BIT-Xiaomi's Simultaneous Translation System for AutoSimTrans 2022

Mengge Liu[1*] **Xiang Li**[2] **Bao Chen**[1] **Yanzhi Tian**[1] **Tianwei Lan**[1]
Silin Li[1] **Yuhang Guo**[1†] **Jian Luan**[2] **Bin Wang**[2]
[1]Beijing Institute of Technology, Beijing, China
[2]Xiaomi AI Lab, Beijing, China
liumengge@bit.edu.cn lixiang21@xiaomi.com
{chenbao,tianyanzhi,lantianwei,lisilin,guoyuhang}@bit.edu.cn
{luanjian,wangbin11}@xiaomi.com

Abstract

This system paper describes the BIT-Xiaomi simultaneous translation system for Autosimtrans 2022 simultaneous translation challenge. We participated in three tracks: the Zh-En text-to-text track, the Zh-En audio-to-text track, and the En-Es test-to-text track. In our system, wait-k is utilized to train prefix-to-prefix translation models. We integrate streaming chunking to detect segmentation boundaries as the source streaming reading in. We further improve our system with data selection, data augmentation, and R-Drop training methods. Results show that our wait-k implementation outperforms the organizer's baseline by at most 8 BLEU score and our proposed streaming chunking method further improves by about 2 BLEU score in the low latency regime.

1 Introduction

Simultaneous translation (Cho and Esipova, 2016; Yarmohammadi et al., 2013; Ma et al., 2019), is a task in Machine Translation (MT), which intends to provide low latency translation in real-time scenarios. To achieve low latency translation, the translation system needs to begin translating before the end of source sentences, which can be viewed as prefix-to-prefix translation (Ma et al., 2019). Simultaneous translation is widely used in real-time translation scenarios such as simultaneous interpretation, online subtitles, and live broadcasting. In these scenarios, low latency may have equal or even higher priority than translation quality.

In simultaneous translation, the most challenge is the balance of translation quality and latency. Low latency translation requires beginning translation with insufficient source information, which may cause incorrect translation results. How to find a simultaneous policy to balance quality and latency is the most challenging question. On another hand, in most cases, the standard machine translation model is trained on full sentences, which can achieve good performance in full-sentence evaluation. But for prefix-to-prefix inference, which is crucial for simultaneous translation, the standard machine translation model always perform poorly.

Previous methods for simultaneous translation can be classified as the fixed policy and the adaptive policy according to different simultaneous policies. Fixed policy uses fixed-latency simultaneous strategy, for example, set value K, and forces the translation to lag behind source for K tokens (Ma et al., 2019). The adaptive policy needs an agent module to perform adaptive simultaneous translation. The agent will consider the current translation state, including the source prefix and the hypothesis prefix, to decide whether to output new tokens at the current state (Gu et al., 2017; Arivazhagan et al., 2019; Ma et al., 2020). Chunk-base (Xiong et al., 2019; Zhang et al., 2020) simultaneous translation is a special adaptive policy, which makes a decision only based on the source prefix.

In our system, we propose a streaming chunking method that can be combined with a fixed wait-k policy. The streaming chunking method can significantly improve translation quality with little latency increase in low latency regions. We train a segmentation model to detect boundaries in streaming sources and employ a wait-k policy to decide output token numbers. We pre-train transformer models with multi-path wait-k on a

[*]The work was done during the author's internship at Xiaomi.
[†] Corresponding author.

Proceedings of the Third Workshop on Automatic Simultaneous Translation, pages 34 - 42

Track	Corpus	#Sentence Pairs
Zh-En	BSTC	38K
	CWMT	9M
En-Es	UN Parallel	22M
Zh ASR	BSTC	68h
	AIshell	150h

Table 1: Data statistics. Parallel corpus is counted by sentence pairs. ASR corpus is counted by audio time (hour).

large general corpus and fine-tune with single k on a small domain corpus. We augment the general corpus and domain corpus with Back-Translation (BT) and Front-Translation (FT), and further augment the domain corpus with character-level pseudo ASR error. In training we incorporate R-Drop (liang et al., 2021) method to improve translation quality. In text-to-text tracks, we use text streaming input provided by the organizer. In the audio-to-text track, we train our ASR system to transcript audio into the streaming text as translation input.

