Infusing Sequential Information into Conditional Masked Translation Model with Self-Review Mechanism

Pan Xie[*1]   **Zhi Cui**[2]   **Xiuying Chen**[3]   **Xiaohui Hu**[4]   **Jianwei Cui**[2]   **Bin Wang**[2]

[1]Beihang University
[2]Xiaomi AI Lab
[3]Peking University
[4]Chinese Academy of Sciences

panxie@buaa.edu.cn   xy-chen@pku.edu.cn   hxh@iscas.ac.cn
{cuizhi,cuijianwei,wangbin11}@xiaomi.com

Abstract

Non-autoregressive models generate target words in a parallel way, which achieve a faster decoding speed but at the sacrifice of translation accuracy. To remedy a flawed translation by non-autoregressive models, a promising approach is to train a conditional masked translation model (CMTM), and refine the generated results within several iterations. Unfortunately, such approach hardly considers the *sequential dependency* among target words, which inevitably results in a translation degradation. Hence, instead of solely training a Transformer-based CMTM, we propose a Self-Review Mechanism to infuse sequential information into it. Concretely, we insert a left-to-right mask to the same decoder of CMTM, and then induce it to autoregressively review whether each generated word from CMTM is supposed to be replaced or kept. The experimental results (WMT14 En↔De and WMT16 En↔Ro) demonstrate that our model uses dramatically less training computations than the typical CMTM, as well as outperforms several state-of-the-art non-autoregressive models by over 1 BLEU. Through knowledge distillation, our model even surpasses a typical left-to-right Transformer model, while significantly speeding up decoding.

1   Introduction

Neural Machine Translation (NMT) models have achieved a great success in recent years (Sutskever et al., 2014; Bahdanau et al., 2015; Cho et al., 2014; Kalchbrenner et al., 2016; Gehring et al., 2017; Vaswani et al., 2017). Typically, NMTs use autoregressive decoders, where the words are generated one-by-one. However, due to the left-to-right dependency, this computationally-intensive decoding process cannot be easily parallelized, and therefore causes a large latency (Gu et al., 2018).

To break the bottleneck of autoregression, several non-autoregressive models have been proposed to induce the decoder to generate all target words simultaneously (Gu et al., 2018; Łukasz Kaiser et al., 2018; Li et al., 2019; Ma et al., 2019). Despite the acceleration of computation efficiency, these models usually suffers from the cost of translation accuracy. Even worse, they decode a target only in one shot, and thus miss a chance to remedy a flawed translation. Against them, a promising research line is to perform refinement on the generated result within several iterations (Lee et al., 2018; Ghazvininejad et al., 2019).

Along this line, Ghazvininejad et al. (2019) propose a Mask-Predict decoding strategy, which iteratively refines the generated translation given the most confident target words predicted from the previous iteration. This model is trained using an objective of conditional masked translation modeling (CMTM), by predicting the masked words conditioned on the rest of observed words. However, CMTM just learns from a subset of words instead of the entire target in terms of a training step. As a result, it will iterate more times over the training dataset to explore the contextual relationship within a sentence, and thus will struggle with a huge cost of the whole training time (Clark et al., 2020). Most importantly, CMTM extensively bases upon the assumption of conditional independence, making it hard to capture the strong correlation across the adjacent words (Gu et al., 2018). Inevitably, this issue will still degrade the translation performance, such as outputting repetitive words (Wang et al., 2019).

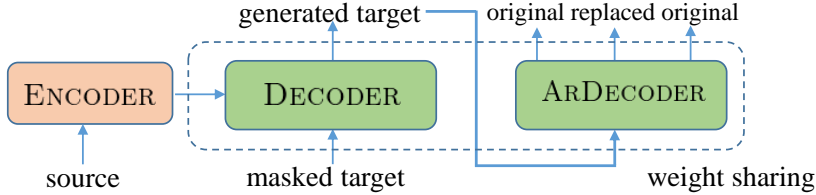[*]The work was done as an intern at Xiaomi.

Figure 1: A simplified architecture of our model based on Transformer. We tie the weights of DECODER and ARDECODER. The inter attention between ENCODER and ARDECODER is omitted for readability.

To address the issues, our idea is to infuse sequential information into CMTM. Accordingly, we propose a Self-Review Mechanism, a discriminative task in which the model learns to autoregressively distinguish the ground truth target from the non-autoregressively generated output of itself. As shown by ARDECODER (short for Autoregressive Decoder) in Figure 1, we firstly switch on the autoregressive mode of the same DECODER with CMTM by inserting a left-to-right mask. More importantly, we then require this ARDECODER to recurrently review whether each generated word from CMTM is supposed to be a replacement or just an original. In this way, this mechanism constrains our model to review each predicted word only based on previous ones, which is able to not only correct the prediction errors, but also facilitates the learning of conditional dependence of the target words. Moreover, this mechanism could also help speed up the whole training by learning from all target words rather than a small masked subset.

