ICT's System for AutoSimTrans 2021: Robust Char-Level Simultaneous Translation

Shaolei Zhang [1,2], **Yang Feng** [1,2*]
[1]Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[2] University of Chinese Academy of Sciences, Beijing, China
{zhangshaolei20z, fengyang}@ ict.ac.cn

Abstract

Simultaneous translation (ST) outputs the translation simultaneously while reading the input sentence, which is an important component of simultaneous interpretation. In this paper, we describe our submitted ST system, which won the first place in the streaming transcription input track of the Chinese-English translation task of AutoSimTrans 2021. Aiming at the robustness of ST, we first propose char-level simultaneous translation and applied wait-k policy on it. Meanwhile, we apply two data processing methods and combine two training methods for domain adaptation. Our method enhance the ST model with stronger robustness and domain adaptability. Experiments on streaming transcription show that our method outperforms the baseline at all latency, especially at low latency, the proposed method improves about 6 BLEU. Besides, ablation studies we conduct verify the effectiveness of each module in the proposed method.

1 Introduction

Automatic simultaneous translation (ST) (Cho and Esipova, 2016; Gu et al., 2017; Ma et al., 2019), a task in machine translation (MT), aims to output the target translation while reading the source sentence. The standard machine translation is a full-sentence MT, which waits for the complete source input and then starts translation. The huge latency caused by full-sentence MT is unacceptable in many real-time scenarios. On the contrary, ST is widely used in real simultaneous speech translation scenarios, such as simultaneous interpretation, synchronized subtitles, and live broadcasting.

Previous methods (Ma et al., 2019; Arivazhagan et al., 2019) for ST are all evaluated on the existing full-sentence MT parallel corpus, ignoring the real speech translation scenario. In the real scene, the paradigm of simultaneous interpretation is Automatic Speech Recognition (ASR) → simultaneous translation (ST) → Text-to-Speech Synthesis (TTS), in which these three parts are all carried out simultaneously. As a downstream task of simultaneous ASR, the input of ST is always not exactly correct and in the spoken language domain. Thus, robustness and domain adaptability become two challenges for the ST system.

For robustness, since the input of the ST system is ASR result (streaming transcription), which is incremental and may be unsegmented or incorrectly segmented, the subword-level segmentation result (Ma et al., 2019) of the streaming transcription seriously affect the ST result. Existing methods (Li et al., 2020) often remove the last token after segmentation to prevent it from being incomplete, which leads to a considerable increase in latency. Table 1 shows an example of the tokenization result of the streaming transcription input with different methods. In steps 4-7 of standard wait-2, the input prefix is different from its previous step, while the previous output prefix is not allowed to be modified in ST, which leads to serious translation errors. Although removing the last token improves the robustness, there is no new input in many consecutive steps, which greatly increases the latency.

For domain adaptability, the existing spoken language domain corpus is lacking, while the general domain corpus for MT and the spoken language domain corpus for ST are quite different in terms of word order, punctuation and modal particles, so ST needs to efficiently complete the domain adaption.

In our system, we propose a *Char-Level Wait-k Policy* for simultaneous translation, which is more robust with streaming transcription input. Besides, we apply data augmentation and combine two training methods to train the model to complete domain adaptation. Specifically, the source of the char-level wait-k policy is a character sequence segmented according to characters, and the target still maintains subword-level segmentation and BPE operations (Sennrich et al., 2016). When decoding,

* Corresponding author: Yang Feng.

Proceedings of the 2nd Workshop on Automatic Simultaneous Translation, pages 1–11
June 10, 2021.

Streaming Transcription	Tokenization of Streaming Transcription Input		
	Standard Wait-2	Standard Wait-2 Remove Last Token	Char-Level Wait-2 (Ours)
他是研究生物的			
他是研究生物的	他/是/	他/	他/是/
他是研究生物的	他/是/研/	他/是/ ▷ Delay	他/是/研/
他是研究生物的	他/是/研究/ ✘	他/是/ ▷ Delay	他/是/研/究/
他是研究生物的	他/是/研究生/ ✘	他/是/ ▷ Delay	他/是/研/究/生/
他是研究生物的	他/是/研究/生物/ ✘	他/是/研究/	他/是/研/究/生/物/
他是研究生物的	他/是/研究/生物/的/	他/是/研究/生物/的/	他/是/研/究/生/物/的/

Table 1: An example of the tokenization result of standard wait-k, standard wait-k+remove last token and char-level wait-k, when dealing with streaming transcription input (take $k = 2$ as an example). Red mark: the source prefix changes during streaming input. Green mark: no input in consecutive steps since the last token is removed.

the char-level wait-k policy first waits for k source characters, then alternately reads a character, and outputs a target subword. Table 1 shows the tokenization results of the char-level wait-k policy, which not only guarantees the stability of the input prefix but also avoids unnecessary latency. To adapt to the spoken language domain, we first pretrain an ST model on the general domain corpus and perform fine-tuning on the spoken language domain corpus. To improve the effect and efficiency of domain adaptation, we carry out data augmentation on both the general domain corpus and spoken language domain corpus and combine two different training methods for training.

