Opportunistic Decoding with Timely Correction for Simultaneous Translation

Renjie Zheng [1,2,*] **Mingbo Ma** [2,*] **Baigong Zheng** [2] **Kaibo Liu**[2] **Liang Huang** [1,2]
[1]Oregon State University, Corvallis, OR, USA
[2]Baidu Research, Sunnyvale, CA, USA
{renjiezheng,mingboma,baigongzheng}@baidu.com
{kaiboliu,lianghuang}@baidu.com

Abstract

Simultaneous translation has many important application scenarios and attracts much attention from both academia and industry recently. Most existing frameworks, however, have difficulties in balancing between the translation quality and latency, i.e., the decoding policy is usually either too aggressive or too conservative. We propose an opportunistic decoding technique with timely correction ability, which always (over-)generates a certain mount of extra words at each step to keep the audience on track with the latest information. At the same time, it also corrects, in a timely fashion, the mistakes in the former overgenerated words when observing more source context to ensure high translation quality. Experiments show our technique achieves substantial reduction in latency and up to +3.1 increase in BLEU, with revision rate under 8% in Chinese-to-English and English-to-Chinese translation.

1 Introduction

Simultaneous translation, which starts translation before the speaker finishes, is extremely useful in many scenarios, such as international conferences, travels, and so on. In order to achieve low latency, it is often inevitable to generate target words with insufficient source information, which makes this task extremely challenging.

Recently, there are many efforts towards balancing the translation latency and quality with mainly two types of approaches. On one hand, Ma et al. (2019a) propose very simple frameworks that decode following a *fixed-latency policy* such as wait-k. On the other hand, there are many attempts to learn an *adaptive policy* which enables the model to decide READ or WRITE action on the fly using various techniques such as reinforcement learning (Gu et al., 2017; Alinejad et al., 2018; Grissom II et al., 2014), supervised learning over pseudo-oracles (Zheng et al., 2019a), imitation learning (Zheng et al., 2019b), model ensemble (Zheng et al., 2020) or monotonic attention (Ma et al., 2019d; Arivazhagan et al., 2019).

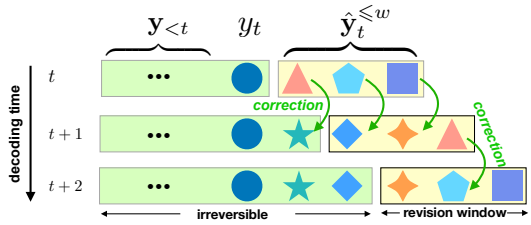

Figure 1: Besides y_t, opportunistic decoding continues to generate additional w words which are represented as $\hat{\boldsymbol{y}}_t^{\leqslant w}$. The timely correction only revises this part in future steps. Different shapes denote different words. In this example, from step t to $t+1$, all previously opportunistically decoded words are revised, and an extra triangle word is generated in opportunistic window. From step $t+1$ to $t+2$, two words from previous opportunistic window are kept and only the triangle word is revised.

Though the existing efforts improve the performance in both translation latency and quality with more powerful frameworks, it is still difficult to choose an appropriate policy to explore the optimal balance between latency and quality in practice, especially when the policy is trained and applied in different domains. Furthermore, all existing approaches are incapable of correcting the mistakes from previous steps. When the former steps commit errors, they will be propagated to the later steps, inducing more mistakes to the future.

Inspired by our previous work on speculative beam search (Zheng et al., 2019c), we propose an opportunistic decoding technique with timely correction mechanism to address the above problems. As shown in Fig. 1, our proposed method always decodes more words than the original policy at each step to catch up with the speaker and

*These authors contributed equally.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 437–442

reduce the latency. At the same time, it also employs a timely correction mechanism to review the extra outputs from previous steps with more source context, and revises these outputs with current preference when there is a disagreement. Our algorithm can be used in both speech-to-text and speech-to-speech simultaneous translation (Oda et al., 2014; Bangalore et al., 2012; Yarmohammadi et al., 2013). In the former case, the audience will not be overwhelmed by the modifications since we only review and modify the last few output words with a relatively low revision rate. In the later case, the revisable extra words can be used in look-ahead window in incremental TTS (Ma et al., 2019b). By contrast, the alternative retranslation strategy (Arivazhagan et al., 2020) will cause non-local revisions which makes it impossible to be used in incremental TTS.

