Learning Adaptive Segmentation Policy for Simultaneous Translation

Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He[*], Hua Wu, Haifeng Wang
Baidu Inc. No. 10, Shangdi 10th Street, Beijing, 100085, China
{zhangruiqing01, zhangchuanqiang, hezhongjun, wu_hua, wanghaifeng}@baidu.com

Abstract

Balancing accuracy and latency is a great challenge for simultaneous translation. To achieve high accuracy, the model usually needs to wait for more streaming text before translation, which results in increased latency. However, keeping low latency would probably hurt accuracy. Therefore, it is essential to segment the ASR output into appropriate units for translation. Inspired by human interpreters, we propose a novel adaptive segmentation policy for simultaneous translation. The policy learns to segment the source text by considering possible translations produced by the translation model, maintaining consistency between the segmentation and translation. Experimental results on Chinese-English and German-English translation show that our method achieves a better accuracy-latency trade-off over recently proposed state-of-the-art methods.

1 Introduction

In recent years, simultaneous translation has attracted increasing interest both in research and industry community. It aims at a real-time translation that demands high translation quality and an as-short-as-possible delay between speech and translation output.

A typical simultaneous translation system consists of an auto-speech-recognition (ASR) system that transcribes the source speech into source streaming text, and a machine translation (MT) system that performs the translation from the source into the target text. However, there is a gap between the output of ASR and the input of MT. The MT system takes sentences as input, while the streaming ASR output has no segmentation boundaries. Therefore, exploring a *policy* to split ASR output into appropriate segments becomes a vital issue for simultaneous translation. If translation starts before adequate source content is delivered, the translation quality degrades. However, waiting for too much source text increases latency.

The *policies* of recent work generally falls into two classes:

- **Fixed Policies** are hard policies that follow a pre-defined schedule independent of the context. They segment the source text based on a fixed length (Ma et al., 2019; Dalvi et al., 2018). For example, the wait-k method (Ma et al., 2019) first reads k source words, and then generates one target word immediately after each subsequent word is received. Policies of this type are simple and easy to implement. However, they do not consider contextual information and usually result in a drop in translation accuracy.

- **Adaptive Policies** learn to do segmentation according to dynamic contextual information. They either use a specific model to chunk the streaming source text (Sridhar et al., 2013; Oda et al., 2014; Cho and Esipova, 2016; Gu et al., 2017; Zheng et al., 2019a, 2020) or jointly learn segmentation and translation in an end-to-end framework (Arivazhagan et al., 2019; Zheng et al., 2019b; Ma et al., 2020). The adaptive methods are more flexible than the fixed ones and achieve state-of-the-art.

In this paper, we propose a novel adaptive segmentation policy for simultaneous translation. Our method is motivated by two widely used strategies in simultaneous interpretation:

- Meaningful Unit (MU) Chunking. While listening to speakers, interpreters usually preemptively group the streaming words into units with clear and definite meaning, referred to as meaningful units that can be directly translated without waiting for more words.

[*] Corresponding author.

	shàngwǔ		diǎn		wǒ	qùle	tàng		gōngyuán
Source:	上午	10	点		我	去了	趟		公园
	morning	10	o'clock		I	go to			park
Text Translation:									I went to the park at 10 a.m.
Source with MU:	上午	10	点	||	我	去了	趟	||	公园
Simul. Interpretation:			At 10 a.m.	||			I went to	||	the park.

Table 1: A comparison of Chinese-English text translation and simultaneous interpretation. A text translator translates the full sentence after reading all the source words and produces a translation with a long-distance reordering by moving the initial part (as underlined) of the source sentence to the end of the target side. But when doing simultaneous interpreting, an interpreter starts to translate as soon as he or she judges that the current received streaming text constitutes an MU ("||") and translate them monotonically.

- Interpreters are often obliged to keep close to the source speech and render the translation of MUs in order, i.e., perform translation monotonically while making the translation grammatically tolerable.

See Table 1 for illustration. Unlike text translator, a simultaneous interpreter dynamically segments the source text into 3 MUs and translates them monotonically.

In our approach, we model the *policy* as an MU segmentation model, which dynamically splits the streaming text into meaning units. Once a meaning unit is detected [1], it is fed to the MT model to generate translation. The MU segmentation is implemented by a classification model under the pre-training & fine-tuning framework (Devlin et al., 2018; Sun et al., 2019). As there are no standard training corpora to train the MU segmentation classifier, we propose a novel translation-prefix based method to generate training data. Basically, the method detects whether the translation of a sequence of words is a prefix of the full sentence's translation. If so, the sequence is considered as an MU. This makes the segmentation model consistent with the translation model. We further propose a refined method to extract fine-grained MUs to reduce latency.

