warm	welcome
k=3				jiang	zemin	expressed	welcome	to	president	bush	's	visit to china
k=3†				jiang	zemin	meets	president	bush	in	china	's	bid to visit china

Figure 11: Chinese-to-English example from online news. Our wait-3 model correctly anticipates both "expressed" and "welcome" (though missing "warm"), and moves the PP ("to ... visit to china") to the very end which is fluent in the English word order. †: test-time wait-k produces nonsense translation.

	1	2	3	4	5	6	7	8	9	10	
	Měiguó	*dāngjú*	*duì*	*Shātè*	*jìzhě*	*shīzōng*	*yī*	*àn*	*gǎndào*	*dānyōu*	
(a)	美国	当局	对	沙特	记者	失踪	一	案	感到	担忧	
	US	authorities	to	Saudi	reporter	missing	a	case	feel	concern	
k=3				the	us	authorities	are	very	concerned	about	the saudi reporter 's missing case
k=3†				the	us	authorities	have	dis-	appeared	from	saudi reporters
										bùmǎn	
(b)	美国	当局	对	沙特	记者	失踪	一	案	感到	不满	
k=3				the	us	authorities	are	very	concerned	about	the saudi reporter 's missing case
k=5						the	us	authorities	have	expressed	**dissatisfaction** with the incident of saudi arabia 's missing reporters

Figure 12: (a) Chinese-to-English example from more recent news, clearly outside of our data. Both the verb "*gǎndào*" ("feel") and the predicative "*dānyōu*" ("concerned") are correctly anticipated, probably hinted by "missing". (b) If we change the latter to *bùmǎn* ("dissatisfied"), the wait-3 result remains the same (which is wrong) while wait-5 translates conservatively without anticipation. †: test-time wait-k produces nonsense translation.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	it	was	learned	that	this	is	the	largest	fire	accident	in	the	medical	and	health	system	nationwide	since	the	founding	of	new	china	
k=3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20			
				jù	*liǎojiě*	,	*zhè*	*shì*	*zhōngguó*	*jìn*	*jǐ*	*nián*	*lái*	*fāshēng*	*de*	*zuì*	*dà*	*yī*	*qǐ*	*yīliáo*	*wèishēng*	*xìtǒng*	*huǒzāi*	*shìgù*
				据	了解	,	这	是	中国	近	几	年	来	发生	的	最	大	一	起	医疗	卫生	系统	火灾	事故
				to	known	,	this	is	China	recent	few	years	since	happen	-	most	big	one	case	medical	health	system	fire	accident
k=3†				*yīnwèi*	*tā*	*shì*	,	*zhège*	,	*shì*	*zuì*	*dà*	*de*	*huǒzāi*	*shìgù*	,	*zhè*	*shì*	*xīn*	*zhōngguó*	*chénglì*	*yǐlái*		
				因为	它	是	,	这个	,	是	最	大	的	火灾	事故	,	这	是	新	中国	成立	以来		
				because	it	is	,	this	,	is	most	big	-	fire	accident	,	this	is	new	China	funding	since		

Figure 13: English-to-Chinese example in the dev set with incorrect anticipation due to mandatory long-distance reorderings. The English sentence-final clause "since the founding of new china" is incorrectly predicted in Chinese as "近 几 年 来"("in recent years"). Test-time wait-3 produces translation in the English word order, which sounds odd in Chinese, and misses two other quantifiers ("in the medical and health system" and "nationwide"), though without prediction errors. The full-sentence translation, "据 了解， 这 是 新 中国 成立 以来 ， 全国 医疗 卫生 系统 发生 的 最 大 的 一 起 火 灾 事故", is perfect.

(†: difference in 4-ref BLEUs, which in our experience reduces by about half in 1-ref BLEUs). We argue that this order roughly characterizes the relative difficulty of *simultaneous* translation in these directions. In our data, we found en→zh to be particularly difficult due to the mandatory long-distance reorderings of English sentence-final temporal clauses (such as "in recent years") to much earlier positions in Chinese; see Fig. 13 for an example. It is also well-known that de→en is more challenging in simultaneous translation than en→de since SOV→SVO involves prediction of the verb, while SVO→SOV generally does not need prediction in our wait-k with a reasonable k, because V is often shorter than O. For example, human evaluation found only 1.4%, 0.1%, and 0% word anticipations in en→de for k=3, 5 and 7, and 4.5%, 1.5%, and 0.6% for de→en.