We also define, for the first time, two metrics for revision-enabled simultaneous translation: a more general latency metric *Revision-aware Average Lagging* (RAL) as well as the *revision rate*. We demonstrate the effectiveness of our proposed technique using fixed (Ma et al., 2019a) and adaptive (Zheng et al., 2019a) policies in both Chinese-to-English and English-to-Chinese translation.

2 Preliminaries

Full-sentence NMT. The conventional full-sentence NMT processes the source sentence $\boldsymbol{x} = (x_1, ..., x_n)$ with an encoder, where x_i represents an input token. The decoder on the target side (greedily) selects the highest-scoring word y_t given source representation $\mathbf{h}$ and previously generated target tokens, $\boldsymbol{y}_{<t} = (y_1, ..., y_{t-1})$, and the final hypothesis $\boldsymbol{y} = (y_1, ..., y_t)$ with $y_t = \texttt{<eos>}$ has the highest probability:

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

Simultaneous Translation. Without loss of generality, regardless the actual design of policy, simultaneous translation is represented as:

$$p_g(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{|\boldsymbol{y}|} p(y_t \mid \boldsymbol{x}_{\leqslant g(t)}, \boldsymbol{y}_{<t}) \quad (2)$$

where $g(t)$ can be used to represent any arbitrary fixed or adaptive policy. For simplicity, we assume the policy is given and does not distinguish the difference between two types of policies.

3 Opportunistic Decoding with Timely Correction and Beam Search

Opportunistic Decoding. For simplicity, we first apply this method to fixed policies. We define the original decoded word sequence at time step t with y_t, which represents the word that is decoded in time step t with original model. We denote the additional decoded words at time step t as $\hat{\boldsymbol{y}}_t^{\leqslant w} = (y_t^1, ..., y_t^w)$, where w denote the number of extra decoded words. In our setting, the decoding process is as follows:

$$p_g(y_t \circ \hat{\boldsymbol{y}}_t^{\leqslant w} \mid \boldsymbol{x}_{\leqslant g(t)}) = p_g(y_t \mid \boldsymbol{x}_{\leqslant g(t)}) \prod_{i=1}^{w} p_g(\hat{y}_t^i \mid \boldsymbol{x}_{\leqslant g(t)}, y_t \circ \hat{\boldsymbol{y}}_t^{<i}) \quad (3)$$

where $\circ$ is the string concatenation operator.

We treat the procedure for generating the extra decoded sequence as opportunistic decoding, which prefers to generate more tokens based on current context. When we have enough information, this opportunistic decoding eliminates unnecessary latency and keep the audience on track. With a certain chance, when the opportunistic decoding tends to aggressive and generates inappropriate tokens, we need to fix the inaccurate token immediately.

Timely Correction. In order to deliver the correct information to the audience promptly and fix previous mistakes as soon as possible, we also need to review and modify the previous outputs.

At step $t+1$, when encoder obtains more information from $\boldsymbol{x}_{\leqslant g(t)}$ to $\boldsymbol{x}_{\leqslant g(t+1)}$, the decoder is capable to generate more appropriate candidates and may revise and replace the previous outputs from opportunistic decoding. More precisely, $\hat{\boldsymbol{y}}_t^{\leqslant w}$ and $y_{t+1} \circ \hat{\boldsymbol{y}}_{t+1}^{\leqslant w-1}$ are two different hypothesis over the same time chunk. When there is a disagreement, our model always uses the hypothesis from later step to replace the previous commits. Note our model does not change any word in y_t from previous step and it only revise the words in $\hat{\boldsymbol{y}}_t^{\leqslant w}$.

Modification for Adaptive Policy. For adaptive policies, the only difference is, instead of committing a single word, the model is capable of generating multiple irreversible words. Thus our proposed methods can be easily applied to adaptive policies.

Correction with Beam Search. When the model is committing more than one word at a time, we can use beam search to further improve the translation quality and reduce revision rate (Murray and Chiang, 2018; Ma et al., 2019c).