Experimental results on NIST Chinese-English and WMT 2015 German-English datasets show that our method outperforms the previous state-of-the-art methods in balancing translation accuracy and latency. The contributions of this paper can be summarized as follows:

- Inspired by human interpreters, we propose a novel adaptive segmentation policy that splits the ASR output into meaning units for simultaneous translation. The meaning units ensure the MT model to produce high-quality translation with low latency.
- We propose a novel prefix-attention method to extract fine-grained MUs by training a neural machine translation (NMT) model that generates monotonic translations.
- Our method is simple yet effective. It can be easily integrated into a practical simultaneous translation system.

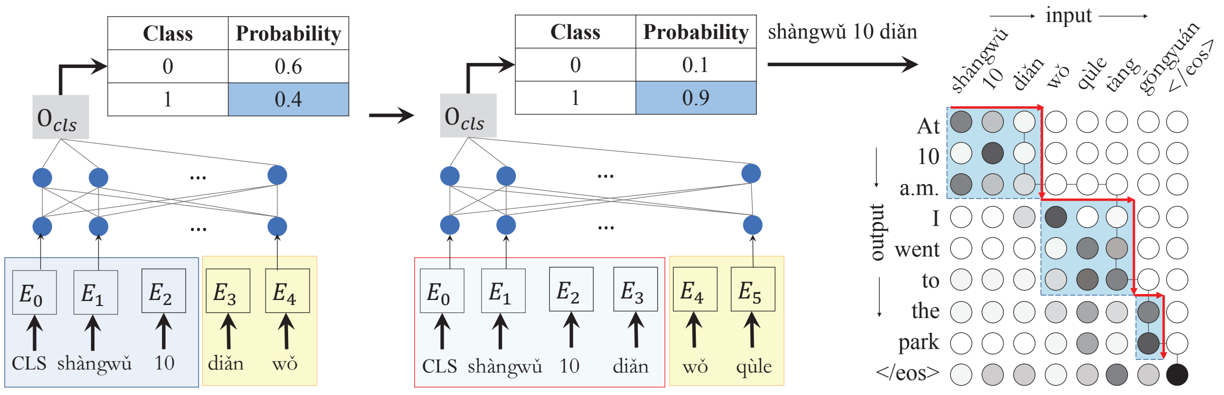
2 Adaptive Segmentation Policy

Our idea is inspired by human interpreters who start translating as soon as they recognize an MU. In this paper, we aim to split the streaming text into MUs to get a trade-off between translation quality and latency. See Figure 1 for illustration. We model the MU segmentation as a classification problem and train a classifier, which receives a streaming text from ASR output and detects whether it constitutes an MU (Figure 1 (a) and (b)). Once an MU is detected, it is sent to the MT model to produce translation (Figure 1 (c)). Meanwhile, the MU segmentation model keeps receiving source words.

To build an MU segmentation model, there are three key issues:

1. What is an MU? Though it is a widely adopted concept in simultaneous interpretation, it has no precise definition. We will discuss it in Section 2.1.
2. How to construct a training corpus for the MU segmentation model? We propose two methods to extract MUs from the training corpus automatically. In the basic method, we propose a generation framework (Section 2.2). To further generate fine-grained MU, we then propose a refined method (Section 2.3).

[1] In this paper, we use segmentation and detection interchangeably.


Class
Probability
0
0.6
1
0.4
O_{cls}
E_0 E_1 E_2 E_3 E_4
CLS shàngwǔ 10 diǎn wǒ
Class
Probability
0
0.1
1
0.9
O_{cls}
E_0 E_1 E_2 E_3 E_4 E_5
CLS shàngwǔ 10 diǎn wǒ qùle
shàngwǔ 10 diǎn
input
shàngwǔ 10 diǎn wǒ qùle tāng gōngyuán </eos>
output
At 10 a.m. I went to the park </eos>

(a). Detect whether "shàngwǔ 10" is an MU (b). Detect whether "shàngwǔ 10 diǎn" is an MU (c). Monotonic translation

Figure 1: Illustration of our simultaneous translation system. The MU segmentation model receives streaming ASR output and detects whether the streaming source text forms a meaning unit. (a). Detect whether the sequence in the left rectangle constitutes an MU, with the reference of two future words in the right rectangle. The output probability of **class 1** (predicts the sequence as an MU) is lower than threshold δ (e.g. $\delta = 0.7$). (b). The model reads more source words and the probability of **class 1** is greater than δ now. (c). As soon as an MU is detected, it is sent to the translation model to generate translation. Once an MU is translated, the translation will not be changed by the incoming source text. We thus use a force decoding to ensure the monotonic translation.

