Simpler and Faster Learning of Adaptive Policies for Simultaneous Translation

Baigong Zheng [1,*] **Renjie Zheng** [2,*] **Mingbo Ma** [1] **Liang Huang** [1,2]
[1]Baidu Research, Sunnyvale, CA, USA
[2]Oregon State University, Corvallis, OR, USA
{baigongzheng, mingboma}@baidu.com zrenj11@gmail.com

Abstract

Simultaneous translation is widely useful but remains challenging. Previous work falls into two main categories: (a) fixed-latency policies such as Ma et al. (2019) and (b) adaptive policies such as Gu et al. (2017). The former are simple and effective, but have to aggressively predict future content due to diverging source-target word order; the latter do not anticipate, but suffer from unstable and inefficient training. To combine the merits of both approaches, we propose a simple supervised-learning framework to learn an adaptive policy from oracle READ/WRITE sequences generated from parallel text. At each step, such an oracle sequence chooses to WRITE the next target word if the available source sentence context provides enough information to do so, otherwise READ the next source word. Experiments on German↔English show that our method, without retraining the underlying NMT model, can learn flexible policies with better BLEU scores and similar latencies compared to previous work.

1 Introduction

Simultaneous translation outputs target words while the source sentence is being received, and is widely useful in international conferences, negotiations and press releases. However, although there is significant progress in machine translation (MT) recently, simultaneous machine translation is still one of the most challenging tasks. This is because it is hard to balance translation quality and latency, especially for the syntactically divergent language pairs, such as English and Japanese.

Researchers previously study simultaneous translation as a part of real-time speech-to-speech translation system (Yarmohammadi et al., 2013; Bangalore et al., 2012; Fügen et al., 2007; Sridhar et al., 2013; Jaitly et al., 2016; Graves et al., 2013). Recent simultaneous translation research focuses on obtaining a strategy, called a *policy*, to decide whether to wait for another source word (READ action) or emit a target word (WRITE action). The obtained policies fall into two main categories: (1) fixed-latency policies (Ma et al., 2019; Dalvi et al., 2018) and (2) context-dependent adaptive policies (Grissom II et al., 2014; Cho and Esipova, 2016; Gu et al., 2017; Alinejad et al., 2018; Arivazhagan et al., 2019; Zheng et al., 2019a). As an example of fixed-latency policies, wait-k (Ma et al., 2019) starts by waiting for the first few source words and then outputs one target word after receiving each new source word until the source sentence ends. It is easy to see that this kind of policy will inevitably need to guess the future content, which can often be incorrect. Thus, an adaptive policy (see Table 1 as an example), which can decides on the fly whether to take READ action or WRITE action, is more desirable for simultaneous translation. Moreover, the widely-used beam search technique become non-trivial for fixed policies (Zheng et al., 2019b).

To represent an adaptive policy, previous work shows three different ways: (1) a rule-based decoding algorithm (Cho and Esipova, 2016), (2) an original MT model with an extended vocabulary (Zheng et al., 2019a), and (3) a separate policy model (Grissom II et al., 2014; Gu et al., 2017; Alinejad et al., 2018; Arivazhagan et al., 2019). The decoding algorithm (Cho and Esipova, 2016) applies heuristics measures and does not exploit information in the hidden representation, while the MT model with an extended vocabulary (Zheng et al., 2019a) needs guidance from a restricted dynamic oracle to learn an adaptive policy, whose size is exponentially large so that approximation is needed. A separate policy model could avoid these issues. However, previous policy-learning

[*] These authors contributed equally.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 1349–1354, Hong Kong, China, November 3–7, 2019.

German	Ich		bin	mit	dem	Bus				nach	Ulm	gekommen						
Gloss	I		am	with	the	bus				to	Ulm	come						
Action	R	W	R	R	R	R	W	W	W	R	R	R	W	W	W	W		
Translation		I					took	the	bus				to	come	to	Ulm		

Table 1: An example for READ/WRITE action sequence. R represents READ and W represents WRITE.

methods either depends on reinforcement learning (RL) (Grissom II et al., 2014; Gu et al., 2017; Alinejad et al., 2018), which makes the training process unstable and inefficient due to exploration, or applies advanced attention mechanisms (Arivazhagan et al., 2019), which requires its training process to be autoregressive, and hence inefficient. Furthermore, each such learned policy cannot change its behaviour according to different latency requirements at testing time, and we will need to train multiple policy models for scenarios with different latency requirements.