We extensively validate our model on the datasets of WMT14 En↔De and WMT16 En↔Ro. The experimental results demonstrate that our model outperforms several state-of-the-art non-autoregressive models by over 1 BLEU. Through knowledge distillation, our model even achieves competitive performance compared with the typical left-to-right Transformer, while significantly reducing the cost of time during inference. Meanwhile, we also prove that the training speed of our model is much faster than the typical CMLM.

2 Background

2.1 Autoregressive NMT

Given a source sentence $\mathbf{x} = \{x_1, x_2, ..., x_{|\mathbf{x}|}\}$, a NMT model is aimed to generate a sentence in target language $\mathbf{y} = \{y_1, y_2, ..., y_{|\mathbf{y}|}\}$ with identical semantics expressed, where $|\mathbf{x}|$ and $|\mathbf{y}|$ are denoted as the length of source and target sentence, respectively. Typically, the training objective of an autogressive NMT model is expressed as a chain of conditional probabilities in a left-to-right manner:

$$\log p_{at}(\mathbf{y}|\mathbf{x}) = \sum_{t=1}^{|\mathbf{y}|+1} \log p(y_t|y_{0:t-1}, \mathbf{x}) \tag{1}$$

where y_0 and $y_{|\mathbf{y}|+1}$ are <SOS> and <EOS>, standing for the start and end of a sentence, respectively. Usually, these probabilities are parameterized using a standard encoder-decoder architecture (Sutskever et al., 2014), where the decoders use autoregressive strategy to capture the left-to-right dependency among the target words.

2.2 Conditional Masked Translation Model

Different from the training objective in Equation 1, we adopt conditional masked translation modeling (CMTM) (Ghazvininejad et al., 2019) to optimize our proposed non-autoregressive NMT model. During training, our model is aimed to predict a set of masked target words $\mathbf{y}_{mask}$ given an source input $\mathbf{x}$ and a set of observed target words $\mathbf{y}_{obs}$. Note that $|\mathbf{y}| = |\mathbf{y}_{mask}| + |\mathbf{y}_{obs}|$. Based on the assumption that the words of $\mathbf{y}_{mask}$ are independent, the training objective of CMTM is formulated as:
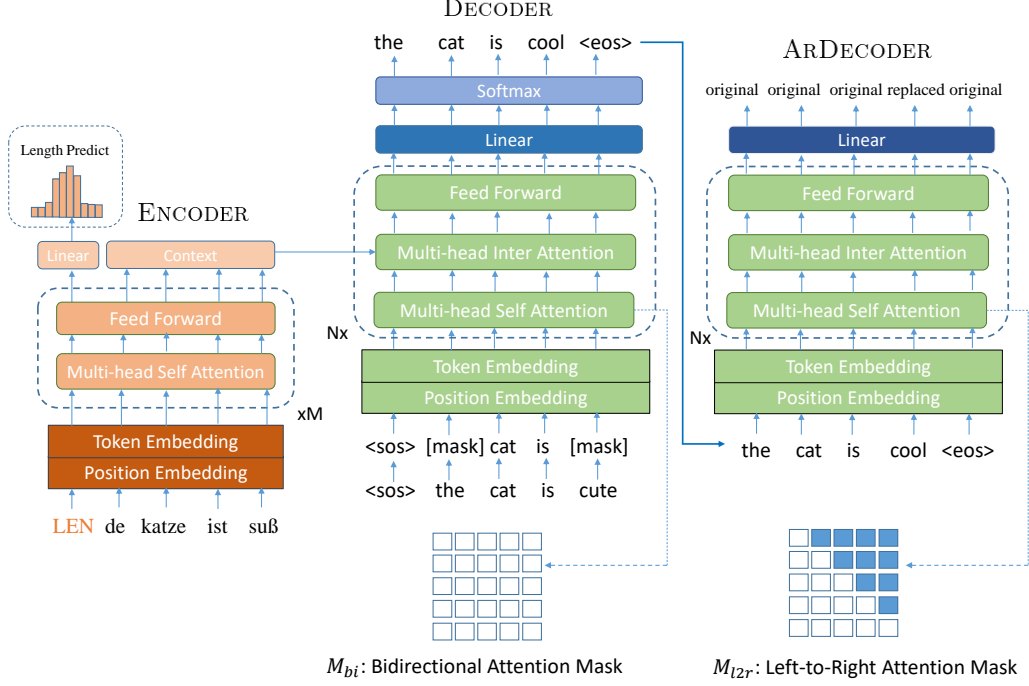
Figure 2: The architecture of our model, which includes an ENCODER, a DECODER and a ARDECODER. During inference, we only perform iterative decoding based on ENCODER-DECODER. The weights are shared across the DECODER and ARDECODER as denoted by the green blocks.

