Simultaneous Neural Machine Translation with Constituent Label Prediction

Yasumasa Kano[1]   **Katsuhito Sudoh**[1,2]   **Satoshi Nakamura**[1,2]

[1]Nara Institute of Science and Technology (NAIST), Japan
[2]Center for Advanced Intelligence Project (AIP), RIKEN, Japan
`{kano.yasumasa.kw4, sudoh, s-nakamura}@is.naist.jp`

Abstract

Simultaneous translation is a task in which translation begins before the speaker has finished speaking, so it is important to decide when to start the translation process. However, deciding whether to read more input words or start to translate is difficult for language pairs with different word orders such as English and Japanese. Motivated by the concept of pre-reordering, we propose a couple of simple decision rules using the label of the next constituent predicted by incremental constituent label prediction. In experiments on English-to-Japanese simultaneous translation, the proposed method outperformed baselines in the quality-latency trade-off.

1 Introduction

Simultaneous machine translation is a task in which the machine starts outputting a translation before reading the entire input sentence. This task is more difficult than full-sentence translation because it translates the initial part of a sentence without the context of the latter part. This involves a trade-off between delay and quality of the translation; using a longer context should improve translation quality at the cost of a longer delay, and vice versa. In practice, we should control the latency so that it's not too large, but we may also need to allow a long latency depending on the situation.

Most of the recent simultaneous translation models (Ma et al., 2019; Arivazhagan et al., 2019; Raffel et al., 2017; Arivazhagan et al., 2019; Ma et al., 2020b; Dalvi et al., 2018; Gu et al., 2017; Alinejad et al., 2018; Cho and Esipova, 2016; Zheng et al., 2020, 2019; Zhang et al., 2020) are based on neural machine translation (NMT), although earlier studies were based on statistical machine translation (Rangarajan Sridhar et al., 2013; Grissom II et al., 2014; Oda et al., 2014, 2015). In simultaneous NMT, there are two major approaches: those in which a latency hyperparameter is given before the training and those in which it is given at the time of inference.

The former approach requires training a model individually for each pre-defined latency setting, while the latter approach uses a single model for different latency conditions. Most human simultaneous interpreters would not need such long training to slightly adjust latency, while it takes much more time to learn other languages to develop their translation skill. Therefore, the latter approach is closer to the learning process of human simultaneous interpreters.

wait-k (Ma et al., 2019) is a simple simultaneous NMT method of the former approach that waits *k* tokens before starting to translate. It also has variants within the latter approach called test-time wait-k, in which k is determined at the inference time. wait-k had better performance than test-time wait-k in that study's experiments.

There is another method in the latter approach that uses Meaningful Unit (Zhang et al., 2020). In this model, chunk-based incremental decoding is done at inference time by segmentation with a boundary predictor. This model outperformed baselines of the former approach. They refined their basic boundary predictor to deal with sentence pairs in which full-sentence translation needs long-distance reordering. However, its training process is very complicated: It first generates monotonic translations, fine-tunes the NMT model with them, then generates an oracle boundary with the model, and finally fine-tunes a boundary-prediction model based on BERT (Devlin et al., 2019).

Simultaneous translation is still difficult for language pairs such as English-Japanese, which often require long-distance reordering. To tackle the reordering problem, we propose an input-segmentation method for simultaneous translation, using a couple of simple rules and incremental prediction of the label of a syntactic constituent coming immediately after the input existing so far. This

1124

Source sentence	I **bought** *a pen.*
Monotonic translation	watashi wa **katta** *pen wo.*
Full-sentence translation	watashi wa *pen wo* **katta**.

Table 1: Translation from English (SVO) to Japanese (SOV)

Boundary prediction	I / **bought** *a pen.*
Simultaneous translation	watashi wa / *pen wo* **katta**.

Table 2: Example of English-to-Japanese translation using proposed method with segment-boundary prediction

is not dependent on the trained NMT. Therefore, once we create it, it is reusable for other models.

Our proposed method is inspired by Head Finalization (Isozaki et al., 2010). Head Finalization reorders words of the source sentence before translating from an SVO (Subject-Verb-Object) language to an SOV language in full-sentence statistical machine translation. This method moves a syntactic head into a later position so that the word order of the source language (e.g., English) becomes similar to that of the target language (e.g., Japanese). This enables us to monotonically translate from English, which is a typical SVO language, to Japanese, a typical SOV language.