To combine the merits of fixed and adaptive policies, and to resolve the mentioned drawbacks, we propose a simple supervised learning framework to learn an adaptive policy, and show how to apply it with controllable latency. This framework is based on sequences of READ/WRITE actions for parallel sentence pairs, so we present a simple method to generate such an action sequence for each sentence pair with a pre-trained neural machine translation (NMT) model.[1] Our experiments on German↔English dataset show that our method, without retraining the underlying NMT model, leads to better policies than previous methods, and achieves better BLEU scores than (the retrained) wait-k models at low latency scenarios.

2 Generating Action Sequences

In this section, we show how to generate action sequences for parallel text. Our simultaneous translation policy can take two actions: READ (receive a new source word) and WRITE (output a new target word). A sequence of such actions for a sentence pair $(\mathbf{s}, \mathbf{t})$ defines one way to translate $\mathbf{s}$ into $\mathbf{t}$. Thus, such a sequence must have $|\mathbf{t}|$ number of WRITE actions. However, not every action sequence is good for simultaneous translation. For instance, a sequence without any READ action will not provide any source information, while a sequence with all $|\mathbf{s}|$ number of READ actions before all WRITE actions usually has large latency. Thus the ideal sequences for simultaneous translation should have the following two properties:

- there is no anticipation during translation, i.e. when choosing WRITE action, there is enough source information for the MT model to generate the correct target word;
- the latency is as low as possible, i.e. the WRITE action for each target word appears as early as possible.

Table 1 gives an example for such a sequence.

Algorithm 1 Generating Action Sequence

Input: sentence pair $(\mathbf{s}, \mathbf{t})$, integer r, model M
$id_s \leftarrow 1, id_t \leftarrow 1$
Seq $\leftarrow [R]$
while $id_t \leq |\mathbf{t}|$ **do**
  if $rank_M(t_{id_t}|\mathbf{s}_{\leq id_s}) \leq r$ or $id_s = |\mathbf{s}|$ **then**
    Seq $\leftarrow$ Seq $+[W]$
    $id_t \leftarrow id_t + 1$
  else
    Seq $\leftarrow$ Seq $+[R]$
    $id_s \leftarrow id_s + 1$
return Seq

In the following, we present a simple method to generate such an action sequence for a sentence pair $(\mathbf{s}, \mathbf{t})$ using a pre-trained NMT model, assuming this model can make reasonable prediction given incomplete source sentence. Our method is based on this observation: *if the rank of the next ground-truth target word is high enough in the prediction of the model, then this implies that there is enough source-side information for the model to make a correct prediction.* Specifically, we sequentially input the source words to the pre-trained model, and use it to predict next target word. If the rank of the gold target word is high enough, we will append a WRITE action to the sequence and then try the next target word; otherwise, we append a READ action and input a new source word. Let r be a positive integer, M be a pre-trained NMT model, $\mathbf{s}_{\leq i}$ be the source sequence consisting of the first i words of $\mathbf{s}$, and $rank_M(t_j|\mathbf{s}_{\leq i})$ be the rank of the target word t_j in the prediction of model M

[1] Previous work (He et al., 2015; Niehues et al., 2018) shows that carefully generated parallel training data can help improve simultaneous MT or low-latency speech translation. This is different from our work in that we generates action sequences for training policy model instead of parallel translation data for MT model.

given sequence $\mathbf{s}_{\leq i}$. Then the generating process can be summarized as Algorithm 1.

Although we can generate action sequences balancing the two wanted properties with appropriate value of parameter r, the latency of generated action sequence may still be large due to the word order difference between the two sentences. To avoid this issue, we filter the generated sequences with the latency metric *Average Lagging* (AL) proposed by Ma et al. (2019), which quantifies the latency in terms of the number of source words and avoids some flaws of other metrics like *Average Proportion* (AP) (Cho and Esipova, 2016) and *Consecutive Wait* (CW) (Gu et al., 2017). Another issue we observed is that, the pre-trained model may be too aggressive for some sentence pair, meaning that it may write all target words without seeing the whole source sentence. This may be because the model is also trained on the same dataset. To overcome this, we only keep the action sequences that will receive all the source words before the last WRITE action. After the filtering process, each action sequence has AL less than a fixed constant α and receives all source words.

