Simultaneous Translation with Flexible Policy via Restricted Imitation Learning

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 one of the most difficult tasks in NLP. Previous work either uses fixed-latency policies, or train a complicated two-staged model using reinforcement learning. We propose a much simpler *single model* that adds a "delay" token to the target vocabulary, and design a *restricted dynamic oracle* to greatly simplify training. Experiments on Chinese↔English simultaneous translation show that our work leads to flexible policies that achieve better BLEU scores and lower latencies compared to both fixed and RL-learned policies.

1 Introduction

Simultaneous translation, which translates sentences before they are finished, is useful in many scenarios such as international conferences, summits, and negotiations. However, it is widely considered one of the most challenging tasks in NLP, and one of the holy grails of AI (Grissom II et al., 2014). A major challenge in simultaneous translation is the word order difference between the source and target languages, e.g., between SOV languages (German, Japanese, etc.) and SVO languages (English, Chinese, etc.).

Simultaneous translation is previously studied as a part of real-time speech recognition 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). Recently, there have been two encouraging efforts in this problem with promising but limited success. Gu et al. (2017) propose a complicated two-stage model that is also trained in two stages. The base model, responsible for producing target words, is a conventional full-sentence seq2seq model, and on top of that, the READ/WRITE (R/W) model decides, at every step, whether to wait for another source word (READ) or to emit a target word (WRITE) using the pretrained base model. This R/W model is trained by reinforcement learning (RL) method without updating the base model. Ma et al. (2018), on the other hand, propose a much simpler architecture, which only need one model and can be trained with end-to-end local training method. However, their model follows a fixed-latency policy, which inevitably needs to guess future content during translation. Table 1 gives an example which is difficult for the fixed-latency (wait-k) policy but easy for adaptive policy.

Chinese	我	得到	有关	方面	的	回应		
pinyin	*wǒ*	*dédào*	*yǒuguān*	*fāngmiàn*	*de*	*huíyìng*		
gloss	I	receive	relevant	party	's	response		
wait-1 policy	I	received	thanks	from	relevant	parties		
wait-5 policy					I	received	responses	from relevant parties
adaptive policy	I	received				responses	from	relevant parties

Table 1: A Chinese-to-English translation example. Wait-1 policy makes a mistake on guessing *thanks from* while wait-5 policy has high latency. The adaptive policy can wait for more information to avoid guesses while maintaining low latency.

We aim to combine the merits of both efforts, that is, we design a single model *end-to-end trained from scratch* to perform simultaneous translation, as with Ma et al. (2018), which can decide on the fly whether to wait or translate as in Gu et al. (2017). There are two key ideas to achieve this: the first is to add a "delay" token (similar to the READ action in Gu et al. (2017), the *empty* token in Press and Smith (2018), and the 'blank' unit in Connectionist Temporal Classification (CTC) (Graves et al., 2006)) to the target-side vocabulary, and if the model emits this delay token, it will read one source word; the second idea is to train the model using (restricted) imitation learning by designing a (restricted) dynamic oracle as the expert policy. Table 2 summarizes different approaches for simultaneous translation using neural machine translation (NMT) model.

*These authors contributed equally.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5816–5822
Florence, Italy, July 28 - August 2, 2019.

	seq-to-seq	prefix-to-prefix
fixed policy	static Read-Write (Dalvi et al., 2018) test-time wait-k (Ma et al., 2018)	wait-k (Ma et al., 2018)
adaptive policy	RL (Gu et al., 2017)	**imitation learning (this work)**

Table 2: Different approaches for simultaneous translation.

2 Preliminaries

Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a sequence of words. For an integer $0 \leq i \leq n$, we denote the sequence consisting of the first consecutive $i-1$ words in $\mathbf{x}$ by $\mathbf{x}_{<i} = (x_1, \ldots, x_{i-1})$. We say such a sequence $\mathbf{x}_{<i}$ is a *prefix* of the sequence $\mathbf{x}$, and define $\mathbf{s} \preceq \mathbf{x}$ if sequence $\mathbf{s}$ is a prefix of $\mathbf{x}$.