The decoder maintains a beam B_t^k of size b at step t, which is ordered list of pairs

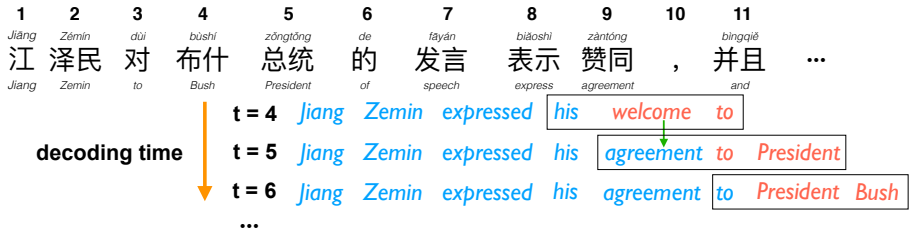

Figure 2: The decoder generates target word y_4 = "his" and two extra words "welcome to" at step $t = 4$ when input x_9 = "zàntóng" ("agreement") is not available yet. When the model receives x_9 at step $t = 5$, the decoder immediately corrects the previously made mistake "welcome" with "agreement" and emits two additional target words ("to President"). The decoder not only is capable to fix the previous mistake, but also has enough information to perform more correct generations. Our framework benefits from opportunistic decoding with reduced latency here. Note though the word "to" is generated in step $t = 4$, it only becomes irreversible at step $t = 6$.

⟨hypothesis, probability⟩, where k denotes the k^{th} step in beam search. At each step, there is an initial beam $B_t^0 = [\langle \boldsymbol{y}_{t-1}, 1 \rangle]$. We denote one-step transition from the previous beam to the next as

$$B_t^{k+1} = \text{next}_1^b(B_t^k)$$
$$= \overset{b}{\textbf{top}}\{\langle \boldsymbol{y}' \circ v,\ u \cdot p(v|\boldsymbol{x}_{\leqslant g(t)}, \boldsymbol{y}') \rangle \mid \langle \boldsymbol{y}', u \rangle \in B_t^k\}$$

where $\textbf{top}^b(\cdot)$ returns the top-scoring b pairs. Note we do not distinguish the revisable and non-revisable output in $\boldsymbol{y}'$ for simplicity. We also define the multi-step advance beam search function with recursive fashion as follows:

$$\text{next}_i^b(B_t^k) = \text{next}_1^b(\text{next}_{i-1}^b(B_t^k))$$

When the opportunistic decoding window is w at decoding step t, we define the beam search over $w + 1$ (include the original output) as follows:

$$\langle \boldsymbol{y}'_t, u_t \rangle = \textbf{top}^1\left(\text{next}_{n+w}^b(B_t^0)\right) \quad (4)$$

where $\text{next}_{n+w}^b(\cdot)$ performs a beam search with $n + w$ steps, and generate $\boldsymbol{y}'_t$ as the outputs which include both original and opportunistic decoded words. n represents the length of $\boldsymbol{y}_t$

4 Revision-aware AL and Revision Rate

We define, for the first time, two metrics for revision-enabled simultaneous translation.

4.1 Revision-aware AL

AL is introduced in (Ma et al., 2019a) to measure the average delay for simultaneous translation. Besides the limitations that are mentioned in (Cherry and Foster, 2019), AL is also not sensitive to the modifications to the committed words. Furthermore, in the case of re-translation, AL is incapable to measure the meaningful latency anymore.

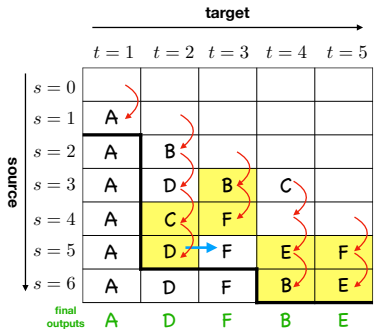

Figure 3: The red arrows represent the changes between two different commits, and the last changes for each output word is highlighted with yellow.