In the streaming transcription track for the Chinese → English translation task of AutoSimTrans 2021, we evaluate the proposed method on the real speech corpus (Zhang et al., 2021). Our method exceeds the baseline model at all latency and performs more prominently at lower latency.

Our contributions can be summarized as follows:

- To our best knowledge, we are the first to propose char-level simultaneous translation, which is more robust when dealing with real streaming input.
- We apply data augmentation and incorporate two training methods, which effectively improve the domain adaptation and overcome the shortage of spoken language corpus.

2 Task Description

We participated in the streaming transcription input track of the Chinese-English translation task of AutoSimTrans 2021 [1]. An example of the task is shown in Table 2. Streaming transcription is manually transcribed without word segmentation. Between each step, the source input adds one more character. The task applies AL and BLEU respectively to evaluate the latency and translation quality of the submitted system.

Streaming Transcript	Translation
大	
大家	
大家好	Hello everyone!
欢	
欢迎	
欢迎大	
欢迎大家	
欢迎大家来	Welcome
欢迎大家来到	everyone
欢迎大家来到这	come
欢迎大家来到这里	here.

Table 2: An example of streaming transcription output track of the Chinese-English translation task.

3 Background

Our system is based on a variant of wait-k policy (Ma et al., 2019), so we first briefly introduce wait-k policy and its training method.

Wait-k policy refers to waiting for k source tokens first, and then reading and writing alternately, i.e., the output always delays k tokens after the input. As shown by 'standard wait-k policy' in Figure 1, if $k = 2$, the first target token was output after reading 2 source tokens, and then output a target token as soon as a source token is read.

Define $g(t)$ as a monotonic non-decreasing function of t, which represents the number of source

[1] https://autosimtrans.github.io/shared

tokens read in when outputting the target token y_t. For the wait-k policy, $g(t)$ is calculated as:

$$g(t) = \min\{k + t - 1, |\boldsymbol{x}|\}, t = 1, 2, \cdots \quad (1)$$

where $\boldsymbol{x}$ is the input subword sequence.

Wait-k policy is trained with "prefix-to-prefix" framework. In "prefix-to-prefix" framework, when generating the t^{th} target word, the source tokens participating in encoder is limited to less than $g(t)$.

4 Methods

To improve the robustness and domain adaptability of ST, we enhance our system from read / write policy, data processing and training methods respectively.

4.1 Char-Level Wait-k Policy

To enhance the robustness of dealing with streaming transcription, we first proposed char-level simultaneous translation and applied the wait-k policy on it.

4.1.1 Char-Level Simultaneous Translation

Character-level neural machine translation (Ling et al., 2015; Lee et al., 2017; Cherry et al., 2018; Gao et al., 2020) tokenizes the source sentence and target sentence according to characters, thereby gaining advantages over subword-level neural machine translation in some specific aspects, such as avoiding out-of-vocabulary problems (Passban et al., 2018) and errors caused by subword-level segmentation (Tang et al., 2020). In terms of translation quality, the character-level MT is still difficult to compare with the subword-level MT. An important reason is that only one wrong generated character will directly cause the entire target word wrong (Sennrich, 2017).

To improve the robustness of the ST system when dealing with unsegmented incremental input, while avoiding the performance degradation caused by character-level MT, we propose *char-level simultaneous translation*, which is more suitable for streaming input. The framework of char-level ST is shown in the lower part of Figure 1.

Different from subword-level ST, given the parallel sentence pair $< X, Y >$, the source of the ST model in the proposed char-level ST is the character sequence $\boldsymbol{c} = (c_1, \cdots, c_n)$ after char-level tokenization, and the target is the subword sequence $\boldsymbol{y} = (y_1, \cdots, y_m)$ after word segmentation and BPE (Sennrich et al., 2016), where n and m are the source and target sequence lengths respectively.