$$\log p_{nat}(\mathbf{y}_{mask}|\mathbf{x}, \mathbf{y}_{obs}) = \sum_{t=1}^{|\mathbf{y}_{mask}|} \log p(y_{mask}^t|\mathbf{x}, \mathbf{y}_{obs}) \tag{2}$$

where the masked words in $\mathbf{y}_{mask}$ are randomly selected and denoted by a special token [mask].

3 Approach

3.1 Model Architecture

Figure 2 illustrates the overall architecture of our proposed model, which is composed of three modules, an ENCODER, a DECODER and an ARDECODER. Notably, ARDECODER is obtained by solely adding a left-to-right mask to DECODER, where their weights are tied. Rather than a pure CMTM, we also propose a Self-Review Mechanism to ask the ARDECODER to review the predicted target from DECODER in a left-to-right manner. In this section, we will detail each module and the Self-Review Mechanism.

Encoder Our ENCODER is identical to the standard Transformer (Vaswani et al., 2017). Built upon self attention, it encodes a source input $\mathbf{x}$ into a series of contextual representations $\mathbf{H}_{enc} = \{h_{enc}^1, h_{enc}^2, ..., h_{enc}^{|\mathbf{x}|}\}$ by:

$$\mathbf{H}_{enc} = \{h_{enc}^1, h_{enc}^2, ..., h_{enc}^{|\mathbf{x}|}\} = \text{encode}(\mathbf{x}) \tag{3}$$

Decoder The non-autoregressive property of our model mainly lies in our DECODER. Different from ENCODER, the DECODER has two sets of attention heads as shown in Figure 2: the inner heads are attending over the target words, and the inter heads are over the hidden outputs of ENCODER. It is worth noting that we use a bidirectional mask (denoted as M_{bi}) as shown in the middle of Figure 2. Such mask allows the DECODER to use both left and right contexts to predict each target word, ensuring that the prediction for t-th position can depend not only on the information before t-th but also right after t-th.

17

Our DECODER is optimized using the objective in Equation 2. Given a source input $\mathbf{x}$ and part of observed target words $\mathbf{y}_{obs}$, the DECODER is required to predict those words of $\mathbf{y}_{mask}$. Firstly, we obtain a series of DECODER hidden outputs $\mathbf{H}_{dec} = \{h_{dec}^1, h_{dec}^2, ..., h_{dec}^{|\mathbf{y}|}\}$, by feeding DECODER with the observed target $\mathbf{y}_{obs}$ and the ENCODER outputs $\mathbf{H}_{enc}$. Mathematically, we parameterize $\mathbf{H}_{dec}$ as:

$$\mathbf{H}_{dec} = \{h_{dec}^1, h_{dec}^2, ..., h_{dec}^{|\mathbf{y}|}\} = \text{decode}(\mathbf{H}_{enc}, \mathbf{y}_{obs}; M_{bi}) \tag{4}$$

Then, we apply a linear projection on the hidden outputs $\mathbf{H}_{dec}$, and obtain the probabilities of target words using softmax. Notably, we only focus on the probabilities of masked words during training. Therefore, the probability $p(y_{mask}^t|\mathbf{x}, \mathbf{y}_{obs})$ in Equation 2 is parameterized by:

$$p(y_{mask}^t|\mathbf{x}, \mathbf{y}_{obs}) = \text{softmax}(W_1^T h_{dec}^t) \tag{5}$$

ARDECODER   The ARDECODER is introduced to serve as a discriminator of adversarial models (Goodfellow et al., 2014), and will play an important role in our Self-Review Mechanism. As shown in Figure 2, ARDECODER is obtained by adding a left-to-right mask (denoted as M_{l2r}) to the DECODER. This mask prevents ARDECODER from attending future words when reading the predicted target from DECODER, ensuring that the prediction for t-th position can only rely on the known outputs before t-th.

Unlike the aim of DECODER, ARDECODER is asked to review the predicted sentence from DECODER, and distinguish whether each word is supposed to be replaced or not. Notably, we tie the weights of ARDECODER and DECODER, to ensure that our DECODER can take advantage of the sequential information learned from this discriminative task.