We hereby propose a new latency, Revision-aware AL (RAL), which can be applied to any kind of translation scenarios, i.e., full-sentence translation, use re-translation as simultaneous translation, fixed and adaptive policy simultaneous translation. Note that for latency and revision rate calculation, we count the target side difference respect to the growth of source side. As it is shown in Fig. 3, there might be multiple changes for each output words during the translation, and we only start to calculate the latency for this word once it agrees with the final results. Therefore, it is necessary to locate the last change for each word. For a given source side time s, we denote the t^{th} outputs on target side as $f(x_{\leqslant s})_t$. Then we are able to find the Last Revision (LR) for the t^{th} word on target side as follows:

$$LR(t) = \underset{s<|x|}{\textbf{argmax}} \left(f(x_{\leqslant(s-1)})_t \neq f(x_{\leqslant s})_t\right)$$

From the audience point of view, once the former words are changed, the audience also needs to take the efforts to read the following as well. Then we also penalize the later words even there are no changes, which is shown with blue arrow in Fig. 3. We then re-formulate the $\overline{LR}(t)$ as follows (assume $\overline{LR}(0) = 0$):

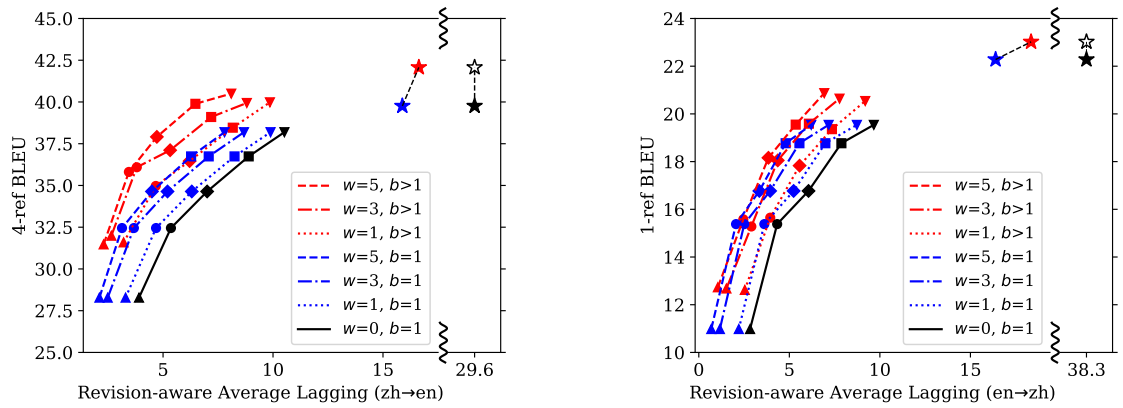

Figure 4: BLEU against RAL using wait-k polocies. ▲ ▲ ▲ : wait-1 policies, ● ● ● : wait-3 policies, ♦ ♦ ♦ : wait-5 policies, ■ ■ ■ : wait-7 policies, ▼ ▼ ▼: wait-9 policies,★ (★): re-translation with pre-trained NMT model with greedy (beam search) decoding, ★ (☆): full-sentence translation with pre-trained NMT model with greedy (beam search) decoding. The baseline for wait-k policies is decoding with $w = 0, b = 1$.

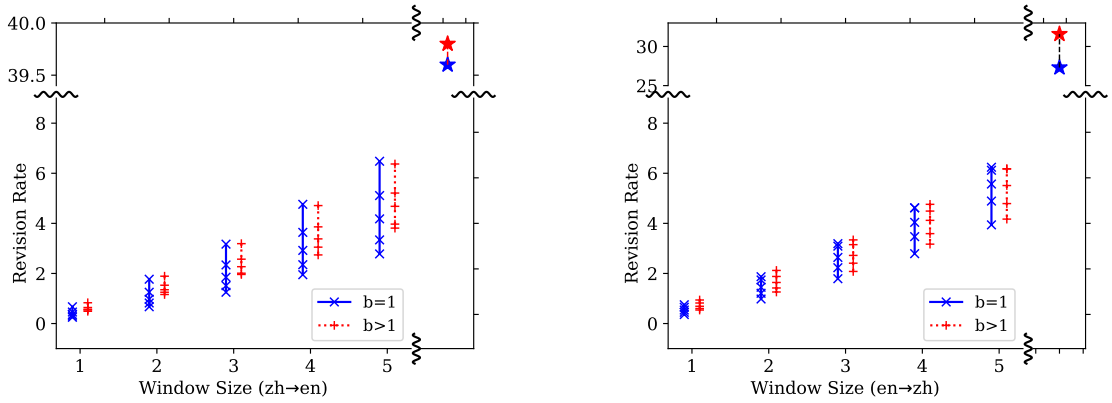

Figure 5: Revision rate against window size with different wait-k policies. ★ (★): re-translation with pre-trained NMT model with greedy (beam search) decoding.