Recent NMT models like Transformer (Vaswani et al., 2017) works well on reordering in general, so this kind of pre-reordering is not usually used. However, simultaneous translation monotonically reads input words one by one, and therefore the difference in word order remains a problem. As shown in Table 1, monotonic translation often becomes unnatural compared to full-sentence translation. The part "bought a pen" should be translated to *pen wo katta* by reversing the word order. Therefore, after reading the word "bought," it is important to wait for future words without starting to translate it. In this case, "I" is the last word that does not require reordering. This word is regarded as a segment boundary to start a partial translation.

Table 2 shows an example of our proposed segmentation. Suppose we predict the next constituent label as a verb phrase (VP) after reading an input word "I." This shows the possibility that the next words should be reordered, so the "I" becomes the boundary. Once detecting the boundary, NMT model starts to translate "I" into "watashi wa." After that, the model restarts to read the remaining input words, then translates "bought a pen" into "pen wo katta." The total output of simultaneous translation based on the proposed segmentation is the same as that of full-translation in this simple

example. By ensuring that the Verb and its Object of the source sentence are included in a single segment, it is possible to output translation while maintaining the SOV-like structure of the target language.

In experiments on English-to-Japanese simultaneous translation, the proposed method outperformed baselines in the quality-latency trade-off.

2 Related work

In statistical machine translation, there are several approaches to finding boundaries of segments for simultaneous translation. Oda et al. (2014) proposed a method to choose segment boundaries that maximize the BLEU score. Rangarajan Sridhar et al. (2013) proposed segmentation strategies based on lexical cues.

In NMT, there have been many studies on simultaneous translation. The amount of latency is decided either before training or at inference time. wait-k (Ma et al., 2019) is the simplest variant using fixed latency: It simply waits for k tokens before starting translation (Ma et al., 2019). The latency policy can be learned from a parallel corpus together with an NMT model. MILk (Arivazhagan et al., 2019) and other approaches (Raffel et al., 2017; Ma et al., 2020b) used a latency-augmented loss function in training to balance latency and accuracy.

In contrast, the latency policy can be learned with a pre-trained NMT model, such as test-time wait-k (Ma et al., 2019) and STATIC-RW (Dalvi et al., 2018). These have fixed policies that wait for the fixed number of tokens before translation, but there are other models that learn a more flexible policy for a given pre-trained NMT model. Some studies use reinforcement learning to learn an adaptive READ/WRITE policy (Grissom II et al., 2014; Satija and Pineau, 2016; Gu et al., 2017; Alinejad et al., 2018). Training by reinforcement learning can be unstable depending on the condition. One

method that does not use reinforcement learning is wait-if-* (Cho and Esipova, 2016), which translates and segments jointly to maximize the translation quality. Zheng et al. (2020) extended wait-k to an adaptive policy by adaptively choosing the strategy at inference. There is another method that generates oracle READ/WRITE actions by a pre-trained NMT model and predicts actions using a neural network model (Zheng et al., 2019). Meaningful Unit (Zhang et al., 2020) works along the same lines and has outperformed baselines such as MILk and wait-k.

With respect to the use of syntactic clues for simultaneous translation, Oda et al. (2015) proposed a method to incrementally parse an incomplete sentence by predicting unseen syntactic constituents on the right and left side of each segment. They concatenated the predicted constituents and the words in a segment and then input the result into tree2string translation. They decided to wait for more tokens or output the translation depending on where the constituents appear in the translation result.

Our proposed method is based on chunk-based simultaneous translation using chunk boundary detection with simple rules on next-constituent labels. It basically segments an input before a verb phrase. This is much simpler and easier to implement than the work by Zhang et al. (2020) and Oda et al. (2015).

3 Proposed Method

Figure 1 shows a step-by-step example of our proposed method described in this section.

3.1 Standard Simultaneous Translation

A standard NMT for full sentences is represented by the following equation:

$$p_{full}(Y|X) = \prod_{t=1}^{|Y|} P(y_t|X, y_{<t}), \qquad (1)$$

where $X = x_1, x_2, ..., x_n$ is an input sentence consisting of n tokens and $Y = y_1, y_2, ..., y_m$ is a predicted target language sentence consisting of m tokens.

A simultaneous NMT uses only a prefix of the input to predict a target language token:

$$p_{simul}(Y|X) = \prod_{t=1}^{|Y|} P(y_t|x_{g(t)}, y_{<t}), \qquad (2)$$

where $g(t)$ is a monotonic non-decreasing function representing the number of read source tokens to output the tth target token.

3.2 Chunk-based Simultaneous Translation

We use chunk-based incremental decoding for our simultaneous translation model and a full-sentence NMT model trained in a standard manner. However, at the time of inference, we translate the current prefix upon chunk segmentation while keeping the previously translated output unchanged.