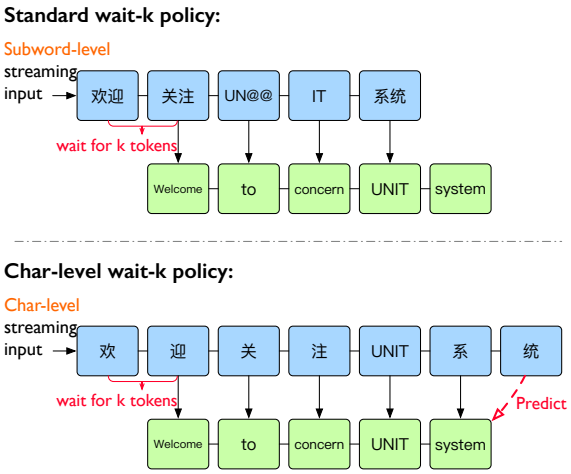

Figure 1: Standard wait-k policy vs. our char-level wait-k policy (take $k = 2$ as an example).

The word segmentation and BPE operation at the target end are the same as subword-level MT (Vaswani et al., 2017), and char-level tokenization is similar to character-level MT (Yang et al., 2016; Nikolov et al., 2018; Saunders et al., 2020) but not completely consistent. The char-level tokenization we proposed divides each source language character into a token, and other characters (such as numbers, other language characters) are still divided into a token according to complete words. An example of char-level tokenization is shown in Table 3. In the result of char-level tokenization, each Chinese character is divided into a token, and the number (12) and English (UNIT) are entirely taken as a token, respectively. Char-level tokenization is more suitable for streaming transcription, which ensures that the newly input content at each step in streaming transcription is a complete token, and the input prefix does not change in any way. The robustness of char-level ST is greatly improved with the complete token and stable prefix.

Why char-level simultaneous translation? Motivating our use of char-level ST we consider three desiderata. 1) With the incremental source input, char-level ST is more robust since it avoids unstable prefixes caused by word segmentation, as shown in Table 1. 2) Char-level ST can obtain a more fine-grained latency, because if one character is enough to express the meaning of a entire word, the ST system does not have to wait for the complete word before translating. 3) Char-level ST only performs char-level tokenization on the source, while the target still retains subword-level tokenization, so its translation performance will not be affected too much, as shown in Table 7.

Input Sentence		欢迎来到UNIT系统的第12期高级课程。
Output Sentence		welcome to the 12th advanced course on UNIT system .
S.	subword-level MT	欢迎 / 来到 / UN@@ / IT / 系统 / 的 / 第 / 12@@ / 期 / 高级 / 课程 / 。
	character-level MT	欢 / 迎 / 来 / 到 / U / N / I / T / 系 / 统 / 的 / 第 / 1 / 2 / 期 / 高 / 级 / 课 / 程 / 。
	char-level tokenization	欢 / 迎 / 来 / 到 / UNIT / 系 / 统 / 的 / 第 / 12 / 期 / 高 / 级 / 课 / 程 / 。
T.	subword-level MT	welcome / to / the / 12@@ / th / advanced / course / on / UNIT / system / .

Table 3: An example of tokenization method applied by the char-level wait-k policy. For the source, we use char-level tokenization, which separates each source language character into separate segments, and divides the others by words. For the target, we apply the same operation as the conventional subword-level MT. The sentences marked in red are the source and target of our proposed ST model.

4.1.2 Read / Write Policy

For the read / write policy, we apply the wait-k policy on the proposed char-level ST. The difference between char-level wait-k policy and standard wait-k policy is that each token in standard wait-k policy is a subword, while each token in char-level wait-k policy is a character (other languages or Numbers are still words), as shown in Figure 1.

We rewrite $g(t)$ in Eq.(1) into $g_k(t)$ for char-level wait-k policy, which represents the number of source tokens (Character) read in when outputting the target token y_t, calculated as:

$$g_k(t) = \min\{k + t - 1, |\boldsymbol{c}|\}, t = 1, 2, \cdots \quad (2)$$

where $\boldsymbol{c}$ is the input character sequence.

Another significant advantage of the standard wait-k policy is that it can obtain some implicit prediction ability in training, and char-level wait-k policy further strengthens the prediction ability and improves the stability of prediction. The reason is that the granularity of the char-level is smaller so that the prediction of char-level is simpler and more accurate than that of subword-level. As shown in Figure 1, it is much simpler and more accurate to predict "系统" given "系", since there are few possible characters that can be followed by "系".

4.2 Domain Adaptation

To improve the quality of domain adaptation, we apply some modifications to all training corpus, including general domain and spoken language domain, to make them more closer to streaming transcription. Besides, we also augment the spoken language corpus to make up for the lack of data.

4.2.1 Depunctuation

For training corpus, including general domain and spoken language domain, the most serious difference from streaming transcription is that each sentence in streaming transcription usually lacks ending punctuation, as shown in Table 2. Since the punctuation in the training corpus is complete, and the ending punctuation is often followed by $< eos >$, the model trained with them tends to wait for the source ending punctuation and then generate the corresponding target ending punctuation and $< eos >$ to stop translating. As a result, given the unpunctuated input in streaming transcription, it is difficult for the model to generate target punctuation and $< eos >$ to stop the translation.

