Data-Driven Adaptive Simultaneous Machine Translation

Anonymous ACL submission

Abstract

In simultaneous translation (SimulMT), the most widely used strategy is the wait-k policy thanks to its simplicity and effectiveness in balancing translation quality and latency. However, wait-k suffers from two major limitations: (a) it is a fixed policy that can not adaptively adjust latency given context, and (b) its training is much slower than full-sentence translation. To alleviate these issues, we propose a novel and efficient training scheme for adaptive SimulMT by augmenting the training corpus with adaptive prefix-to-prefix pairs, while the training complexity remains the same as that of training full-sentence translation models. Experiments on two language pairs show that our method outperforms all strong baselines in terms of translation quality and latency.

1 Introduction

Simultaneous Machine Translation (SimulMT) which starts translation before the source sentence finishes, is broadly used in many scenarios which require much shorter translation latency for spontaneous communication such as international conferences, traveling and negotiations. Most existing SimulMT models (Ma et al., 2019a,b; Arivazhagan et al., 2019, 2020; Ma et al., 2020; Ren et al., 2020) exploit the wait-k policy (Ma et al., 2019a) as a fundamental design of their framework due to the simplicity and effectiveness of the wait-k policy.

However, there are two major drawbacks of wait-k policy. Firstly, the wait-k policy is computationally expensive and requires substantial GPU memory. Given a certain value of k, the wait-k policy iterates over all possible prefix pairs between source and target sentences to enforce the model to learn to anticipate future information of the source sentence. This process escalates the original time complexity from quadratic in full-sentence translation to cubic in SimulMT. Secondly, the wait-k policy is a fixed decoding strategy which can not adjust the latency given difference circumstances. Some researchers adapted the wait-k policy into adaptive policies (Zheng et al., 2019, 2020). But those methods need to train multiple wait-k models with different k, which escalate the training complexity even further.

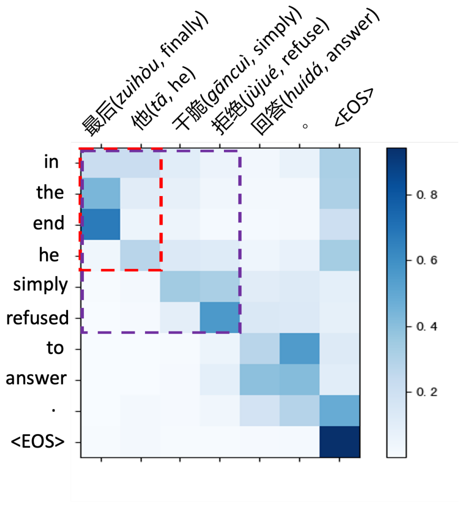

Figure 1: Attention map generated by AdaData for a Zh→En full-sentence pair. The dashed boxes illustrate two pairs of self-translatable prefixes. Given a source prefix, a target prefix is translatable if the dashed attention block contains enough information for translation.

To perform adaptive SimulMT training without escalating the training time and computation resource, we propose a data-driven adaptive SimulMT system, named AdaData. Our AdaData system automatically generates self-translatable prefix pairs. We define a self-translatable prefix pair if the source prefix reaches a certain information threshold to obtain the target translation prefix. Then those generated prefix pairs enable us to train an adaptive SimulMT model in the fashion of full-sentence training. The advantages are twofold: first, the SimulMT knowledge resides in the training data instead of in the model and hence not necessary for model modifications; second, by sampling over the generated prefix pairs, the training complexity remains the same as full-sentence translation training. Figure 1 illustrates how to generate two pairs of self-translatable prefixes based on the source-target attention map. Moreover, the generated data allows us to fine-tune a pre-trained full-sentence model and dramatically improves the training efficiency. Experiments on De→En and Zh→En simultaneous translation show substantial improvements for both AdaData and AdaData fine-tuning over strong baselines.

2 Adaptive Prefix Pair Generation

SimulMT consists of two subtasks: the partial translation task when the source sentence is incomplete and the full-sentence translation task when the source sentence is fully observed. This inspires us to treat SimulMT as a multi-task learning problem, where the learned model is able to perform both partial and full-sentence translation depending on the completeness of the source sentence.

In order to learn partial translation, we need to generate artificial prefix pairs from full-sentence pairs to train the model. The major challenge in prefix pair generation is to determine the lengths of the source prefix and the target prefix. As both translation quality and latency are required by SimulMT systems, the generated prefix pairs should meet two basic rules. First, the source prefix should carry necessary information to generate the target prefix for translation quality. Second, the target prefix should contain as many words as possible to minimize translation latency.