Suppose we have already translated input chunks $\boldsymbol{X}^{i-1} = X_1, X_2, ..., X_{i-1}$ into an output prefix also represented by chunks: $\widetilde{\boldsymbol{Y}}^{i-1} = \widetilde{Y}_1, \widetilde{Y}_2, ..., \widetilde{Y}_{i-1}$, while translating the next input chunk X_i into $\widetilde{Y}_i$. We restart the translation from the beginning using all of the available input chunks X_1^i. This is similar to an approach called *re-translation* that generates translations from scratch for every new input word (Niehues et al., 2016; Arivazhagan et al., 2020), but we apply forced decoding to $\widetilde{\boldsymbol{Y}}^{i-1}$ in the output prefix. The probability of the prefix $\widetilde{\boldsymbol{Y}}^i$ can be denoted as follows:

$$p_{prefix}(\widetilde{\boldsymbol{Y}}^i|\boldsymbol{X}^i) =$$
$$p_{full}(\widetilde{\boldsymbol{Y}}^{i-1}|\boldsymbol{X}^i) \times p_{chunk}(\widetilde{Y}_i|\boldsymbol{X}^i, \widetilde{\boldsymbol{Y}}^{i-1}). \quad (3)$$

The first term is calculated in the same way as the standard full-sentence NMT in Eq. (1) through forced decoding, and the second term is decomposed as follows, letting $\widetilde{Y}_i = y_1^i, y_2^i, ..., y_{|\widetilde{Y}_i|}^i$:

$$p_{chunk}(\widetilde{Y}_i|\boldsymbol{X}^i, \widetilde{\boldsymbol{Y}}^{i-1}) = \prod_{t=1}^{|\widetilde{Y}_i|} P(y_t^i|\boldsymbol{X}^i, \widetilde{\boldsymbol{Y}}^{i-1}, y_{<t}^i).$$
$$(4)$$

This can be more efficient than an incremental Transformer (Ma et al., 2019) that refreshes the encoder for every input word, since our chunk-based translation refreshes the encoder for every input *chunk*, which usually consists of multiple words.

3.3 Chunk Segmentation

We use constituent labels for our rule-based chunk segmentation as follows.

3.3.1 Incremental Constituent Label Prediction

We predict the label of a syntactic constituent coming after a sentence prefix at the current time-step. We call this process *Incremental Constituent Label Prediction* (ICLP). Here, we define this *next*
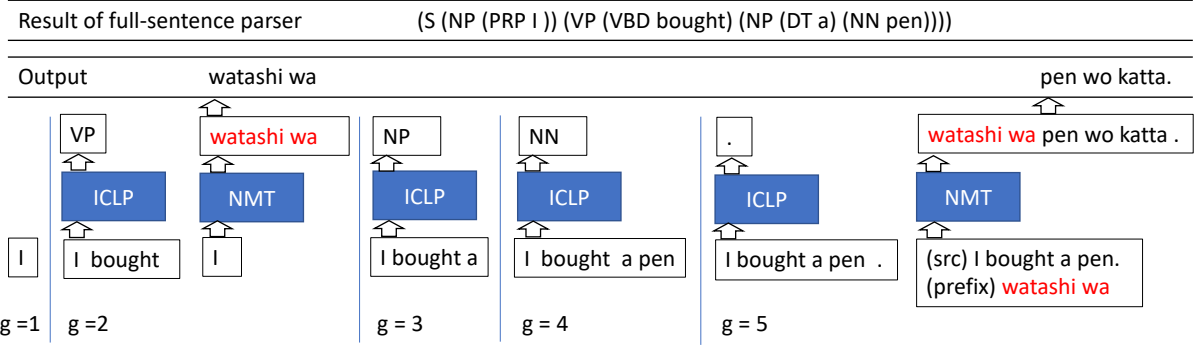
Result of full-sentence parser	(S (NP (PRP I)) (VP (VBD bought) (NP (DT a) (NN pen))))

Output   watashi wa   pen wo katta.

VP	watashi wa	NP	NN	.	watashi wa pen wo katta .
ICLP	NMT	ICLP	ICLP	ICLP	NMT
I	I bought	I	I bought a	I bought a pen	I bought a pen .	(src) I bought a pen. (prefix) watashi wa

g =1   g =2   g = 3   g = 4   g = 5

Figure 1: One look-ahead ICLP gives constituent labels. When a boundary is detected based on the label and rules, NMT starts to translate the source subsequence. The previous translation, which is red in the figure, is used as prefix words for the next translation. EOS is omitted for simplicity in the figure.

segmentation	You	/ can	save	time	by	/ doing	this	.
constituent label		VP	VP	NP	PP	S	NP	.
syntax tree	(S (NP (PRP You)) (VP (MD can) (VP (VB save) (NP (NN time)) (PP (IN by) (S (VP (VBG doing) (NP (DT this))))))) (. .))							

