Anticipation-free Training for Simultaneous Translation

Anonymous ACL submission

Abstract

Simultaneous translation (SimulMT) speeds up the translation process by starting to translate before the source sentence is completely available. It is difficult due to limited context and word order difference between languages. Existing methods increase latency or introduce adaptive read-write policies for SimulMT models to handle local reordering and improve translation quality. However, the long-distance reordering would make the SimulMT models learn translation mistakenly. Specifically, the model may be forced to predict target tokens when the corresponding source tokens have not been read. This leads to aggressive anticipation during inference, resulting in the hallucination phenomenon. To mitigate this problem, we propose a new framework that decompose the translation process into the monotonic translation step and the reordering step, and we model the latter by the auxiliary sorting network (ASN). The ASN rearranges the hidden states to match the order in the target language, so that the SimulMT model could learn to translate more reasonably. The entire model is optimized end-to-end and does not rely on external aligners or data. During inference, ASN is removed to achieve streaming. Experiments show the proposed framework could outperform previous methods with less latency.[1]

1 Introduction

Simultaneous translation (SimulMT) is an extension of neural machine translation (NMT), aiming to perform streaming translation by outputting the translation before the source input has ended. It is more applicable to real-world scenarios such as international conferences, where people could communicate fluently without delay.

However, SimulMT faces additional difficulties compared to full-sentence translation – such a model needs to translate with limited context, and the different word order between languages would make streaming models learn translation mistakenly. The problems can often be alleviated by increasing the context. Using more context allows the model to translate with more information, trading off speed for quality. But the word order could be very different among languages. Increasing the context could only solve the local reordering problem. If long-distance reordering exists in training data, the model would be forced to predict tokens in the target language when the corresponding source tokens have not been read. this is called *anticipation* (Ma et al., 2019a). Ignoring the long-distance reordering may cause unnecessarily high latency, or encourage aggressive anticipation, resulting in the hallucination phenomenon (Müller et al., 2020).

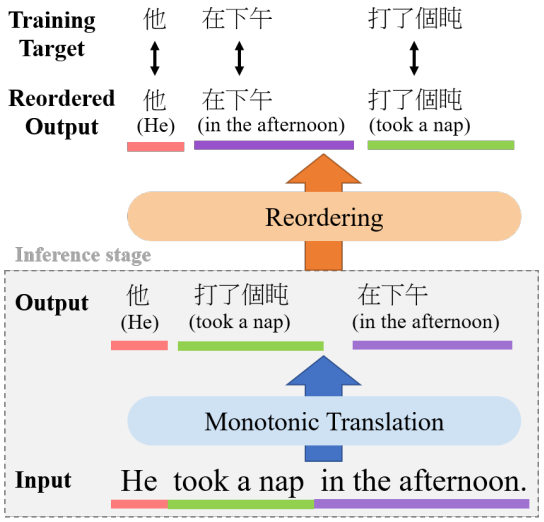


Figure 1: Illustration of the training process. The translated output is rearranged to match the order of training target, reducing anticipation. We use the gray part during inference.

It sheds light on the importance of matching the word order between the source and target languages. Existing methods aim to reduce anticipation by using syntax-based rules to rewrite the translation target (He et al., 2015). It requires addi-

[1]The source code is available. See Appendix A

tional language-specific prior knowledge and constituent parse trees. Other approaches pre-train a full-sentence model, then incrementally feed the source sentence to it to generate monotonic translation target (pseudo reference) (Chen et al., 2021b; Zhang et al., 2020). However, the full-sentence model was not trained to translate incrementally, which create a train-test mismatch, resulting in varying prediction quality. They require combining with the original data to be effective.

To this end, this work aims to address long-distance reordering by incorporating it directly into the training process, as Figure 1 shows. We decompose the typical translation process into the *monotonic translation* step and the *reordering* step. Inspired by the Gumbel-Sinkhorn network (Mena et al., 2018), we proposed an auxiliary sorting network (ASN) for the *reordering* step. During training, the ASN explicitly rearranges the hidden states to match the target language word order. The ASN will not be used during inference, so that the model could translate monotonically. The proposed method reduces anticipation, thus increases the lexical precision (He et al., 2015) of the model without compromising its speed. We apply the proposed framework to a simple model – a causal Transformer encoder trained with connectionist temporal classification (CTC) (Graves et al., 2006). The CTC loss can learn an adaptive policy (Chousa et al., 2019), which performs local reordering by predicting blank symbols until enough information is read, then write the information in the target order. Even so, it still suffers from high latency and under-translation due to long-distance reordering in training data. Our ASN handles these long-distance reordering, improving both the latency and the quality of the CTC model. We conduct experiments on CWMT English to Chinese and WMT15 German to English translation datasets. Our contributions are summarized below:

- We proposed a new framework for SimulMT. The ASN could apply on various causal models to handle long-distance reordering.
- Experiments showed that the proposed method could outperform the pseudo reference method. It indicated the proposed method could better handle the long-distance reordering.
- The proposed model is a causal encoder, which is parameter efficient and could outperform wait-k Transformer with less latency.

Our implementation is based on fairseq (Ott et al., 2019). The instructions to access our source code is provided in Appendix A.

2 Related Works

2.1 Simultaneous Translation

SimulMT is first achieved by applying fixed read-write policies on NMT models. Wait-if-worse and Wait-if-diff (Cho and Esipova, 2016) form decisions based on the next prediction's probability or its value. Static Read and Write (Dalvi et al., 2018) first read several tokens, then repeatedly read and write several tokens at a time. Wait-k (Ma et al., 2019a) trains end-to-end models for SimulMT. Its policy is similar to Static Read and Write.

On the other hand, adaptive policies seek to learn the read-write decisions. Some works explored training agents with reinforcement learning (RL) (Gu et al., 2017; Luo et al., 2017). Others design expert policies and apply imitation learning (IL) (Zheng et al., 2019a,b). Monotonic attention (Raffel et al., 2017) integrates the read-write policy into the attention mechanism to jointly train with NMT. MoChA (Chiu and Raffel, 2018) enhances monotonic attention by adding soft attention over a small window. MILk (Arivazhagan et al., 2019) extends such window to the full encoder history. MMA (Ma et al., 2019b) extends MILk to multi-head attention. Connectionist temporal classification (CTC) were also explored for adaptive policy by treating the blank symbol as wait action (Chousa et al., 2019). Recently, making read-write decisions based on segments of meaningful unit (MU) (Zhang et al., 2020) improves the translation quality. Besides, an adaptive policy can also be derived from an ensemble of fixed-policy models (Zheng et al., 2020).

When performing simultaneous interpretation, humans avoid long-distance reordering whenever possible (Al-Khanji et al., 2000; He et al., 2016). Thus, some works seek to reduce the anticipation in data to ease the training of simultaneous models. These include syntax-based rewriting (He et al., 2015), or generating pseudo reference by test-time wait-k (Chen et al., 2021b) and prefix-attention (Zhang et al., 2020). We reduce anticipation from a different approach: instead of rewriting the target, we let the model match its hidden states to the target on its own. As shown in experiments, our method is comparable or superior to the pseudo reference method.

2.2 Gumbel-Sinkhorn Network

The Sinkhorn Normalization (Adams and Zemel, 2011) is an iterative procedure that converts a matrix into doubly stochastic form. It was initially proposed to perform gradient-based rank learning. Gumbel-Sinkhorn Network (Mena et al., 2018) combines the Sinkhorn Normalization with the Gumbel reparametrization trick (Kingma and Welling, 2013). It approximates sampling from a distribution of permutation matrices. Subsequently, Sinkhorn Transformer (Tay et al., 2020) applied this method to the Transformer (Vaswani et al., 2017) to model long-distance dependency in language models with better memory efficiency. This work applies the Gumbel-Sinkhorn Network to model the reordering between languages, in order to reduce anticipation in SimulMT.

3 Proposed Method

For a source sentence $\mathbf{x} = \langle x_1, x_2, ..., x_{|\mathbf{x}|} \rangle$ and a target sentence $\mathbf{y} = \langle y_1, y_2, ..., y_{|\mathbf{y}|} \rangle$, in order to perform SimulMT, the conditional probability of translation $p(\mathbf{y}|\mathbf{x})$ is modeled by the prefix-to-prefix framework (Ma et al., 2019a). Formally,

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

where $g(t)$ is a monotonic non-decreasing function. This way, the t-th token $\hat{\mathbf{y}}_t$ can be predicted with a limited context $\mathbf{x}_{\leq g(t)}$. However, if long-distance reordering exists in the training data, the model is forced to generate target tokens whose corresponding source tokens have not been revealed yet. This issue is known as anticipation.