Original	各位开发者、各位朋友们，大家下午好！
Depunctuation	各位开发者、各位朋友们，大家下午好

Table 4: An example of depunctuation operation, where the ending punctuation of the source sentence is deleted.

To this end, to strengthen the model's ability to translate punctuation from unpunctuated sentences, we delete the ending punctuation of the source sentence, and the target sentence remains unchanged, as shown in Table 4. Note that our depunctuation operation is limited to the ending punctuation at the end of the source sentence ('。','！','？').

4.2.2 Data Augmentation

For the spoken language domain corpus, since the data size is too small, we perform data augmentation on the source sentence. For each source sentence, we perform 5 operations: add a comma, add a tone character, copy an adjacent character, replace a character with its homophone, or delete a character. Meanwhile, the target sentence remains unchanged. The proposed method improves the robustness of the model while augmenting the data. An example of data augmentation is shown in Table 5.

Original	1957年我到北京上大学
Add Comma	1957年，我到北京上大学
Add Tone character	1957年我啊到北京上大学
Copy Character	1957年我到北北京上大学
Replace Homophone	1957年我到北经上大学
Delete Character	1957年我到北京上大学

Table 5: An example of data augmentation.

4.3 Training Methods

Our method is based on Transformer (Vaswani et al., 2017), and the training is divided into two stages. First, we pre-train an ST model on the general domain MT corpus, and then fine-tune the ST model on the spoken language domain corpus. For pre-training, we apply *multi-path* (Elbayad et al., 2020) and *future-guided* (Zhang et al., 2020b), to enhance the predict ability and avoid the huge consumption caused by training different models for each k. For fine-tuning, we apply the original prefix-to-prefix framework (Ma et al., 2019).

4.3.1 Pre-training

To improve the predictive ability of the ST model, we apply the future-guided training proposed by (Zhang et al., 2020b). Besides the incremental Transformer for simultaneous translation with char-level wait-k policy, we introduce a full-sentence Transformer, used as the teacher of the incremental Transformer for ST through knowledge distillation. The full-sentence Transformer is trained with cross-entropy loss:

$$\mathcal{L}\left(\theta_{full}\right) = -\sum_{(\boldsymbol{c},\boldsymbol{y})\in D_g}\sum_{t=1}^{|\boldsymbol{y}|}\log p_{\theta_{full}}\left(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{c}\right) \tag{3}$$

where θ_{full} is the parameter of full-sentence Transformer, D_g is the general domain corpus.

For the incremental Transformer for ST, since it applies char-level wait-k policy, the source tokens participating in translating are limited to less than $g_k\left(t\right)$ when decoding the t^{th} target token. For each k, the decoding probability is calculated as:

$$p\left(\boldsymbol{y} \mid \boldsymbol{c}, k\right) = \prod_{t=1}^{|\boldsymbol{y}|} p_{\theta_{incr}}\left(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{c}_{\leq g_k(t)}\right) \tag{4}$$

where $\boldsymbol{c}$ and $\boldsymbol{y}$ are the input character sequence and the output subword sequence, respectively. $\boldsymbol{c}_{\leq g_k(t)}$ represents the first $g_k\left(t\right)$ tokens of $\boldsymbol{c}$. θ_{incr} is the parameter of incremental Transformer.

Following Elbayad et al. (2020), to cover all possible k during training, we apply multi-path training. k is not fixed during training, but randomly and uniformly sampled from K, where $K = [1, \cdots, |\boldsymbol{c}|]$ is the set of all possible values of k. Incremental Transformer is also trained with cross-entropy loss:

$$\mathcal{L}\left(\theta_{incr}\right) = -\sum_{(\boldsymbol{c},\boldsymbol{y})\in D_g}\sum_{t=1,k\sim\mathcal{U}(\mathrm{K})}^{|\boldsymbol{y}|}\log p_{\theta_{incr}}\left(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{c}_{\leq g_k(t)}\right) \tag{5}$$

For the knowledge distillation between full-sentence Transformer and incremental Transformer, we apply L_2 regularization term between their encoder hidden states, calculated as:

$$\mathcal{L}\left(\boldsymbol{z}^{incr}, \boldsymbol{z}^{full}\right) = \frac{1}{|\boldsymbol{c}|}\sum_{i=1}^{|\boldsymbol{c}|}\left\|z_i^{incr} - z_i^{full}\right\|^2 \tag{6}$$

where $\boldsymbol{z}^{incr}$ and $\boldsymbol{z}^{full}$ represent the hidden states of incremental Transformer and full-sentence Transformer, respectively.