Table 3: Example result of one look-ahead ICLP with a minimum segment size of one.

constituent as the one coming next to the sentence prefix in pre-order tree traversal. However, this label prediction is not easy without observations on the next constituent. In this work, we allow one look-ahead, where we read one more word and predict the label of the constituent starting from that word. This causes an additional delay by one word but improves the prediction accuracy. Suppose we have an input sequence $W = [w_1, w_2, ..., w_{|W|}]$. The one look-ahead ICLP predicts the constituent label c_i upon the observation of w_i, as follows:

$$c_i = \operatorname*{argmax}_{c' \in C} p(c'|w_{\leq i}), \qquad (5)$$

where C is a set of constituent labels. Only a prefix word subsequence is fed into the ICLP, so previous label predictions do not affect later ones.

We can train the ICLP model as a multi-class classifier using a set of training instances in the form of prefix-label pairs. One sentence generates several instances for training data: (w_1, c_1), (w_1, w_2, c_2), (w_1, w_2, w_3, c_3), $(w_1, w_2, w_3, w_4, c_4)$, and so on. We implemented the ICLP model in two different ways using LSTM (Hochreiter and Schmidhuber, 1997) and BERT (Devlin et al., 2019).

3.3.2 Segmentation Rules

Table 3 shows an example of a result by the one look-ahead ICLP. We use one basic and two supplemental rules for chunk segmentation as follows.

- Segment the input coming just before constituents labeled S and VP.

- If the previous label is S or VP, do not segment the input.

- If the chunk is shorter than the minimum length, do not segment the input.

In incremental translation from *Subject-Verb-Object* to *Subject-Object-Verb*, the subject can be translated before observing the verb coming next, but the verb should be translated after observing the object. Therefore, the chunk boundary should be between the subject and verb, not between verb and object. To achieve this, we employ a simple rule to segment a chunk just before VP. We also include S in the rule just as with VP because S (simple declarative clause) often appears in the form of a unary branch "(S (VP ...))" as shown in Table 3.

However, in cases such as "can save" in the example, VP occurs again immediately after the segmentation before "can." The basic rule suggests segmentation before "save," but it does not seem appropriate. Therefore, we introduce the minimum segment size to avoid such over-segmentation as a hyperparameter to control the accuracy-latency trade-off. If the hyperparameter is larger than one, the chunk segmentation after "You" in the example does not occur.

4 Experimental setup

4.1 Dataset and preprocessing

We conducted experiments on English-Japanese (En-Ja) translation. We also tried English-German (En-De) translation to investigate the difference in language pairs.

For En-Ja, the model was trained on 17.9 M sentence pairs from WMT2020 and fine-tuned on 223 K sentence pairs from IWSLT2017. We used 5312 sentence pairs for the development set from dev2010, tst2011, tst2012, and tst2013 of IWSLT. We evaluated the model on 1442 sentence pairs from dev2021 of IWSLT.

For En-De, the model was trained on 4.5 M sentence pairs from WMT2014 and fine-tuned on 206 K sentence pairs from IWSLT2017. We used 5589 sentence pairs for the development set from dev2010, tst2011, tst2012, and tst2013 of IWSLT. We evaluated the model on 1,080 sentence pairs from tst2015 of IWSLT.

We tokenized English and German sentences with `tokenizer.perl` in Moses (Koehn et al., 2007) and Japanese sentences with MeCab (Kudo, 2005). For each language pair, we used subwords based on Byte Pair Encoding (BPE) (Sennrich et al., 2016) with a shared vocabulary of 16 K entries. To develop the subword vocabulary, we used all of the in-domain training sentences (IWSLT) and one million out-of-domain sentences (WMT).

We trained the ICLP models using Penn Treebank 3 (Marcus et al., 1993) for training, excluding a randomly selected one percent of sentences reserved for the development set. We used NAIST-NTT TED Talk Treebank (Neubig et al., 2014) for the evaluation set. The number of training, development, and test instances (e.g., the number of labels to be predicted) were 2.8 M, 27.9 K, and 21.9 K, respectively. Note that multiple ICLP instances are induced from what a single parse tree generates.

4.2 Model settings

We compared the following four models. All of them were based on the Transformer-base (Vaswani et al., 2017).

wait-k

The range of k is [2, 4, 6,..., 30].