3 Supervised-Learning Framework for Simultaneous Translation Policy

Given a sentence pair and an action sequence for this pair, we can apply supervised learning method to learn a parameterized policy for simultaneous translation. For the policy to be able to choose the correct action, its input should include information from both source and target sides. Since we use Transformer (Vaswani et al., 2017) as our underlying NMT model in this work, we need to recompute encoder hidden states for all previous seen source words, which is the same as done for wait-k model training (Ma et al., 2019).[2] The policy input o_i at step i consists of three components from this model:

- h_i^s: the last-layer hidden state from the encoder for the first source word at step i;
- h_i^t: the last-layer hidden state from the decoder for the first target word at step i;[3]
- c_i: cross-attention scores at step i for the current input target word on all attention layers in decoder, averaged over all current source words.

That is $o_i = [h_i^s, h_i^t, c_i]$.

Let a_i be the i-th action in the given action sequence $\mathbf{a}$. Then the decision of our policy on the i-th step depends on all previous inputs $\mathbf{o}_{\leq i}$ and all taken actions $\mathbf{a}_{<i}$. We want to maximize the probability of the next action a_i given those information:

$$\max p_\theta(a_i|\mathbf{o}_{\leq i}, \mathbf{a}_{<i})$$

where p_θ is the action distribution of our policy parameterized by θ.

4 Decoding with Controllable Latency

To apply the learned policy for simultaneous translation, we can choose at each step the action with higher probability . However, different scenarios may have different latency requirements. Thus, this greedy policy may not always be the best choice for all situations. Here we present a simple way to implicitly control the latency of the learned policy without retraining the policy model.

Let ρ be a probability threshold. For each step in translation, we choose READ action only if the probability of READ is greater than ρ; otherwise we choose WRITE action. Thus, this threshold balances the tradeoff between latency and translation quality: with larger ρ, the policy prefers to take WRITE actions, providing lower latency; and with smaller ρ, the policy prefers to take READ actions, and provides more conservative translation with larger latency.

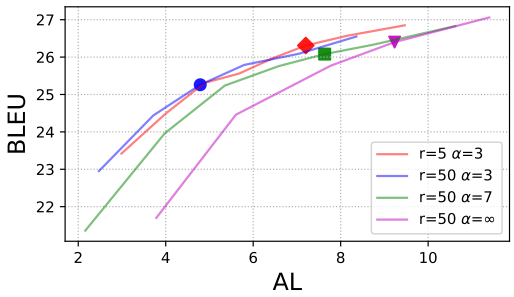

Figure 1: Translation quality against latency on DE→EN dev set. Lines are obtained with different probability thresholds ρ. Markers represent $\rho = 0.5$.

5 Experiments

Dataset We conduct experiments on English↔German (EN↔DE) simultaneous translation. We use the parallel corpora from

[2] In our experiments, the decoder and policy model combined need about 0.0445 seconds on average to generate one target word (this might include multiple READ's), while it takes on average only about 0.0058 seconds to recompute all encoder states for each new source word. So we think this re-computation might not be a serious issue for system efficiency.

[3] The hidden states of the first target word will be recomputed at each step.

