Transformer-based Double-token Bidirectional Autoregressive Decoding in Neural Machine Translation

Kenji Imamura and **Eiichiro Sumita**
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289, Japan
{kenji.imamura,eiichiro.sumita}@nict.go.jp

Abstract

This paper presents a simple method that extends a standard Transformer-based autoregressive decoder, to speed up decoding. The proposed method generates a token from the head and tail of a sentence (two tokens in total) in each step. By simultaneously generating multiple tokens that rarely depend on each other, the decoding speed is increased while the degradation in translation quality is minimized. In our experiments, the proposed method increased the translation speed by around 113%–155% in comparison with a standard autoregressive decoder, while degrading the BLEU scores by no more than 1.03. It was faster than an iterative non-autoregressive decoder in many conditions.

1 Introduction

Most neural machine translation systems are based on an encoder–decoder architecture. Although there are some frameworks, such as recurrent neural network-based translation (Sutskever et al., 2014; Bahdanau et al., 2014) and Transformer-based translation (Vaswani et al., 2017), they employ autoregressive decoding for high-quality translation. However, autoregressive decoding requires a decoding time that depends on sentence length because it generates a single token in each step.

To solve this problem, non-autoregressive decoding, which generates all tokens in one step, has been proposed (Gu et al., 2017; Lee et al., 2018; Ghazvininejad et al., 2019). However, the translation quality of non-autoregressive decoding has not yet matched the quality of autoregressive decoding. To improve the quality, some methods, such as Mask-Predict (Ghazvininejad et al., 2019), apply non-autoregressive decoding iteratively. This is a trade-off between quality and speed because the decoding time depends on the number of iterations.

This paper presents an autoregressive decoder that simultaneously generates two tokens in each step, as an intermediate solution between autoregressive and non-autoregressive decoding. The proposed method is based on the Transformer decoder and generates a token from the head and tail of a sentence (two tokens in total) in each step. Although this is a simple extension of standard autoregressive decoding, it has some notable features as follows:

- By simultaneously generating multiple tokens that rarely depend on each other, our method increases decoding speed while minimizing the degradation of translation quality.
- Because it is an extension of standard autoregressive decoding, our proposed method inherits its merits.
  - All tokens can be learned in parallel in the training phase.
  - In contrast to non-autoregressive decoders, our method does not determine generation lengths in advance.

In the following sections, we first briefly review autoregressive and non-autoregressive decoding. We then explain the proposed method and evaluate it, from the viewpoints of translation quality and speed.

2 Related Work

2.1 Transformer-based Autoregressive Decoding

Autoregressive decoding infers a token y_t of the current timestep t from the sequence of generated tokens $\boldsymbol{y}_{<t}$.

$$\hat{y}_t = \mathrm{argmax}_{y_t} \Pr(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x}), \quad (1)$$

Proceedings of the 7th Workshop on Asian Translation, pages 50–57

where $\boldsymbol{x}$ denotes the sequence of source tokens. When a test phase of translation, the argmax operation is replaced with a beam search, and the highest-probability hypothesis is selected from multiple candidates.

Transformer-based decoding considers generated tokens as a context using the self-attention mechanism. Namely,

$$\Pr(y_t) = Y_t = \mathrm{Out}(h_t^L), \quad (2)$$

$$h_t^\ell = \mathrm{TFLayer}^\ell(\boldsymbol{h}_{\leq t}^{\ell-1}, \boldsymbol{h}_{enc}), \quad (3)$$

$$h_t^0 = \mathrm{Emb}(y_{t-1}, t), \quad (4)$$

$$\boldsymbol{h}_{enc} = \mathrm{Encoder}(\boldsymbol{x}), \quad (5)$$

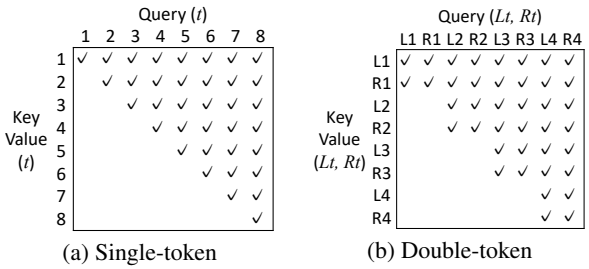
where Y_t denotes the posterior probability distribution of the token y_t. The Out, TFLayer, Emb, and Encoder functions denote the mapping from a hidden state h to the probability distribution, the function of a Transformer layer (L is the number of layers), the function that computes the word and positional embeddings, and the encoder function, respectively.