The remainder of this paper is organized as follows. We describe the techniques employed in our system and the methods we propose in Section 2. In Section 3 we show our experiment settings and results, including data and model. Finally, we conclude this paper.

2 Methods

In this section, we describe the data, the utilized prefix-to-prefix translation model, and the proposed streaming chunking method.

2.1 Data

We describe the data used in our system from the following aspects: statistics, preprocessing, filtering and data-augmentation.

All allowed bilingual training sets are employed, including the BSTC (Zhang et al., 2021) and the CWMT21 for the Zh-En track, the UN Parallel Corpus for the En-Es track. For the ASR model in the Zh-En audio-to-text track, we use the BSTC and the AIshell (Hui Bu, 2017) corpus for training. Data statistics are shown in Table 1.

Pre-processing. `Sacremoses`[1] is conducted to normalize and tokenize English and Spanish sentences. `Jieba`[2] is used to segment Chinese sentences. And redundant spaces in the text are removed. After tokenization, we apply `Subword-nmt`[3] to learn byte-pair encoding with 32K operations.

Data filtering. The noises in the original data may bring a negative impact on translation quality, so we filter the training set as following steps:

- First, the parallel corpus is filtered by hand-crafted rules. Sentences that contain less than 30% linguistic words will be viewed as noise sentences. When any sentence in a sentence pair is judged as noise, this pair is discarded. For Chinese sentences, we consider Chinese characters as linguistic words. For En or Es, we consider words only containing alphabet characters as linguistic words.
- Second, we utilize `fast_align`[4] to filter out poorly aligned sentence pairs. We calculate align scores for each sentence pair and filter out sentence pairs with low scores. Align score threshold is set as -7.
- Third, language identification is applied with `langid`[5]. Sentences in the wrong languages are viewed as low-quality samples and removed.
- Finally, we discard duplicate pairs and remove the pair with a length ratio greater than 3.0 or the sentence with a length more than 200.

Data selection Because the bilingual corpus utilized in training is not all from the speech domain, we use a language-model-based data selection method select domain data, which is similar to methods proposed by Moore and Lewis (2010). We train two 5-gram language model on source sentences with `KenLM`[6], one on the BSTC corpus (denoted as lm^{in}), another on the CWMT corpus (denoted as lm^{out}). Than for each sentence in the CWMT corpus, we compute the perplexity distances with two language model, which denoted as domain

[1] https://github.com/alvations/sacremoses

[2] https://github.com/fxsjy/jieba

[3] https://github.com/rsennrich/subword-nmt

[4] https://github.com/clab/fast_align

[5] https://github.com/saffsd/langid.py

[6] https://github.com/kpu/kenlm

score for the sentence $ppl\_score = -(ppl^{in} - ppl^{out})$. We sort the corpus by domain score and remove the pair with a large domain distance.

Data augmentation As the training corpus is limited, we utilize back-translation (BT) and front-translation (FT) to augment the training corpus. We first train two translation models in two directions: Zh-En and En-Zh, then generate pseudo training corpus in two directions.

2.2 R-Drop

R-Drop[7] is a method to improve translation quality in machine translation, which can be easily incorporated with our translation model. All models in our system are trained with the R-Drop algorithm proposed by liang et al. (2021).

2.3 Wait-k

Wait-k is a simple and effective method for fixed-policy simultaneous translation, which can train prefix-to-prefix translation ability for transformer models. We build our system based on `fairseq`, which provides a wait-k baseline similar to efficient wait-k (Elbayad et al., 2020). Two-stage training is employed to achieve better performance in the speech domain. Model is firstly trained on large scale parallel corpus with multi-path wait-k, which randomly selects a value of k within the interval (for example, [k, k+n]) for each training batch (denoted as `wait(k)-(k+n)`). Secondly, we fine-tune the model with a small speech domain parallel corpus with simple wait-k (denoted as `wait(k)`) or multi-path wait-k.