$$\overline{LR}(t) = max\{\overline{LR}(t-1), LR(t)\} \quad (5)$$

The above definition can be visualized as the thick black line in Fig. 3. Similar with original AL, our proposed RAL is defined as follows:

$$\mathrm{RAL}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\tau(|\boldsymbol{x}|)} \sum_{t=1}^{\tau(|\boldsymbol{x}|)} \overline{LR}(t) - \frac{t-1}{r} \quad (6)$$

where $\tau(|\boldsymbol{x}|)$ denotes the cut-off step, and $r = |y|/|x|$ is the target-to-source length ratio.

4.2 Revision Rate

Since each modification on the target side would cost extra effort for the audience to read, we penalize all the revisions during the translation. We define the revision rate as follows:

$$\Big(\sum_{s=1}^{|x|-1} \mathrm{dist}\Big(f(x_{\leqslant s}), f(x_{\leqslant s+1})\Big)\Big) \Big/ \Big(\sum_{s=1}^{|x|} |f(x_{\leqslant s})|\Big)$$

where dist can be arbitrary distance measurement between two sequences. For simplicity, we design a modified Hamming Distance to measure the difference:

$$\mathrm{dist}(a, b) = \mathrm{hamming}\big(a,\ b_{\leq |a|} \circ \langle \mathrm{pad} \rangle^{\max(|a|-|b|,0)}\big)$$

where $\langle \mathrm{pad} \rangle$ is a padding symbol in case b is shorter than a.

5 Experiments

Datasets and Implementation We evaluate our work on Chinese-to-English and English-to-Chinese simultaneous translation tasks. We use the NIST corpus (2M sentence pairs) as the training data. We first apply BPE (Sennrich et al., 2015) on all texts to reduce the vocabulary sizes. For evaluation, we use NIST 2006 and NIST 2008 as our dev and test sets with 4 English references. We re-implement wait-k model (Ma et al., 2019a) and adaptive policy (Zheng et al., 2019a). We use Transformer (Vaswani et al., 2017) based wait-k model and pre-trained full-sentence model for learning adaptive policy.

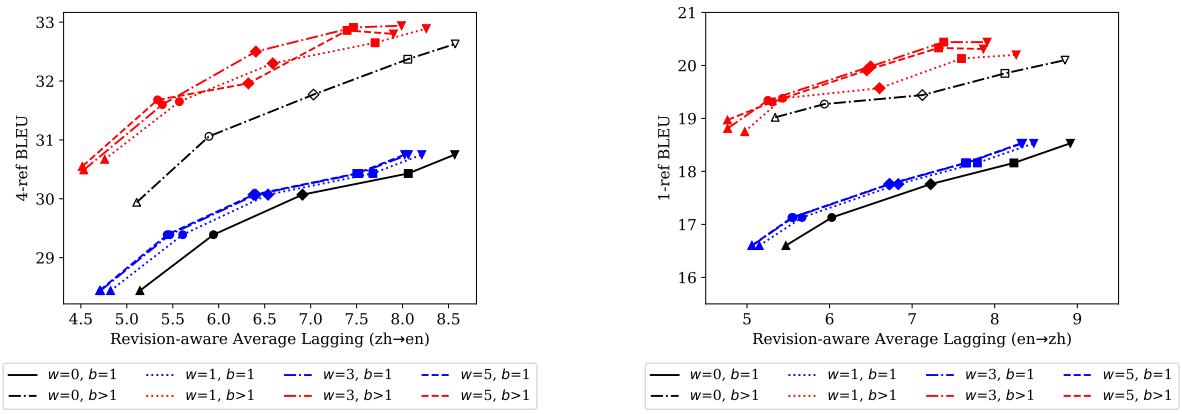

Figure 6: BLEU against RAL using adaptive policies. Baseline is decoded with $w = 0, b = 1$ and $w = 0, b > 1$.