Decoding has a direction. Generation from the head of a sentence to the tail is called left-to-right (L2R) decoding, and generation in the opposite direction is called right-to-left (R2L) decoding. In L2R decoding, the decoder only refers to the left context. The decoding finishes when the end-of-sentence (EOS) token is generated.

In Equation 3, the system requires the previous states $\boldsymbol{h}_{<t}^{\ell-1}$, to compute the current state h_t^ℓ. However, the previous states have already been computed while predicting the previous tokens. Therefore, only the state $h_t^{\ell-1}$ needs to be computed if we preserve the previous states. We call this *inner state preservation* in this paper.

When training, all tokens can be learned in parallel using a mask of the triangular matrix, which restricts the tokens for the self-attention mechanism (Figure 1(a)).

Another strategy for speeding up autoregressive decoding is to substitute the self-attention mechanism with the other units. The Average Attention Network (AAN) (Zhang et al., 2018; Junczys-Dowmunt et al., 2018) and Simpler Simple Recurrent Unit (SSRU) (Kim et al., 2019) are faster than the self-attention mechanism because they depend only on the last state. Similar to our method, which is described in this paper, Zhou et al. (2019) proposed a model that simultaneously decodes two tokens from the head and tail of a sentence. They modified the self-attention mechanism to mix two contexts that are output from the forward and backward mechanisms.



(a) Single-token (b) Double-token

Figure 1: Examples of self-attention masks for training. Queries only refer to key–value pairs of the checked timesteps.

2.2 Non-autoregressive Decoding

Non-autoregressive decoding generates all tokens simultaneously, utilizing the parallelism of Transformer (Gu et al., 2017; Lee et al., 2018; Ghazvininejad et al., 2019). For example, the Mask-Predict method (Ghazvininejad et al., 2019) recovers masked tokens (`[mask]`) using the left and right contexts, like BERT encoders (Devlin et al., 2019). The initial tokens are all masks. Because of the parallel generation, the generation lengths must be determined in advance. The Mask-Predict method predicts these lengths from the encoder output.

Although non-autoregressive decoding performs fast generation, the translation quality in one step is relatively low. We can improve the quality by iteratively applying the parallel decoding. However, iterative decoding causes the following problems.

- The decoding speed reduces as the number of iterations increases. This is a trade-off between quality and speed.
- Iterative non-autoregressive decoding must recompute all states because it refers to whole contexts in a sentence, in contrast to autoregressive decoding, which can utilize inner state preservation.

Because of the above problems, high-quality and fast non-autoregressive decoding, which outperforms autoregressive decoding, has not yet been realized.

3 Proposed Method

3.1 Double-token Bidirectional Decoding

The proposed method of decoding combines L2R and R2L autoregressive decoding. That is, it generates each token from the head and tail of a sentence. This approach aims to realize fast decoding while maintaining translation quality by simultaneously generating multiple tokens that are adjacent to the fixed tokens but rarely depend on each other (that is, they are almost mutually independent).

Concretely, Equation 3 is replaced by the following equation.

$$(h^{\ell}_{Lt}, h^{\ell}_{Rt}) = \mathrm{TFLayer}^{\ell}(\boldsymbol{h}^{\ell-1}_{\leq Lt}, \boldsymbol{h}^{\ell-1}_{\leq Rt}, \boldsymbol{h}_{enc}), \quad (6)$$

where h^{ℓ}_{Lt} and h^{ℓ}_{Rt} denote the left and right outputs from layer ℓ at timestep t, respectively. Similarly, $\boldsymbol{h}^{\ell-1}_{\leq Lt}$ and $\boldsymbol{h}^{\ell-1}_{\leq Rt}$ denote the left and right contexts output from layer $\ell - 1$.