Meaningful Unit

The hyperparameter is p, which is the threshold of the probability of a boundary. The ranges of p are [0.5, 0.1, 0.15,..., 0.95], [0.99, 0.991, 0.992,..., 0.999], and [0.9991, 0.9992,..., 0.9999]. Monotonic translation of Meaningful Unit was generated from the fine-tuning dataset by the fine-tuned NMT model. We used their refined Meaningful Unit method, which improved the translation quality at low latency (Zhang et al., 2020)[1]. They used a two look-ahead boundary predictor in their experiments. We additionally tried a one look-ahead predictor because it is not certain how many future words should be used for the predictor.

Fixed-size segmentation This simply segments an input with a fixed length specified by a hyperparameter f, which means the boundary comes every f subwords or words. The range of f is [2, 4, 6,..., 30] for words and [4, 8, 12,..., 60] for subwords.

ICLP

The hyperparameter is m, which means the minimum number of words in one segment. The range of m is [1, 2, 3, . . . , 29].

We controlled hyperparameters to adapt to a wide range of latency. The hyperparameter is given both in the training and at the inference time for wait-k, but it is given only at the inference time for other models. Therefore, we trained the wait-k model for each k while in other approaches a single NMT model is commonly used.

We used fairseq (Ott et al., 2019) to implement these models and basically followed the official baseline for IWSLT 2021[2,3] to set the hyperparameters. We saved checkpoints every 5000 updates for pre-training and every 200 updates for fine-tuning. Other hyperparameters were the same for pre-training and fine-tuning. We stopped training early with patience 4. The max-tokens for the mini batch size was 4096, and weights were updated

---

[1]Zhang et al. (2020) removed the monotonic translations with a lower score than full-sentence translation. However, it is rare for a monotonic translation to have a higher score than full-sentence translation. Consequently, few sentences remained in our setting. Therefore, we improved the translation quality by preventing over-translation instead of removing it. Once the same words are output four times continuously or the target length becomes four times longer than the source length, we expand the source prefix.

[2]https://github.com/pytorch/fairseq/blob/master/examples/simultaneous_translation/docs/enja-waitk.md

[3]https://github.com/pytorch/fairseq/issues/346

every 4 mini batches. We set the learning rate to 0.0007 and trained the model on a single GPU. The last three models used the same NMT model. We used beam search within chunks in a standard way and chose 1-best hypotheses at the end of chunk translation. The beam size was four for the chunk-based and full-sentence models. We used greedy decoding for wait-k.

We implemented two types of ICLP models as mentioned earlier. For the LSTM-based ICLP, we used two-layered unidirectional LSTMs to encode an input sentence with a fully connected layer for the constituent label prediction. The numbers of dimensions for embedding and hidden states are 512. We tokenized English sentences using `tokenizer.perl` in Moses and Byte Pair Encoding (Sennrich et al., 2016) with a vocabulary of 16 K entries. For the BERT-based ICLP, we used a BERT-based classifier with an additional fully connected layer over the `[CLS]` token, implemented using Huggingface transformers (Wolf et al., 2020) with a pre-trained model `bert-base-uncased` and the corresponding subword tokenizer. For both models, the input was a subword sequence, so the constituent label prediction was made upon the observation of an *end-of-word* subword. The following training conditions were commonly applied to both models: learning rate of 5e-5, training batch size of 512 instances, checkpoints saved at the end of every epoch, and early stopping with the patience of three epochs.

4.3 Evaluation

We used SimulEval (Ma et al., 2020a) to evaluate the quality and latency of simultaneous translation. BLEU (Papineni et al., 2002) was used to evaluate quality. We used Average Lagging (AL) (Ma et al., 2019) to evaluate the latency. AL is widely used and defined by the following equation:

$$AL_g(X, Y) = \frac{1}{\tau_g(|X|)} \sum_{t=1}^{\tau_g(|X|)} g(t) - \frac{t-1}{\gamma}. \quad (6)$$

$\tau_g(|X|)$ is the decoding step when the source sentence finishes. It counts latency up to the $\tau_g(|X|)$ th target token predicted just after reading the final source token. γ is defined as $|Y|/|X|$. When the source length $|X|$ equals target length $|Y|$, AL of wait-k equals its k. In this experiment, the latency was calculated on character level for En-Ja, and word level for En-De.
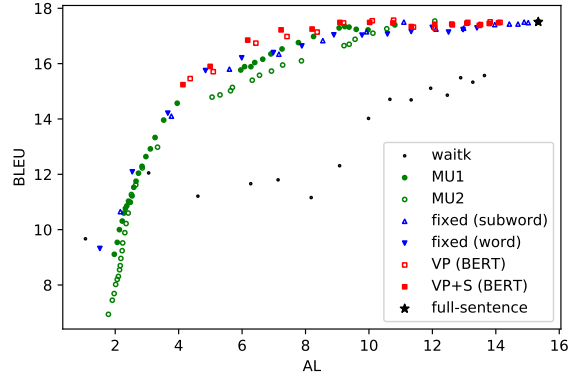