3.1 Training Framework

To overcome this, we introduce a latent variable $\mathbf{Z}$: a permutation matrix capturing the reordering process from $\mathbf{x}$ to $\mathbf{y}$. Thus, the translation probability can be expressed as a marginalization over $\mathbf{Z}$:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{Z}} \underbrace{p_g(\mathbf{y}|\mathbf{x}, \mathbf{Z})}_{\substack{\text{monotonic} \\ \text{translation}}} \underbrace{p(\mathbf{Z}|\mathbf{x})}_{\text{reordering}}. \quad (2)$$

During training, since $\mathbf{Z}$ captures reordering, the $p_g(\mathbf{y}|\mathbf{x}, \mathbf{Z})$ corresponds to monotonic translation, which can be correctly modeled by a prefix-to-prefix model without anticipation. During inference, we can translate monotonically by simply removing the effect of $\mathbf{Z}$:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p_g(\mathbf{y}|\mathbf{x}, \mathbf{Z} = \mathbf{I}). \quad (3)$$

where $\mathbf{I}$ is the identity matrix. However, equation 2 is intractable due to the factorial search space of permutations. One could select the most likely permutation using an external aligner (Ran et al., 2021), but such a method requires an external tool, and it could not be end-to-end optimized. Instead, we use the ASN to learn the permutation matrix $\mathbf{Z}$ associated with source-target reordering. By doing this, the entire model is optimized end-to-end.

Figure 2 shows the proposed framework applied on the CTC model. It is composed of a causal Transformer encoder, an ASN, and a length projection network. We describe each component in detail below.

3.2 Causal Encoder

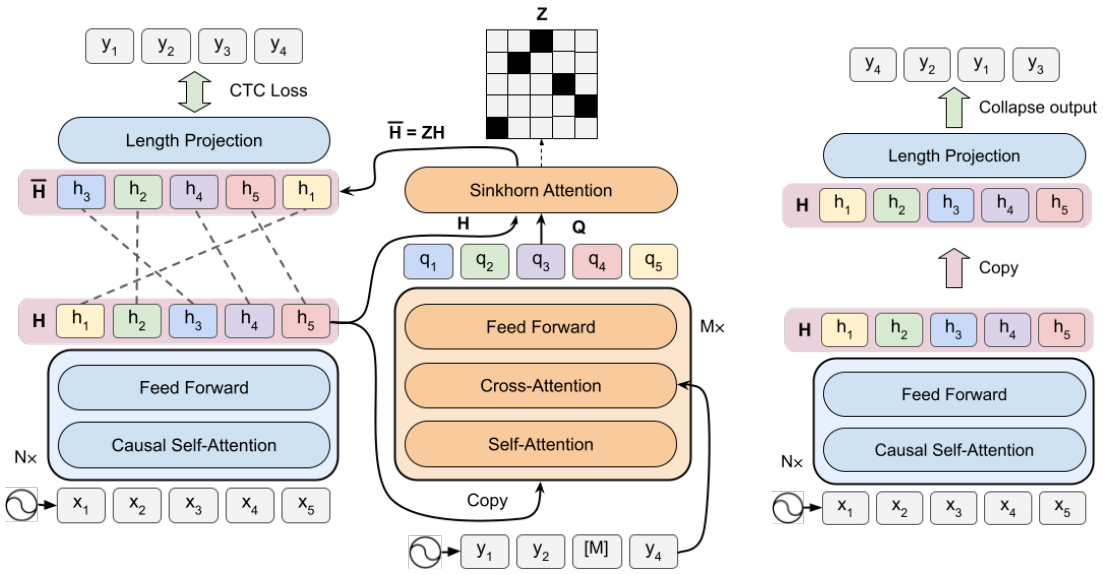
The encoder maps the source sequence $\mathbf{x}$ to hidden states $\mathbf{H} = \langle h_1, h_2, ..., h_{|\mathbf{x}|} \rangle$. During training, the encoder uses a causal attention mask so that it can be streamed during inference. To enable the trade-off between quality and latency, we introduce a tunable delay in the causal attention mask of the first encoder layer. We define the delay in a similar sense to wait-k: For delay-k, the t-th hidden state h_t is computed after observing the $(t + k - 1)$-th source token.

We pre-train the encoder with CTC loss (Libovický and Helcl, 2018). Since the CTC is an adaptive policy already capable of local reordering, initializing from it encourages the ASN to only handle long-distance reordering. We study the effectiveness of this technique in Section 5.2.

3.3 Auxiliary Sorting Network (ASN)

The ASN samples a permutation matrix $\mathbf{Z}$, which would sort the encoder hidden states $\mathbf{H}$ into the target order. To do so, the ASN first computes intermediate variables $\mathbf{Q} = \langle q_1, q_2, ..., q_{|\mathbf{x}|} \rangle$ using a stack of M non-causal Transformer decoder layers. These layers use the target token embeddings as the context for cross attention. Providing this context guides the reordering process[2], inspired by the word alignment task (Zhang and van Genabith, 2021; Chen et al., 2021a). We randomly mask out

[2] Although ASN has decoder layers and takes target tokens as input, which are unavailable during inference, they are only used to assist training.



(a) The model consists of a causal encoder (lower left, blue), an ASN (right, orange), and a length projection network (upper left, blue).

(b) During inference, only the encoder and length projection (blue) are used.

Figure 2: The architecture of the proposed model. Add & Norm layers are omitted for simplicity.

$\gamma\%$ of the context in ASN to avoid collapsing to a trivial solution.

Subsequently, the Sinkhorn Attention in ASN computes the attention scores between $\mathbf{Q}$ and $\mathbf{H}$ using the scaled dot-product attention:

$$\mathbf{A} = \frac{\mathbf{Q}\mathbf{H}^T}{\sqrt{d_h}}, \tag{4}$$

where d_h is the last dimension of $\mathbf{H}$. To convert the attention scores $\mathbf{A}$ to a permutation matrix $\mathbf{Z}$, ASN applies the Gumbel-Sinkhorn operator. Such operator approximates sampling from a distribution of permutation matrices (Mena et al., 2018). It is described by first adding the Gumbel noise (equation 5), then scaling by a positive temperature τ, and finally applying the l-iteration Sinkhorn normalization (denoted by $S^l(\cdot)$) (Adams and Zemel, 2011). We also add a scaling factor δ to adjust the Gumbel noise level (equation 6). The output would be doubly stochastic (Sinkhorn, 1964), which is a relaxation of permutation matrix. We leave the detailed description of the Gumbel-Sinkhorn operator in Appendix F.

$$\boldsymbol{\mathcal{E}} \in \mathbb{R}^{N\times N} \overset{i.i.d.}{\sim} Gumbel(0,1), \tag{5}$$

$$\mathbf{Z} = S^l\left(\left(\mathbf{A} + \delta\boldsymbol{\mathcal{E}}\right)/\tau\right), \tag{6}$$

Next, we use a matrix multiplication of $\mathbf{Z}$ and $\mathbf{H}$ to reorder $\mathbf{H}$, the result is denoted by $\overline{\mathbf{H}}$:

$$\overline{\mathbf{H}} = \mathbf{Z}\mathbf{H} \tag{7}$$

Since $\mathbf{Z}$ approximates a permutation matrix, using matrix multiplication is equivalent to permuting the vectors in $\mathbf{H}$. This preserves the content of its individual vectors, and is essential to our method as we will show in Section 5.1.

3.4 Length Projection

To optimize the model with CTC loss function, we tackle the length mismatch between $\overline{\mathbf{H}}$ and $\mathbf{y}$ by projecting $\overline{\mathbf{H}}$ to a μ-times longer sequence via an affine transformation (Libovický and Helcl, 2018). The μ represents the upsample ratio. For ASN to learn reordering effectively, it is required that the projection network and the loss must not perform reordering. Our length projection is time-independent, and CTC is monotonic, both satisfy our requirement.

3.5 Inference Strategy

To enable streaming, we remove the ASN during inference[3] (Figure 2(b)). Specifically, when a new input token x_t arrives, the encoder computes the hidden state h_t, then we feed h_t directly to the length projection to predict the next token(s). The prediction is post-processed by the CTC collapse function in an online fashion. Namely, we only output a new token if 1) it is not the blank symbol and 2) it is different from the previous token.

[3]While this seemingly creates a train-test discrepancy, we address this in FAQ

4 Experiments

4.1 Datasets

We conduct experiments on English-Chinese and German-English datasets. For En-Zh, we use a subset[4] of CWMT (Chen and Zhang, 2019) parallel corpora as training data (7M pairs). We use NJU-newsdev2018 as the development set and report results on CWMT2008, CWMT2009, and CWMT2011. The CWMT test sets have up to 3 references. Thus we report the 3-reference BLEU score. For De-En, we use WMT15 (Bojar et al., 2015) parallel corpora as training data (4.5M pairs). We use newstest2013 as the development set and report results on newstest2015.

We use SentencePiece (Kudo and Richardson, 2018) on each language separately to obtain its vocabulary of 32K subword units. We filter out sentence pairs that have empty sentences or exceed 1024 tokens in length.