Finally, the total loss $\mathcal{L}$ is calculated as:

$$\mathcal{L} = \mathcal{L}\left(\theta_{incr}\right) + \mathcal{L}\left(\theta_{full}\right) + \lambda\mathcal{L}\left(\boldsymbol{z}^{incr}, \boldsymbol{z}^{full}\right) \tag{7}$$

where λ is the hyper-parameter, and we set $\lambda = 0.1$ in our system.

4.3.2 Fine-tuning

After pre-training an ST model, we use spoken language domain corpus for fine-tuning. The spoken language domain corpus is a small dataset, and meanwhile most of the word order between the target and the source is the same, so we do not continue to use multi-path and future-guided training methods. We fix k and use the original prefix-to-prefix framework for training, and train different models for each k. Given k, the incremental Transformer is trained with cross-entropy loss:

$$\mathcal{L}\left(\theta_{incr}, k\right) = -\sum_{(\boldsymbol{c},\boldsymbol{y})\in D_s}\sum_{t=1,}^{|\boldsymbol{y}|}\log p_{\theta_{incr}}\left(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{c}_{\leq g_k(t)}\right) \tag{8}$$

where D_s is the spoken language domain corpus. Finally, for each k, we fine-tuned a ST model.

Datasets	Domain	#Sentence Pairs
CWMT19	General	9,023,708
Transcription	Spoken	37,901
Dev. Set	Spoken	956

Table 6: Statistics of Chinese → English datasets.

		AL	BLEU	
			Greedy	Beam4
subword level	Pre-train	24.93	20.24	20.35
	+ FT	24.93	24.79	25.39
char level	Pre-train	24.93	20.14	20.28
	+ FT	24.93	24.60	25.13

Table 7: Results of offline model. '+FT': +fine-tuning.

5 Experiments

5.1 Dataset

The dataset for Chinese → English task provided by the organizer contains three parts, shown in Table 6. **CWMT19** [2] is the general domain corpus that consists of 9,023,708 sentence pairs. **Transcription** consists of 37,901 sentence pairs and **Dev. Set** consists of 956 sentence pairs [3], which are both spoken language domain corpus collected from real speeches (Zhang et al., 2021).

We use **CWMT19** to pre-train the ST model, then use **Transcription** for fine-tuning, and finally evaluate the latency and translation quality of our system on **Dev. Set**. Note that we use the streaming transcription provided by the organizer for testing. Streaming transcription consists of 23,836 lines, which are composed by breaking each sentence in **Dev. Set** into lines whose length is incremented by one word until the end of the sentence.

We eliminate the corpus with a huge ratio in length between source and target from **CWMT19**, and finally got 8,646,245 pairs of clean corpus. We augment the **Transcription** data according to the method in Sec.4.2.2, and get 227,406 sentence pairs. Meanwhile, for both **CWMT19** and **Transcription**, we remove the ending punctuation according to the method in Sec.4.2.1.

Given the processed corpus after cleaning and augmentation, we first perform char-level tokenization (Sec.4.1) on the Chinese sentences, and tokenize and lowercase English sentences with the Moses[4]. We apply BPE (Sennrich et al., 2016) with 16K merge operations on English.

5.2 System Setting

We set the standard wait-k policy as the baseline and compare our method with it. We conducted experiments on the following systems:

Offline: offline model, full-sentence MT based on Transformer. We report the results of the subword-level / char-level offline model with greedy / beam search respectively in Table 7.

Standard Wait-k: standard subword-level wait-k policy proposed by Ma et al. (2019), used as our baseline. For comparison, we apply the same training method as our method (Sec.4.3) to train it.

Standard Wait-k + rm Last Token: standard subword-level wait-k policy. In the inference time, the last token after the word segmentation is remove to prevent it from being incomplete.

Char-Level Wait-k: our proposed method, refer to Sec.4 for details.

The implementation of all systems is based on Transformer-Big, and adapted from Fairseq Library (Ott et al., 2019). The parameters are the same as the original Transformer (Vaswani et al., 2017). All systems are trained on 4 RTX-3090 GPUs.

5.3 Evaluation Metric

For evaluation metric, we use BLEU [5] (Papineni et al., 2002) and AL[6] (Ma et al., 2019) to measure translation quality and latency, respectively.

Latency metric AL of char-level wait-k policy is calculated with $g_k(t)$ in Eq.(2):

$$\mathrm{AL} = \frac{1}{\tau}\sum_{t=1}^{\tau} g_k(t) - \frac{t-1}{\frac{|\boldsymbol{y}|}{|\boldsymbol{c}|}} \tag{9}$$

$$\text{where } \tau = \underset{t}{\mathrm{argmax}}\left(g_k(t) = |\boldsymbol{c}|\right) \tag{10}$$

where $\boldsymbol{c}$ and $\boldsymbol{y}$ are the input character sequence and the output subword sequence, respectively. Note that since the streaming transcription provided by the organizer adds a source character at each step, for all systems, we use character-level AL to evaluate the latency.