The decoding starts from the initial tokens, $y_{L0} = \mathrm{BOS}$ and $y_{R0} = \mathrm{EOS}$, and sequentially generates a pair of left and right tokens (Figure 2). Therefore, the decoder generates pairs of tokens from left to right, although each of the two tokens has a different meaning.

The decoding stops when a pair contains the end-of-decoding (EOD) token (Gu et al., 2019). At this time, the L2R and R2L tokens are separated from the sequence of pairs, and the final sentence is output after reordering them. During the decoding process, inner state preservation is also applied.

In this method, sinusoidal positional embedding (Vaswani et al., 2017) is not suitable because the tokens in each pair are not contiguous in the actual order. We use a learned positional embedding, which refers to learned parameters similar to the word embedding.

The difference from Zhou et al. (2019)'s method is that their method requires the model (specifically, the self-attention mechanism) to be modified. On the contrary, our method mainly realizes double-token decoding in the preprocessing and postprocessing phases and the modified beam search, except for the positional embedding. Therefore, another speedup method involving model modification, such as the AAN (Zhang et al., 2018), can be applied to our method.

3.2 Implementation

Algorithm 1 Beam search with double-token bidirectional decoding

Input: encoder output $\boldsymbol{h}_{enc}$, beam width W, number of outputs K
Output: K-best translation hypotheses H
B_t: beam of timestep t
Y_t: probability distribution according to the beam
(y_{Lt}, y_{Rt}): output token pair
1: $B_0 \leftarrow \{(y_{L0} = \mathrm{BOS}, y_{R0} = \mathrm{EOS})\}$
2: $H \leftarrow \emptyset$
3: $t \leftarrow 1$
4: **while** $|H| < K$ **do**
5: $B_t \leftarrow \emptyset$
6: $(Y_{Lt}, Y_{Rt}) \leftarrow \textsc{DecodeBeam}(B_{t-1}, \boldsymbol{h}_{enc})$
7: $(y_{Lt}, y_{Rt})^{1..3W} \leftarrow \textsc{DoubleTopK}(Y_{Lt}, Y_{Rt}, W)$
8: **for** $i \leftarrow 1$ **to** W **do**
9: **if** $y^i_{Lt} = \mathrm{EOD}$ **or** $y^i_{Rt} = \mathrm{EOD}$ **then**
10: $H \leftarrow H \cup \textsc{Finalize}(y^i_{Lt}, y^i_{Rt}, B_{t-1})$
11: **end if**
12: **end for**
13: **for** $i \leftarrow 1$ **to** $3W$ **do**
14: **if** $y^i_{Lt} \neq \mathrm{EOD}$ **and** $y^i_{Rt} \neq \mathrm{EOD}$ **then**
15: $B_t \leftarrow B_t \cup \textsc{ExpandBeam}(y^i_{Lt}, y^i_{Rt}, B_{t-1})$
16: **end if**
17: **if** $|B_t| \geq W$ **then**
18: **break**
19: **end if**
20: **end for**
21: $t \leftarrow t + 1$
22: **end while**

Algorithm 1 shows the beam search performed by our method. The number of iterations of lines 4–22 is half of that of standard autoregressive decoding, and the speed is increased.

The DecodeBeam function on line 6 sequentially computes Equations 4, 6, and 2. The DoubleTopK function on line 7 obtains $3W$ pairs (y_{Lt}, y_{Rt}) from the probability distributions (Y_{Lt}, Y_{Rt}). The Finalize function on line 10 completes a sentence from generated tokens. The ExpandBeam function generates beams at timestep t from the selected tokens and the previous beam. The DecodeBeam and DoubleTopK functions are executed mainly on GPUs. The Finalize and ExpandBeam functions are executed mainly on CPUs, but partially on GPUs.