3. How to train the MU segmentation model? We train the classifier under a pre-training & fine-tuning framework (Section 2.4).

Finally, the MU segmentation model is integrated into a cascaded simultaneous translation system. It receives ASR output and produces MUs as MT input.

2.1 MU Definition

As mentioned, an MU in simultaneous interpretation refers to a group of streaming words with definite or clear meaning. However, it is not easy to give it a precise definition. Even human interpreters cannot determine the exact boundary of MUs during interpreting.

Before we describe our definition, we first try to list the properties of an ideal MU:

1. An MU should be short to reduce latency.

2. The translation of an MU should not be changed (or affected) by the incoming source words. [2] This requires that an MU should contain enough information to produce a translation.

Accordingly, we define an MU as *the minimum segment whose translation will not be changed by subsequent text.*

[2]Once an MU is detected, the simultaneous translation system should output its translation immediately, and the translation cannot be modified. Rewriting the generated MU translation in a practical system will hurt user experience.

Formally, we can take a pre-trained MT system M_{nmt} to extract MUs. Given a streaming source sequence $\mathbf{x} = \{x_1, x_2, ...x_T\}$, we want to find a list of MU segments $\mathbf{S_{MU}} = \{S_1, S_2, ...S_K\}$ i.e., to split $\mathbf{x}$ into K MUs, satisfying the above properties that each partial translation $M_{nmt}(S_k)$ will not change by the incoming words. And our goal is to find a segmentation $\mathbf{S_{MU}}$ with appropriate granularity.

2.2 Basic Method for Constructing Training Data

We propose a simple method to generate MUs for a source sentence $\mathbf{x} = \{x_1, x_2, ...x_T\}$. The main idea is that, for a prefix $\mathbf{x}_{\leq t} = \{x_1, x_2, ...x_t\}$ $(1 \leq t \leq T)$, if its translation $\mathbf{y}^t = M_{nmt}(\mathbf{x}_{\leq t})$ is also a prefix of the full sentence translation $\tilde{\mathbf{y}} = M_{nmt}(\mathbf{x})$, we take x_t as a boundary of MU. The reason is that, in this case, the translation of $\mathbf{x}_{\leq t}$ is not affected by more source words, indicating that the information of the current source sequence is sufficient to generate an accurate partial translation. To keep the MU as short as possible, we incrementally input the source text word-by-word to an MT model and detect whether the translation $\mathbf{y}^t$ of current source sequence is a prefix of the full-sentence translation $\tilde{\mathbf{y}}$. If the answer is true, then we segment the current source sequence as an MU. Otherwise, the model continues reading more source words.

Note that once an MU is detected, its translation

```
Algorithm 1: Extract MUs
    Input: x = $x_1, ..., x_T$    ▷ streaming input
    Output: $\mathbf{S_{MU}}$    ▷ list of MU segmentation
1   $k = 0$    ▷ position of last MU boundary
2   $\widetilde{\mathbf{y}} = M_{nmt}(src = \mathbf{x},\ tgt_{force} = None)$
        ▷ full sentence decoding
3   while Reading $x_t$ do
4       $\mathbf{y}^t = M_{nmt}(src = \mathbf{x}_{\leq t},\ tgt_{force} = \mathbf{y}^k)$
5       if $\mathbf{y}^t$ is a prefix of $\widetilde{\mathbf{y}}$ then
6           $\mathbf{S_{MU}} = \mathbf{S_{MU}} \cup \{x_{k+1}, ..., x_t\}$
7           $k = t$
8   return $\mathbf{S_{MU}}$
```

is fixed. To keep consistency, when detecting a new MU, we first force decode the translation of previous MUs and then decode the new sequence. The whole process is described in Algorithm 1. The algorithm reads source sequence word-by-word (Line 3), and generates translation by force decoding using the history translation of previous detected MUs, denoting as tgt_{force}(Line 4). The sequence is detected as an MU if its translation is a prefix of the full-sentence translation (Line 5).

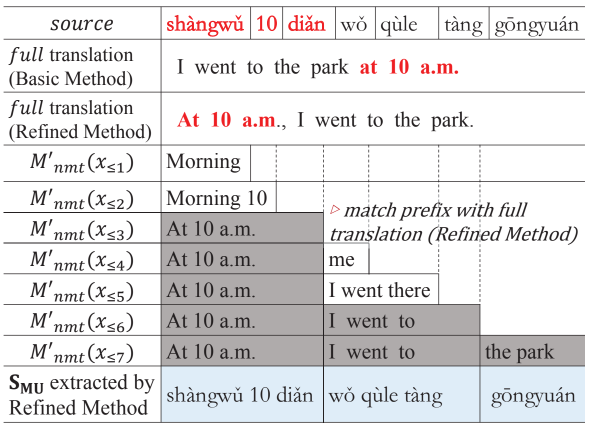
The above algorithm is simple, however, there are two main problems. First, the constraint that $\mathbf{y}^t$ is a prefix of $\widetilde{\mathbf{y}}$ (Line 5) is too strict. To alleviate this problem, we expand the full-sentence translation $\widetilde{\mathbf{y}}$ to a set of candidates through beam search [3].