2.4 Streaming Chunking

In a streaming translation system, the source is received token by token. The wait-k policy will try to translate each time source is ahead of target for k tokens, which may bring some mistakes when the source stops at a partial phrase. Especially for Chinese streaming input, in which source streaming is growing by character. So some source prefixes may contain incomplete word pieces which may cause misunderstanding and incorrect translation. A stream case with error source prefixes is shown in Table 2. We propose a streaming chunking method, which employs a streaming segmentation model to detect word boundaries on-the-fly in streaming input.

2.4.1 Streaming Segmentation Model

We build our streaming segmentation model base on `chinese-roberta-wwm-ext`[8] proposed by Cui et al. (2021). Compared with a vanilla Chinese word segmentation model, the streaming segmentation model does not need to obtain the complete sentence and can segment words without introducing an additional delay. We treat the streaming word segmentation task as a sequence classification task and use the final hidden state of the classification token ([CLS]) to perform binary classification through a 3-layer fully connected network to determine whether the current source sentence prefix end with complete words. We construct training data using transcribed sentences from the BSTC training set. The complete sentences in the training data are segmented using `pkuseg` (Luo et al., 2019). The source sentence prefixes ending with word boundaries are considered positive examples, while the rest of the source sentence prefixes are negative examples.

2.4.2 Combine with wait-k

We utilize the streaming segmentation model to detect word boundaries and only enable the wait-k policy at the word boundaries to determine word numbers that need to translate. Then the prefix-to-prefix translation is performed, which can avoid translating on source prefix containing incomplete words. Algorithm 1 gives the pseudo code of our proposed method. And Figure 1 shows how the streaming segmentation model works with the wait-k inference.

2.5 Evaluation

We evaluate our simultaneous translation model in two aspects. First is translation quality, we compute BLEU (Papineni et al., 2002) score with merged document translation results. Second, for latency, we utilize Average Lagging (AL) (Ma et al., 2019) to represent the text lagging of our model compared to

[7] https://github.com/dropreg/R-Drop

[8] https://huggingface.co/hfl/chinese-roberta-wwm-ext

stream-id	char-stream	word-stream
1	那	那
2	那首	那
3	那首先	那首先
4	那首先呢	那首先呢
5	那首先呢我	那首先呢我
6	那首先呢我先	那首先呢我先
7	那首先呢我先介	那首先呢我先
8	那首先呢我先介绍	那首先呢我先介绍
9	那首先呢我先介绍一	那首先呢我先介绍
10	那首先呢我先介绍一下	那首先呢我先介绍一下
11	那首先呢我先介绍一下我	那首先呢我先介绍一下我
12	那首先呢我先介绍一下我自	那首先呢我先介绍一下我
13	那首先呢我先介绍一下我自己	那首先呢我先介绍一下我自己
full sentence	nà | shǒu xiān | nē | wǒ | xiān | jiè shào | yí xià | wǒ zì jǐ 那 | 首先 | 呢 | 我 | 先 | 介绍 | 一下 | 我自己 then | first | - | I | - | introduce | - | myself	

Table 2: Case analysis of incomplete streaming in a Chinese sentence. Char-stream presents sentences by characters. Word-stream presents sentence by word. The prefixes in red color mean error in char-stream, which contains incomplete word-piece. The partial word piece may cause misunderstanding and incorrect translation.