To fulfill the two requirements, one key point is to measure the information carried by each source word to predict a target word. The idea of weighted information is similar to the attention mechanism where each word is assigned an attention weight. Therefore, we utilize the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) architecture with the attention mechanism (Bahdanau et al., 2014) to help us generate prefix pairs. In this way, the cumulative information of a source prefix to predict a specific target word can be measured by the sum of its attention weights. In order to obtain accurate cumulative information of a source prefix, we propose the Monotonic LSTM model (MonoLSTM). Compared with the regular attention LSTM model, MonoLSTM utilizes a unidirectional encoder and cut off the connection between the last

Algorithm 1: Prefix pair generation

```
Input: Full-sentence pairs, threshold $e$ and
       MonoLSTM
Output: Adaptive prefix pairs
for each full pair $(\{x_{1:S}\}, \{y_{1:T}\})$ do
    generate source-target attention matrix $\mathbb{A}$
      using MonoLSTM;
    for $s \in \{1, ..., S\}$ do
        for $t \in \{1, ..., T\}$ do
            if $\sigma_{t,1:s} < e$ then
                break;
            add $(\{x_{1:s}\}, \{y_{1:t-1}\})$ to prefix pairs;
return prefix pairs
```

encoder hidden state and the initial decoder hidden state. By doing so, MonoLSTM prevents source word information from flowing backward and from directly flowing into the decoder. Consequently, the attention module is the only information pathway from the source side to the target side, enabling us to generate self-translatable prefix pairs based on the attention weights.

Specifically, given a full sentence pair of a source sentence $\{x_{1:S}\}$ and a target sentence $\{y_{1:T}\}$, MonoLSTM can generate a $T \times S$ attention matrix $\mathbb{A}$ where each attention score α_{ts} is only affected by prefixes $\{x_{1:s}\}$ and $\{y_{1:t}\}$. For more details about MonoLSTM, refer to Appendix. So the cumulative information a source prefix $\{x_{1:s}\}$ provides to generate the target word y_t can be measured by the sum of attention weights:

$$\sigma_{t,1:s} = \sum_{i=1}^{s} \alpha_{ti}. \tag{1}$$

Given a hyperparameter threshold e, we mathematically define that a target prefix $\{y_{1:t}\}$ is translatable from a prefix $\{x_{1:s}\}$ if all words in the target prefix are translatable from the source prefix, i.e., $\forall j \leq t$, we have $\sigma_{j,1:s} \geq e$. Hence, we can generate adaptive prefix pairs based on this cumulative information. The data generation process is illustrated in Algorithm 1. In addition, we can also adjust the latency trade-off by changing the threshold e.

3 Data-driven Adaptive Prefix Training

3.1 Adaptive Training

In order to balance the performance on both partial translation and full-sentence translation, we subsample the generated prefix pairs and train the model on a 1:1 mix of prefix pairs and full-sentence pairs for each training epoch. Since prefix pairs are shorter than full-sentence pairs, the multi-tasking training is more efficient than training two full-sentence translation models.

3.2 Adaptive Inference

Since the simultaneous translation model is jointly trained for partial translation and full-sentence translation, it is able to adaptively generate the translation at each decoding step. To be more specific, when a new source word arrives, the system starts to decode until the '<eos>' token is generated. This adaptive inference not only closes the gap between training and inference, but also allows the model to adaptively tune the simultaneous translation latency based on the source sequence.

3.3 Adaptive Fine-tuning

Another advantage of data-driven adaptive training is that we can significantly improve the training efficiency by fine-tuning a full-sentence translation model using the generated adaptive prefix pairs. In our experiments, we found that 1 epoch of fine-tuning on the 1:1 mixed data is able to achieve comparable performances as the models trained from scratch.

4 Experiments

4.1 Datasets and Evaluation Metrics

We conduct experiments on the WMT15 German-to-English dataset (De $\rightarrow$ En, 4.5M sentence pairs) and the NIST Chinese-to-English dataset (Zh $\rightarrow$ En, 2M sentence pairs). For De $\rightarrow$ En, we use newstest-2013 for validation and use newstest-2015 for test. For Zh $\rightarrow$ En, we use NIST 2006 for validation and use NIST 2008 for test. Both datasets are tokenized using BPE (Sennrich et al., 2015).