The second problem is that the translation model M_{nmt} is trained on sentence pairs used for text translation rather than simultaneous translation. There are often long-distance reorderings in the training corpus, which have been learned by the translation model and prevent the basic method from extracting fine-grained MUs. See Figure 2 for illustration, the initial part of the *source* is translated to a sequence at the end of the target (in bold) in the basic method. This makes all the translation of **x** prefixes fail to match the *full translation*, resulting in only one MU could be extracted, as the whole sentence itself. For this problem, we propose a **refined method** to train an NMT model M'_{nmt} with fewer reorderings.

2.3 Refined Method for Constructing Training Data

The process of the **refined method** is described as below:

[3] In this paper, we keep top $N = 10$ results as candidates



source	shàngwǔ	10	diǎn	wǒ	qùle	tàng	gōngyuán
full translation (Basic Method)	I went to the park **at 10 a.m.**						
full translation (Refined Method)	**At 10 a.m.**, I went to the park.						
$M'_{nmt}(x_{\leq 1})$	Morning						
$M'_{nmt}(x_{\leq 2})$	Morning 10			▷ *match prefix with full translation (Refined Method)*			
$M'_{nmt}(x_{\leq 3})$	At 10 a.m.						
$M'_{nmt}(x_{\leq 4})$	At 10 a.m.			me			
$M'_{nmt}(x_{\leq 5})$	At 10 a.m.			I went there			
$M'_{nmt}(x_{\leq 6})$	At 10 a.m.			I went to			
$M'_{nmt}(x_{\leq 7})$	At 10 a.m.			I went to			the park
$\mathbf{S_{MU}}$ extracted by Refined Method	shàngwǔ 10 diǎn			wǒ qùle tàng			gōngyuán

Figure 2: A running example of extracting MUs. Using the **refined method**, we obtain three MUs according to the matching of partial translation and full translation. While due to the long-distance reordering of full translation in the **basic method**, we cannot extract short MUs. The gray blocks denotes the tgt_{force} parts.

1. Use standard sentence aligned parallel corpus to pre-train an NMT model M_{nmt};

2. Generate monotonic translation for each source sentence in the corpus using M_{nmt} with *prefix-attention*. [4]

3. Use the generated training data to train a monotonic translation model M'_{nmt} by fine-tuning on M_{nmt}.

4. Use M'_{nmt} to extract MUs on the training corpus according to Algorithm 1.

Prefix-attention. To generate monotonic translation, we propose a method that each target word y_j is generated by a prefix source sequence rather than by the full source sentence. Formally, given a source sentence **x**=$\{x_1, x_2, ..., x_T\}$, we define $g(j)$ as a monotonic non-decreasing function that denotes the current source position the encoder observed from the beginning. At decoding step j, only a prefix source sequence $x_{\leq g(j)} = \{x_1, x_2, ..., x_{g(j)}\}$ can be used to generate y_j, where $0 < g(j) \leq T$.

The key issue is how to carefully choose $g(j)$ for each target word y_j. Our main idea is that, to generate target word y_j, we expect the model to

[4] From the translation results produced by M_{nmt} with prefix attention, we filter out two kinds of low-quality sentences: 1) Remove those sentences whose word orders are identical with their counterparts in corresponding full sentence translations. 2) Remove the translation whose score is lower than full-sentence translation.

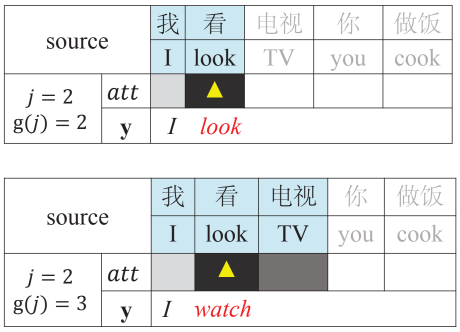


Figure 3: A Chinese-English example for our prefix attention. In the upper case, the model fails to produce correct translation because of a lack of future contextual information. By expanding the source prefix (lower case), the model produces correct translation. *att* is the encoder-decoder attention, and the triangle indicates the location of maximum attention for the current decoding step.

observe limited but adequate contextual information to produce correct translation. See Figure 3 for illustration where the full-sentence translation should be "*I watch TV, you cook*". For the upper case of Figure 3, the current decoding step is $j = 2$, and $g(j) = 2$, meaning that the NMT model uses prefix $x_{\leq 2}$ to generate y_2. However, in this case, the model makes an error to generate y_2. Without observing more context, the model is difficult to make a decision whether y_2 should be "*look*" or "*watch*" or "*see*", etc. For the lower case, the model expands the source prefix by one more source word and produces correct translation.