3.2   Self-Review Mechanism

As discussed previously, solely a CMTM is insufficient to capture the sequential dependency of target words, and thus still inevitably results in a disappointing translation. To remedy the issue, the core of our work is how to better *infuse* sequential information into the model.

Concretely, we propose a Self-Review Mechanism for CMTM to learn strong correlations among the target words. During training, ENCODER-DECODER firstly predicts a target $\hat{\mathbf{y}}$ given a gold observed target $\mathbf{y}_{obs}$ as well as an input source $\mathbf{x}$. Then, the ARDECODER is asked to review the predicted target $\hat{\mathbf{y}}$, and distinguish whether a predicted word $\hat{y}_t$ is supposed to be replaced by the ground truth y_t as:

$$p(\hat{y}_t = y_t|\hat{y}_{1:t-1}, \mathbf{x}) = \sigma(W_2^T h_{rev}^t) \tag{6}$$

$$\mathbf{H}_{rev} = \{h_{rev}^1, h_{rev}^2, ..., h_{rev}^{|\mathbf{y}|}\} = \text{review}(\mathbf{H}_{enc}, \hat{\mathbf{y}}; M_{l2r}) \tag{7}$$

where $\sigma(\cdot)$ is a sigmoid function. Finally, the objective of the self-reviewing becomes:

$$\mathcal{L}_{rev} = -\sum_{t=1}^{|\mathbf{y}|}((\hat{y}_t = y_t)\log(\sigma(W_2^T h_{rev}^t)) + (\hat{y}_t \neq y_t)\log(1 - \sigma(W_2^T h_{rev}^t))) \tag{8}$$

Here, we do not back-propagate the learning errors from ARDECODER to ENCODER-DECODER due to the difficulty of applying adversarial learning to text (Caccia et al., 2018). By adding up $\mathcal{L}_{rev}$, our model sees the entire target sentence rather than a small subset of words in terms of a training step, and thus it does not need to iterate more times to explore the contextual relationships among the words, which is beneficial to speeding up the whole training compared with a pure CMTM (Clark et al., 2020). Besides, our work can also be regarded as a multi-task learning, where we enhance our DECODER with the bidirectional contextual information as well as the left-to-right correlations of target words.

3.3 Length Prediction

Typically, an autoregressive NMT model generates the target sentence word-by-word, and thus it decides the length of target sentence by encountering a special token <EOS>. However, our model adopts the strategy of non-autoregressive decoding, namely, it predicts the entire target sentence in a parallel way. Following (Devlin et al., 2019; Ghazvininejad et al., 2019), we add a special token <LEN> to the begining of source input. In this sense, our ENCODER is also required to predict the length of target sentence L, i.e., predict a token from $[1, N]$ given the source input $\mathbf{x}$, where N is the maximum length of target sentences in our corpus. Mathematically, we define the loss of length prediction as:

$$\mathcal{L}_{len} = \sum_{i}^{N} -(L = i) \log p(L|\mathbf{x}) \tag{9}$$

3.4 Optimization and Inference

Overall, the whole model is jointly trained by minimizing the total loss $\mathcal{L}$, which is a combination of Equation 2,8,9:

$$\mathcal{L} = \mathcal{L}_{dec} + \mathcal{L}_{len} + \mathcal{L}_{rev} \tag{10}$$

where $\mathcal{L}_{dec} = -\log p_{nat}(\mathbf{y}_{mask}|\mathbf{x}, \mathbf{y}_{obs})$.

During inference, we abandon ARDECODER and perform iterative refinement only based on ENCODER-DECODER. Following Mask-Predict (Ghazvininejad et al., 2019), we generate a raw sequence starting with an entirely masked target given a new input source. Upon this raw sequence, we conduct refinement by masking-out and re-predicting a subset of words whose probabilities are under a threshold. This refinement is repeated within a heuristic number of iterations. For more details, please refer to (Ghazvininejad et al., 2019).

4 Experiments

4.1 Setting

Datasets   We conduct experiments on two benchmark datasets, WMT14 En↔De (4.5M sentence pairs) and WMT16 En↔Ro (610k pairs). After preprocessing the two datasets following (Lee et al., 2018), we tokenize them into subword units using BPE (Sennrich et al., 2016). We use newstest-2013 and newstest-2014 as our development and test datasets for WMT14 En↔De, while use newsdev-2016 and newstest-2016 as our development and test datasets for WMT16 En↔Ro.