[2] casia2015, casict2011, casict2015, datum2015, datum2017 and neu2017. http://mteval.cipsc.org.cn:81/agreement/AutoSimTrans

[3] https://dataset-bj.cdn.bcebos.com/qianyan%2FAST_Challenge.zip

[4] http://www.statmt.org/moses/

[5] The script for calculating BLEU is provided by the organizer from https://dataset-bj.cdn.bcebos.com/qianyan%2FAST_Challenge.zip.

[6] The calculation of AL is as https://github.com/autosimtrans/SimulTransBaseline/blob/master/latency.py.

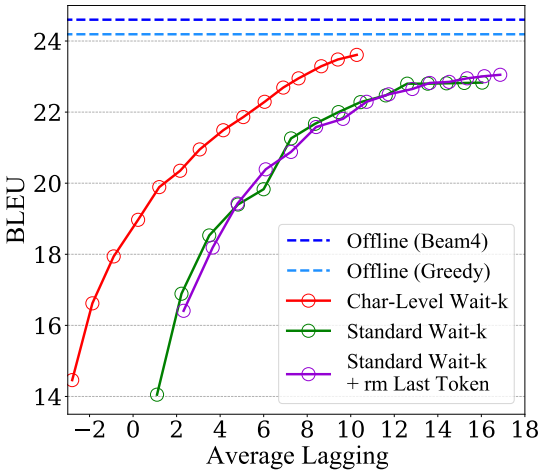

Figure 2: Translation quality (BLEU) against latency (AL) on Chinese → English simultaneous translation, showing the results of proposed char-level wait-k, standard wait-k, standard wait-k+rm last token and offline model with greedy/beam search.

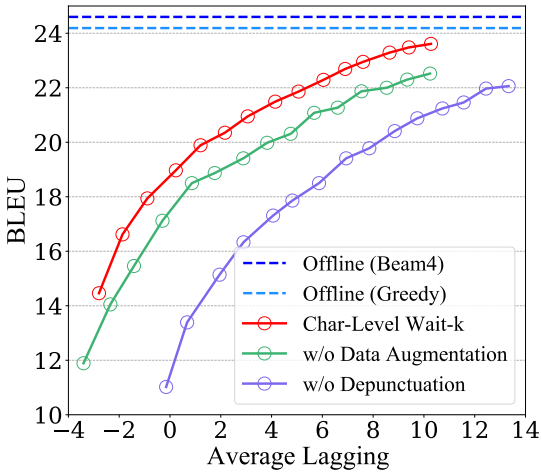

Figure 3: Result of our method without depunctuation or data augmentation.

5.4 Main Result

We compared the performance of our proposed char-level wait-k policy and subword-level wait-k policy, and set $k = 1, 2, \ldots, 15$ to draw the curve of translation quality against latency, as shown in Figure 2. Note that the same value of k for char-level wait-k policy and subword-level wait-k policy does not mean that the latency of the two are similar, because lagging k tokens in char-level wait-k means strictly waiting for k characters, while for subword-level wait-k, it waits for k subwords, which contain more characters.

'Char-Level Wait-k' outperforms 'Standard Wait-k' and 'Standard Wait-k+rm Last Token' at all latency, and improves about 6 BLEU at low latency (AL=1.10). Besides, char-level wait-k performs more stable and robust than standard wait-k when dealing with streaming transcription input, because char-level wait-k has a stable prefix while the prefix of standard wait-k may change between adjacent steps due to the different word segmentation results. 'Standard Wait-k+rm Last Token' solves the issue that the last token may be incomplete, so that the translation quality is higher than Standard Wait-k under the same k, which improves about 0.56 BLEU (average on all k). However, 'Standard Wait-k+rm Last Token' increases the latency. Compared with 'Standard Wait-k', it waits for one more token on average under the same k. Therefore, from the overall curve, the improvement of 'Standard Wait-k+rm Last Token' is limited.

Char-level wait-k is particularly outstanding at low latency, and it achieves good translation quality even when the AL is less than 0. It is worth mentioning that the reason why the AL is less than 0 is that the generated translation is shorter and $\frac{|\boldsymbol{y}|}{|\boldsymbol{c}|}$ in Eq.(9) is greater than 1.

5.5 Effect of Data Processing

To analyze the effect of data processing, including 'Depunctuation' and 'Data Augmentation', we show the results without them in Figure 3.