4.2 Experimental Setup

All SimulMT models use causal encoders. During inference, the encoder states are computed incrementally after each read, similar to (Elbayad et al., 2020). The causal encoder models follow a similar training process to non-autoregressive translation (NAT) (Gu et al., 2018; Libovický and Helcl, 2018; Lee et al., 2018; Zhou et al., 2019). We adopt sequence level knowledge distillation (Seq-KD) (Kim and Rush, 2016) for all systems. The combination of Seq-KD and CTC loss has been shown to achieve state-of-the-art performance (Gu and Kong, 2020) and could deal with the reordering problem (Chuang et al., 2021). Specifically, we first train a full-sentence model as a teacher model on the original dataset, then we use beam search with beam width 5 to decode the Seq-KD set. We use the Seq-KD set in subsequent experiments. We list the Transformer and ASN hyperparameters separately in Appendix C and D.

We use Adam (Kingma and Ba, 2014) with an inverse square root schedule for the optimizer. The max learning rate is 5e-4 with 4000 warm-up steps. We use gradient accumulation to achieve an effective batch size of 128K tokens for the teacher model and 32K for others. We optimize the model with the 300K steps. Early stopping is applied when the validation BLEU does not improve within 25K steps. Label smoothing (Szegedy et al., 2016) with $\epsilon_{ls} = 0.1$ is applied on cross-entropy and CTC loss. For CTC, this reduces excessive blank symbol predictions (Suyoun et al., 2017). Random seeds are set in training scripts in our source code. For the hardware information and environment settings, see Appendix E.

For latency evaluation, we use SimulEval (Ma et al., 2020a) to compute Average Lagging (AL) (Ma et al., 2019a) and Computation Aware Average Lagging (AL-CA) (Ma et al., 2020b). AL is measured in words or characters, whereas AL-CA is measured in milliseconds. We describe these metrics in detail in Appendix G. For quality evaluation, we use BLEU (Papineni et al., 2002) calculated by SacreBLEU (Post, 2018). We conduct statistical significance test for BLEU using paired bootstrap resampling (Koehn, 2004). For multiple references, we use the first reference to run SimulEval[5] and use all available references to run SacreBLEU. The language-specific settings for SimulEval and SacreBLEU can respectively be found in Appendix H and I.

4.3 Baselines

We compare our method with two target rewrite methods which generate new datasets:

- **Pseudo reference** (Chen et al., 2021b): This approach first trains a full-sentence model and uses it to generate monotonic translation. The approach applies the test-time wait-k policy (Ma et al., 2019a), and performs beam search with beam width 5 to generate pseudo references. The pseudo reference set is the combination of original dataset and the pseudo references. We made a few changes 1) instead of the full-sentence model, we use the wait-9 model[6]. 2) instead of creating a new dataset for each k, we only use $k = 9$ since it has the best quality.
- **Reorder**: We use the word alignments to reorder the target sequence. We use *awesome-align* (Dou and Neubig, 2021) to obtain word alignments on the Seq-KD set, and we sort the target tokens based on their corresponding source tokens. Target tokens that did not align to a source token are placed at the position after their preceding target token.

[4] We use casia2015, casict2011, casict2015, neu2017.

[5] we use SimulEval for latency metrics only. Only one reference is required to run it.

[6] our wait-9 model has higher training set BLEU score than applying test-time wait-k on full-sentence model.

We train two types of models on either the Seq-KD set, the pseudo reference set or the reorder set:

- **wait-k**: an encoder-decoder model. It uses a fixed policy that first reads k tokens, then repeatedly reads and writes a single token.
- **CTC**: a causal encoder trained with CTC loss. The policy is adaptive, i.e., it outputs blank symbols until enough content is read, outputs the translated tokens, then repeats.

4.4 Quantitative Results

Figure 3 shows the latency-quality trade-off on the CWMT dataset, each node on a line represents a different value of k. Due to space limit, the significant test results are reported in Appendix J.

First of all, although the vanilla CTC model has high latency in terms of AL, they are comparable to or faster than the wait-k model according to AL-CA. This is due to the reduced parameter size. Besides, CTC models outperform wait-k in low latency settings. The pseudo reference method improves the quality of wait-k and CTC models, and it slightly improves the latency of the CTC model. In contrast, the reorder method harms the performance of both models. Meanwhile, our method significantly improves both the quality and latency of the CTC model across all latency settings, outperforming the pseudo reference method and the reorder method. In particular, our $k = 1, 3$ models outperform wait-1 by around 13-15 BLEUs with a faster speed in terms of AL-CA. This shows that our models are more efficient than wait-k models under low latency regimes.

Figure 4 shows the latency-quality trade-off on the WMT15 De-En dataset. The vanilla CTC model is much more competitive in De-En. It outperforms vanilla wait-k in low latency settings in BLEU and AL-CA, and its AL is much less than those in En-Zh. Our method improves the quality of the CTC model, comparable to the pseudo reference method. However, our method does not require combining with the original dataset to improve the performance.

To understand why our method is more effective on CWMT, we calculate the k-Anticipation Rate (k-AR) (Chen et al., 2021b) on the evaluation sets of both datasets. For the definition of k-AR, see Appendix G. Intuitively, k-AR describes the amount of anticipation (or reordering) in the corpus whose range is longer than k source tokens. We report k-AR across $1 \leq k \leq 9$ in Figure 5. En-Zh has much higher k-AR in general, and it decreases slower as k increases. When $k = 9$, over 20% of anticipations remain in En-Zh, while almost none remains in De-En. We conclude that En-Zh has much more reordering, and over 20% of them are longer than 9 words. The abundance of long-distance reordering gives our method an advantage, which explains the big improvement observed on CWMT. On the other hand, De-En reordering is less common and mostly local, so ASN has limited effect. Indeed, we found that ASN predicts matrices close to the identity matrix on De-En, whereas, on En-Zh, it predicts non-identity matrices throughout training.

4.5 Qualitative Results

We show some examples from the CWMT test set. We compare the predictions from wait-k, CTC, and CTC+ASN models in Figure 6. In the first example, wait-k predicts the sentence *"demonstrative is one of the major languages in the world's languages,"* which is clearly hallucination. CTC failed to translate *"8000"* and *"assets,"* which shows that CTC may under-translate and ignore source information. In the second example, wait-k hallucinates the sentence *"this is the world's best contest, but to a earthquake without earthquake, it's the opening remarks."* CTC under-translates *"silver said in a telephone interview."* Our method generally provides translation that preserves the content. Although our model prediction is a bit less fluent than wait-k, they are generally comprehensible. See Appendix N for more examples.

We study the output of the ASN to verify that reordering information is being learned. Figure 7 shows an example of the permutation matrix $\mathbf{Z}$ predicted by the ASN. The horizontal axis is labeled with the source tokens. The vertical axis is the output positions, each are labeled with 2 target tokens (due to the length projection). In the example, the English phrase *"for all green hands"* come late in the source sentence, but their corresponding Chinese tokens appear early in target, which causes anticipation. Our ASN permutes the hidden states of this phrase to early positions, so anticipation no longer happens, and provides the correct training signal for the model. We provide additional examples in Appendix M.

5 Ablation Study

We perform ablation studies on the CWMT dataset.

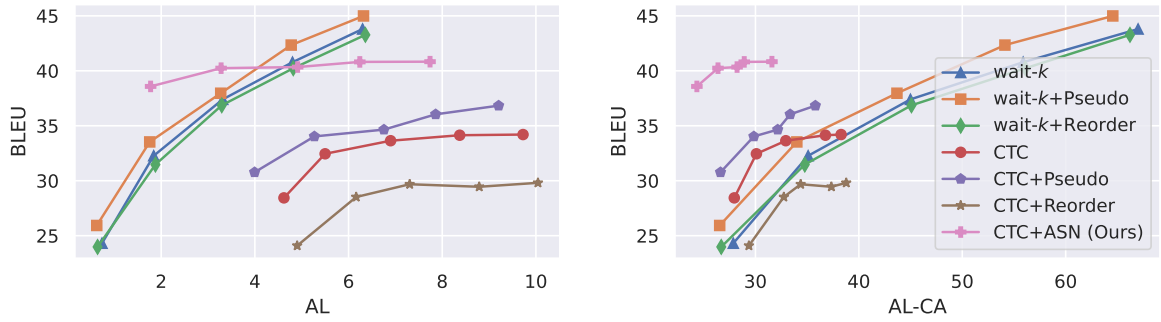


Figure 3: Latency-quality trade off on the CWMT En-Zh dataset. Each line represents a system, and the 5 nodes from left to right corresponds to $k = 1, 3, 5, 7, 9$. The figures share the legend.

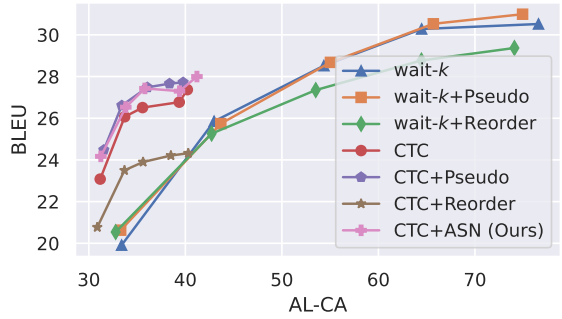


Figure 4: Latency-quality trade off on the WMT15 De-En dataset.