Evaluation Metrics   We adopt the widely-used **BLEU**[1] (Papineni et al., 2002) to evaluate the translation accuracy. To compare the training speed, we also use **Floating-Point Operations per second** (FLOPs)[2] to measure the computational complexity.

Implementation Details   We follow the base configuration of Transformer (Vaswani et al., 2017): The dimension of model is set to 512, and the dimension of inner layers is set to 2048. The ENCODER is consisted of a stack of 6 layers , as well as the DECODER and ARDECODER. The weights of our model are all randomly initialized with a uniform distribution $\mathcal{N}(0, 0.02)$. Besides, we set the parameters of layer normalization as $\beta = 0, \gamma = 1$. We use Adam optimizer (Kingma and Ba, 2015) with 98k tokens per batch. We increase the learning rate from 0 to 5e-4 within the first 10,000 warmup steps, and gradually decay it with respect to the inverse square root of training steps. Note that we share the weights of DECODER and ARDECODER only except the output layer ($W_1 \neq W_2$ in Equation 5 and 6). During inference, we set length candidates as 5 for non-autoregressive decoding, where the max length N is defined as 10,000. The number of iteration for refinement is set as 10. To compare with autoregressive models, we adopt a beam width of 5 for beam search decoding. The training speed is measured on 8 NVIDIA Tesla P100 GPUs and decoding speed is just on one.

---

[1]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl
[2]https://github.com/google-research/electra/blob/master/flops_computation.py

Knowledge Distillation Previous works on non-autoregressive NMT models have proved that knowledge distillation can substantially improve the performance (Gu et al., 2018; Lee et al., 2018; Stern et al., 2019; Zhou et al., 2020). Commonly, a student model is trained on a distilled dataset which is generated by a teacher model, where the teacher model usually adopts a much larger configuration of parameters than its student. Different from this common setting, we will investigate if it is still useful to tie the configuration of the teacher and its student model. We train our model on a distilled corpus (EN↔DE and EN↔RO), where the distilled target are generated by a typcial left-to-right Transformer with a base configuration. In the followings, we will identify the effect of knowledge distillation to our model.

4.2 Baselines

To demonstrate the effectiveness of our work, we compare with several state-of-the-art NMT models:

Seq2Seq (Bahdanau et al., 2015): It is a LSTM-based sequence-to-sequence model, where the decoder adopts beam search strategy.

ConvS2S (Gehring et al., 2017): It is a convolution-based sequence-to-sequence model, and it decodes the target words in a left-to-right manner.

Transformer (Vaswani et al., 2017): It is a state-of-the-art autoregressive model, and it adopts beam search decoding to generate target translation.

FTNAT (Gu et al., 2018): It is a non-autogressive Transformer model using fertitilies, and adopts noisy parallel decoding (NPD) to generate target translation.

FlowSeq (Ma et al., 2019): It is also a non-autogressive model, which introduces a latent variable to model the generative flow. During inference, it generates a target translation using argmax decoding.

HintNAT (Li et al., 2019): It is also a non-autoregressive model, which leverags alignments and hidden states of a teacher autoregressive model.

IRNAT (Lee et al., 2018): It is a non-autogressive model trained with a conditional denoising autoencoder. During inference, it iteratively devises the generated translation. We set the number of iterations as 10.

Mask-Predict (Ghazvininejad et al., 2019): It is a typical CMTM model. During inference, it adopts Mask-Predict on the translation within 10 iterations. By comparing with it comprehensively, we aim to examine the effectiveness of our proposed Self-Review Mechanism.

4.3 Comparison Against Baselines

The experimental results are summarized in Table 1. We firstly examine the non-autoregressive models with different decoding strategies, i.e., one-shot decoding vs iterative decoding. As shown in Table 1, FTNAT, HintNAT and FlowSeq achieve the lowest score of BLEU. Such degradation is mainly due the problem of multimodality (Gu et al., 2018) that these models hardly considers the left-to-right dependency. Even worse, they do not have a chance to remedy the translations. The same thing happens to the first iteration of IRNAT, Mask-Predict and our model as well, where the results are similar to the one-shot decoding models. From this comparison, we can conclude that iterative decoding is an effective technique for non-autoregressive NMTs.

Although IRNAT and MaskPredict are able to turn the initial bad translation into a much better one through multiple iterations of decoding, there is still a gap of the translation accuracy when comparing against the SOTA autoregressive model, i.e., Transformer. Still, this deficiency is attributed to the lack of a mechanism or strategy to capture the strong correlations among the target words, which is also the root cause why non-autoregressive models are hard to generated satisfactory translation (Ren et al., 2020).