Figure 2: Scatter plot of BLEU and AL (En-Ja). MU1 and MU2 correspond to Meaningful Unit with one and two look-ahead respectively.
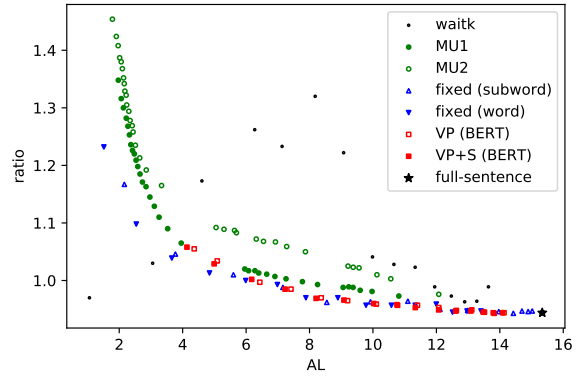

Figure 3: Scatter plot of length ratio and AL (En-Ja)

4.4 Results

We illustrate the results of English-Japanese translation in Figure 2. Our proposed method outperformed baselines in a wide range of AL. Most of the points of the proposed method appear to the upper-left of the other methods, thus showing the best performance. We compared the use of segmentation rules based on VP and VP+S. The points shifted to the left by adding S as boundary because it increased the number of boundaries and decreased latency. Although we tried the different look-ahead lengths of one and two for the boundary predictor of Meaningful Unit, our proposed model outperformed both of these models in a wide range of latency.

The difference between wait-k and the models using the full-sentence translation model was large in the quality-latency trade-off. Surprisingly, the fixed-size segmentation was also effective. When the segment size was fixed, it did not make a large
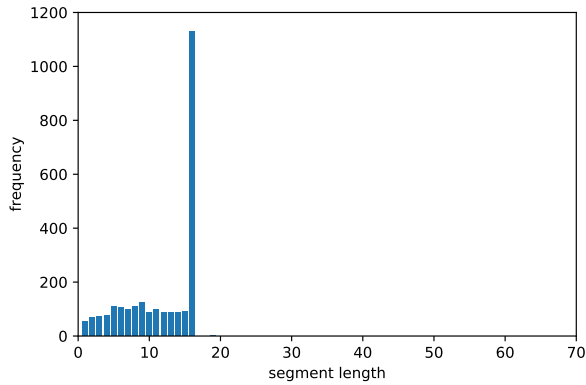
Figure 4: Segment length distribution of fixed-size segmentation with 16 subwords for AL 7.16 (En-Ja test)

Label	AL	BLEU
Fixed (16 subwords)	7.16	16.34
1 look-ahead MU	7.26	16.53
1 look-ahead ICLP (VP+S)	7.23	17.22

Table 4: BLEU results for AL close to 7

difference in the result, regardless of whether the unit was a subword or a word.

5 Analysis

5.1 Length ratio

Figure 3 shows the length ratios of translation hypotheses and references with different latency parameters. Too large a ratio decreases the BLEU score and makes the content delivery difficult both in text (subtitles) and speech (text-to-speech).

The length ratio of wait-k was unstable compared to other models because it was trained individually for each k.

Except for wait-k, the length ratios were large in the range of small latency, probably due to the condition mismatch between training and inference. These NMT models were trained on full sentences, but they were used to translate short segments in the inference. Therefore, they tend to output longer segment translations than expected. Their ratios gradually decrease as AL increases and the length of segments becomes closer to the length of full sentences.

5.2 Segment length distribution

Figures 4,5 and 6 show the distributions of source segment length in the En-Ja test set for which AL is close to 7.2. Table 4 shows their corresponding AL and BLEU of each model. The length was
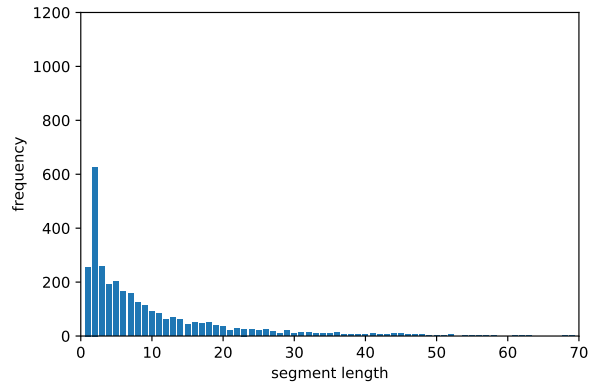