Conventional Machine Translation Given a sequence $\mathbf{x}$ from the source language, the conventional machine translation model predicts the probability distribution of the next target word y_j at the j-th step, conditioned on the full source sequence $\mathbf{x}$ and previously generated target words $\mathbf{y}_{<j}$, that is $p(y_j \mid \mathbf{x}, \mathbf{y}_{<j})$. The probability of the whole sequence $\mathbf{y}$ generated by the model will be $p(\mathbf{y} \mid \mathbf{x}) = \prod_{j=1}^{|\mathbf{y}|} p(y_j \mid \mathbf{x}, \mathbf{y}_{<j})$.

To train such a model, we can maximize the probability of ground-truth target sequence conditioned on the corresponding source sequence in a parallel dataset D, which is equivalent to minimize the following loss:

$$\ell(D) = -\sum_{(\mathbf{x},\mathbf{y})\in D} \log p(\mathbf{y} \mid \mathbf{x}). \quad (1)$$

In this work, we use *Transformer* (Vaswani et al., 2017) as our NMT model, which consists of an encoder and a decoder. The encoder works in a self-attention fashion and maps a sequence of words to a sequence of continuous representations. The decoder performs attention over the predicted words and the output of the encoder to generate next prediction. Both encoder and decoder take as input the sum of a word embedding and its corresponding positional embedding.

Prefix-to-Prefix Framework Previous work (Gu et al., 2017; Dalvi et al., 2018) use seq2seq models to do simultaneous translation, which are trained with full sentence pairs but need to predict target words based on partial source sentences. Ma et al. (2018) proposed a *prefix-to-prefix* training framework to solve this mismatch. The key idea of this framework is to train the model to predict the next target word conditioned on the partial source sequence the model has seen, instead of the full source sequence.

As a simple example in this framework, Ma et al. (2018) presented a class of policies, called *wait-k policy*, that can be applied with local training in the prefix-to-prefix framework. For a positive integer k, the wait-k policy will wait for the first k source words and then start to alternate generating a target word with receiving a new source word, until there is no more source words, when the problem becomes the same as the full-sequence translation. The probability of the j-th word is $p_k(y_j \mid \mathbf{x}_{<j+k}, \mathbf{y}_{<j})$, and the probability of the whole predicted sequence is $p_k(\mathbf{y} \mid \mathbf{x}) = \prod_{j=1}^{|\mathbf{y}|} p_k(y_j \mid \mathbf{x}_{<j+k}, \mathbf{y}_{<j})$.

3 Model

To obtain a flexible and adaptive policy, we need our model to be able to take both READ and WRITE actions. Conventional translation model already has the ability to write target words, so we introduce a "delay" token $\langle\varepsilon\rangle$ in target vocabulary to enable our model to apply the READ action. Formally, for the target vocabulary V, we define an extended vocabulary

$$V_+ = V \cup \{\langle\varepsilon\rangle\}. \quad (2)$$

Each word in this set can be an action, which is applied with a transition function δ on a sequence pair $(\mathbf{s}, \mathbf{t})$ for a given source sequence $\mathbf{x}$ where $\mathbf{s} \preceq \mathbf{x}$. We assume $\langle\varepsilon\rangle$ cannot be applied with the sequence pair $(\mathbf{s}, \mathbf{t})$ if $\mathbf{s} = \mathbf{x}$, then we have the transition function δ as follows,

$$\delta((\mathbf{s}, \mathbf{t}), a) = \begin{cases} (\mathbf{s} \circ x_{|\mathbf{s}|+1}, \mathbf{t}) & \text{if } a = \langle\varepsilon\rangle \\ (\mathbf{s}, \mathbf{t} \circ a) & \text{otherwise} \end{cases}$$

where $\mathbf{s} \circ x$ represents concatenating a sequence $\mathbf{s}$ and a word x.

Based on this transition function, our model can do simultaneous translation as follows. Given the currently available source sequence, our model continues predicting next target word until it predicts a delay token. Then it will read a new source word, and continue prediction. Since we use Transformer model, the whole available source sequence needs to be encoded again when reading in a new source word, but the predicted target sequence will not be changed.