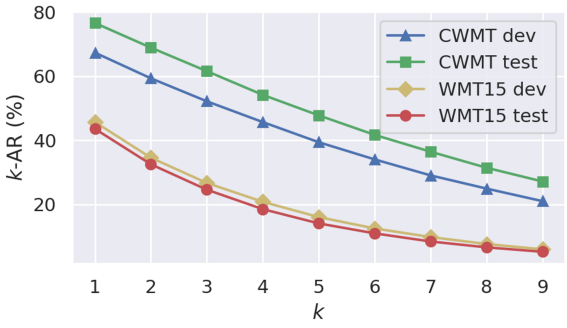


Figure 5: The k-anticipation rate computed on CWMT En-Zh and WMT15 De-En development and test sets.

5.1 Gumbel-Sinkhorn Network

We show that the Gumbel-Sinkhorn Network is crucial to our method. We train CTC+ASN models with $k = 3$ under the following settings:[7]

- **No temperature**: Set the temperature τ to 1.
- **No noise**: Set the Gumbel noise factor δ to 0.
- **Gumbel softmax**: Replace Sinkhorn normalization with softmax.
- **Default**: The Gumbel-Sinkhorn Network.

[7] we do not use weight initialization in this subsection.

Table 1 shows the result of these settings. Without low temperature, the ASN output $\mathbf{Z}$ is not sparse, which means the content of individual vectors in $\mathbf{H}$ is not preserved after applying ASN. Because ASN is removed during inference, this creates a train-test mismatch for the projection network, which is detrimental to the prediction quality ((a) v.s. (d)). Removing the noise ignores the sampling process, which hurts the robustness of the model ((b) v.s. (d)). Using softmax instead of Sinkhorn normalization makes $\mathbf{Z}$ not doubly stochastic, which means $\overline{\mathbf{H}}$ might not cover every vector in $\mathbf{H}$. Those not covered are not optimized for generation during training. However, during inference, all vectors in $\mathbf{H}$ are passed to length projection to generate tokens. This mismatch is also harmful to the result ((c) v.s. (d)).

Settings	BLEU(↑)
(a) No temperature	28.39
(b) No noise	27.88
(c) Gumbel softmax	36.54
(d) Default	**38.92**

Table 1: Test set BLEU scores of different settings.

5.2 Weight Initialization

We investigate the effectiveness of initializing encoder parameters from the CTC baseline model. Specifically, we train the CTC+ASN model from scratch to compare it with the weight initialized setting. As Figure 8 reveals, the weight initialization significantly improves the translation quality while slightly increasing the latency.

This improvement comes from what was already learned by the CTC baseline model. The CTC baseline model learns to perform reordering, i.e., it

Input	the adic is one of the world's richest sovereign funds, with an estimated $800bn of assets under management.
wait-*k*	指示语 是 语言 世界的 主要 语言 之一，它 被 许多 最富有的 投资者 估计为 800亿 美元 资产。 demonstrative is language world's major language one of, it by many richest investor estimated 80billion USD asset.
CTC	迪奇 是 是 世界上 最富有的 主权 基金 之一，估计 亿 美元 的 管理 迪奇 is is world's richest sovereign funds one of, estimated (SOS) 0.1 billion USD 's (exec) management
CTC+ASN	ad陀 是 是 世界上 最富有的 主权 基金 之一，估计 为 000亿 美元 的 资产 正在 管理 中 ad陀 is is world's richest sovereign funds one of, estimated is (8)00billion USD 's asset under management (under)
Input	it's the opening up of cracks before an earthquake, silver said in a telephone interview.
wait-*k*	“这 是 世界上 最好的 比赛，但 对于 一个 没有 地震的 地震 中 银 来说，这 是 开场白。 this is world's best contest, but for a without earthquake earthquake in silver (for), this is opening remarks.
CTC	“这 是 地震 前 裂缝 裂缝 开放 this is earthquake before crack crack opening up
CTC+ASN	“这 是 开放的 裂缝 地震 前,” 西尔弗 说 :“ 在 在 一次 电话 采访 中 this is open crack earthquake before silver said in in a telephone interview (in)

Figure 6: Examples from CWMT En→Zh. Text in red are hallucinations unrelated to source. We use $k = 3$ models.

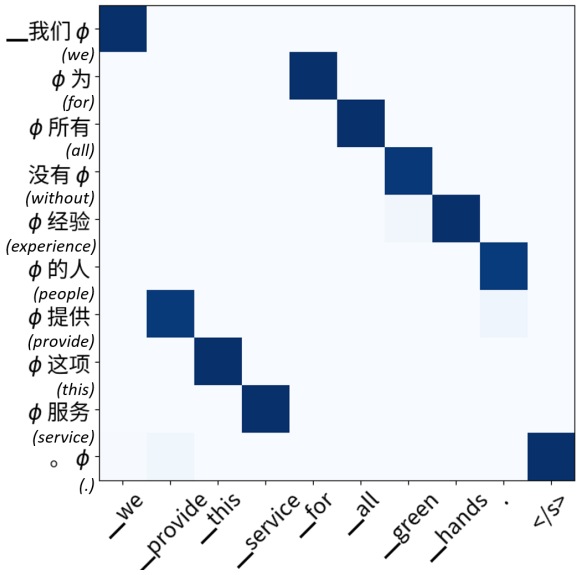


Figure 7: The **Z** predicted by ASN. The horizontal axis is the source tokens. The vertical axis is the output positions, each corresponds to 2 target tokens.

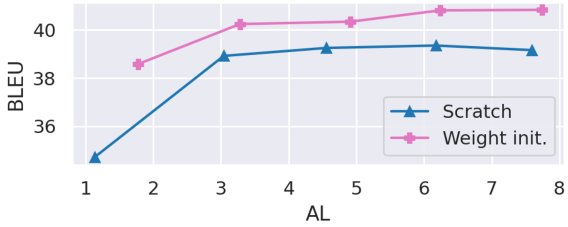


Figure 8: Latency and quality comparison between the model trained from scratch and one with weight initialization.

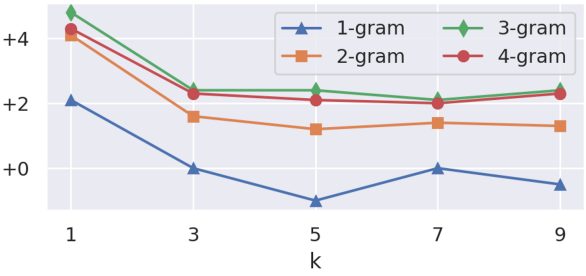


Figure 9: The n-gram precision improvement of weight initialization compared to Scratch across different delays (k).

outputs blank symbols when reading the information, then outputs the content in the target language order. Such information might span several source tokens, so the AL of the CTC baseline model is high (Figure 3). In our weight initialized setting, ASN handles the long-distance reordering that CTC was struggling with, while the local reordering already learned by CTC is preserved. In contrast, when trained from scratch, ASN would learn most of the reordering, so the encoder would not learn to perform local reordering. We hypothesize that if the model performs local reordering during inference, its latency might increase, but the higher order n-grams precision can improve, which benefits its quality. Indeed, Figure 9 indicates that the weight initialization mostly improves the 2,3,4-gram precision of the BLEU score.

6 Conclusion

We proposed a framework to alleviate the impact of long-distance reordering on simultaneous translation. We apply our method to the CTC model and show that it improves the translation quality and latency, especially English to Chinese translation. We verified that the ASN indeed learns the correct alignment between source and target. Besides, we showed that a single encoder can perform simultaneous translation with competitive quality in low latency settings and enjoys the speed advantage over wait-k Transformer.

References

Ryan Prescott Adams and Richard S Zemel. 2011. Ranking via sinkhorn propagation. *arXiv preprint arXiv:1106.1925*.

Raja Al-Khanji, Said El-Shiyab, and Riyadh Hussein. 2000. On the use of compensatory strategies in simultaneous interpretation. *Meta: Journal des traducteurs/Meta: Translators' Journal*, 45(3):548–557.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Ond rej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Chi Chen, Maosong Sun, and Yang Liu. 2021a. Mask-align: Self-supervised neural word alignment. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4781–4791, Online. Association for Computational Linguistics.

Jiajun Chen and Jiajun Zhang. 2019. *Machine Translation: 14th China Workshop, CWMT 2018, Wuyishan, China, October 25-26, 2018, Proceedings*, volume 954. Springer.

Junkun Chen, Renjie Zheng, Atsuhito Kita, Mingbo Ma, and Liang Huang. 2021b. Improving simultaneous translation by incorporating pseudo-references with fewer reorderings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5857–5864.

Chung-Cheng Chiu and Colin Raffel. 2018. Monotonic chunkwise attention. In *International Conference on Learning Representations*.

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

Katsuki Chousa, Katsuhito Sudoh, and Satoshi Nakamura. 2019. Simultaneous neural machine translation using connectionist temporal classification.