We use BLEU (Papineni et al., 2002) to evaluate the translation quality and use Average Lagging (AL) (Ma et al., 2019a) to measure the translation latency. Since each Chinese sentence in the Zh $\rightarrow$ En dataset has 4 English references, we report 4-reference BLEU scores for this dataset.

4.2 Baselines and Model Configurations

The baseline models we include in the experiments are Wait-k (Ma et al., 2019a), MILk (Arivazhagan et al., 2019), MMA-H, MMA-IL (Ma et al., 2019b), Proportional re-translation (Niehues et al., 2018; Arivazhagan et al., 2020), and a policy-based adaptive model (Zheng et al., 2020). All baseline results are copied from their original papers except for Proportional re-translation. Since the Proportional re-translation paper reports DAL (Cherry and Foster, 2019) instead of AL as their latency metric, we replicate the model and report results in AL.

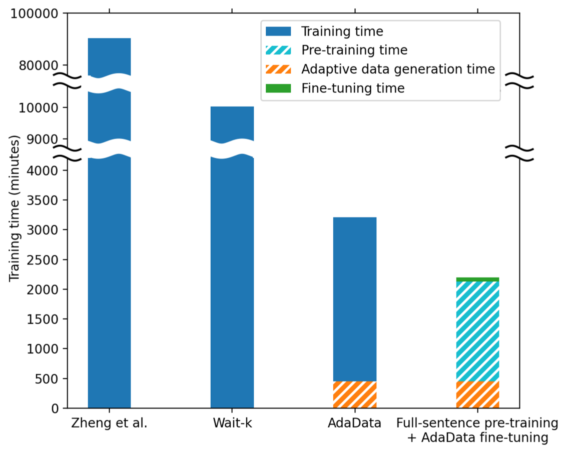

Figure 2: Training efficiency comparison of Wait-k, AdaData and AdaData fine-tuning on the De$\rightarrow$En dataset using 10 $\times$ 1080Ti GPUs.

To be consistent with the baseline models, we adopt the Transformer-base (Vaswani et al., 2017) architecture as our simultaneous translation model. It consists of a 6-layer encoder and a 6-layer decoder. The model dimension is set to 512 and the number of attention heads is 8. Since the MMA-H model and the MMA-IL model are based on 1024-dimensional Transformers, we also carry out experiments with a wider configuration of our model to make a fair comparison with MMA-H and MMA-IL. Specifically, as with MMA-H and MMA-IL, the wider version of our model has 1024 dimensions and the number of attention heads is set 16. The number of layers remains to be 6.

For the adaptive prefix generation part, our proposed MonoLSTM is based on a 2-layer, 512-dimensional unidirectional LSTM. Please find more training details in Appendix.

4.3 Training Efficiency

Figure 2 illustrates the training efficiency of Wait-k, (Zheng et al., 2020) and our methods. Note that, the hatched bars represent "run only once". For example, we can pre-train a full-sentence model once and fine-tune multiple times with different in-

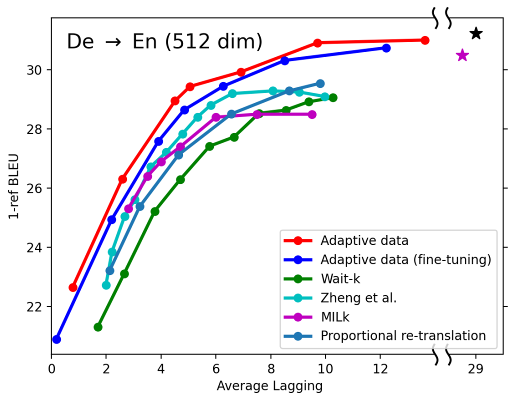

Figure 3: Translation quality against latency curves for De → En test sets with the basic 512 dimension configuration.

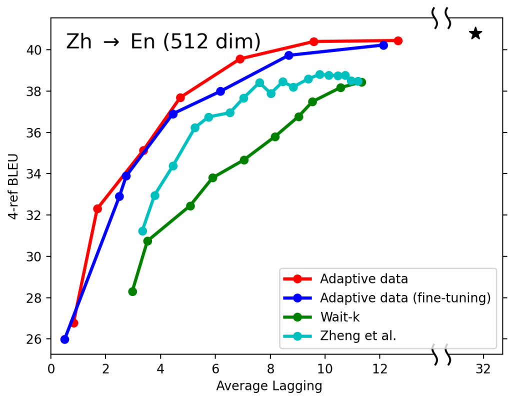

Figure 5: Translation quality against latency curves for Zh → En test sets with the basic 512 dimension configuration.