Note that the predicted delay tokens do not provide any semantic information, but may introduce

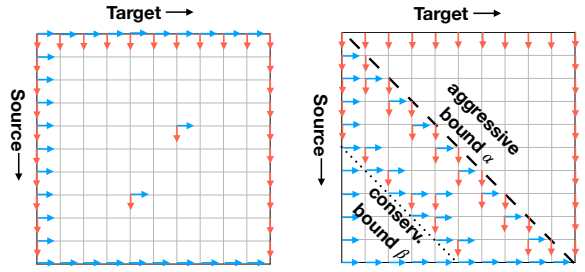


Figure 1: Illustration of our proposed dynamic oracle on a prefix grid. The blue right arrow represents choosing next ground-truth target word, and the red downward arrow represents choosing the delay token. The left figure shows a simple dynamic oracle without delay constraint. The right figure shows the dynamic oracle with delay constraints.

some noise in attention layer during the translation process. So we propose to remove those delay token in the attention layers except for the current input one. However, this removal may reduce the explicit latency information which will affect the predictions of the model since the model cannot observe previous output delay tokens. Therefore, to provide this information explicitly, we embed the number of previous delay tokens to a vector and add this to the sum of the word embedding and position embedding as the input of the decoder.

4 Methods

4.1 Training via Restricted Imitation Learning

We first introduce a restricted dynamic oracle (Cross and Huang, 2016) based on our extended vocabulary. Then we show how to use this dynamic oracle to train a simultaneous translation model via imitation learning. Note that we do not need to train this oracle.

Restricted Dynamic Oracle Given a pair of full sequences $(\mathbf{x}, \mathbf{y})$ in data, the input state of our restricted dynamic oracle will be a pair of prefixes $(\mathbf{s}, \mathbf{t})$ where $\mathbf{s} \preceq \mathbf{x}$, $\mathbf{t} \preceq \mathbf{y}$ and $(\mathbf{s}, \mathbf{t}) \neq (\mathbf{x}, \mathbf{y})$. The whole action set is V_+ defined in the last section. The objective of our dynamic oracle is to obtain the full sequence pair $(\mathbf{x}, \mathbf{y})$ and maintain a reasonably low latency.

For a prefix pair $(\mathbf{s}, \mathbf{t})$, the difference of the lengths of the two prefixes can be used to measure the latency of translation. So we would like to bound this difference as a latency constraint. This idea can be illustrated in the prefix grid (see Figure 1), where we can define a band region and always keep the translation process in this band. For simplicity, we first assume the two full sequences have the same lengths, i.e. $|\mathbf{x}| = |\mathbf{y}|$. Then we can bound the difference $d = |\mathbf{s}| - |\mathbf{t}|$ by two constants: $\alpha < d < \beta$. The conservative bound (β) guarantees relatively small difference and low latency; while the aggressive bound (α) guarantees there are not too many target words predicted before seeing enough source words. Formally, this dynamic oracle is defined as follows.

$$\pi^{\star}_{\mathbf{x},\mathbf{y},\alpha,\beta}(\mathbf{s}, \mathbf{t}) = \begin{cases} \{\langle\varepsilon\rangle\} & \text{if } \mathbf{s} \neq \mathbf{x} \text{ and } |\mathbf{s}| - |\mathbf{t}| \leq \alpha \\ \{y_{|\mathbf{t}|+1}\} & \text{if } \mathbf{t} \neq \mathbf{y} \text{ and } |\mathbf{s}| - |\mathbf{t}| \geq \beta \\ \{\langle\varepsilon\rangle, y_{|\mathbf{t}|+1}\} & \text{otherwise} \end{cases}$$

By this definition, we know that this oracle can always find an action sequence to obtain $(\mathbf{x}, \mathbf{y})$. When the input state does not satisfy any latency constraint, then this dynamic oracle will provide only one action, applying which will improve the length difference. Note that this dynamic oracle is restricted in the sense that it is only defined on the prefix pair instead of any sequence pair. And since we only want to obtain the exact sequence from data, this oracle can only choose the next ground-truth target word other than $\langle\varepsilon\rangle$.

In many cases, the assumption $|\mathbf{x}| = |\mathbf{y}|$ does not hold. To overcome this limitation, we can utilize the length ratio $\gamma = |\mathbf{x}|/|\mathbf{y}|$ to modify the length difference: $d' = |\mathbf{s}| - \gamma|\mathbf{t}|$, and use this new difference d' in our dynamic oracle. Although we cannot obtain this ratio during testing time, we may use the averaged length ratio obtained from training data (Huang et al., 2017).