Our method assumes that two generated tokens are almost independent of each other. This means that, on line 7 of Algorithm 1, y_{Lt} and y_{Rt} can be selected independently. Using this assumption, the DoubleTopK function first selects $2W$ tokens for y_{Lt}, and then selects $3W$ tokens for y_{Rt} considering the left probabilities. The processing time is almost proportional to the beam width W. Notably, the DoubleTopK function obtains $3W$ candidates from the probability distributions. This is because at least W unfinished candidates, which

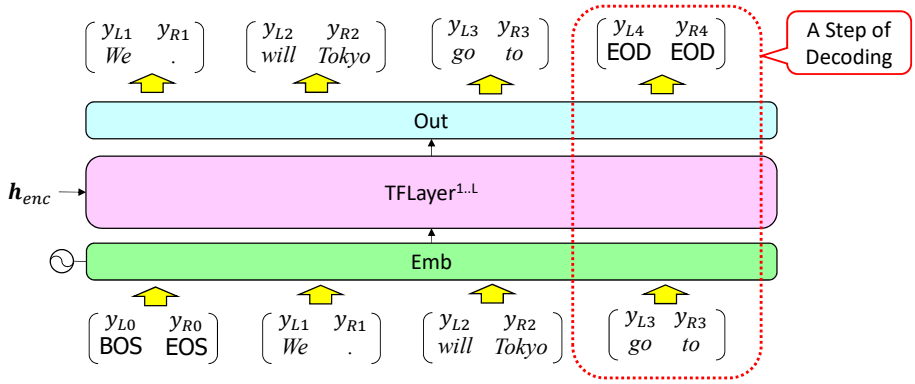


Figure 2: An example of double-token bidirectional decoding: "*We will go to Tokyo .*"

do not contain EODs, must remain to continue the search.

During training, the model is learned from data in which the order of tokens in target sentences is reconstructed as follows.

1. Divide the sequence of tokens into left and right halves, reverse the right half, and alternately fold the halves.
2. Supply one or two EOD tokens, to make the total number of tokens even.

The self-attention mask for training is a triangular matrix in which the unit is a pair of tokens (Figure 1(b)).

4 Experiments

4.1 Experimental Settings

Systems: We modified the fairseq translation system (Ott et al., 2019)[1] for the proposed method. For comparison, we considered the following three system types.

- Single-token (i.e., standard) autoregressive decoding. We used the original fairseq as the baseline. Both L2R and R2L directions were evaluated.
- Double-token unidirectional decoding. This method generates two contiguous tokens in each step, to evaluate the effect of bidirectional decoding.
- Mask-Predict (Ghazvininejad et al., 2019), which is one of the non-autoregressive decoding methods.[2] In our experiments, we did not apply knowledge distillation, to coordinate the setting with that of the other methods.[3]

Corpora: We used two corpora: the English–German (en-de) corpus of WMT-14 (4.5M sentences) (Bojar et al., 2014) and the Japanese–English (ja-en) corpus of ASPEC (3M sentences) (Nakazawa et al., 2016).

To make the evaluation stable, we concatenated all test sets in the corpora, except for validation sets. That is, we used 19,666 sentences (newstest2010-2016) for the WMT-14 corpus and 3,596 sentences (devtest and test) for the ASPEC corpus, as the test sets. The newstest2009 set in WMT-14 and the dev set in ASPEC were used as the validation sets.

All corpora were segmented into subwords (Sennrich et al., 2016). We used 37K shared vocabulary in WMT-14 and 16K vocabularies for the source and target languages in ASPEC.

Models and Hyperparameters: We used two model types: the Transformer base model (six layers, eight heads, 512 model dimensions, and 2,048 FFN dimensions) and the Transformer big model (six layers, 16 heads, 1,024 model dimensions, and 4,096 FFN dimensions).

Table 1 shows the details of the hyperparameters. All models were trained using almost the same settings, except for the learning rates and stopping criteria.

In the test phase, we used a beam width of 10 for autoregressive decoding. For Mask-Predict, we used a beam width of 5.[4] The mini-batch sizes

[1] https://github.com/pytorch/fairseq

[2] https://github.com/facebookresearch/Mask-Predict
This code is based on the fairseq translation system.

[3] We additionally applied knowledge distillation. As a result, the BLEU scores of all the methods, including Mask-Predict, were similarly improved.