We notice that data augmentation improves the translation quality of the model by 1.61 BLEU (average on all k), and the model becomes more stable and robust. 'Depunctation' is even more important. If we keep the ending punctuation in the training corpus, the translation quality of the model drops by 2.27 BLEU, and the latency increase by 2.83 (average on all k). This is because streaming transcription input has no ending punctuation, which makes the model hard to generate target ending punctuation and tend to translate longer translations since it is difficult to generate $< eos >$ without target ending punctuation.

5.6 Ablation Study on Training Methods

To enhance the performance and robustness under low latency, we combine future-guided and multi-path training methods in pre-training. To verify the effectiveness of the two training methods, we conducted an ablation study on them, and show the results of removing one of them in Figure 4.

When removing one of them, the translation quality decreases, especially at low latency. When the 'Future-guided' is removed, the translation quality decreases by 1.49 BLEU (average on all

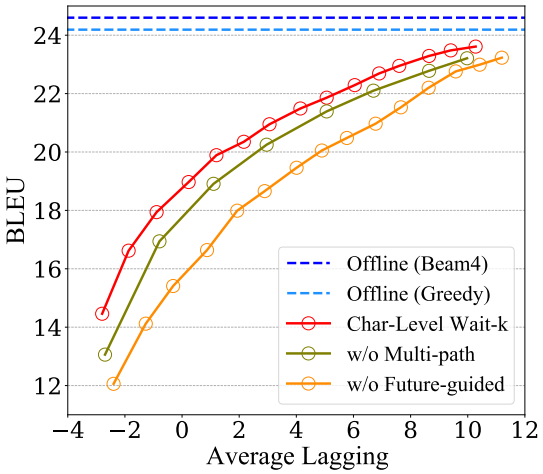

Figure 4: Ablation study on two training methods.

k), and when the 'Multi-path' is removed, the translation quality decreases by 0.76 BLEU (average on all k). This shows that two training methods can both effectively improve the translation quality under low latency, especially 'Future-guided'.

6 Related Work

Previous ST methods are mainly divided into precise read / write policy and stronger predictive ability.

For read / write policy, early policies used segmented translation, and applied full sentence translation to each segment (Bangalore et al., 2012; Cho and Esipova, 2016; Siahbani et al., 2018). Gu et al. (2017) trained an agent through reinforcement learning to decide read / write. Dalvi et al. (2018) proposed STATIC-RW, which first performing S's READs, then alternately performing RW's WRITEs and READs. Ma et al. (2019) proposed wait-k policy, wherein first reads k tokens and then begin synchronizing write and read. Wait-k policy has achieved remarkable performance because it is easy to train and stable, and is widely used in simultaneous translation. Zheng et al. (2019a) generated the gold read / write sequence of input sentence by rules, and then trained an agent with the input sentences and gold read / write sequence. Zheng et al. (2019b) introduces a "delay" token $\{\varepsilon\}$ into the target vocabulary to read one more token. Arivazhagan et al. (2019) proposed MILK, which uses a Bernoulli distribution variable to determine whether to output. Ma et al. (2020) proposed MMA, the implementation of MILK based on Transformer. Zheng et al. (2020) proposed a decoding policy that uses multiple fixed models to accomplish adaptive decoding. Zhang et al. (2020a) propose a novel adaptive segmentation policy for ST.

For predicting future, Matsubara et al. (2000) applied pattern recognition to predict verbs in advance. Grissom II et al. (2014) used a Markov chain to predict the next word and final verb. (Oda et al., 2015) predict unseen syntactic constituents to help generate complete parse trees and perform syntax-based simultaneous translation. Alinejad et al. (2018) added a Predict operation to the agent based on Gu et al. (2017), predicting the next word as an additional input. Elbayad et al. (2020) enhances the wait-k policy by sampling different k to train. Zhang et al. (2020b) proposed future-guided training, which introduces a full-sentence Transformer as the teacher of the ST model and uses future information to guide training through knowledge distillation.

Although the previous methods performed well, they were all evaluated on the traditional MT corpus instead of the real streaming spoken language corpus. Therefore, the previous methods all ignore the robustness and domain adaptation of the ST model in the face of real streaming input. Our method bridgs the gap between the MT corpus and the streaming spoken language domain input, and is more robust and adaptable to the spoken language domain.

7 Conclusion and Future Work

Our submitted system won the first place in AutoSimTrans 2021, which is described in this paper. For streaming transcription input from the real scenarios, our proposed char-level wait-k policy is more robust than standard subword-level wait-k. Besides, we also propose two data processing operations to improve the spoken language domain adaptability. For training, we combine two existing training methods that have been proven effective. The experiment on the data provided by the organizer proves the superiority of our method, especially at low latency.