This raises a question of how do we know whether a source prefix is sufficient or not for producing a target word? Let's take a look at the encoder-decoder attention. To generate a target word y_j, the NMT model computes probabilities between each source word x_t ($1 \leq t \leq g(j)$) and y_j via encoder-decoder attention α_{jt}. The higher the attention weight is, the greater contribution the corresponding source words make in decoding. Therefore, we can find the source words that contribute the most by locating the highest attention weight. For example, in Figure 3, the source word x_2 contributes the most for y_2. When the source word with maximum attention appears at the end of a prefix span, the model takes a risk that it cannot observe future context for translation. In this case, we should expand the span to reduce the risk.

Algorithm 2 shows the whole process of prefix-attention decoding. Initially, we set $g(j) = 1$ for $j = 1$. For each decoding step j, the algorithm

Algorithm 2: Prefix-attention Decoding

```
   Input: $\mathbf{x} = x_1, ..., x_T$        ▷ streaming input
   Output: $\mathbf{y}$                      ▷ monotonic translation
1  $j = 1$                                   ▷ decoding step
2  $g(j) = 1$                                ▷ initialize $g$
3  while Decoding step $j$ do
4      $a_j = \arg\max_{t \in [1, g(j)]} \alpha_{jt}$
                     ▷ the position with max attention
5      if $1 \leq a_j < g(j)$ then
6          $p(y_j|\mathbf{x}) = p(y_j|\mathbf{x}_{\leq g(j)}, \mathbf{y}_{<j}; M_{nmt})$
7          $j \leftarrow j + 1$                 ▷ next step
8          $g(j) = g(j-1)$          ▷ non-decreasing
9      else
10         $g(j) += 1$ ▷ expand $g(j)$ by 1 word
11 return y
```

first locates the maximum attention to a_j (Line 4), according to the following equation:

$$a_j = \arg \max_{t \in [1, g(j)]} (\alpha_{jt}) \tag{1}$$

where,

$$\alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{t'=1}^{g(j)} \exp(e_{jt'})} \tag{2}$$

If $1 \leq a_j < g(j)$, it means that the model can observe both history and future source context to generate y_j. Otherwise, the model faces the risk of lacking future context. In this case, we expand $g(j)$ by one more word.

2.4 The MU Segmentation Model

Our MU segmentation model is illustrated in Figure 1 (a) & (b). Given a streaming source sequence $x = \{x_1, x_2, ...\}$, the model aims to detect whether a prefix of x constitutes an MU on-the-fly. The model takes two inputs: the source sequence $c_t = \{x_{\leq t}\}$ and future words $f_t = \{x_{t+1}, ..., x_{t+m}\}$, and outputs the probability of predicted label l_t, denoting the context c_t being an MU (class 1) or not (class 0). m is a hyper-parameter as the number of future words. Larger m means to wait for more future words at inference time. In this paper, we set $m = 2$. c_t is considered as an MU if $p(l_t = 1|c_t, f_t; \theta_{model})$ is larger than a threshold δ.

In the training stage, we first extract the MUs in the training corpus according to the basic method (Section 2.3) or refined method (Section 2.2). Then we generate the training data for the MU detection model. For each sentence $x = \{x_1, x_2, ..., x_N\}$ in

t	c_t	$f_t(m=2)$	l_t
1	shàngwǔ	10 diǎn	0
2	shàngwǔ 10	diǎn wǒ	0
3	shàngwǔ 10 diǎn	wǒ qùle	1
4	shàngwǔ 10 diǎn wǒ	qùle tàng	0
5	shàngwǔ 10 diǎn wǒ qùle	tàng gōngyuán	0
6	shàngwǔ 10 diǎn wǒ qùle tàng	gōngyuán	1
	...		

Table 2: The training samples for the MU detection model generated according to the MU segmentation result in Figure 2.

the training corpus, we generate N samples. Each sample is a triple $<c_t, f_t, l_t>$ for $t = \{1, 2, ..., N\}$. If x_t is a boundary of MU, we set l_t to 1; otherwise 0. Take the extracted MUs in Figure 2 as example, we generate training samples as illustrated in Table 2. Note that for t larger than $N - m$, we only use the remaining words in the sentence as future words, which is less than m. Our training follows the pre-training and fine-tuning framework (Devlin et al., 2018; Sun et al., 2019).