6.4 Examples and Discussion

We showcase some examples in de→en and zh→en from the dev sets and online news in Figs. 9 to 12. In all these examples except Fig. 12(b), our wait-k models can generally anticipate correctly, often producing translations as good as the full-sentence baseline. In Fig. 12(b), when we change the last word, the wait-3 translation remains unchanged (correct for (a) but wrong for (b)), but wait-5 is more conservative and produces the correct translation without anticipation.

Fig. 13 demonstrates a major limitation of our fixed wait-k policies, that is, sometimes it is just impossible to predict correctly and you have to wait for more source words. In this example, due to the required long-distance reordering between English and Chinese (the sentence-final English clause has to be placed very early in Chinese), any wait-k model would not work, and a good policy should wait till the very end.

7 Related Work

The work of Gu et al. (2017) is different from ours in four (4) key aspects: (a) by design, their model does not anticipate; (b) their model can not achieve any specified latency metric at test time while our wait-k model is guaranteed to have a k-word latency; (c) their model is a combination of two models, using a full-sentence base model to translate, thus a mismatch between training and testing, while our work is a genuine simultaneous model, and (d) their training is also two-staged, using RL to update the R/W model, while we train from scratch.

In a parallel work, Press and Smith (2018) propose an "eager translation" model which also outputs target-side words before the whole input sentence is fed in, but there are several crucial differences: (a) their work still aims to translate full sentences using beam search, and is therefore, as the authors admit, "not a simultaneous translation model"; (b) their work does not anticipate future words; and (c) they use word alignments to learn the reordering and achieve it in decoding by emitting the ϵ token, while our work integrates reordering into a single wait-k prediction model that is agnostic of, yet capable of, reordering.

In another recent work, Alinejad et al. (2018) adds a prediction action to the work of Gu et al. (2017). Unlike Grissom II et al. (2014) who predict the source verb which might come after several words, they instead predict the *immediate* next source words, which we argue is not as useful in SOV-to-SVO translation. [4] In any case, we are the first to predict directly *on the target side*, thus integrating anticipation in a single translation model.

Jaitly et al. (2016) propose an online neural transducer for speech recognition that is conditioned on prefixes. This problem does not have reorderings and thus no anticipation is needed.

8 Conclusions

We have presented a prefix-to-prefix training and decoding framework for simultaneous translation with integrated anticipation, and a wait-k policy that can achieve arbitrary word-level latency while maintaining high translation quality. This prefix-to-prefix architecture has the potential to be used in other sequence tasks outside of MT that involve simultaneity or incrementality. We leave many open questions to future work, e.g., adaptive policy using a single model (Zheng et al., 2019).

Acknowledgments

We thank Colin Cherry (Google Montreal) for spotting a mistake in AL (Eq. 8), Hao Zhang (Google NYC) for comments, the bilingual speakers for human evaluations, and the anonymous reviewers for suggestions.

[4] Their codebase on Github is not runnable, and their baseline is inconsistent with Gu et al. (2017) which we compared to, so we did not include their results for comparison.

References

Ashkan Alinejad, Maryam Siahbani, and Anoop Sarkar. 2018. Prediction improves simultaneous neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *Proc. of NAACL-HLT*.

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

T Fujita, Graham Neubig, Sakriani Sakti, T Toda, and S Nakamura. 2013. Simple, lexicalized choice of translation timing for simultaneous speech translation. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* .

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on empirical methods in natural language processing (EMNLP)*. pages 1342–1352.

Alvin Grissom II, Naho Orita, and Jordan Boyd-Graber. 2016. Incremental prediction of sentence-final verbs: Humans versus machines. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O. K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. pages 1053–1062. https://aclanthology.info/papers/E17-1099/e17-1099.

He He, Jordan Boyd-Graber, and Hal Daumé III. 2016. Interpretese vs. translationese: The uniqueness of human strategies in simultaneous interpretation. In *North American Association for Computational Linguistics*.

He He, Alvin Grissom II, Jordan Boyd-Graber, and Hal Daumé III. 2015. Syntax-based rewriting for simultaneous machine translation. In *Empirical Methods in Natural Language Processing*.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *EMNLP*.

Navdeep Jaitly, David Sussillo, Quoc V Le, Oriol Vinyals, Ilya Sutskever, and Samy Bengio. 2016. An online sequence-to-sequence model using partial conditioning. In *Advances in Neural Information Processing Systems*. pages 5067–5075.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints* .

S Matsubarayx, K Iwashimaz, N Kawaguchizx, K Toyama, and Yasuyoshi Inagaki. 2000. Simultaneous japanese-english interpretation based on early prediction of english verb .