In this competition, we implemented the char-level wait-k policy on the Chinese source. For some language pairs with a large length ratio between the source (char) and the target (bpe), we can read multiple characters at each step to prevent the issue caused by the excessively long char-level source. We put the char-level simultaneous translation on other languages (such as German and English) for both fixed and adaptive policy into our future work.

Acknowledgements

We thank all the anonymous reviewers for their insightful and valuable comments.

References

Ashkan Alinejad, Maryam Siahbani, and Anoop Sarkar. 2018. Prediction improves simultaneous neural machine translation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3022–3027, Brussels, Belgium. Association for Computational Linguistics.

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. 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 Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 437–445, Montréal, Canada. Association for Computational Linguistics.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting Character-Based Neural Machine Translation with Capacity and Compression. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4295–4305, Brussels, Belgium. Association for Computational Linguistics.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation?

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.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2020. Efficient wait-k models for simultaneous machine translation.

Yingqiang Gao, Nikola I. Nikolov, Yuhuang Hu, and Richard H.R. Hahnloser. 2020. Character-Level Translation with Self-attention. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1591–1604, Online. Association for Computational Linguistics.

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, Doha, Qatar. Association for Computational Linguistics.

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: Volume 1, Long Papers, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully Character-Level Neural Machine Translation without Explicit Segmentation. Transactions of the Association for Computational Linguistics, 5:365–378.

Minqin Li, Haodong Cheng, Yuanjie Wang, Sijia Zhang, Liting Wu, and Yuhang Guo. 2020. BIT's system for the AutoSimTrans 2020. In Proceedings of the First Workshop on Automatic Simultaneous Translation, pages 37–44, Seattle, Washington. Association for Computational Linguistics.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015. Character-based Neural Machine Translation. arXiv:1511.04586 [cs]. ArXiv: 1511.04586.

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, Florence, Italy. Association for Computational Linguistics.

Xutai Ma, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020. Monotonic multihead attention. In International Conference on Learning Representations.

Keiichi Matsubara, Shigeki Iwashima, Nobuo Kawaguchi, Katsuhiko Toyama, and Yasuyoshi Inagaki. 2000. Simultaneous japenese-english interpretation based on early predictoin of english verb. In Proceedings of the 4th Symposium on Natural Languauge Processing(SNLP-2000), pages 268–273.

Nikola I. Nikolov, Yuhuang Hu, Mi Xue Tan, and Richard H.R. Hahnloser. 2018. Character-level Chinese-English Translation through ASCII Encoding. In Proceedings of the Third Conference on Machine Translation: Research Papers, pages 10–16, Brussels, Belgium. Association for Computational Linguistics.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Syntax-based simultaneous translation through prediction of unseen syntactic constituents. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 198–207, Beijing, China. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Peyman Passban, Qun Liu, and Andy Way. 2018. Improving Character-Based Decoding Using Target-Side Morphological Information for Neural Machine Translation. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 58–68, New Orleans, Louisiana. Association for Computational Linguistics.

Danielle Saunders, Weston Feely, and Bill Byrne. 2020. Inference-only sub-character decomposition improves translation of unseen logographic characters. In Proceedings of the 7th Workshop on Asian Translation, pages 170–177, Suzhou, China. Association for Computational Linguistics.

Rico Sennrich. 2017. How Grammatical is Character-level Neural Machine Translation? Assessing MT Quality with Contrastive Translation Pairs. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

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, Berlin, Germany. Association for Computational Linguistics.

Maryam Siahbani, Hassan Shavarani, Ashkan Alinejad, and Anoop Sarkar. 2018. Simultaneous translation using optimized segmentation. In Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers), pages 154–167, Boston, MA. Association for Machine Translation in the Americas.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2020. Understanding Pure Character-Based Neural Machine Translation: The Case of Translating Finnish into English. In Proceedings of the 28th International Conference on Computational Linguistics, pages 4251–4262, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2016. A Character-Aware Encoder for Neural Machine Translation. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3063–3070, Osaka, Japan. The COLING 2016 Organizing Committee.

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. arXiv preprint arXiv:2104.03575.

Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2020a. Learning adaptive segmentation policy for simultaneous translation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2280–2289, Online. Association for Computational Linguistics.

Shaolei Zhang, Yang Feng, and Liangyou Li. 2020b. Future-guided incremental transformer for simultaneous translation.

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, pages 2847–2853, Online. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simpler and faster learning of adaptive policies for simultaneous translation. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 1349–1354, Hong Kong, China. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019b. Simultaneous translation with flexible policy via restricted imitation learning. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5816–5822, Florence, Italy. Association for Computational Linguistics.