In contrast, our model, which is additionally optimized with our proposed Self-Review Mechanism, significantly achieves a performance boost over these non-autoregressive models. Meanwhile, our model has a huge lead in BLEU on the dataset of WMT 14 EN→DE compared with Seq2Seq and ConvS2s, and even accomplishes comparable performance with Transformer. More specifically, compared with Transformer, our model (w/o kd) achieves 34.54 (+0.26 gains) and 34.36 (+0.37 gains) of BLEU on WMT16 EN→RO and WMT16 RO→EN, respectively. Even with the help of knowledge distillation, our model outperforms Transformer on almost all the benchmark datasets except WMT16 EN→RO. More importantly, our model dramatically reduces the cost in decoding, which is at least 5.16x faster than Transformer. If we sacrifice a certain translation accuracy by reducing iteration number, we could obtain

Models	Iterations	WMT'14		WMT'16		Speedup
		EN→DE	DE→EN	EN→RO	RO→EN	
Autoregressive Models						
LSTM Seq2Seq	N	24.60	-	-	-	-
ConvS2S	N	26.42	-	-	-	-
Transformer	N	<u>27.30</u>	<u>31.09</u>	<u>34.28</u>	<u>33.99</u>	1.00x
Non-Autoregressive Models						
FTNAT(w/ kd)	1	19.17	23.20	29.79	31.44	2.36x
HintNAT(w/ kd)	1	25.20	29.52	-	-	17.80x
FlowSeq-large (w/ kd)	1	23.72	28.39	-	29.73	-
IRNAT(w/ kd)	1	13.91	16.77	24.45	25.73	-
	10	21.61	25.48	29.32	30.19	5.78x
Mask-Predict(w/ kd)	1	18.05	21.83	27.32	28.20	27.51x
	4	25.94	29.90	32.53	33.23	11.21x
	10	<u>27.03</u>	<u>30.53</u>	<u>33.08</u>	<u>33.31</u>	5.16x
Ours(w/o kd)	1	9.01	8.76	20.76	20.25	27.51x
	4	23.33	26.58	33.08	33.26	11.21x
	10	25.50	29.27	**34.54**	**34.36**	5.16x
Ours(w/ kd)	1	17.48	21.47	23.03	24.30	27.51x
	4	26.99	30.67	32.83	33.55	11.21x
	10	**27.97**	**31.59**	33.98	**34.34**	5.16x

Table 1: The BLEU scores of all models on the benchmark datasets, where "kd" is denoted as knowledge distillation. In the column of speedup, we adopt seconds/sentence to measure the decoding speed, where Transformer is set as the baseline (beam size = 5). We present the best BLEU scores of the baseline models reported in their original paper.

even higher decoding efficiency. In brief, this comparison results validate the effectiveness of Self-Review Mechanism.

4.4 Effect of Knowledge Distillation

The comparison results are listed in the last 6 rows of Table 1. In terms of the large-scale dataset, i.e., WMT14 EN↔De, our model with the knowledge distillation gains a remarkable improvement, especially at the early iterations. Under the same size of configuration, it is widely believed that that the autoregressive model is better that capturing the alignment relationship between a source-target pair (Gu et al., 2018), and thus the autoregressive model as a teacher model is able to reduce the redundant and irrelevant alignment "modes" in the raw corpus. In this way, our proposed model benefits from learning such kind of distilled dataset. However, the improvement is not concurrent on the small-scale dataset, i.e., WMT16 EN↔RO. At the end of 10th iteration, our model even has a decrease of BLEU on WMT16 EN↔RO. We conjecture that a small-scale dataset is statistically likely to contain less redundant "modes" than a large-scale dataset. As a result, distillation for a small-scale dataset might not be more beneficial for a student model compared with the a raw dataset, probably no matter how large the teacher model is. Therefore, it is useful and more efficient to adopt a teacher model with the same size of configuration as the student model for knowledge distillation on a large-scale dataset.

5 Ablation Study and Analysis

Upon CMLM, we additionally introduce a Self-Review Mechanism during training, whereas Mask-Predict (Ghazvininejad et al., 2019) is optimized with only the first two terms in Equation 10. During inference, we abandon ARDECODER, and our model performs decoding as same as Mask-Predict. In this section, we will compare closely to Mask-Predict to validate the contribution of our proposed Self-Review Mechanism.