Training with Restricted Dynamic Oracle We apply imitation learning to train our translation model, using the proposed dynamic oracle as the expert policy. Recall that the prediction of our model depends on the whole generated prefix including $\langle\varepsilon\rangle$ (as the input contains the embedding of the number of $\langle\varepsilon\rangle$), which is also an action sequence. If an action sequence $\mathbf{a}$ is obtained from our oracle, then applying this sequence will result in a prefix pair, say $\mathbf{s_a}$ and $\mathbf{t_a}$, of $\mathbf{x}$ and $\mathbf{y}$. Let $p(a \mid \mathbf{s_a}, \mathbf{t_a})$ be the probability of choosing action a given the prefix pair obtained by applying action sequence $\mathbf{a}$. Then the averaged probability of choosing the oracle actions conditioned on the action sequence $\mathbf{a}$ will be

$$f(\mathbf{a}, \pi^{\star}_{\mathbf{x},\mathbf{y},\alpha,\beta}) = \frac{\sum_{a \in \pi^{\star}_{\mathbf{x},\mathbf{y},\alpha,\beta}(\mathbf{s_a},\mathbf{t_a})} p(a \mid \mathbf{s_a}, \mathbf{t_a})}{|\pi^{\star}_{\mathbf{x},\mathbf{y},\alpha,\beta}(\mathbf{s_a}, \mathbf{t_a})|}.$$

To train a model to learn from the dynamic oracle, we can sample from our oracle to obtain a set, say $S(\mathbf{x}, \mathbf{y})$, of action sequences for a sentence pair $(\mathbf{x}, \mathbf{y})$. The loss function for each sampled sequence $\mathbf{a} \in S(\mathbf{x}, \mathbf{y})$ will be

$$\ell(\mathbf{a}|\mathbf{x}, \mathbf{y}) = -\sum_{i=1}^{|\mathbf{a}|} \log f(\mathbf{a}_{<i}, \pi^{\star}_{\mathbf{x},\mathbf{y},\alpha,\beta}).$$

For a parallel text D, the training loss is

$$\ell(D) = \sum_{(\mathbf{x},\mathbf{y}) \in D} \sum_{\mathbf{a} \in S(\mathbf{x},\mathbf{y})} \frac{1}{|S(\mathbf{x},\mathbf{y})|} \ell(\mathbf{a}|\mathbf{x}, \mathbf{y}).$$

Directly optimizing the above loss may require too much computation resource since for each pair of $(\mathbf{x}, \mathbf{y})$, the size of $S(\mathbf{x}, \mathbf{y})$ (i.e. the number of different action sequences) can be exponentially large. To reduce the computation cost, we propose to use two special action sequences as our sample set so that our model can learn to do translation within the two latency constraints. Recall that the latency constraints of our dynamic oracle $\pi^{\star}_{\mathbf{x},\mathbf{y},\alpha,\beta}$ are defined by two bounds: α and β. For each bound, there is a unique action sequence, which corresponds to a path in the prefix grid, such that following it can generate the most number of prefix pairs that make this bound tight. Let $\mathbf{a}^{\alpha}_{(\mathbf{x},\mathbf{y})}$ ($\mathbf{a}^{\beta}_{(\mathbf{x},\mathbf{y})}$) be such an action sequence for $(\mathbf{x}, \mathbf{y})$ and α (β). We replace $S(\mathbf{x}, \mathbf{y})$ with $\{\mathbf{a}^{\alpha}_{(\mathbf{x},\mathbf{y})}, \mathbf{a}^{\beta}_{(\mathbf{x},\mathbf{y})}\}$, then the above loss for dataset D becomes

$$\ell_{\alpha,\beta}(D) = \sum_{(\mathbf{x},\mathbf{y}) \in D} \frac{\ell(\mathbf{a}^{\alpha}_{(\mathbf{x},\mathbf{y})}|\mathbf{x},\mathbf{y}) + \ell(\mathbf{a}^{\beta}_{(\mathbf{x},\mathbf{y})}|\mathbf{x},\mathbf{y})}{2}.$$

This is the loss we use in our training process.