[4] The beam width for the Mask-Predict method is usually

Type	Autoregressive	Mask-Predict
Model	Base model: 6 layers, 8 heads, 512 model dimensions, 2,048 FFN dimensions, dropout: 0.1	
	Big model: 6 layers, 16 heads, 1,024 model dimensions, 4,096 FFN dimensions, dropout: 0.3, attention_dropout: 0.1, layernorm_before	
Training	warmup: 5 epochs, annealing: inverse square-root, weight decay: 0.0001, clip norm: 5, loss function: labeled smoothed cross-entropy ($\epsilon = 0.1$), batch size: approx. 500 sentences, optimization: Adam ($\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-6}$), 10 best checkpoint averaging	
	learning rate: 0.0004, early stopping (10 epochs)	learning rate: 0.0001, 300,000 updates
Test	batch size: 32 sentences sorted by the source length, length penalty: 1.0, half-precision floating point computation	
	beam width: 10	beam width: 5

Table 1: Details of hyperparameters.

were all 32 sentences.

Evaluation Metrics: We used case-sensitive BLEU (Papineni et al., 2002) to evaluate translation quality. The MultEval tool was used for significance testing (Clark et al., 2011).[5] For the evaluation of speed, we measured translation time (which does not include loading and initialization) five times, and computed the average number of tokens translated per second. An NVIDIA V100 GPU was used during the evaluation.

4.2 Results

The results are shown in Table 2. The "Ratio" column of the table shows the speed ratio, compared with single-token L2R decoding.

The BLEU scores of the proposed method (double-token bidirectional) were slightly lower than those of single-token decoding. However, the differences were between 0.08 and 1.03 in the cases of both WMT-14 and ASPEC.

When we compare bidirectional (proposed) and unidirectional (L2R and R2L) double-token decoding, the BLEU scores of unidirectional decoding were lower than those of the proposed method. This phenomenon indicates that it is difficult to simultaneously generate two contiguous tokens because they depend on each other.

Focusing on speed, the proposed method was faster than single-token decoding. Speedups of 13%–24% for WMT-14 and 49%–55% for ASPEC were achieved. The speed of double-token unidirectional decoding was almost equal to that of the proposed method because they utilized the same algorithm and model structure. The speed of the Mask-Predict method differed dramatically depending on the number of iterations. In the case of four iterations, the speed of the proposed method was equal for the ASPEC corpus, but Mask-Predict was faster for the WMT-14 corpus.

Comparing the Transformer base and big models, the speed ratios of the big models were greater than those of the base models in the proposed method. This phenomenon will be discussed in Section 5.2.

5 Analysis

5.1 N-gram Precision Rates

A feature of our method is bidirectional decoding. To analyze this feature, we evaluated n-gram precision rates when the hypotheses were limited to a certain number of tokens from the head and tail.

Table 3 shows the results for the WMT-14 and ASPEC corpora, which were restricted to sentences over 30 tokens (8,251 sentences for WMT-14 and 1,320 sentences for ASPEC). When we evaluated left tokens, there were no great differences between the unigram precision rates of the baseline (single-token L2R) and the proposed method (double-token bidirectional). However, evaluating right tokens, the unigram precision rates of the proposed method were 0.4%–1.6% (WMT-14) and 1.1%–2.6% (ASPEC) higher than those of the baseline. These results demonstrate the effect of bidirectional decoding.

Despite these good results, the final BLEU score of the proposed method was less than that of the baseline. This is because the 4-gram precision of the left tokens was worse then the baseline, whereas that of the right tokens was better. The proposed method changes the original token order and connects the L2R and R2L hypotheses at the center of the sentence; these modifications have a detrimental effect on long n-grams, and will be the subject of future improvement.

5.2 Translation Speed for Settings

The speedup effect of the proposed method is influenced by the model and evaluation settings. In this section, we discuss the speedup effect from the viewpoints of model size, vocabulary size, and beam width.

small because it decisively generates a translation for a predicted length.