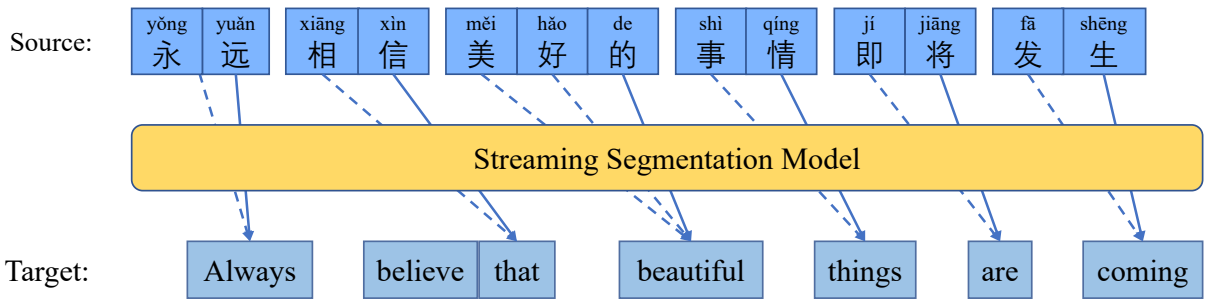

Figure 1: This example shows how the streaming segmentation model works with a wait-k model. The solid lines are the translation points of our proposed method, and the dashed lines are the additional possible translation points of the wait-k model.

Algorithm 1: Wait-k decoding with the streaming chunking method

Input: the translation model M_t, the chunking model M_c, the source sequence x, wait-k lagging K

Output: The translated sentence $\hat{y}$

Initialization: the read token sequence $\hat{x} = []$, the output sentence $\hat{y} = []$,the incomplete word read $x_p = ''$

while $|\hat{y}| \neq$ '</s>' **do**
  if $|\hat{x}| - |\hat{y}| \geq K$ **then**
    $token_{next} = M_t(\hat{x}, \hat{y})$
    $y = y + token_{next}$
  else
    $x_p = x_p + x.next\_char()$
    // x_p is a complete word
    if $M_c(\hat{x}, x_p)$ **then**
      $= + x_p$
      $x_p = ''$
    end
  end
end
return

ideal simultaneous interpretation, which is calculated in the following equation:

$$AL = \frac{1}{\tau}\sum_{j=1}^{\tau} g(j) - \frac{j-1}{\gamma}$$
$$\text{where} \quad \tau = \arg\min_{t}[\, g(j) = |X|\,] \qquad (1)$$
$$\gamma = |Y|/|X|$$

3 Experiments and Results

In this section, we describe our experiment settings and results on all the three tracks we participate in.

3.1 Zh-En text-to-text track

For the Zh-En text-to-text track, we introduce our experiments in detail, including model configurations, data, as well as results of a strong wait-k baseline and streaming chunking method.

3.1.1 Model Configurations

In our experiment, we train transformer-big models with the same parameters in Vaswani et al. (2017). The token-level batch size is about 100k on 8 GPUs for pre-training in all experiments. The learning rate is set as 5e-4 for pre-training and 5e-6 for fine-tuning, controlled by Adam optimizer (Kingma and Ba, 2015). We pre-train the model for 100000 steps and save the model every 2000 steps. We fine-tune the model for 10000 steps and save every 200 steps (batch size is about 30k).

3.1.2 Data

We filter the BSTC corpus and the CWMT corpus with methods described in Section 2.1 and apply language-model-based data selection to the CWMT corpus. For the first edition standard transformer model, we mix the BSTC corpus and the CWMT corpus for pre-training, using the BSTC corpus for fine-tuning (denoted as M1). And following is the detail of the M1 model.

For the pre-training stage, we show our results in each filtering step in Table 3. We directly mix the CWMT and the BSTC parallel data as the D0 corpus. The rules-filter discards noise data containing few linguistic words, which improves about 1.3 BLEU. In align-langid-filter, we drop sentence pairs with a align score less than -7 and sentences in the wrong languages. In PPL-selection, we use *ppl_score* computed by the language model to sort sentence pairs and drop sentence pairs with a *ppl_score* larger than 8000. With align-langid-filter and PPL-selection, 1.5M sentence pairs are dropped and nearly no BLEU descend is observed. We get the D1 corpus after all the filtering and selection. Further, we up-sample the BSTC corpus 5 times to enlarge the proportion of domain data. The R-Drop method is incorporated and we choose a larger dropout value (default dropout 0.1). Results in 4 show that the R-Drop ($\alpha = 5$) method significantly improves BLEU, and more increase is observed as we employ these methods together. For fine-tuning, we filter the BSTC corpus by hand-crafted rules and train with the consistent R-Drop method in the pre-training. Finally, we integrate the pre-training and the fine-tuning to train the M1 model, and the performance on the development set is shown in Table 7.