5.1 Training Speed

To better understand the comparison of training speed between Mask-Predict and our model, we measure the FLOPS of one single step and the whole training steps as shown in Table 2 and Table 3, respectively. In terms of the time usage of one training step, Table 2 shows that Mask-Predict is about 1.6x faster than our model, since our model has to optimize ARDECODER together. However, such result of one training step cannot lead to a conclusion that it will take more time to train our model than Mask-Predict. Instead, the results from Table 3 illustrate that our model effectively speeds up the whole training especially on a large-scale dataset WMT14 EN↔DE (at least 5x faster). This discrepancy between one step and overall steps might be resulted from several reasons. We conjecture that our model is able to see whole target sentence, where the ARDECODER needs to review each word generated from DECODER. On the contrary, Mask-Predict only learns from a subset of masked words, and thus it has to take much more steps to discover the semantic relationships among the words.

Module	Encoder	Decoder	Reviewer
FLOPs	19332M	24017M	19468M
Model	**MaskPredict**	**Ours**	
FLOPs	43349M	72484M	

Table 2: The comparison of FLOPs per training step between Mask-Predict and Ours, where MaskPredict is just composed of ENCODER-DECODER, while Ours is composed of ENCODER-DECODER-ARDECODER.

Dataset/	WMT14 EN→DE	WMT14 DE→EN	WMT16 EN→RO	WMT16 RO→EN
Model	FLOPs(speedup)	FLOPs(speedup)	FLOPs(speedup)	FLOPs(speedup)
MaskPredict (w/ kd)	1.12e19 (1.00x)	6.84e18 (1.00x)	1.33e18 (1.00x)	1.45e18 (1.00x)
Ours (w/ kd)	1.43e18(**7.83x**)	1.30e18 (**5.26x**)	0.78e18 (**1.71x**)	0.79e18 (**1.83x**)

Table 3: Comparison of overall training FLOPs (speedup) between Ours and MaskPredict on each dataset.

5.2 Sentence Length

Compared with Mask-Predict, we step further to examine the influence of Self-Review Mechanism on different sentence lengths. We conduct comparative experiments on WMT14 EN→DE, and divide the reference target by length into different buckets. As shown in Figure 3, Mask-Predict performs similar or slightly better than our model when the sentence length is small. However, the performance of our model is significantly improved as the sentence length increases, even leading to a wide gap with Mask-Predict when the sentence length is quite large. This result supports that our proposed Self-Review Mechanism is better at capturing the long-term dependency among the target words.
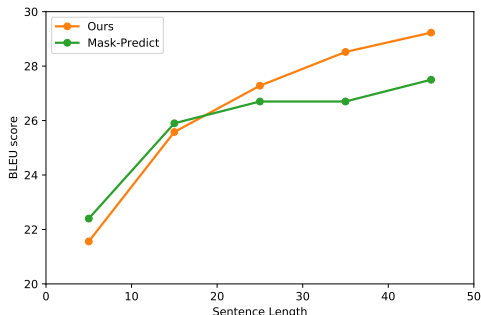

Figure 3: The Effect of Sentence Length

Iterations	WMT14 EN→DE	
	MaskPredict	**Ours**
T = 1	16.72%	9.67%
T = 2	5.40%	3.34%
T = 3	2.03%	1.48%
T = 4	1.07%	0.89%
T = 5	0.72%	0.60%

Table 4: The percentage of repetitive words at different number of decoding iterations (T).