Note that there are some steps where our oracle will return two actions, so for such steps we will have a multi-label classification problem where labels are the actions from our oracle. In such cases, Sigmoid function for each action is more appropriate than the Softmax function for the actions will not compete each other (Ma et al., 2017; Zheng et al., 2018; Ma et al., 2019). Therefore, we apply Sigmoid for each action instead of using Softmax function to generate a distribution for all actions.

4.2 Decoding

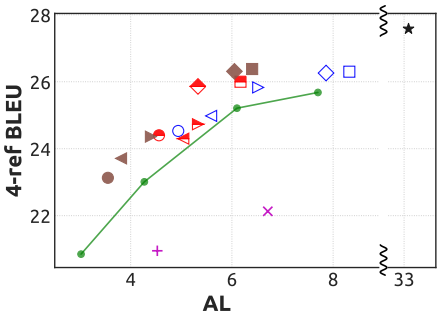
We observed that the model trained on the two special action sequences occasionally violates the latency constraints and visits states outside of the designated band in prefix grid. To avoid such case, we force the model to choose actions such that it will always satisfy the latency constraints. That is, if the model reaches the aggressive bound, it must choose a target word other than $\langle\varepsilon\rangle$ with highest score, even if $\langle\varepsilon\rangle$ has higher score; if the model reaches the conservative bound, it can only choose $\langle\varepsilon\rangle$ at that step. We also apply a temperature constant e^t to the score of $\langle\varepsilon\rangle$, which can implicitly control the latency of our model without retraining it. This improves the flexibility of our trained model so that it can be used in different scenarios with different latency requirements.

5 Experiments

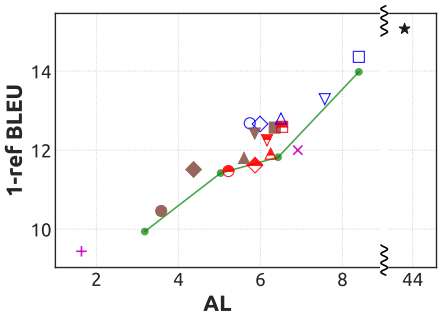
To investigate the empirical performance of our proposed method, we conduct experiments on NIST corpus for Chinese-English. We use NIST 06 (616 sentence pairs) as our development set and NIST 08 (691 sentence pairs) as our testing set. We apply tokenization and byte-pair encoding (BPE) (Sennrich et al., 2015) on both source and target languages to reduce their vocabularies. For training data, we only include 1 million sentence pairs with length larger than 50. We use Transformer (Vaswani et al., 2017) as our NMT model, and our implementation is adapted from PyTorch-based OpenNMT (Klein et al., 2017). The architecture of our Transformer model is the same as the base model in the original paper.

We use BLEU (Papineni et al., 2002) as the translation quality metric and *Average Lagging* (AL) introduced by Ma et al. (2018) as our latency metrics, which measures the average delayed words. AL avoids some limitations of other existing metrics, such as insensitivity to actual lagging like Consecutive Wait (CW) (Gu et al., 2017), and sensitivity to input length like Average Proportion (AP) (Cho and Esipova, 2016) .

Results We tried three different pairs for α and β: (1, 5), (3, 5) and (3, 7), and summarize the results on testing sets in Figure 2. Figure 2 (a) shows the results on Chinese-to-English translation. In this direction, our model can always achieve higher BLEU scores with the same latency, compared with the wait-k models and RL models. We notice the model prefers conservative policy during decoding time when $t = 0$. So we apply negative values of t to encourage the model to choose actions other than $\langle\varepsilon\rangle$. This can effectively reduce latency without sacrificing much



(a) Chinese-to-English



(b) English-to-Chinese

Figure 2: Translation quality (BLEU) against latency (AL) on testing sets. Markers ●: wait-k models for $k \in \{1, 3, 5, 7\}$, +: RL with CW = 5, ×: RL with CW = 8, ★: full-sentence translation. Markers for our models are given in the right table.