Shun-Po Chuang, Yung-Sung Chuang, Chih-Chiang Chang, and Hung-yi Lee. 2021. Investigating the reordering capability in CTC-based non-autoregressive end-to-end speech translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1068–1077, Online. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *NAACL-HLT (2)*.

Zi-Yi Dou and Graham Neubig. 2021. Word alignment by fine-tuning embeddings on parallel corpora. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

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

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu and Xiang Kong. 2020. Fully non-autoregressive neural machine translation: Tricks of the trade. *arXiv preprint arXiv:2012.15833*.

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

He He, Jordan Boyd-Graber, and Hal Daumé III. 2016. Interpretese vs. translationese: The uniqueness of human strategies in simultaneous interpretation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 971–976.

He He, Alvin Grissom II, John Morgan, Jordan Boyd-Graber, and Hal Daumé III. 2015. Syntax-based rewriting for simultaneous machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 55–64.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 388–395.

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

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Jindřich Libovickỳ and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021. Association for Computational Linguistics.

Yuping Luo, Chung-Cheng Chiu, Navdeep Jaitly, and Ilya Sutskever. 2017. Learning online alignments with continuous rewards policy gradient. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2801–2805. IEEE.

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

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020a. Simuleval: An evaluation toolkit for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 144–150.

Xutai Ma, Juan Pino, and Philipp Koehn. 2020b. Simulmt to simulst: Adapting simultaneous text translation to end-to-end simultaneous speech translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 582–587.

Xutai Ma, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2019b. Monotonic multihead attention. In *International Conference on Learning Representations*.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. 2018. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations*.

Mathias Müller, Annette Rios Gonzales, and Rico Sennrich. 2020. Domain robustness in neural machine translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (AMTA 2020)*, pages 151–164.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

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

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

Maja Popović. 2016. chrf deconstructed: beta parameters and n-gram weights. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 499–504.

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

Colin Raffel, Minh-Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *International Conference on Machine Learning*, pages 2837–2846. PMLR.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2021. Guiding non-autoregressive neural machine translation decoding with reordering information. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13727–13735.

Richard Sinkhorn. 1964. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879.

Kim Suyoun, L Seltzer Michael, Jinyu Li, and Rui Zhao. 2017. Improved training for online end-to-end speech recognition systems. In *INTERSPEECH*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR.

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

Jingyi Zhang and Josef van Genabith. 2021. A bidirectional transformer based alignment model for unsupervised word alignment. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 283–292, Online. Association for Computational Linguistics.

Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2020. Learning adaptive segmentation policy for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2280–2289.

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020. Simultaneous translation policies: From fixed to adaptive. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2847–2853, Online. Association for Computational Linguistics.

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

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

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2019. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations*.

A Source Code

Our source code is available on Anonymous Github at https://anonymous.4open.science/r/sinkhorn-simultrans, and will be made publicly available on Github upon acceptance. Please follow the instructions in `README.md` to reproduce the results.

B Datasets

We use the CWMT English to Chinese and WMT15 German to English datasets for experiments. They can be downloaded in the following links: 1) CWMT http://nlp.nju.edu.cn/cwmt-wmt/) 2) WMT15 http://www.statmt.org/wmt15/translation-task.html. The WMT15 De-En is a widely used corpus for simultaneous machine translation, in the news domain. Another popular dataset is the NIST En-Zh corpus, however, NIST is not publicly available, thus we use CWMT corpus instead. CWMT is also in the news domain.

Both datasets are publicly available. We didn't find any license information for both. We adhered to the terms of use for both. We didn't find any information on names or uniquely identified individual people or offensive content and the steps taken to protect or anonymize them.

C Transformer Hyperparameters

Our architecture related hyperparameters are listed in Table 2. We follow the *base* configuration of Transformer for encoder-decoder models. For models without decoder, we follow the same configuration for its encoder. The total parameter count for Transformer is 76.9M. For encoder-only models without ASN, it is 52.2M. The ASN has 12.6M parameters.

Hyperparameter	(A)	(B)
encoder layers	6	6
decoder layers	6	0
embed dim	512	512
feed forward dim	2048	2048
num heads	8	8
dropout	0.1	0.1

Table 2: Transformer architecture related hyperparameters for each model. (A) full-sentence and wait-k model (B) CTC encoder model.

D ASN Hyperparameters

We perform a Bayesian hyperparameter optimization on both datasets using the sweep utility provided by Weights & Biases (Biewald, 2020). Table 3 shows the search range and the selected values. We found a well performing set in the 7th run for CWMT and 1st run for WMT15. It is possible that different k might prefer different hyperparameters. However, we use the same set to fairly compare to wait-k, and to reduce the cost. All subsequent results are obtained using this set of values if not specified.

Hyperparameter	CWMT	WMT15	Range
layers M	3	3	1, 3
iterations l	16	16	4, 8, 16
temperature τ	0.25	0.13	[0.05, 0.3]
noise factor δ	0.3	0.45	[0.1, 0.3]
upsample ratio μ	2	2	2, 3
mask ratio γ	0.5	0.5	[0., 0.7]

Table 3: ASN related hyperparameters and the search range. We use Bayesian hyperparameter optimization, so the combinations are not exhaustively searched.

E Hardware and Environment

For training, each run are conducted on a container with a single Tesla V100-SXM2-32GB GPU, 4 CPU cores and 90GB memory. The operating system is `Linux-3.10.0-1127.el7.x86_64-x86_64-with-glibc2.10`. The version of Python is 3.8.10, and version of PyTorch is 1.9.0. We use a specific version of fairseq (Ott et al., 2019) toolkit, the instructions are provided in `README.md` of our source code. All run uses mixed precision (i.e. fp16) training implemented by fairseq. All training took 10-15 hours to converge (early stopped).

For inference, the evaluation are conducted on another machine with 12 CPU cores (although we restrict the evaluation to only use 2 threads), 32GB memory and no GPU is used. The operating system is `Linux-5.11.0-25-generic-x86_64-with-glibc2.10`.

F Gumbel-Sinkhorn Operator

The Sinkhorn normalization (Adams and Zemel, 2011) iteratively performs row-wise and column-wise normalization on a matrix, converting it to a

doubly stochastic matrix. Formally, for a N dimensional square matrix $X \in \mathbb{R}^{N \times N}$, the Sinkhorn normalization $S(X)$ is defined as:

$$S^0(X) = \exp(X), \tag{8}$$

$$S^l(X) = \mathcal{T}_c\left(\mathcal{T}_r\left(S^{l-1}(X)\right)\right), \tag{9}$$

$$S(X) = \lim_{l \to \infty} S^l(X). \tag{10}$$

where $\mathcal{T}_r$ and $\mathcal{T}_c$ are row-wise and column-wise normalization operators on a matrix, defined below:

$$\mathcal{T}_r(X) = X \oslash (X\mathbf{1}_N\mathbf{1}_N^\top), \tag{11}$$

$$\mathcal{T}_c(X) = X \oslash (\mathbf{1}_N\mathbf{1}_N^\top X). \tag{12}$$

The $\oslash$ denotes the element-wise division, and $\mathbf{1}_N$ denotes a column vector full of ones. As the number of iterations l grows, $S^l(X)$ will eventually converge to a doubly stochastic matrix (equation 10) (Sinkhorn, 1964). In practice, we often consider the truncated version, where l is finite.

On the other hand, the Gumbel-Sinkhorn operator adds the Gumbel reparametrization trick (Kingma and Welling, 2013) to the Sinkhorn normalization, in order to approximate the sampling process. It can be used to estimate marginal probability via sampling. Formally, suppose that a noise matrix ε is sampled from independent and identically distributed (i.i.d.) Gumbel distributions:

$$\boldsymbol{\mathcal{E}} \in \mathbb{R}^{N \times N} \overset{i.i.d.}{\sim} Gumbel(0, 1). \tag{13}$$

The Gumbel-Sinkhorn operator is described by first adding the Gumbel noise $\boldsymbol{\mathcal{E}}$, then scaling by a positive temperature τ, and finally applying the Sinkhorn normalization:

$$S((X + \boldsymbol{\mathcal{E}})/\tau). \tag{14}$$

By taking the limit $\tau \to 0^+$, the output converges to a permutation matrix. The Gumbel-Sinkhorn operator approximates sampling from a distribution of permutation matrices. Thus, the equation 2 can be estimated through sampling:

$$p(\mathbf{y}|\mathbf{x}) = \mathbb{E}_{\mathbf{Z} \sim p(\mathbf{Z}|\mathbf{x})}\left[p_g(\mathbf{y}|\mathbf{x}, \mathbf{Z})\right]. \tag{15}$$

In practice, we sample from $p(\mathbf{Z}|\mathbf{x}, \mathbf{y})$ instead, as it is easier to perform word alignment ($p(\mathbf{Z}|\mathbf{x}, \mathbf{y})$) than directly predicting order ($p(\mathbf{Z}|\mathbf{x})$).