As the training corpus is limited, we utilize data augmentation methods. We perform data augmentation with the M1 model, containing forward-translation (FT) and backward-

Pre-training (data)	Data statistic	dev (SacreBleu)
Orig BSTC+CWMT (D0)	9.1M	16.82
+rules-filter	7.7M	18.09
+align-langid-filter	7.2M	18.04
+PPL-selection (D1)	6.2M	17.99

Table 3: Data filtering and selection in the pre-training stage. BLEU is computed by ScareBleu in sentence-level. Filtering and selection methods are applied incrementally.

Pre-training (method)	Data statistic	dev (SacreBleu)
BSTC+CWMT (D1)	6.2M	17.99
+up-sampling	6.34M	18.40
+dropout 0.25	6.2M	18.59
+R-Drop ($\alpha = 5$)	6.2M	19.72
+up-sampling + dropout 0.25 + R-Drop	6.34M	21.48

Table 4: Data statistic and BLEU on the development of our pre-training methods. BLEU is computed by ScareBleu in sentence-level.

translation (BT) on the pre-training and the fine-tuning corpus. For the pre-training corpus, we leverage the M1 model to perform FT and BT on the D1 corpus, mixed with D1 corpus as the augmented pre-training corpus. Results in Table 5 show FT has better performance than BT. For fine-tuning corpus, we employ the M1 model to translate BSTC corpus in forward and backward paths and add all 5 beam results to the fine-tuning corpus. What's more, to strengthen the robustness of the model, we add char-level augmentation into the fine-tuning corpus, which contains insertion, deletion, duplication, and homophone substitution. For homophone substitution, we use `python-pinyin`[9] to extract homophone dictionary and substitute homophone characters according to character frequency. Results on the fine-tuning corpus are shown in Table 6, which indicates that each augmentation method is useful.

Finally, we add FT augmentation in pre-training, add FT, and BT as well as character augmentation in fine-tuning. The model trained with augmented pre-training and fine-tuning is denoted as the M2 models. Significant improvement of the M2 model against the M1 model could be observed in Table 7.

3.1.3 Wait-k Baseline

To improve prefix-to-prefix translation quality, we use wait-k training described in Section 2.3. Using the same training data of the M2 model, we pre-train the model with multi-path wait-k and fine-tune with simple wait-k or multi-path wait-k. We report the results of our model on the BSTC development set. All trained model is listed in Table 8, and we show the AL-BLEU curve of several models. We achieve good performance according to Figure 2, in which our `M2_wait1-9_wait5` model exceeds the PaddlePaddle wait-5 model by at most 8 BLEU. The model trained with small k may achieve better performance in the low-latency regime, but not perform well in the high-latency regime. What's more, we ensemble the top-3 model in each inference k, which shows benefits across all latency regimes. Same as Guo et al. (2022), standard beam-search is utilized after the source stream is finished. Our models achieve almost consistent performance in high latency regime.

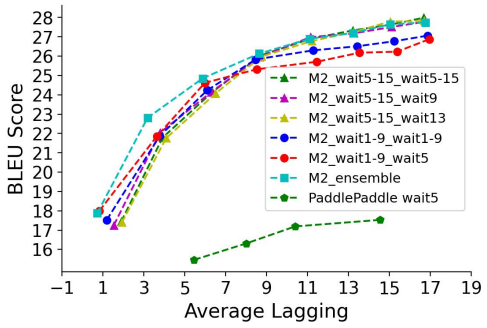

Figure 2: Results of M2 wait-k models. Models are list in Table 8. PaddlePaddle_wait5 is wait-k model provided by organizer.

[9] https://github.com/mozillazg/python-pinyin

Pre-training (Augmentation)	Data statistic (Pre-training)	dev (SacreBleu)
M1 (only pre-train)	6.34M	21.48
+FT pre-train	10.95M	22.32
+BT pre-train	11.03M	19.90

Table 5: Results of data augmentation in the pre-training stage. We use the M1 model to generate the FT and BT augment data and mixed with the D1 corpus for pre-training.