5.3 Adjacent Words

According previous work (Wang et al., 2019), non-autoregressive models usually suffer from repetitive words at adjacent positions. To validate if such inappropriate pattern is remedied by Self-Review Mechanism, we conduct a statistical study of the repetitive words to compare Mask-Predict and our model. The results in Table 4 show that our model has substantially less repetitive words than Mask-Predict. For better understanding, we visualize the cosine similarities of two generated targets by Mask-Predict and our model respectively given a same input source, where the similarities are measured between decoder hidden states of the last layer. From the heatmaps of the resulting cosine similarities in Figure 4, we can see that there are observably more yellow blocks in (a) than those in (b), indicating that Mask-Predict shares much more similar hidden states across the positions of the generated sentence, especially illustrated along the diagonal parts in Figure 4. The results of Table 4 and Figure 4 demonstrate that our proposed Self-Review Mechanism is beneficial for the model to reduce repetitive words, which further indicate that Self-Review is also an effective technique for CMTM to capture the strong correlations among the target words.
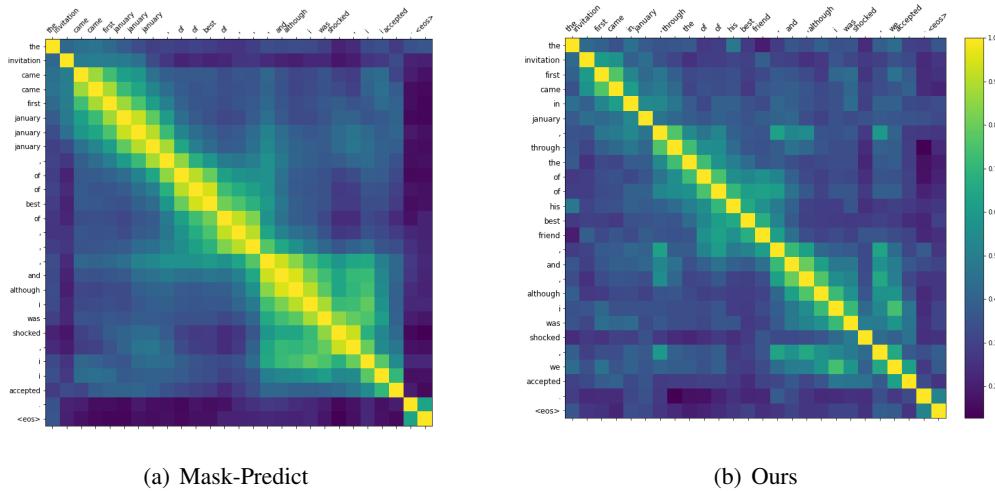

(a) Mask-Predict   (b) Ours

Figure 4: The heatmaps of cosine similarities. The axis corresponds to the generated target word.

6 Related Work

To tackle the high latency of autoregression, many researchers attempt to use one-shot parallel decoding for machine translation. Gu et al. (2018) firstly proposed a non-autoregressive model by using fertilities as a latent variable. Later on, several works introduced different kinds of latent information to improve the performance. Łukasz Kaiser et al. (2018) used a sequence of discrete latent variables as the decoder inputs, Li et al. (2019) utilized the hints from the hidden states and word alignments of an autoregressive model, and Ma et al. (2019) modeled a meaningful generative flow using latent variables. Although these methods are able to decode the target in one shot, they usually suffer from the cost of translation accuracy (Ren et al., 2020). Worse still, they will never have a chance to remedy the flawed translation.

Our work resides in the research line of iterative parallel decoding. Lee et al. (2018) iteratively refined the generated outputs through a denoising autoencoder. However, the optimization is complicated, as they resort to a heuristic method of stochastic corruption on the training data. Still along this line, our work is most relevant to (Ghazvininejad et al., 2019), where they proposed a simple yet effective method, i.e., Mask-Predict decoding strategy. A major difference is that Ghazvininejad et al. (2019) resorts to a typical conditional masked translation model (CMTM), which is highly based upon the assumption of conditional independence. However, this assumption goes against the highly multimodal distribution of true target translations (Gu et al., 2018). To alleviate the issue, we develop a Self-Review Mechanism to infuse sequential information into the CMTM model.

We also get inspired by the idea of augmenting the model with a discriminative task (Clark et al., 2020),

in order to solve the computational inefficiency of CMTM. Clark et al. (2020) introduced a discriminator (similar to our ARDECODER) that learns from all input words rather than a small masked subset. Then, they further finetuned the discriminator for the downstream tasks. The difference lies in that we throw out ARDECODER and only perform iterative decoding on ENCODER-DECODER. Besides, we tie the weights of ARDECODER and DECODER to ensure that our DECODER can take advantage of the sequential information learned from the discriminative task.

7 Conclusion

In this paper, we identify the drawback of CMTM that it is insufficient to capture the sequential correlations among target words. To tackle it, we propose a Self-Review Mechanism that is able to infuse sequential information into CMTM. On several benchmark datasets, we demonstrate that our approach achieves a huge improvement against previous non-autoregressive models and a competitive result to the state-of-the-art Transformer model. Through ablation study, our proposed mechanism is also proved to speed up the training of a CMTM model.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language gans falling short. *arXiv preprint arXiv:1811.02549*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *ArXiv*, abs/1406.1078.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. *ArXiv*, abs/1705.03122.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP-IJCNLP*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *ICLR*.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *ArXiv*, abs/1610.10099.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *EMNLP*.

Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Hint-based training for non-autoregressive machine translation. In *EMNLP-IJCNLP*.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *EMNLP-IJCNLP*.

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

Yi Ren, Jinglin Liu, Xu Tan, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Mitchell Stern, William Chan, Jamie Ryan Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

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

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *AAAI*.

Chunting Zhou, Graham Neubig, and Jiatao Gu. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *ICLR*.

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *ICML*.