3 Experiments

We carry out experiments on two translation tasks: the NIST Chinese-English (Zh-En) translation task (2M sentences), and the WMT 2015 German-English (De-En) translation task (4.5M sentences).we use BLEU (Papineni et al., 2002) score to evaluate translation quality, and Average Lagging [5] (Ma et al., 2019) to measure latency.

3.1 Data Preprocess

We use an open-source Chinese Tokenizer [6] to preprocess Chinese and apply Moses Tokenizer [7] to preprocess English and German. For Zh-En, we validate on NIST newstest 2006 and report results on newstest 2002, 2003, 2004, 2005, and 2008. We use SententcePiece [8] to implement byte-pair encoding (BPE) (Sennrich et al., 2016) for both Chinese and English by setting the vocabulary size to 20K and 18K, respectively. For De-En, we validate on newstest 2013 and then report results on newstest 2015. We utilize a joint vocabulary, with a vocabulary size of 32K. Notably, translation quality in all experiments is measured using detokenized, cased BLEU.

[5]https://github.com/SimulTrans-demo/STACL

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

[7]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

[8]https://github.com/google/sentencepiece

3.2 Model Settings

We compare our methods with previous state-of-the-art methods:

- *wait-k* (Ma et al., 2019): first waiting for k words, then emiting one token after reading each word.
- *chunk*: extracting the training segments by segmenting the source sentence into minimally sized chunks such that crossing and one-to-many links between source and target words in an optimal GIZA++ alignment occur only within individual chunks. We borrow this idea of training samples generation from Rangarajan Sridhar et al. (2013).
- *MILk* (Arivazhagan et al., 2019): using hard attention to schedule the policy and train the policy together with the NMT model in an end-to-end framework. It uses a weight λ in the loss function to balance translation quality and latency. [9]
- *MU*: our proposed **basic method** of translating after detecting a meaning unit.
- *MU++*: our proposed **refined method** to detect fine-grained meaning units.

The training of segmentation models for *chunk*, *MU* and *MU++* are based on the classification task of BERT [10] and ERNIE [11], with the pre-trained language model of German and Chinese, respectively. We use the base model and take the learning rate of $2e^{-5}$ at the fine-tuning stage.

Our translation models are trained on big Transformer (Vaswani et al., 2017). All the approaches share the same machine translation corpus except *MU++*, which is trained on the augmented training corpus generated by *prefix-attention* (Section 2.3).

3.3 Overall Results

3.3.1 Chinese-English Translation

Figure 4 shows the translation quality and latency on Chinese-English translation tasks. We have the following observations:

- Our methods, both *MU* and *MU++*, outperform *wait-k* and *chunk* method in terms of translation quality and latency.

[9]We compared MILk on German-English translation task since they do not report Chinese-English results.

[10]https://github.com/google-research/bert

[11]https://github.com/PaddlePaddle/ERNIE

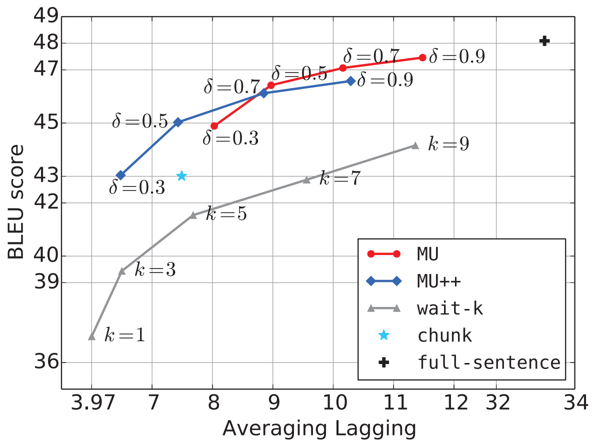


Figure 4: Quality-latency results on the NIST dataset. We report the averaged results on NIST02, NIST03, NIST04, NIST05, and NIST08. Note that each method has its own performance on full-sentence translation, which is denoted as "+" with the same color with the corresponding method. δ is the threshold of the MU detection model (Section 2.4).

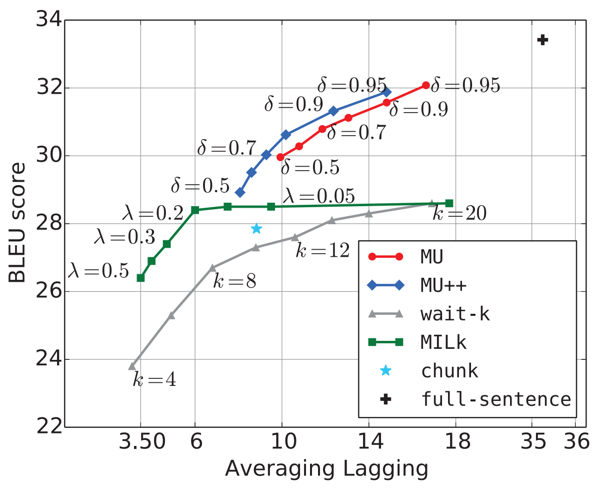


Figure 5: Quality-latency results on the WMT15 German-English dataset.