Fine-tuning (Augmentation)	Data statistic (Fine-tuning)	dev (SacreBleu)
M1 (fine-tuned on BSTC)	36K	22.41
+5FT	197K	22.92
+5BT	211K	22.59
+char-aug	185K	22.80
+5BT +5FT +char-aug	525K	23.05

Table 6: Results of data augmentation in the fine-tuning stage. The M1 model is leveraged to generate FT and BT augment data, and beam 5 results are saved. For the char-aug, we use character-level augmentations including insertion, deletion, duplication, and homophone substitution. The models in this table are all based on the same pre-trained model.

Model	dev (SacreBleu)	dev (Mteval-v13a)
M1	22.43	27.26
M2	23.62	28.96

Table 7: Results of data augmentation on standard transformer model. The M1 model is trained with pre-training and fine-tuning. The M2 model leverage data augmentation in both the pre-training and the fine-tuning stage.

Model name	Pre-train	Fine-tune
M2_wait5-15_wait5	$K \in [5, 15]$	$K = 5$
M2_wait5-15_wait7	$K \in [5, 15]$	$K = 7$
M2_wait5-15_wait9	$K \in [5, 15]$	$K = 9$
M2_wait5-15_wait11	$K \in [5, 15]$	$K = 11$
M2_wait5-15_wait13	$K \in [5, 15]$	$K = 13$
M2_wait5-15_wait15	$K \in [5, 15]$	$K = 15$
M2_wait5-15_wait5-15	$K \in [5, 15]$	$K \in [5, 15]$
M2_wait1-9_wait1	$K \in [1, 9]$	$K = 1$
M2_wait1-9_wait3	$K \in [1, 9]$	$K = 3$
M2_wait1-9_wait5	$K \in [1, 9]$	$K = 5$
M2_wait1-9_wait1-9	$K \in [1, 9]$	$K \in [1, 9]$

Table 8: Our wait-k models are pre-trained and fine-tuned on the same data of the M2 model in Section 3.1. We show the K value settings in pre-training and fine-tuning wait-k training for all M2 wait-k models. Take M2_wait5-15_wait5 for example, we use multi-path wait-k training with $K \in [5, 15]$ for pre-training and use simple wait-k with $K = 5$ for fine-tuning.

3.1.4 Streaming Chunking

In this section, we add streaming chunking methods. We first fine-tune our segmentation model based on `chinese-roberta-wwm-ext` on BSTC train set and get 92.0% accuracy and 93.7% F-score on the BSTC development set. Then we employ our segmentation to perform online source chunking to detect word boundaries. The results in Figure 3 show about 2 BLEU improvements in the low-latency regime with a little increase in AL.

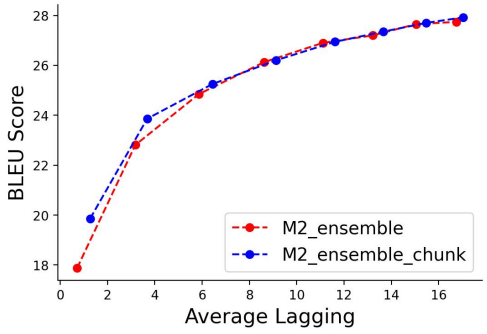

Figure 3: Results of streaming chunking method. M2_ensemble_chunk add streaming segmentation model compare to M2_ensemble.

3.2 En-Es text-to-text track

For En-Es text-to-text track, we use the same data filtering rules on the UN-parallel corpus. Because of lacking speech corpus, we didn't perform data selection and augmentation. Standard and wait1-11 transformers are trained and we report our results on the devel-

opment set in Figure 4.