G Details on Evaluation Metrics

G.1 Average Lagging (AL)

The AL measures the degree the user is out of sync with the speaker (Ma et al., 2019a). It measures the system's lagging behind an oracle wait-0 policy. For a read-write policy $g(\cdot)$, define the cut-off step $\tau_g(|\mathbf{x}|)$ as the decoding step when source sentence finishes:

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

Then the AL for an example $\mathbf{x}, \mathbf{y}$ is defined as:

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

The second term in the summation represents the ideal latency of an oracle wait-0 policy in terms of target words (or characters for Chinese). The AL averaged across the test set is reported.

G.2 Computation Aware Average Lagging (AL-CA)

Originally proposed for simultaneous speech-to-text translation (Ma et al., 2020b), the AL-CA is similar to AL, but takes the actual computation time into account, and is measured in milliseconds.

$$\mathrm{AL}_g^{CA}(\mathbf{x}, \mathbf{y}) = \frac{1}{\tau_g(|\mathbf{x}|)} \sum_{i=1}^{\tau_g(|\mathbf{x}|)} d_{CA}(y_i) - \frac{(i-1) \cdot T_s}{|\mathbf{y}|/|\mathbf{x}|} \tag{16}$$

The $d_{CA}(y_i)$ is the the time that elapses from the beginning of the process to the prediction of y_i, **which considers computation**. T_s represents the actual duration of each source feature. The second term in the summation represents the ideal latency of an oracle wait-0 policy in terms of milliseconds, **without considering computation**. In speech-to-text translation, T_s corresponds to the duration of each speech feature. However, since our source feature is text, the "actual duration" for a word is unavailable, so we set $T_s = 1$.

The motivation behind using AL-CA here is to show the speed advantage of CTC models. When calculating AL-CA, we account for variance by running the evaluation 3 times and report the average.

G.3 Character n-gram F-score (chrF)

The general formula for the chrF score is given by:

$$\text{chrF}\beta = (1+\beta^2)\frac{\text{chrP}\cdot\text{chrR}}{\beta^2\cdot\text{chrP}+\text{chrR}}. \quad (17)$$

where

- chrP: percentage of character n-grams in the hypothesis which have a counterpart in the reference.
- chrR: percentage of character n-grams in the reference which are also present in the hypothesis.
- β: a parameter which assigns β times more importance to recall than to precision.

The maximum n-gram length N is optimal when $N = 6$ (Popović, 2015), and the optimal β is shown to be $\beta = 2$ (Popović, 2016).

The motivation behind using chrF2 is that 1) as machine translation researchers, we are encouraged to report multiple automatic evaluation metrics. 2) BLEU is purely precision-based, while chrF2 is F-score based, which takes recall into account. 3) chrF2 is shown to correlate better with human rankings than the BLEU score.

G.4 k-Anticipation Rate (k-AR)

For each sentence pair, we first use *awesome-align* (Dou and Neubig, 2021) to extract word alignments, then for each aligned target word y_j, it is considered a k-anticipation if it is aligned to a source word x_i that is k words behind, in other words, if $i - k + 1 > j$. See Figure 10 for an example of 2-anticipation. The k-AR is calculated as the percentage of k-anticipation among all aligned word pairs.

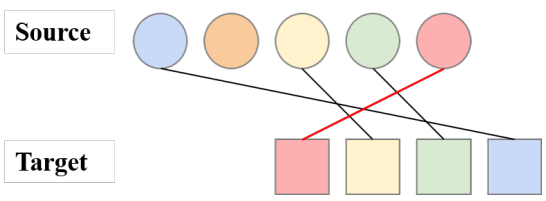


Figure 10: An example of 2-anticipation. The links are alignments, and the red link is an instance of anticipation.

H SimulEval Configuration

Table 4 show the language specific options for latency evaluation on SimulEval, which affect the AL calculation.

Options	En	Zh
–eval-latency-unit	word	char
–no-space	false	true

Table 4: Configuration for SimulEval under different target languages.

I SacreBLEU Signatures

Table 5 shows the signatures of SacreBLEU evaluation.

Lang	Metric	Signature
Zh	BLEU	nrefs:var|bs:1000|seed:12345 |case:lc|eff:no|tok:zh |smooth:exp|version:2.0.0
Zh	chrF2	nrefs:var|bs:1000|seed:12345 |case:lc|eff:yes|nc:6 |nw:0 |space:no|version:2.0.0
En	BLEU	nrefs:1|bs:1000|seed:12345 |case:lc|eff:no|tok:13a |smooth:exp|version:2.0.0
En	chrF2	nrefs:1|bs:1000|seed:12345 |case:lc|eff:yes|nc:6 |nw:0 |space:no|version:2.0.0

Table 5: The SacreBLEU signatures for each target language and each metric.

J Detailed Statistics of Quality Metrics

Table 7 shows the detailed distributional statistics of the quality metrics evaluated on the CWMT and WMT15 datasets. All settings are trained once, but we use statistical significant test using bootstrap resampling.

K Latency-quality results with chrF

Figure 11 show the quality-latency trade off with chrF on the CWMT En-zh dataset. Figure 12 show the quality-latency trade off with chrF on the WMT15 De-En dataset. These results have similar trends with BLEU score.

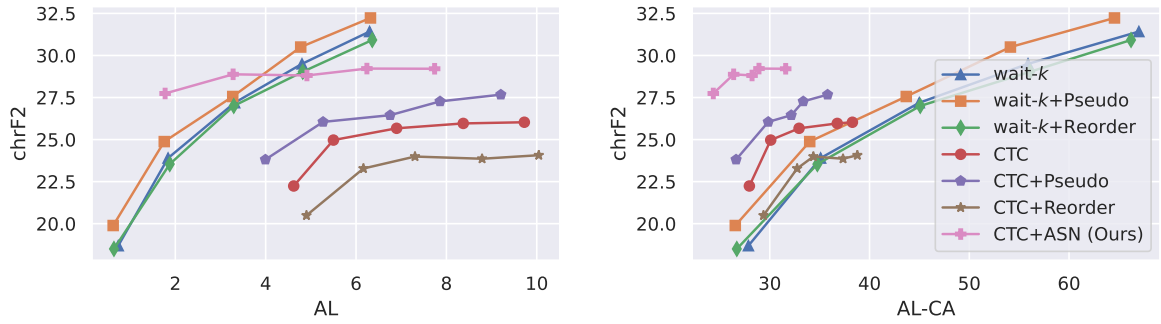


Figure 11: Latency-quality trade off with chrF score on the CWMT En-Zh dataset. Each line represents a system, and the 5 nodes corresponds to $k = 1, 3, 5, 7, 9$, from left to right. The figures share the same legend.

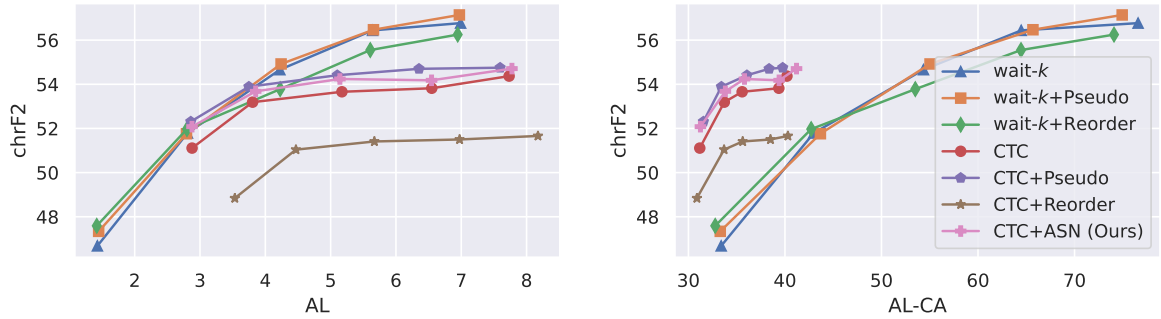


Figure 12: Latency-quality trade off with chrF score on the WMT15 De-En dataset. Each line represents a system, and the 5 nodes corresponds to $k = 1, 3, 5, 7, 9$, from left to right. The figures share the same legend.

L Performance with Oracle Reordering

We study our encoder models' performance when the oracle reordering is provided. To achieve this, we re-use the ASN during inference, and fed the (first) reference translation as the context to ASN to estimate $\mathbf{Z}$. The results compared to default setting is shown in Table 6. This result serves as a upperbound for the performance of CTC-based encoder models.

M More on ASN Output

We describe how the target tokens are placed on the vertical axis of the ASN output illustration. Since the length projection upsamples $\overline{\mathbf{H}}$ to 2 times longer, each position of $\overline{\mathbf{H}}$ corresponds to two target tokens (including repetition and blank symbols introduced by CTC). To find the optimal position for each target tokens and blank symbols, we use the Viterbi alignment (an implementation is publicly available at https://github.com/rosinality/imputer-pytorch) to align the model's logits and the actual target tokens.