[5] https://github.com/jhclark/multeval

Method	WMT-14 en-de BLEU	WMT-14 en-de Speed Tokens/s (Ratio)	ASPEC ja-en BLEU	ASPEC ja-en Speed Tokens/s (Ratio)
Double-token bidir. (proposed)	25.62 †‡	2,251 (113%)	28.23	2,811 (149%)
Single-token L2R	26.45 ‡	1,997 (—)	28.65	1,891 (—)
Single-token R2L	25.79 †	2,004 (100%)	28.31	1,775 (94%)
Double-token L2R	24.55 †‡	2,349 (118%)	26.47 †‡	2,717 (144%)
Double-token R2L	24.43 †‡	2,327 (117%)	27.26 †‡	2,725 (144%)
Mask-Predict (10 iterations)	22.31 †‡	1,564 (78%)	25.05 †‡	1,476 (78%)
Mask-Predict (4 iterations)	19.62 †‡	2,910 (146%)	22.08 †‡	2,836 (150%)

(a) Transformer base models

Method	WMT-14 en-de BLEU	WMT-14 en-de Speed Tokens/s (Ratio)	ASPEC ja-en BLEU	ASPEC ja-en Speed Tokens/s (Ratio)
Double-token bidir. (proposed)	25.56 †‡	1,888 (124%)	27.50 ‡	2,241 (155%)
Single-token L2R	26.59 ‡	1,520 (—)	27.85	1,449 (—)
Single-token R2L	26.14 †	1,537 (101%)	28.23	1,376 (95%)
Double-token L2R	24.16 †‡	1,915 (126%)	26.17 †‡	2,229 (154%)
Double-token R2L	24.42 †‡	1,958 (129%)	26.57 †‡	2,208 (152%)
Mask-Predict (10 iterations)	23.47 †‡	1,092 (72%)	25.00 †‡	1,095 (76%)
Mask-Predict (4 iterations)	20.34 †‡	2,234 (147%)	22.30 †‡	2,060 (142%)

(b) Transformer big models

Table 2: Translation quality and speed of each method. The symbols † and ‡ indicate the scores that are significantly different from single-token L2R and R2L, respectively ($p < 0.05$).

Tokens Left	Tokens Right	1-gram precision Single-token L2R	1-gram precision Double-token bidir.	4-gram precision Single-token L2R	4-gram precision Double-token bidir.
∞	∞	59.8%	59.9% (+0.1%)	18.9%	18.2% (-0.7%)
5	0	53.8%	53.8% ($\pm$0.0%)	16.8%	16.2% (-0.6%)
10	0	49.1%	49.0% (-0.1%)	15.1%	14.5% (-0.6%)
15	0	45.1%	45.0% (-0.1%)	13.7%	13.2% (-0.5%)
0	5	62.9%	64.5% (+1.6%)	14.7%	15.3% (+0.6%)
0	10	62.2%	63.0% (+0.8%)	16.5%	16.8% (+0.3%)
0	15	61.8%	62.2% (+0.4%)	16.7%	16.6% (-0.1%)

(a) WMT-14 corpus

Tokens Left	Tokens Right	1-gram precision Single-token L2R	1-gram precision Double-token bidir.	4-gram precision Single-token L2R	4-gram precision Double-token bidir.
∞	∞	65.8%	66.2% (+0.4%)	18.5%	17.8% (-0.7%)
5	0	59.2%	59.4% (+0.2%)	16.2%	15.6% (-0.6%)
10	0	53.7%	53.9% (+0.2%)	14.5%	13.9% (-0.6%)
15	0	49.3%	49.5% (+0.2%)	13.1%	12.5% (-0.6%)
0	5	73.5%	76.1% (+2.6%)	17.5%	18.9% (+1.4%)
0	10	72.4%	74.6% (+2.2%)	17.3%	18.9% (+1.6%)
0	15	71.5%	72.6% (+1.1%)	17.3%	18.3% (+1.0%)

(b) ASPEC corpus

Table 3: Unigram and four-gram precision rates when the number of tokens was limited to n from the head and tail of the hypotheses (Transformer base models, over 30 tokens). Values in parentheses indicate differences between the single-token L2R and double-token bidirectional methods.