Training		Decoding Policy						
α	β	wait-α	wait-β	$t=-2$	$t=-0.5$	$t=0$	$t=4.5$	$t=9$
1	5	●	■	◀	▶	◆	▲	▼
3	5	◓	◓	◀	▶	◆	△	▼
3	7	○	□	◁	▷	◇	△	▽

translation quality, implying that our model can implicitly control latency during testing time.

Figure 2 (b) shows our results on English-to-Chinese translation. Since the English source sentences are always longer than the Chinese sentences, we utilize the length ratio $\gamma = 1.25$ (derived from the dev set) during training, which is the same as using "catchup" with frequency $c = 0.25$ introduced by Ma et al. (2018). Different from the other direction, models for this direction works better if the difference of α and β is bigger. Another difference is that our model prefers aggressive policy instead of conservative policy when $t = 0$. Thus, we apply positive values of t to encourage it to choose $\langle\varepsilon\rangle$, obtaining more conservative policies to improve translation quality.

Example We provide an example from the development set of Chinese-to-English translation in Table 3 to compare the behaviours of different models. Our model is trained with $\alpha = 3, \beta = 7$ and tested with $t = 0$. It shows that our model can wait for information "*Ōuméng*" to translates "eu", while the wait-3 model is forced to guess this information and made a mistake on the wrong guess "us" before seeing "*Ōuméng*".

Ablation Study To analyze the effects of proposed techniques on the performance, we also provide an ablation study on those techniques for our model trained with $\alpha = 3$ and $\beta = 5$ in Chinese-to-English translation. The results are given in Table 4, and show that all the techniques are important to the final performance and using Sigmoid function is critical to learn adaptive policy.

Model	Decoding Policy					
	Wait-3		Wait-5		t=0	
	BLEU	AL	BLEU	AL	BLEU	AL
Wait-3	29.32	4.60	-	-	-	-
Wait-5	-	-	30.97	6.30	-	-
keep $\langle\varepsilon\rangle$ in attention	29.55	4.50	30.68	6.49	30.74	6.53
no $\langle\varepsilon\rangle$ number embedding	30.20	4.76	30.98	6.36	30.65	6.29
use Softmax instead of Sigmoid	29.23	5.11	31.46	6.79	29.99	4.79
Full	29.45	4.71	31.72	6.35	31.59	6.28

Table 4: Ablation study on Chinese-to-English development set with $\alpha = 3$ and $\beta = 5$.

6 Conclusions

We have presented a simple model that includes a delay token in the target vocabulary such that the model can apply both READ and WRITE actions during translation process without a explicit policy model. We also designed a restricted dynamic oracle for the simultaneous translation problem and provided a local training method utilizing this dynamic oracle. The model trained with this method can learn a flexible policy for simultaneous translation and achieve better translation quality and lower latency compared to previous methods.

Chinese	一	名	不	愿	具名	的	欧盟	官员	指出	...
pinyin	*yì*	*míng*	*bú*	*yùan*	*jùmíng*	*de*	*Ōuméng*	*gūanyúan*	*zhǐchū*	
gloss	a	-	not	willing	named	's	EU	official	point out	...
wait-3			a		us	official	who	declined	to	be named said that ...
our work			a					eu	official	, who declined to be named , pointed out ...

Table 3: A Chinese-to-English development set example. Our model is trained with $\alpha = 3$ and $\beta = 7$.

References

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

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

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of EMNLP*.

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, New Orleans, Louisiana. Association for Computational Linguistics.

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

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. ACM.

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, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1053–1062.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2134–2139.

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, Bing Xiang, and Bowen Zhou. 2017. Group sparse CNNs for question classification with answer sets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 335–340, Vancouver, Canada. Association for Computational Linguistics.

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. 2018. STACL: Simultaneous translation with integrated anticipation and controllable latency. *arXiv preprint arXiv:1810.08398, to appear ACL 2019*.

Mingbo Ma, Renjie Zheng, and Liang Huang. 2019. Learning to stop in structured prediction for neural machine translation. *arXiv preprint arXiv:1904.01032, to appear NAACL 2019*.

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.

Ofir Press and Noah A. Smith. 2018. You may not need attention. *arXiv preprint arXiv:1810.13409*.

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

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 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*.

Renjie Zheng, Mingbo Ma, and Liang Huang. 2018. Multi-reference training with pseudo-references for neural translation and text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3188–3197.