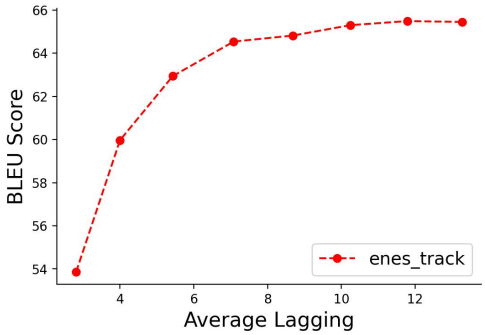

Figure 4: Results of En-Es text-to-text track. BLEU is computed in document level with Mteval-v13a.

3.3 Zh-En audio-to-text track

In Zh-En audio-to-text track, we train a simple transformer ASR model[10] with audio from BSTC and AIshell. The audio wav files are segmented by Silero-VAD(Team, 2021) and we achieve 0.38 WER on development and 0.28 WER on the test. And we perform simultaneous decoding on the ASR transcriptions with the same model and settings in the text-to-text track. Results show on development Figure 5 shows that the translation BLEU dropped by about 10 BLEU on audio input.

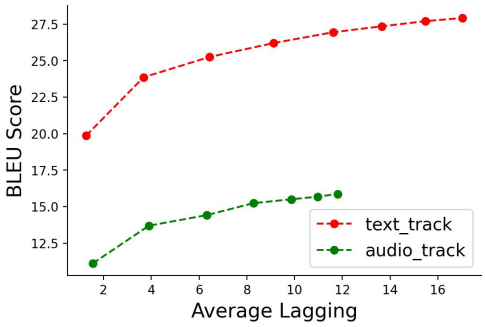

Figure 5: Results of Zh-En audio-to-text track. BLEU is computed in document level with Mteval-v13a.

4 Conclusion

We elaborate on the BIT-Xiaomi simultaneous translation system in this paper. We investigate data filtering and augmentation to enlarge high-quality corpus and utilize the R-Drop method to improve translation quality. We train our simultaneous translation models based on the wait-k strategy, and the streaming chunking method is employed to avoid segmentation errors in the source stream. The results on Zh-En text-to-text track indicate that the streaming chunking method can be integrated with the streaming decoding and improves translation quality. The slightly worse quality on the audio track suggests that the ASR error may affect translation quality much. In the future, we will explore better streaming ASR models and try more interesting simultaneous policies to get better latency and quality.

[10] https://github.com/facebookresearch/fairseq/blob/main/examples/speech_to_text/docs/mustc_example.md

Acknowledgements

This work is supported by the National Key RD Program of China (No. 2020AAA0106600).

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. In *Proc. of ACL*.

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

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2021. Pre-training with whole word masking for chinese bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

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

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proc. of EACL*.

Bao Guo, Mengge Liu, Wen Zhang, Hexuan Chen, Chang Mu, Xiang Li, Jianwei Cui, Bin Wang, and Yuhang Guo. 2022. The xiaomi text-to-text simultaneous speech translation system for IWSLT 2022. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*.

Xingyu Na Bengu Wu Hao Zheng Hui Bu, Jiayu Du. 2017. Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline. In *Oriental COCOSDA 2017*.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

xiaobo liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-drop: Regularized dropout for neural networks. In *Proc. of NeurIPS*.

Ruixuan Luo, Jingjing Xu, Yi Zhang, Xuancheng Ren, and Xu Sun. 2019. Pkuseg: A toolkit for multi-domain chinese word segmentation. *CoRR*.

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 *Proc. of ACL*.

Xutai Ma, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020. Monotonic multihead attention. In *Proc. of ICLR*.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Silero Team. 2021. Silero vad: pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Proc. of NeurIPS*.

Hao Xiong, Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Dutongchuan: Context-aware translation model for simultaneous interpreting. *arXiv preprint arXiv:1907.12984*.

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*.

Ruiqing Zhang, Xiyang Wang, Chuanqiang Zhang, Zhongjun He, Hua Wu, Zhi Li, Haifeng Wang, Ying Chen, and Qinfei Li. 2021. BSTC: A large-scale Chinese-English speech translation dataset. In *Proceedings of the Second Workshop on Automatic Simultaneous Translation*.

Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2020. Learning adaptive segmentation policy for simultaneous translation. In *Proc. of EMNLP*.