Figure 5: Segment length distribution of one look-ahead Meaningful Unit for AL 7.26 (En-Ja test)
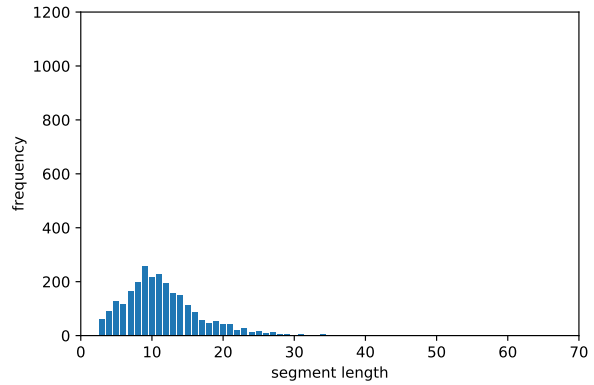

Figure 6: Segment length distribution of one look-ahead ICLP model dividing with label S and VP for AL 7.23 (En-Ja test)

calculated as the number of subwords in a segment, and the previous segment was concatenated to the next segment when the previous segment has no translation output.

Segmentation with fixed size 16 has some segments shorter than size 16 because the sentence length is not always a multiple of 16.

Compared with ICLP model, Meaningful Unit has wider distribution, and the most segments consist of two subwords. These short segments have less context information and can output longer segment translation than expected. This would be one of the reason why our proposed method outperformed Meaningful Unit.

5.3 Controlling latency

In Figures 7 and 8, each plot is labeled by the corresponding value of the hyperparameter of inference. It is difficult to control latency for Meaningful Unit as shown in the figure. BLEU scores of hyperpa-
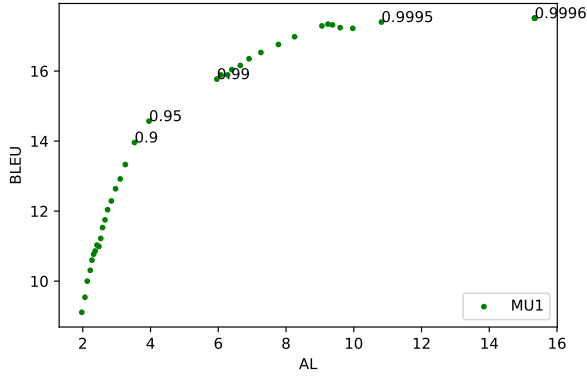
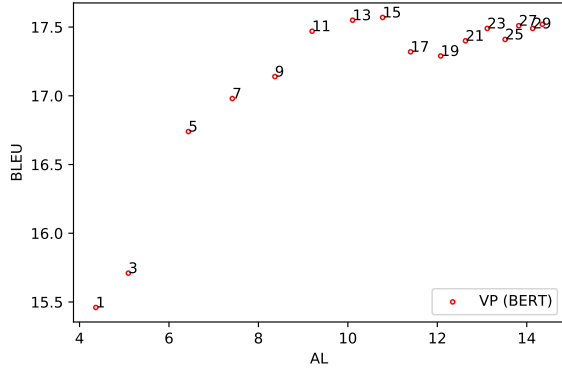Figure 7: BLEU and AL with different chunk segmentation thresholds for Meaningful Unit

Figure 8: BLEU and AL with different chunk size thresholds for the proposed method

Label	Precision	Recall	F1
NP	0.90	0.94	0.92
VP	0.89	0.97	0.93
NN	0.95	0.97	0.96
,	0.98	1.00	0.99
PP	0.85	0.93	0.89
S	0.87	0.52	0.65

Table 5: Results of label prediction (BERT)

Label	Precision	Recall	F1
NP	0.85	0.89	0.87
VP	0.91	0.94	0.92
NN	0.93	0.92	0.92
,	0.98	1.00	0.99
PP	0.78	0.94	0.86
S	0.84	0.52	0.64

Table 6: Results of label prediction (LSTM)

Label	Precision	Recall	F1
NP	0.62	0.85	0.72
VP	0.75	0.80	0.78
NN	0.60	0.78	0.68
,	0.41	0.34	0.37
PP	0.50	0.47	0.48
S	0.77	0.62	0.69

Table 7: Results of label prediction (BERT) without look-ahead

rameters from 0.9996 to 0.9999 were also the same as that of a full-sentence translation model.

In contrast, our proposed method can easily control latency because it uses the minimum chunk length as an intuitive hyperparameter to adjust it.

5.4 How many words to wait

Compared with the fixed-size segmentation model, our proposed model and Meaningful Unit have a disadvantage in AL, which is caused by the look-ahead approach. Despite this disadvantage, our proposed approach outperformed the fixed-size segmentation in a wide range of AL. This means the benefit of looking at the future words and finding a better boundary outweighed the above disadvantage.