- With the increase of δ (the threshold for MU detection, in Section 2.4), the quality is improved while the latency is also increased. In practice, δ can be tuned to obtain a trade-off between quality and latency according to real requirement.

- Compared to *MU*, *MU++* significantly reduces latency while causing a drop in quality. A possible reason is that the references in the test set are produced via text translation and contain many long-distance reorderings. But *MU++* is designed to produce translation with less reordering. We'll further analyze this issue in Section 3.4.

3.3.2 German-English Translation

Figure 5 shows the De-En translation results. When the average lagging is larger than 8, our model's translation quality outperforms the other models. Note that low latency in other models performance causes a large decrease in BLEU scores.

For the joint learning method, *MILk*, its full-sentence performance is limited by the RNN architecture, which is inferior to the Transformer. Furthermore, its full-sentence translation model uses a bidirectional encoder, while the streaming model uses unidirectional encoders, resulting in the performance gap in its full-sentence model and streaming model. Both models in our approaches, on the contrary, use the bidirectional encoder, thus avoiding such gaps.

It's interesting to find that the trend of *MU* and *MU++* is different from that of the Zh-En experiment. According to Figure 4, *MU++* is inferior to *MU*, achieving low latency while impairing the translation quality. But in De-En translation, *MU++* performs better than *MU*. We analyze this in the next section.

3.4 Test on Reference with Simulated Simultaneous Interpretation

We randomly select 200 sentences in Zh-En and De-En, respectively, from the corresponding test set and ask human translators to translate them in the way that they do simultaneous interpretation. For example in Figure 6 ("*Simul-Ref*"), "*next week*" appears at the initial position of the target sentence, keeping the order of the "*Source*". We also list the translation process of the comparing methods.

Using the re-translated text as references, we evaluate both *MU* and *MU++* with flexible latency ($\delta = 0.3, 0.5, 0.7, 0.9$) on the test sets. The performance on the new Zh-En test set is depicted in Figure 7. *MU++* presents shorter latency as well as more promising quality on this dataset compared to *MU*. Another finding is the quality of *MU* degrades even with a larger δ. We attribute this to the inconsistency between reference and translation of *MU*, because longer MU may further cause long-distance reordering. This also explains that the superiority of *MU* to *MU++* in the original test set is due to the distribution inconsistency.

The performance of De-En is illustrated in 8, in which the performance of the two methods is in line with that on the original test set: *MU++* performs slightly better than *MU*. The reason is that in

Source:	jǐngfāng 警方 police	xiàzhoū 下周 next week	jiāng 将 will	duì 对 for	bù fèn 部分 part	shè àn 涉案 involved	rén yuán 人员 people	tí qǐ gōng sù 提起 公诉 indict
Text-Ref:	Police will indict some of the people involved in the case next week .							
Simul-Ref:	Next week, police will indict some of the people involved in the case.							
chunk		Police next week						will prosecute part of the suspects.
Wait-3:			Police	to	part	of		the people involved will be charged next week.
MU	The police							will bring lawsuits against some of the suspects next week.
MU++		Next week, the police	will					bring lawsuits against some of the suspects.

Figure 6: A Chinese-English example in the test set with the original text translation reference ("*Text-Ref*") and the simultaneous interpretation reference ("*Simul-Ref*"). Both *chunk* and *wait-3* generates incorrect translation. But *MU* and *MU++* translates accurately. Furthermore, *MU++* avoids long-distance reordering by keeping "xiàzhoū (next week)" in order with the source sentence, and thus reduces latency.

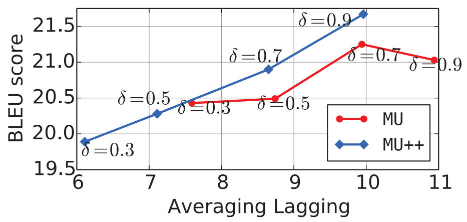


Figure 7: Performance of Zh-En on 200 sentences with simultaneous interpretation reference (*Simul-Ref*).

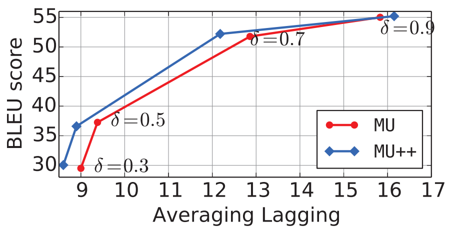


Figure 8: Performance of De-En on 200 sentences with simultaneous interpretation reference (*Simul-Ref*).