Figure 13 shows more examples of the approximated permutation matrix predicted by the ASN. The sentence pairs are from CWMT En-Zh test set.

k	Method	BLEU	1/2/3/4-gram	BP
1	Default	38.58	76.7 / 51.0 / 32.5 / 20.6	0.96
	+ Oracle	41.59	76.0 / 52.7 / 35.9 / 23.9	0.96
3	Default	40.24	79.5 / 53.7 / 34.8 / 22.6	0.94
	+ Oracle	41.75	77.5 / 53.7 / 36.5 / 24.4	0.95
5	Default	40.34	78.8 / 53.5 / 35.0 / 22.7	0.94
	+ Oracle	41.70	76.0 / 52.4 / 35.5 / 23.6	0.98
7	Default	40.81	80.0 / 54.2 / 35.2 / 22.9	0.94
	+ Oracle	43.37	78.8 / 55.2 / 37.9 / 25.8	0.96
9	Default	40.83	79.5 / 54.1 / 35.4 / 23.1	0.94
	+ Oracle	41.77	76.3 / 52.7 / 35.5 / 23.6	0.98

Table 6: The BLEU score on the CWMT dataset, including n-gram precision and brevity penalty (BP), of the CTC+ASN system for each k with and without oracle order.

N More CWMT Examples

Figure 14 shows more examples from CWMT test set and the predictions of wait-k, CTC and CTC+ASN models.

O FAQ

Q1 The trained ASN cannot be used during inference, how to guarantee the model can still perform reordering?

We categorize reordering into local reordering and long-distance reordering. Our goal is for the ASN to primarily deal with long-distance reordering. In Section 5.2, we observed that employing the weight initialization improves the 2,3,4-gram precision (but not the unigram), and slightly increases the latency. This suggest that CTC+ASN model can indeed perform local reordering during inference.

As for long-distance reordering, we stress that in simultaneous interpretation, humans actively avoid long-distance reordering in order to reduce latency, which is also the goal of SimulMT. This provides the justification for removing the ASN during inference. (equation 3)

We additionally provide the performance when $\mathbf{Z}$ is available during inference in Appendix L.

Q2 Using ASN during training may cause the model to rely on Z, which may cause train-test discrepancy during inference?

In terms of the mismatch of hidden representation, because Gumbel-Sinkhorn gaurantees that $\mathbf{Z}$ is doubly stochastic (and almost permutation, depending on τ), the representation before and after ASN would only differ by a permutation. This is also discussed in Section 5.1 where removing Sinkhorn nomalization indeed negatively impact the performance.

As for the mismatch of the order of the representation, we note that the length projection network is merely a position-wise affine transformation, which means it is independent of time, so the mismatch of order between training and testing would not negatively impact the prediction made by the length projection network.

Q3 Proposed method underperform wait-k in high latency.

Simultaneous translation aims to translate in a short time, hence our work focuses on improving the translation quality under low latency setting. The higher latency model is less acceptable in practice. For instance, a $k = 9$ model decodes a single word after seeing 9 words. We included the results for experimental completeness purpose.

For the reason why proposed method underperform wait-k model: Based on the observation in Appendix L, 43.37 is the best performance of CTC+ASN method. It is inferior to the wait-9 model's 43.80. We suspect that it is caused by the inherent difference between non-autoregressive (NAR) model and auto-regressive (AR) model. However, CTC+ASN method's performance is relatively consistent when the latency decreases, while wait-k's performance decreases drastically. Therefore, to fit the simultaneous translation setting, our proposed method is more suitable than wait-k.

Q4 Explanation for why ASN could outperform Reorder and Pseudo reference baselines?

For the Reorder baseline, we suspect that since the external aligner is fixed and not jointly optimized, it may produce incorrect alignments, or miss correct ones, producing wrongful training targets.

As for the Pseudo reference baseline, there are two problems that might limit its effectiveness. For one, the pseudo reference is produced from a full-sentence model while using a wait-k decoding strategy, which is a train-test discrepancy. For another, in order to compensate for the first issue, the original translation is included as a second target for each example. This leads to the infamous multi-modality problem for non-autoregressive models, which might be harmful to our CTC-based encoder.

Q5 What are the limitations of the proposed method?

First of all, for SimulMT to be applicable to a conference setting, we assume a streaming ASR is available. However, we did not account for ASR errors in our SimulMT models.

Second, as discussed in Section 4.4, our method is only effective if the language pair includes sufficient long-distance reordering. For instance, when translation from English to Spanish, we there's hardly any reason to employ our method.

Finally, as discussed in Q3, our method is less advantageous when the latency budget is high.

Q6 What are the risks of the proposed method?

One risk is that our method may favor low-latency over high precision, which means that erroneous translation may occur, which might twist the meaning of source sentence. However, latency and quality is inherently a trade-off, and erroneous translation could be mitigated by refinement or post-editing techniques.

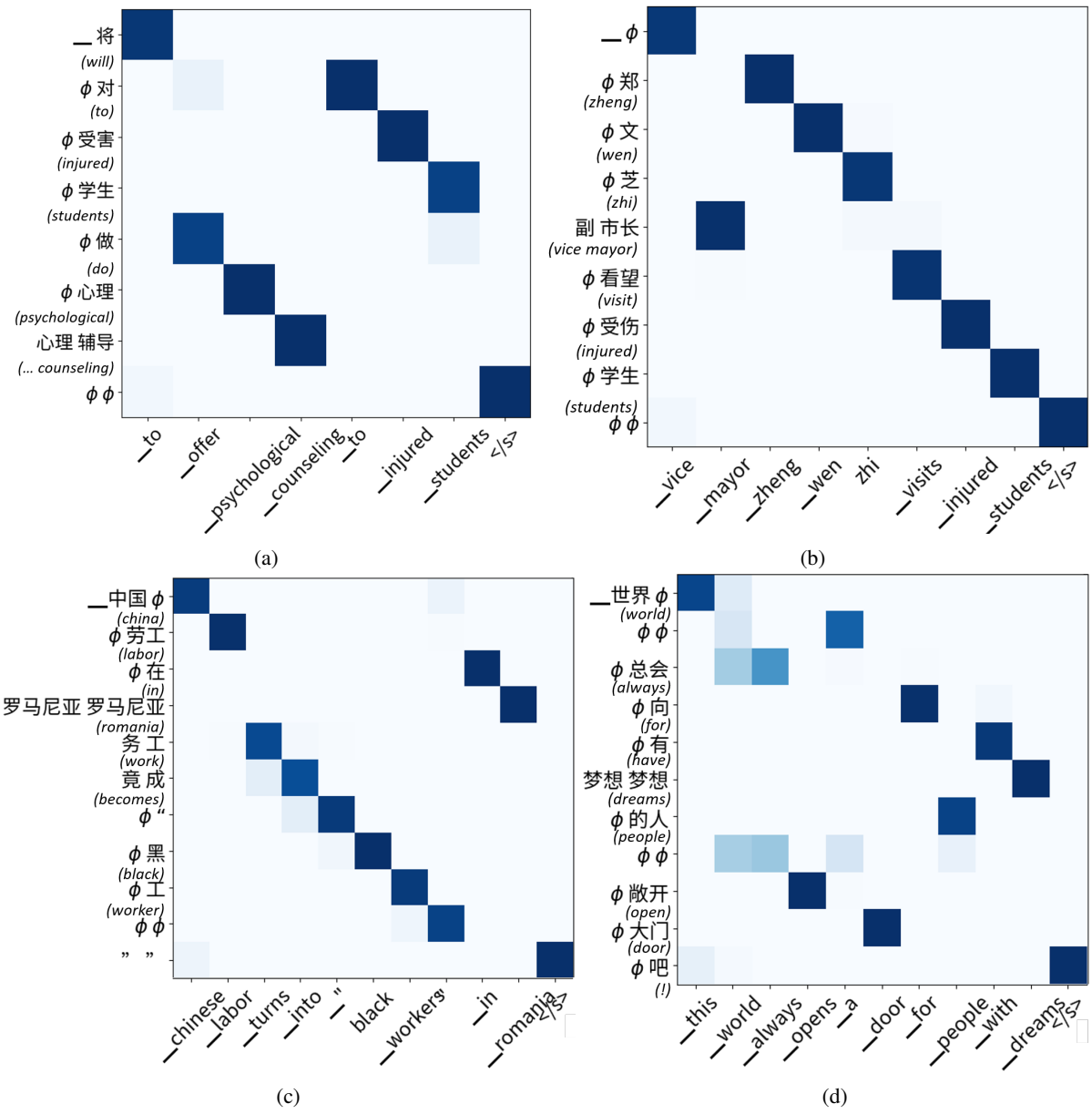


Figure 13: More approximated permutation matrices predicted by ASN.