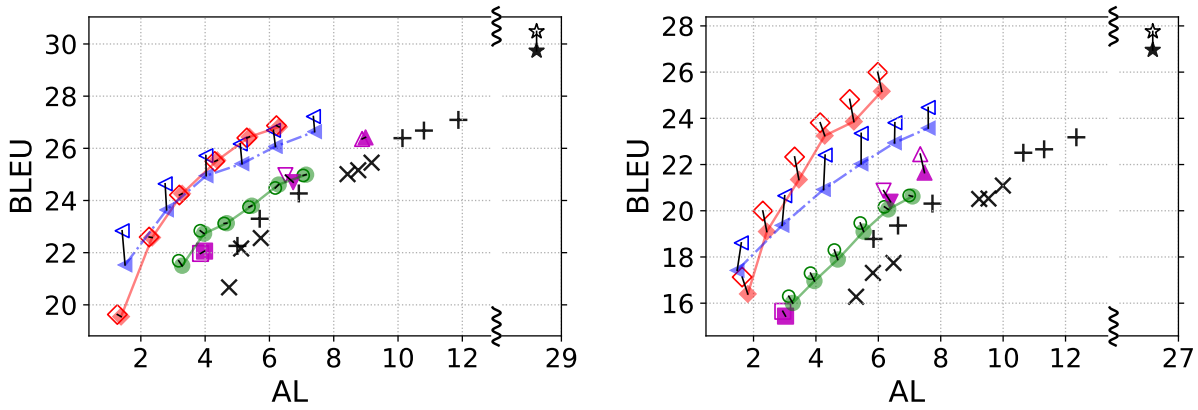

Figure 2: Comparing performances of different methods on testing sets. Left: DE→EN. Right: EN→DE. The shown pairs are results of greedy decoding (solid shapes) and beam search (empty shapes, beam-size = 5). ◆: wait-k models for $k \in \{1,2,3,4,5,6\}$, ●: test-time wait-k for $k \in \{1,2,3,4,5,6\}$, ◀: our SL model with threshold $\rho \in \{0.65, 0.6, 0.55, 0.5, 0.45, 0.4\}$, ★: full-sentence translation model, ■: RL with CW = 2, ▼: RL with CW = 5, ▲: RL with CW = 8, ×: WID for $s_0 \in \{2,4,6\}$ and $\delta \in \{2,4\}$, +: WIW for $s_0 \in \{2,4,6\}$ and $\delta \in \{1,2\}$.

WMT 15 for training, newstest-2013 for validation and newstest-2015 for testing.[4] All datasets are tokenized and segmented into sub-word units with byte-pair encoding (BPE) (Sennrich et al., 2016), and we only use the sentence pairs of lengths less than 50 (on both sides) for training.

Model Configuration We use Transformer-base (Vaswani et al., 2017) as our NMT model and our implementation is based on PyTorch-based OpenNMT (Klein et al., 2017). We add an <eos> token on the source side, which is not included in the original OpenNMT codebase. Our recurrent policy model consists of one GRU layer with 512 units, one fully-connected layer of dimension 64 followed by ReLU activation, and one fully-connected layer of dimension 2 followed by a softmax function to produce the action distribution. We use BLEU (Papineni et al., 2002) as the translation quality metric and Averaged Lagging (AL) (Ma et al., 2019) as the latency metric.

Effects of Generated Action Sequences We first analyze the effects of the two parameters in the generation process of action sequences: the rank r and the filtering latency α. We fix $\alpha = 3$ and choose the rank $r \in \{5, 50\}$; then we fix $r = 50$ and choose the latency $\alpha \in \{3, 7, \infty\}$ to generate action sequences on DE→EN direction.Figure 1 shows the performances of resulting models with different probability thresholds ρ. We find that smaller α helps achieve better performance and our model is not very sensitive to values of rank. Therefore, in the following experiments, we report results with $r = 50$ and $\alpha = 3$.

Performance Comparison We compare our method on EN↔DE directions with different methods: greedy decoding algorithms Wait-If-Worse/Wait-If-Diff (WIW/WID) of Cho and Esipova (2016), RL method of Gu et al. (2017), wait-k models and test-time wait-k methods of Ma et al. (2019). Both of WIW and WID only use the pre-trained NMT model, and the algorithm initially reads s_0 number of source words and chooses READ only if the probability of the most likely target word decreases when given another δ number of source words (WIW) or the most likely target word changes when given another δ number of source words (WID). For RL method, we use the same kind of input and architecture for policy model.[5] Test-time wait-k means decoding with wait-k policy using the pre-trained NMT model. All methods share the same underlying pre-trained NMT model except for the wait-k method, which retrains the NMT model from scratch, but with the same architecture as the pre-trained model.

Figure 2 shows the performance comparison. Our models on both directions can achieve higher BLEU scores with the similar latency than WIW, WID, RL model and test-time wait-k method, implying that our method learns a better policy model than the other methods when the underlying NMT model is not retrained.[6] Compared with wait-k models, our models with beam search achieve

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

[5] We only report results obtained with CW-based reward functions for they outperform those with AP-based ones.