German, there are a lot of "SOV" structures, while English is an "SVO" language. In this case, both *MU* and *MU++* should wait until a verb at the end of a sentence before generating an accurate translation. Thus the performance of *MU* and *MU++* is similar.

We further ask human translators to evaluate the quality of *MU* and *MU++*. They rated each translation in **Bad**, **OK** and **Good** based on the translations' adequacy, correctness and fluency:

- **Bad** indicates the translation is unacceptable and incorrect or inadequate.
- **OK** denotes the translation is comprehensible and adequate, but with minor errors such as incorrect function words and less fluent phrases.
- **Good** means a translation is correct and contains no obvious errors.

	Method	**Bad**	**OK**	**Good**	**Acceptablity**
Zh-En	*MU*	28.5%	49%	22.5%	71.5%
	MU++	30.0%	46.5%	23.5%	70.0%
De-En	*MU*	23.5%	54.5%	22.0%	76.5%
	MU++	22.0%	57.0%	21.0%	78.0%

Table 3: The human evaluation of the Zh-En and De-En translation on 200 sentences with $\delta = 0.7$.

We evaluate the performance of *MU* and *MU++* at $delta = 0.7$, which is the point of achieving relatively high translation quality with limited latency. The evaluated performance of the 200 sentences in Zh-En and De-En is reported in Table 3. We define the overall acceptability as a percentage of the sum of **OK** and **Good** cases. It is obvious that the acceptability of *MU* and *MU++* shows a consistent trend with their BLEU in Figure 7 and Figure 8 that *MU++* performs slightly worse in Zh-En but the opposite in De-En. However in both language pairs, *MU++* achieves a lower latency.

4 Related Work

Recent simultaneous translation work focuses on exploring a policy to decide whether to wait for another source word or generate a target word. Rangarajan Sridhar et al. (2013) investigated a variety of policies depending on lexical cues. Oda et al. (2014) proposed to optimize a segmentation model with the target of achieving better translation quality. However, their performance is limited largely by weak features such as N-gram and POS. Some research learns the policy depending on reinforcement learning, with the goal of good translation quality and low latency (Grissom II et al., 2014; Satija and Pineau, 2016; Gu et al., 2017; Alinejad et al., 2018). But reinforcement learning is notorious for its unstable training process. Cho and Esipova (2016) proposed a heuristic measure to determine the policy at inference time, without using a deep model. Ma et al. (2019) and Dalvi et al. (2018) applied fixed policy independent of contextual information, which inevitably need to guess the future context in translation (Zheng et al., 2019a). Some work applied advanced attention mechanisms that replace the softmax attention with a stepwise Bernoulli selection probability (Raffel et al., 2017). Arivazhagan et al. (2019) proposed infinite lookback to integrate the hard monotonic attention with soft attention. Ma et al. (2020) proposed multi-head monotonic attention and obtained further improvements. However, the autoregressive training process makes its exploration inefficient.

5 Conclusions

In this paper, we propose a novel adaptive segmentation policy for simultaneous translation. Motivated by human interpreters, the model constantly reads streaming text and dynamically segments it into meaning units. We first generate training data for MU via a translation-prefix based method, keeping consistency between the segmentation model and the translation model. Further, we propose a refined-method to extract fine-grained MUs to reduce latency. Experimental results on both Chinese-English and German-English show that our model outperforms the previous state-of-the-art. The method obtains a good trade-off between translation accuracy and latency and can be easily implemented into a practical simultaneous translation system.

Acknowledgements

We would like to thank our colleague, Chen Ying, for her valuable advice in writing articles and organizing relevant data evaluation.

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.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

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

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. *arXiv preprint arXiv:1806.03661*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor OK 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.

Mingbo Ma, Liang Huang, Hao Xiong, Kaibo Liu, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, and Haifeng Wang. 2019. STACL: simultaneous translation with integrated anticipation and controllable latency. In *ACL 2019*, volume abs/1810.08398.

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

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 551–556.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Colin Raffel, Minh-Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2837–2846. JMLR. org.

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

Harsh Satija and Joelle Pineau. 2016. Simultaneous machine translation using deep reinforcement learning. In *Proceedings of the Workshops of International Conference on Machine Learning, New York*, pages 110–119.

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)*, volume 1, pages 1715–1725.

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

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020. Simultaneous translation policies: From fixed to adaptive. *arXiv preprint arXiv:2004.13169*.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2014 Conference on empirical methods in natural language processing (EMNLP)*.

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.