		CWMT En→Zh				WMT15 De→En			
Delay	**Method**	**BLEU**	**μ±95%CI**	**chrF2**	**μ±95%CI**	**BLEU**	**μ±95%CI**	**chrF2**	**μ±95%CI**
offline	Transformer	45.85	45.85±0.60	32.46	32.46±0.45	31.67	31.70±0.77	57.65	57.67±0.61
$k=1$	wait-k	24.31	24.29±0.62	18.69	18.67±0.43	19.91	19.91±0.68	46.68	46.70±0.69
	wait-k+Pseudo	*25.93	25.91±0.66	*19.89	19.87±0.46	*20.63	20.63±0.68	*47.34	47.35±0.68
	wait-k+Reorder	23.98	23.96±0.59	18.50	18.49±0.39	*20.54	20.55±0.65	*47.59	47.61±0.68
	CTC	28.44	28.42±0.56	22.24	22.24±0.35	23.08	23.09±0.69	51.11	51.13±0.56
	CTC+Pseudo	†30.77	30.75±0.61	†23.81	23.81±0.38	†**24.48**	24.49±0.69	†**52.31**	52.32±0.56
	CTC+Reorder	†24.09	24.08±0.58	†20.49	20.48±0.36	†20.77	20.78±0.65	†48.84	48.85±0.56
	CTC+ASN	†**38.58**	38.57±0.45	†**27.74**	27.73±0.32	†24.17	24.19±0.70	†52.08	52.10±0.54
$k=3$	wait-k	32.27	32.25±0.65	23.90	23.90±0.43	25.85	25.87±0.78	51.79	51.81±0.67
	wait-k+Pseudo	*33.53	33.52±0.64	*24.88	24.87±0.44	25.74	25.76±0.77	51.76	51.78±0.66
	wait-k+Reorder	*31.47	31.46±0.66	*23.54	23.54±0.45	*25.26	25.28±0.73	51.97	51.99±0.65
	CTC	32.45	32.44±0.61	24.97	24.96±0.39	26.07	26.09±0.69	53.19	53.21±0.58
	CTC+Pseudo	†34.03	34.03±0.61	†26.05	26.05±0.39	†**26.61**	26.63±0.68	†**53.89**	53.91±0.55
	CTC+Reorder	†28.52	28.50±0.62	†23.28	23.28±0.40	†23.50	23.52±0.71	†51.04	51.06±0.55
	CTC+ASN	†**40.24**	40.23±0.51	†**28.88**	28.87±0.34	†26.53	26.55±0.73	†53.68	53.70±0.57
$k=5$	wait-k	37.40	37.39±0.65	27.19	27.19±0.44	**28.52**	28.54±0.82	**54.66**	54.68±0.64
	wait-k+Pseudo	*37.96	37.95±0.67	*27.56	27.56±0.46	**28.68**	28.71±0.78	**54.92**	54.95±0.60
	wait-k+Reorder	*36.86	36.84±0.65	27.00	26.99±0.44	*27.35	27.38±0.75	*53.78	53.81±0.63
	CTC	33.64	33.63±0.62	25.67	25.66±0.39	26.51	26.53±0.77	53.66	53.68±0.58
	CTC+Pseudo	†34.65	34.64±0.61	†26.45	26.45±0.40	†27.48	27.49±0.76	†**54.41**	54.43±0.60
	CTC+Reorder	†29.68	29.68±0.61	†23.99	23.98±0.38	†23.90	23.91±0.72	†51.41	51.44±0.57
	CTC+ASN	†**40.34**	40.33±0.50	†**28.81**	28.81±0.36	†27.43	27.45±0.75	†54.24	54.27±0.57
$k=7$	wait-k	40.78	40.76±0.67	29.50	29.50±0.48	**30.28**	30.32±0.80	**56.44**	56.47±0.62
	wait-k+Pseudo	***42.34**	42.34±0.62	***30.50**	30.50±0.45	**30.53**	30.56±0.82	**56.47**	56.49±0.64
	wait-k+Reorder	*40.23	40.23±0.61	*29.03	29.03±0.45	*28.77	28.79±0.75	*55.55	55.58±0.57
	CTC	34.14	34.12±0.58	25.96	25.95±0.40	26.77	26.78±0.72	53.82	53.84±0.62
	CTC+Pseudo	†36.04	36.04±0.63	†27.27	27.27±0.41	†27.66	27.67±0.75	†54.70	54.72±0.58
	CTC+Reorder	†29.45	29.44±0.64	†23.86	23.85±0.40	†24.21	24.23±0.70	†51.50	51.53±0.57
	CTC+ASN	†40.81	40.80±0.49	†29.22	29.21±0.35	†27.30	27.32±0.74	†54.18	54.21±0.57
$k=9$	wait-k	43.80	43.79±0.63	31.42	31.42±0.45	30.52	30.55±0.77	56.77	56.79±0.61
	wait-k+Pseudo	***44.99**	44.98±0.57	***32.23**	32.23±0.45	***30.99**	31.02±0.79	***57.14**	57.16±0.62
	wait-k+Reorder	*43.27	43.27±0.62	*30.92	30.92±0.44	*29.37	29.39±0.80	*56.25	56.27±0.58
	CTC	34.20	34.18±0.60	26.03	26.02±0.41	27.37	27.38±0.74	54.37	54.39±0.59
	CTC+Pseudo	†36.83	36.83±0.64	†27.67	27.66±0.41	†27.72	27.74±0.75	†54.75	54.77±0.58
	CTC+Reorder	†29.81	29.79±0.65	†24.07	24.06±0.40	†24.32	24.33±0.71	†51.66	51.68±0.58
	CTC+ASN	†40.83	40.82±0.51	†29.21	29.20±0.35	†28.00	28.02±0.78	†54.71	54.74±0.60

Table 7: Detailed quality metrics statistics on both datasets. Significance tests are conducted with paired bootstrap resampling. "*" suggests **significantly different (better or worst)** from the wait-k baseline with p-value < 0.05. "†" suggests significantly different from the CTC baseline. **Bold** text suggests the best value in the same k. If multiple values are in bold, it means that these values are not significantly different according to paired bootstrap resampling.

Input	it took a huge leap of faith to travel to india.
wait-*k*	花了 很大的 劲 才 把 这条鱼 带到 印度 去. took huge strength have this fish brought to india
CTC	旅行 巨大的 信心 飞跃 travel huge faith leap (india)
CTC+ASN	花了 巨大的 飞跃 信心 旅行 去 印度 took huge leap faith travel to india
Input	this is the first of a five-part travelogue recounting that journey.
wait-*k*	这 是 第一次, 一个 五星级 酒店, 一个 豪华的 酒店, 一个 豪华的 酒店。 this is the first time a five star hotel, a luxurious hotel, a luxurious hotel.
CTC	这 是 五 这次 旅行的 this is five this time journey's
CTC+ASN	这 是 这 是 的 第一个 五部分 旅游记 中, 述 这次 旅行中的 this is this is 's first five-part travelogue in describe this time in the journey
Input	one man had his foot stitched up with nothing to kill the pain but his son's embrace.
wait-*k*	有一个人 脚 被缝好了, 什么也杀不了 疼痛, 只有 他的 儿子 的 脚 被拥抱着。 someone foot is stitched up, nothing can kill pain, only his his son 's foot is embraced.
CTC	一个 男人 把 脚 缝,, 但 他儿子 one man had foot stitch but his son
CTC+ASN	有一个人 把 脚 缝, 无 任何 杀死 疼痛 除了, 儿子的 拥抱 someone had foot stitch no anything kill pain except, son's embrace
Input	in recent months, a number of iconic buildings on manhattan's skyline have been the target of middle eastern investors.
wait-*k*	近几个月来, 一些 标志性 建筑 在 曼哈顿的 地平线上 被建造成 一座大型的 中东 投资者的 目标。 in recent months, some iconic buildings on manhattan's horizon is built into a large scale middle eastern investors' target.
CTC	近几个月来, 一些 曼哈顿 地平线上 一些 标志性 建筑 中东 投资者的 目标 in recent months, some manhattan horizon some iconic buildings middle eastern investors' target.
CTC+ASN	近几个月来, 一些 标志性 建筑 在 曼哈顿的 天际线 一直是 的 目标 中东 投资者的 目标 in recent months, some iconic buildings on manhattan's skyline has been 's target middle eastern investors' target.
Input	it would be boring for the other teams because they would be racing only for second place.
wait-*k*	"如果 人们 不 注意 的话, 对 其他 球队 来说, 这 太无聊了。" if people don't pay attention (if…), for other teams (for…) this too boring.
CTC	"其他 球队 来说 无聊, 因为 他们 只 为 第二名 other team (for…) boring, because they only for second place
CTC+ASN	"将 太无聊 来说 其他 球队的, 因为 他们 只 为 第二名 比赛 would be too boring (for…) other team's, because they only for second place racing
Input	then he looks at me and says, 'jens, read my lips: stay together.'
wait-*k*	利5 又 看我 、说 、 约拿、念 给 我的 嘴、 要 在一起。 again look at me says Jonah read to my mouth must be together.
CTC	,他 看着 我, 说,、 念 我的 嘴唇, 住 , he look at me, says, read my lips, stay
CTC+ASN	"那么, 他 看了看 我, 说 '斯、听 我 我的 嘴唇、同住 then, he looks at me says s listen me my lips live together

Figure 14: More examples from CWMT En→Zh. Text in red are hallucinations unrelated to source. We use $k = 3$ models.