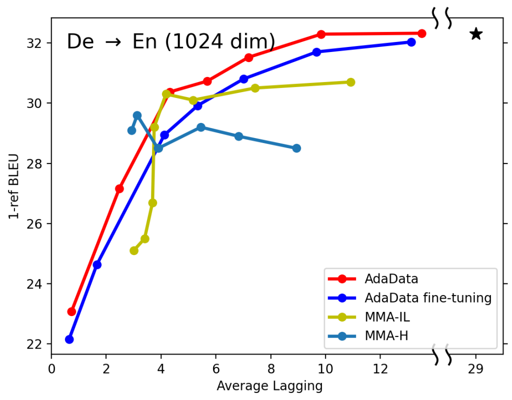

Figure 4: Translation quality against latency curves for De → En test sets with the wide 1024 dimension configuration.

formation threshold e for different quality-latency trade-offs. The fine-tuned models are only fine-tuned for 1 epoch. We can see that our methods achieve significantly better training efficiency, especially AdaData fine-tuning.

4.4 Performance Comparison

In order for our model to obtain different quality-latency trade-offs, we set the cumulative information threshold e ranging from 0.1 to 0.7 with a step size of 0.1. The 512-dimensional model comparison results and the 1024-dimensional model comparison results on the De → En test sets are reported in Figure 3 and Figure 4 respectively. MILk is based on LSTMs, resulting in a lower full-sentence BLEU score compared with other Transformer based models. The erasure rate of Proportional re-translation is set to 0 to match other models. Proportional re-translation is also a data-driven method. In their papers, they discussed two ways to generate prefix pairs: proportion and word alignments. They also found that both methods achieve similar performances. So we report the proportion method here for simplicity. Since neither ways can accurately measure the cumulative information in prefixes, their performances are worse than ours. To sum up, compared with the baseline models, our proposed model achieves better quality-latency trade-offs and larger areas under curves.

The comparison results on the Zh → En test sets are presented in Figure 5. We can see that our proposed model is still able to outperform the baseline models.

It is noteworthy that our fine-tuned models are also able to achieve better performances than the baseline models. Considering the models are fine-tuned for only 1 epoch, this demonstrates that the generated adaptive data can help us greatly improve the training efficiency without significantly hurting the model performance.

5 Conclusions

We have designed an efficient data-driven algorithm for adaptive SimulMT. The proposed method is able to measure the information requirement for each decoding step and generate self-translatable prefix pairs for partial translation. Experiments validate the effectiveness of the proposed data-driven adaptive SimulMT system. Moreover, the generated prefix pairs enable us to fine-tune pre-trained full-sentence translation models and dramatically improve the training efficiency.

References

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. *arXiv preprint arXiv:1906.05218*.

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

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

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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.

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

Xutai Ma, Juan Pino, and Philipp Koehn. 2020. 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*.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-latency neural speech translation. *arXiv preprint arXiv:1808.00491*.

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.

Yi Ren, Jinglin Liu, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. SimulSpeech: End-to-end simultaneous speech to text translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

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*.

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. 2019. Simultaneous translation with flexible policy via restricted imitation learning. In *ACL*.

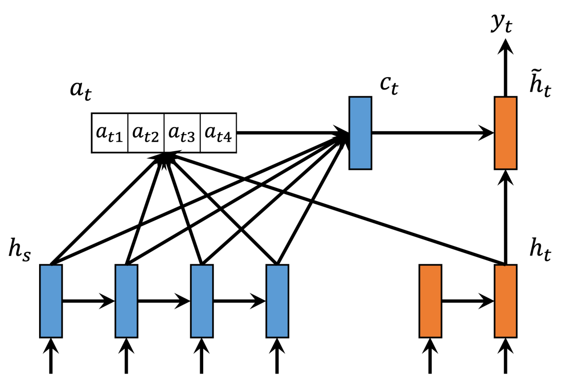

Figure 6: MonoLSTM.

A Example Appendix

B Details of MonoLSTM

The framework of MonoLSTM is shown in Figure 6. Compared with the regular attention LSTM model, MonoLSTM utilizes a unidirectional encoder and cut off the connection between the last encoder hidden state and the initial decoder hidden state. By doing so, MonoLSTM prevents source word information from flowing backward and from directly flowing into the decoder. Thus, the attention module is the only information pathway from the source side to the target side, ensuring that at each decoding step, the attention score of each source word is only affected by the current word and the previous words.