Performance on Wait-k Policy We perform experiments using opportunistic decoding on wait-k policies with $k \in \{1, 3, 5, 7, 9\}$, opportunistic window $w \in \{1, 3, 5\}$ and beam size $b \in \{1, 3, 5, 7, 10, 15\}$. We select the best beam size for each policy and window pair on dev-set.

We compare our proposed method with a baseline called re-translation which uses a full-sentence NMT model to re-decode the whole target sentence once a new source word is observed. The final output sentences of this method are identical to the full sentence translation output with the same model but the latency is reduced.

Fig. 4 (left) shows the Chinese-to-English results of our proposed algorithm. Since our greedy opportunistic decoding doesn't change the final output, there is no difference in BLEU compared with normal decoding, but the latency is reduced. However, by applying beam search, we can achieve 3.1 BLEU improvement and 2.4 latency reduction on wait-7 policy.

Fig. 4 (right) shows the English-to-Chinese results. Compare to the Chinese-to-English translation results in previous section, there is comparatively less latency reduction by using beam search because the output translations are slightly longer which hurts the latency. As shown in Fig. 5(right), the revision rate is still controlled under 8%.

Fig. 5 shows the revision rate with different window size on wait-k policies. In general, with opportunity window $w \leq 5$, the revision rate of our proposed approach is under 8%, which is much lower than re-translation.

Performance on Adaptive Policy Fig. 6 shows the performance of the proposed algorithm on adaptive policies. We use threshold $\rho \in \{0.55, 0.53, 0.5, 0.47, 0.45\}$. We vary beam size $b \in \{1, 3, 5, 7, 10\}$ and select the best one on dev-set. Comparing with conventional beam search on consecutive writes, our decoding algorithm achieves even much higher BLEU and less latency.

5.1 Revision Rate vs. Window Size

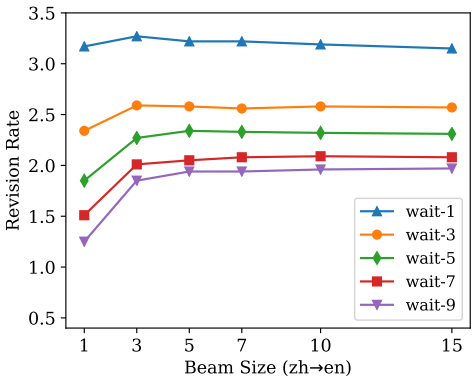

Figure 7: Revision rate against beam size with window size of 3 and different wait-k policies.

We further investigate the revision rate with different beam sizes on wait-k policies. Fig. 7 shows that the revision rate is higher with lower wait-k policies. This makes sense because the low k policies are always more aggressive and easy to make mistakes. Moreover, we can find that the revision rate is not very sensitive to beam size.

6 Conclusions

We have proposed an opportunistic decoding timely correction technique which improves the latency and quality for simultaneous translation. We also defined two metrics for revision-enabled simultaneous translation for the first time.

Acknowledgments

L. H. was supported in part by NSF IIS-1817231.

References

Ashkan Alinejad, Maryam Siahbani, and Anoop Sarkar. 2018. Prediction improves simultaneous neural machine translation. In *EMNLP*.

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*, Florence, Italy. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George Foster. 2020. Re-translation versus streaming for simultaneous translation. *arXiv preprint arXiv:2004.03643*.

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*.

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

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 *EMNLP*, pages 1342–1352.

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.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 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, Florence, Italy. Association for Computational Linguistics.

Mingbo Ma, Baigong Zheng, Kaibo Liu, Renjie Zheng, Hairong Liu, Kainan Peng, Kenneth Church, and Liang Huang. 2019b. Incremental text-to-speech synthesis with prefix-to-prefix framework. *arXiv preprint arXiv:1911.02750*.

Mingbo Ma, Renjie Zheng, and Liang Huang. 2019c. Learning to stop in structured prediction for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1884–1889.

Xutai Ma, Juan Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2019d. Monotonic multihead attention. *arXiv preprint arXiv:1909.12406*.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *Proceedings of WMT 2018*.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *ACL*.

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*.

Mahsa Yarmohammadi, Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. 2013. Incremental segmentation and decoding strategies for simultaneous translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

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*.

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.

Renjie Zheng, Mingbo Ma, Baigong Zheng, and Liang Huang. 2019c. Speculative beam search 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 1395–1402.