5.2.1 Model Size

In the experiments in Section 4, we evaluated translation speed using the Transformer base and big models. The absolute speed of the base models was greater than that of the big models, in all methods. From the perspective of the speed ratio, which compares the proposed method (double-token bidirectional) with the baseline (single-token L2R), the speedup effect of the big model was greater than that of the base model. For example, the speed ratio of the big model was 124% for the WMT-14 corpus, whereas that of the base

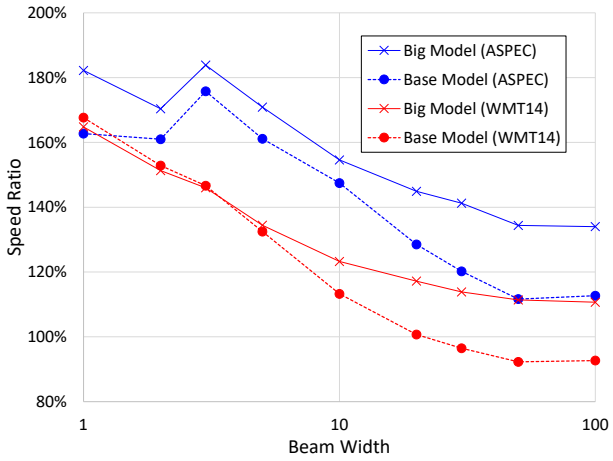


Figure 3: Variation of speed ratio with beam width.

model was 113%. This tendency was the same for the ASPEC corpus. We can conclude that the speedup effect of the proposed method is greater for larger models.

5.2.2 Vocabulary Size

Table 4 shows the translation speed of the base model as the vocabulary size was increased from 4K to 64K. The number of tokens per sentence is also shown in the table because it changes if we change the vocabulary based on subwords.

In addition to the translation speed decreasing, the speed ratio decreased as the vocabulary size increased. This means that the speedup effect of the proposed method was reduced as the vocabulary size increased.

We expected the speed ratio to increase as the vocabulary size increased, because the model size increased. However, the opposite result was actually observed. One of the possible reasons is that the sentence length (number of tokens) increased when the vocabulary size was small, and therefore the effect of the double-token decoding increased.

5.2.3 Beam Width

Figure 3 shows the speed ratio for each value of beam width. A greater ratio means that the proposed method is more effective. As a result, the speed ratio was greater when the beam width was small, in this experiment.

It is difficult to consistently explain the phenomena shown in this section because the processing times of the CPU and GPU are unknown even though the effectiveness must depend on them. However, we can summarize that the speedup effect of the proposed method is increased when we use 1) big models, 2) small vocabulary, and 3) small beam width.

6 Conclusions

This paper presented a bidirectional decoding method that simultaneously generates two tokens. The proposed method achieves fast decoding while minimizing quality degradation by generating tokens that rarely depend on each other. It is faster than both the standard autoregressive decoder and the Mask-Predict method in many conditions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA.

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.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. 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 6112–6121, Hong Kong, China.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *CoRR*, abs/1711.02281.

Vocabulary Size	WMT-14 en-de Length (tokens/sent.) en	de	Speed (tokens/s) Single-token L2R	Double-token bidir.	ASPEC ja-en Length (tokens/sent.) ja	en	Speed (tokens/s) Single-token L2R	Double-token bidir.
4K	52.85	58.10	2,444	3,959 (162%)	41.30	34.44	1,458	3,493 (240%)
8K	37.03	41.33	2,561	3,701 (144%)	31.79	30.17	1,527	3,500 (229%)
16K	30.91	34.22	2,474	3,244 (131%)	29.25	27.67	1,891	2,811 (149%)
32K	27.46	29.67	2,183	2,422 (111%)	28.32	26.43	1,404	2,209 (157%)
64K	25.58	26.85	1,721	1,599 (93%)	27.91	25.81	1,136	1,438 (127%)

Table 4: Variation of translation speed for vocabulary size.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order. *CoRR*, abs/1902.01370.

Marcin Junczys-Dowmunt, Kenneth Heafield, Hieu Hoang, Roman Grundkiewicz, and Anthony Aue. 2018. Marian: Cost-effective high-quality neural machine translation in C++. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 129–135, Melbourne, Australia.

Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. 2019. From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 280–288, Hong Kong.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2204–2208, Portorož, Slovenia.

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 NAACL-HLT 2019: Demonstrations*.

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 (ACL)*, pages 311–318, Philadelphia, Pennsylvania, USA.

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.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Melbourne, Australia.

Long Zhou, Jiajun Zhang, Chengqing Zong, and Heng Yu. 2019. Sequence generation: From both sides to the middle. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5471–5477.