More specifically, given a full sentence pair of a source sentence $\{x_{1:S}\}$ and a target sentence $\{y_{1:T}\}$, at each decoding time step t, target word y_t is predicted as:

$$p(y_t|y_{<t}, x) = \text{softmax}(\boldsymbol{W}\tilde{\boldsymbol{h}}_t), \tag{2}$$

where $\tilde{\boldsymbol{h}}_t$ is the concatenation of the target hidden state $\boldsymbol{h}_t$ and the source-side context vector $\boldsymbol{c}_t$. The context vector $\boldsymbol{c}_t$ represents the summary of the source sentence at time step t, which is computed as the weighted sum of the source hidden states based on their attention scores:

$$\boldsymbol{c}_t = \sum_s \alpha_{ts}\bar{\boldsymbol{h}}_s, \tag{3}$$

where $\bar{\boldsymbol{h}}_s$ is the source hidden state and α_{ts} denotes the attention score between target word y_t and source word x_s:

$$\alpha_{ts} = \frac{\exp(\boldsymbol{h}_t^\intercal \bar{\boldsymbol{h}}_s)}{\sum_{i=1}^{S}\exp(\boldsymbol{h}_t^\intercal \bar{\boldsymbol{h}}_i)}. \tag{4}$$

The attention scores are normalized and sum up to 1. We can see that, in MonoLSTM $\bar{\boldsymbol{h}}_s$ does not contain information for the source words after x_s by removing the backward LSTM and $\boldsymbol{h}_t$ does not contain information for the source sentence by cutting of the connection between source hidden states and target hidden initialization. Therefore, the unnormalized attention score, i.e., the numerator of α_{ts} is only affected by the source prefix $\{x_{1:s}\}$ and the target prefix $\{y_{1:t}\}$. Then we can use the attention scores to measure the information a source prefix provides to produce a target prefix.

C Details of Training Configurations

For the 512-dimensional models on De $\rightarrow$ En, they are trained on 10 $\times$ 1080Ti GPUs. It consists of a 6-layer encoder and a 6-layer decoder. The model dimension is set to 512, the intermediate dimension is 2048 and the number of attention heads is 8. The number of max tokens is 6144. The optimizer is Adam with betas set to (0.9, 0.98). The learning rate is 0.00015 and the warm-up steps are 2000. We also reduce the learning rate on plateaus with a patience of 4. Weight decay is 0.000001. Label smoothing is 0.1 and dropout rate is 0.3.

For the 1024-dimensional models on De $\rightarrow$ En, they are trained on 8 $\times$ TitanX GPUs. It consists of a 6-layer encoder and a 6-layer decoder. The model dimension is set to 1024, the intermediate dimension is 4096 and the number of attention heads is 16. The training configurations follow (Ma et al., 2019b). The number of max tokens is 3584. The optimizer is Adam with betas set to (0.9, 0.98). The learning rate is 0.0005 and the warm-up steps are 6000. The learning rate scheduler is inverse_sqrt. The warm-up initial learning rate is 1e-7. Label smoothing is 0.1 and dropout rate is 0.3.

For the 512-dimensional models on Zh $\rightarrow$ En, they are trained on 10 $\times$ 1080Ti GPUs. It consists of a 6-layer encoder and a 6-layer decoder. The model dimension is set to 512, the intermediate dimension is 2048 and the number of attention heads is 8. The number of max tokens is 10240. The optimizer is Adam with betas set to (0.9, 0.98). The learning rate is 0.00025 and the warm-up steps are 500. We also reduce the learning rate on plateaus with a patience of 4. Weight decay is 0.000001. Label smoothing is 0.1 and dropout rate is 0.3.

For the adaptive prefix generation part, our proposed MonoLSTM is based on a 2-layer, 512-

dimensional unidirectional LSTM. It is also trained on 10 $\times$ 1080Ti GPUs. The number of max tokens is 20480. The optimizer is Adam with betas set to (0.9, 0.997) and epsilon set to 1e-9. The learning rate is 0.005 and the warm-up steps are 4000. We also reduce the learning rate on plateaus with a patience of 4. Weight decay is 0.000001. Label smoothing is 0.1 and dropout rate is 0.2.

For De $\rightarrow$ En, the BPE vocabulary for De and En are 16k and 16k, respectively. For Zh $\rightarrow$ En, the BPE vocabulary for Zh and En are 20k and 10k, respectively.

For fine-tuning, all training configuration remain the same except that learning rate is multiplied by 0.1.