[6] Our test-time wait-k results are better than those in Ma et al. (2019), because the added source side <eos> token helps the full-sentence model to learn when the source sentence

German	die	deutsche	bahn	will		im	kommenden	jahr		die	kin-		zi-		g-	tal-	–		bahn-		strecke	verbessern	.			
gloss	the	German	train	want		in the	coming	year		the		Kinzigtal					–		railroad track			improve	.			
wait-3				deutsche		bahn	wants		to		make		the		cinema		show	a	success		this		coming		year		.
wait-5							deutsche		bahn		wants		to		introduce		the		kin-	zi-	g-		tal		railway		line
test-time wait-3				deutsche		bahn	wants		the		german		railways		to		be		the	kin-	z-		ig-		tal		railway
test-time wait-5									the		german		railways		wants		to		take		the	german	train		to		the
SL policy	the		german		railway		wants the																				
RL policy																											

Table 2: German-to-English example from validation set.

higher BLEU scores when latency AL is small, which we think will be the most useful scenarios of simultaneous translation. Furthermore, this figure also shows that our model can achieve good performance on different latency conditions by controlling the threshold ρ, so we do not need to train multiple models for different latency requirements. We also provide a translation example in Table 2 to compare different methods.

Learning Process Analysis We analyze two aspects of the learning processes of different methods: stability and training time. Figure 3 shows the learning curves of the training processes of RL method and our SL method, averaged over four different runs with different random seeds on DE→EN direction. We can see that the training process of our method is more stable and converges faster than the RL method. Although there are some steps where the RL training process can achieve better BLEU scores than our SL method, the corresponding latencies are usually very big, which are not appropriate for simultaneous translation. We present the training time of different methods in Table 3. Our method only need about 12 hours to train the policy model with 1 GPU, while the wait-k method needs more than 600 hours with 8 GPUs to finish the training process, showing that our method is very efficient. Note that this table does not include the time needed to generate action sequences. The time for this process could be very flexible since we can parallelize this by dividing the training data into separating parts. In our experiments, we need about 2 hours to generate all action sequences in parallel.

6 Conclusions

We have proposed a simple supervised-learning framework to learn an adaptive policy based on generated action sequences for simultaneous translation, which leads to faster training and better policies than previous methods, without the need to retrain the underlying NMT model.

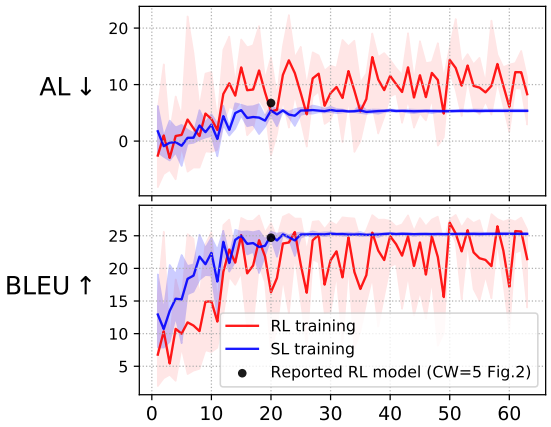

Figure 3: Learning curves averaged over four independent training runs on DE→EN direction. The x-axis represents training steps ($\times$ 50).

DE→EN	pre-train: 64		wait-1	wait-3	wait-5
	SL: +12	RL: +14.3	938	966	945
EN→DE	pre-train: 68		wait-1	wait-3	wait-5
	SL: +11	RL: +11.2	665	665	630

Table 3: Training time (in hours) of different methods. Wait-k training uses 8 GPUs, while others use 1 GPU. The RL time is the average time needed to obtain the three RL models in Figure 2.

Acknowledgments

We thank Hairong Liu for helpful discussion, Kaibo Liu for helping training baseline models, and the anonymous reviewers for suggestions. We also thank Kaibo Liu for making the AL script available at https://github.com/SimulTrans-demo/STACL.

References

Ashkan Alinejad, Maryam Siahbani, and Anoop Sarkar. 2018. Prediction improves simultaneous neural ma-

ends. Without this token, test-time wait-k generates either very short target sentences or many punctuations for small k's.

chine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3022–3027.

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 Conference of the Association for Computational Linguistics, ACL 2019*, pages 1313–1323.

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

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

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 493–499.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine translation*, 21(4):209–252.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

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

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*, pages 1053–1062.

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.

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

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

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-latency neural speech translation. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association.*, pages 1293–1297.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadephia, USA.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238.

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.

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, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. 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. 2019b. Speculative beam search for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*.