5.5 Performance of ICLP

Tables 5 and 6 show the results in precision and recall of the one look-ahead ICLP models. The LSTM-based ICLP was better in precision, but the

BERT-based ICLP was better in recall for VP. Figure 9 compares them in the downstream simultaneous translation. The lines connected by dots nearly overlapped, so there was no large difference in BLEU score. LSTM is more efficient than BERT in incremental processes, so it is suitable for practical usage.

Table 7 shows the results by the ICLP model without one look-ahead approach. Compared with Table 5, the scores are much lower. One look-ahead approach was important to improve its performance.

5.6 En-De translation

We conducted additional experiments in En-De translation to investigate the performance in a different language. German is another language with different word order from English especially in verbs and also suffers from the reordering problem. Figure 10 shows the results. This is almost the opposite of the results of the En-Ja translation. The
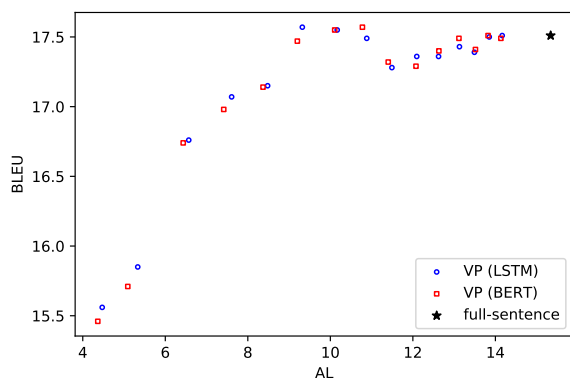
1131

Figure 9: Comparison between the use of LSTM- and BERT-based ICLP
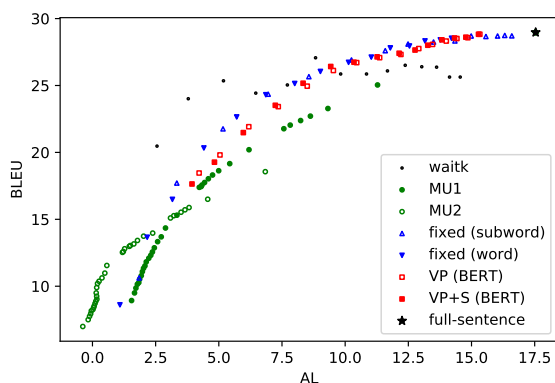

Figure 10: Scatter plot of BLEU and AL (En-De)

proposed boundary decision rules used for En-Ja translation were not so effective for En-De translation, so we need to find other rules to detect boundaries in En-De translation.

6 Conclusion

We proposed a novel segmentation method for simultaneous translation that uses simple rules and ICLP. Our proposed method is simple, and it outperformed the baselines in the quality-latency trade-off in En-Ja translation. On the other hand, the proposed method did not work effectively in En-De translation due to the smaller word order differences than those in En-Ja translation.

In future work, we expect to extract segmentation rules automatically and apply these rules to other language pairs as well.

7 Acknowledgements

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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George Foster. 2020. Re-translation versus streaming for simultaneous translation. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 220–227, Online. 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. 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. 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.

Sepp Hochreiter and J¨urgen Schmidhuber. 1997. *Long short-term memory*. Neural Computation.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010. Head finalization: A simple reordering rule for SOV languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 244–251, Uppsala, Sweden. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Taku Kudo. 2005. Mecab : Yet another part-of-speech and morphological analyzer. `http://mecab.sourceforge.net/`.

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, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020a. SIMULEVAL: An evaluation toolkit for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 144–150, Online. Association for Computational Linguistics.

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

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Graham Neubig, Katsuhito Sudoh, Yusuke Oda, Kevin Duh, Hajime Tsukada, and Masaaki Nagata. 2014. The NAIST-NTT TED talk treebank. In *Proceedings of the 11th International Workshop on Spoken Language Translation (IWSLT)*, Lake Tahoe, USA.

Jan Niehues, Thai Son Nguyen, Eunah Cho, Thanh-Le Ha, Kevin Kilgour, Markus Müller, Matthias Sperber, Sebastian Stüker, and Alex Waibel. 2016. Dynamic transcription for low-latency speech translation. In *Interspeech 2016*, pages 2513–2517.

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)*, pages 551–556, Baltimore, Maryland. 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.

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

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 *Workshops of International Conference on Machine Learning*, page 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)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszko-reit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and andIllia Polosukhin. 2017. Attention is all you need. *CoRR*, page Vol.abs/1706.03762.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2020. 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.

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. 2019. 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.