Barbara Moser-Mercer, Alexander Künzli, and Marina Korac. 1998. Prolonged turns in interpreting: Effects on quality, physiological and psychological stress (pilot study). *Interpreting* 3(1):47–64.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Syntax-based simultaneous translation through prediction of unseen syntactic constituents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 198–207.

Ofir Press and Noah A. Smith. 2018. You may not need attention. https://arxiv.org/abs/1810.13409.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* .

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 30*.

Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the beam search curse: A study of (re-) scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019. Simultaneous translation with flexible policy via restricted imitation learning. In *ACL*.

Appendix

A Supplemental Material: Model Refinement with Catchup Policy

As mentioned in Sec. 3, the wait-k decoding is always k words behind the incoming source stream. In the ideal case where the input and output sentences have equal length, the translation will finish k steps after the source sentence finishes, i.e., the tail length is also k. This is consistent with human interpreters who start and stop a few seconds after the speaker starts and stops.

However, input and output sentences generally have different lengths. In some extreme directions such as Chinese to English, the target side is significantly longer than the source side, with an average gold tgt/src ratio, $r = \overline{|\mathbf{y}^\star|/|\mathbf{x}|}$, of around 1.25 (Huang et al., 2017; Yang et al., 2018). In this case, if we still follow the vanilla wait-k policy, the tail length will be $0.25|\mathbf{x}| + k$ which increases with input length. For example, given a 20-word Chinese input sentence, the tail of wait-3 policy will be 8 word long, almost half of the source length. This brings two negative effects: (a) as decoding progresses, the user will be effectively lagging behind further and further (becomes each Chinese word in principle translates to 1.25 English words), rendering the user more and more out of sync with the speaker; and (b) when a source sentence finishes, the rather long tail is displayed immediately, causing a cognitive burden on the user.[5] These problems become worse with longer input sentences (see Fig. 14).

To address this problem, we devise a "wait-k+catchup" policy so that the user is still k word behind the input in terms of real information content, i.e., always k source words behind the ideal perfect synchronization policy denoted by the diagonal line in Fig. 14. For example, assume the tgt/src ratio is $r = 1.25$, we will output 5 target words for every 4 source words; i.e., the catchup frequency, denoted $c = r - 1$, is 0.25. See Fig. 14.

More formally, with catchup frequency c, the new policy is:

$$g_{\text{wait-}k,\,c}(t) = \min\{k + t - 1 - \lfloor ct \rfloor,\ |\mathbf{x}|\} \quad (13)$$

and our decoding and training objectives change accordingly (again, we *train* the model to catchup using this new policy).

[5] It is true that the tail can in principle be displayed concurrently with the first k words of the next input, but the tail is now much longer than k.

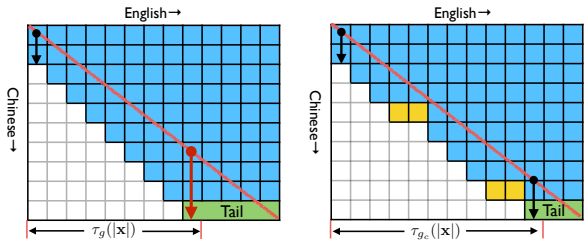

Figure 14: Left (wait-2): it renders the user increasingly out of sync with the speaker (the diagonal line denotes the ideal perfect synchronization). Right (+catchup): it shrinks the tail and is closer to the ideal diagonal, reducing the effective latency. Black and red arrows illustrate 2 and 4 words lagging behind the diagonal, resp.

On the other hand, when translating from longer source sentences to shorter targets, e.g., from English to Chinese, it is very possible that the decoder finishes generation before the encoder sees the entire source sentence, ignoring the "tail" on the source side. Therefore, we need "reverse" catchup, i.e., catching up on encoder instead of decoder. For example, in English-to-Chinese translation, we encode one extra word every 4 steps, i.e., encoding 5 English words per 4 Chinese words. In this case, the "decoding" catcup frequency $c = r - 1 = -0.2$ is negative but Eq. 13 still holds. Note that it works for any arbitrary c, such as 0.341, where the catchup pattern is not as easy as "1 in every 4 steps", but still maintains a rough frequency of c catchups per source word.

Fig. 15 shows the comparison between wait-k model and catchup policy which enables one extra word decoding on every 4^{th} step. For example, for wait-3 policy with catchup, the policy is R R (R W R W R W R W W)$^+$ W$^+$.

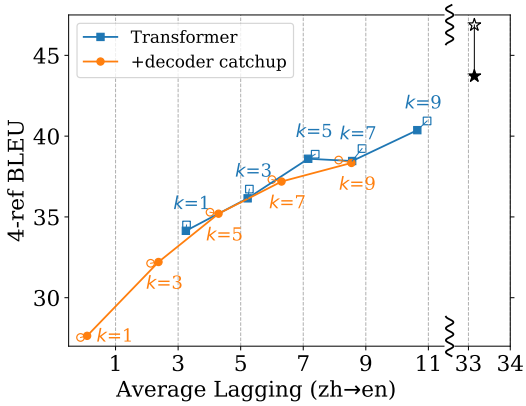

Figure 15: BLEU scores and AL comparisons with different wait-k models on Chinese-to-English on dev set. □ and ○ are decoded with tail beam search. ★☆ and ★☆ are greedy decoding and beam-search baselines.

B Supplemental Material: Evaluations with AP

We also evaluate our work using Average Proportion (AP) on both de↔en and zh↔en translation comparing with full sentence translation and Gu et al. (2017).

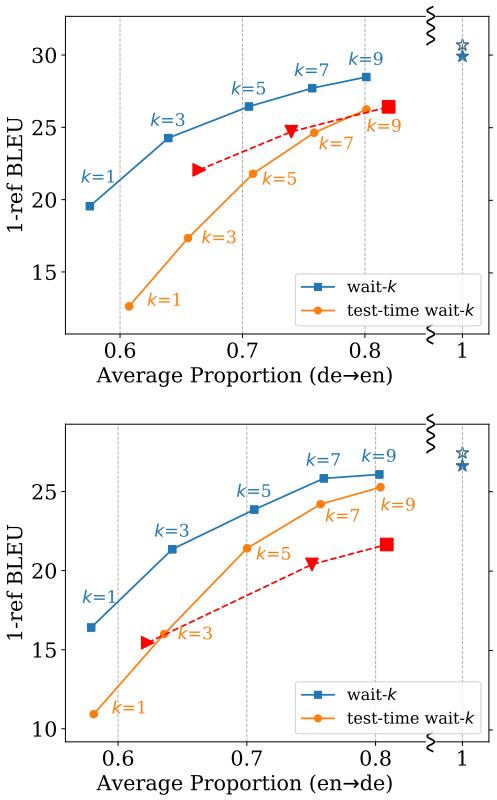

Figure 16: Translation quality against AP on de↔en simultaneous translation, showing wait-k models (for k=1, 3, 5, 7, 9), test-time wait-k results, full-sentence baselines, and our reimplementation of Gu et al. (2017), all based on the same Transformer. ★☆:full-sentence (greedy and beam-search), Gu et al. (2017): ▶:CW=2; ▼:CW=5; ■:CW=8.

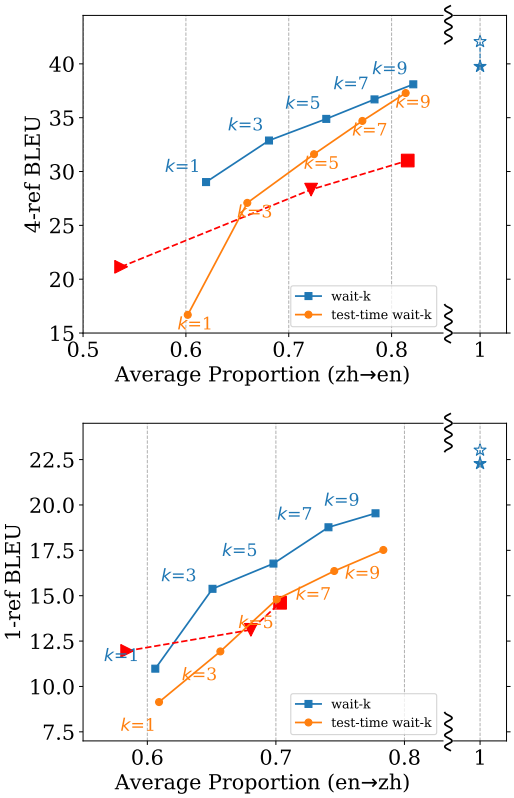

Figure 17: Translation quality against AP on zh↔en simultaneous translation, showing wait-k models (for k=1, 3, 5, 7, 9), test-time wait-k results, full-sentence baselines, and our reimplementation of Gu et al. (2017), all based on the same Transformer. ★☆:full-sentence (greedy and beam-search), Gu et al. (2017): ▶:CW=2; ▼:CW=5; ■:CW=8.