Sequence-Level Training for Non-Autoregressive Neural Machine Translation

Chenze Shao
Key Laboratory of Intelligent
Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
shaochenze18z@ict.ac.cn

Yang Feng
Key Laboratory of Intelligent
Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
fengyang@ict.ac.cn

Jinchao Zhang
Pattern Recognition Center
WeChat AI, Tencent Inc
dayerzhang@tencent.com

Fandong Meng
Pattern Recognition Center
WeChat AI, Tencent Inc
fandongmeng@tencent.com

Jie Zhou
Pattern Recognition Center
WeChat AI, Tencent Inc
withtomzhou@tencent.com

In recent years, Neural Machine Translation (NMT) has achieved notable results in various translation tasks. However, the word-by-word generation manner determined by the autoregressive mechanism leads to high translation latency of the NMT and restricts its low-latency applications. Non-Autoregressive Neural Machine Translation (NAT) removes the autoregressive

---

mechanism and achieves significant decoding speedup by generating target words independently and simultaneously. Nevertheless, NAT still takes the word-level cross-entropy loss as the training objective, which is not optimal because the output of NAT cannot be properly evaluated due to the multimodality problem. In this article, we propose using sequence-level training objectives to train NAT models, which evaluate the NAT outputs as a whole and correlates well with the real translation quality. First, we propose training NAT models to optimize sequence-level evaluation metrics (e.g., BLEU) based on several novel reinforcement algorithms customized for NAT, which outperform the conventional method by reducing the variance of gradient estimation. Second, we introduce a novel training objective for NAT models, which aims to minimize the Bag-of-N-grams (BoN) difference between the model output and the reference sentence. The BoN training objective is differentiable and can be calculated efficiently without doing any approximations. Finally, we apply a three-stage training strategy to combine these two methods to train the NAT model. We validate our approach on four translation tasks (WMT14 En↔De, WMT16 En↔Ro), which shows that our approach largely outperforms NAT baselines and achieves remarkable performance on all translation tasks. The source code is available at `https://github.com/ictnlp/Seq-NAT`.

1. Introduction

Machine translation used to be one of the most challenging tasks in natural language processing, but recent advances in neural machine translation make it possible to translate with an end-to-end model architecture. NMT models are typically built on the encoder-decoder framework. The encoder network encodes the source sentence to distributed representations, and the decoder network reconstructs the target sentence from these representations in an autoregressive manner. The target sentence is generated word-by-word where the previously predicted words are fed back to the decoder as context. In the past few years, autoregressive NMT models have achieved notable results in various translation tasks (Cho et al. 2014; Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2015; Wu et al. 2016; Vaswani et al. 2017). However, the word-by-word generation manner determined by the autoregressive mechanism leads to high translation latency of the NMT and restricts its low-latency applications.

Non-Autoregressive Neural Machine Translation (NAT) (Gu et al. 2018) is proposed to reduce the latency of NMT. By removing the autoregressive mechanism, NAT can generate target words independently and simultaneously, thereby achieving significant decoding speedup. Nevertheless, NAT still takes the word-level cross-entropy loss as the training objective, which is not optimal because the output of NAT cannot be properly evaluated. Due to the multimodality of language, the reference sentence may have many variants that are composed of different words but have the same semantics. For the autoregressive model, the teacher forcing algorithm (Williams and Zipser 1989) can provide it with sequential information that guides the model to generate the reference sentence. However, the sequential information is not available during the training of NAT, so NAT may generate any translation variant with the target semantics. Once the NAT model tends to generate a variant that is not aligned verbatim with the reference sentence, the cross-entropy loss will give it a large penalty with no regard to the translation quality. Consequently, the correlation between the cross-entropy loss and translation quality becomes weak, which has a negative impact on the NAT performance.
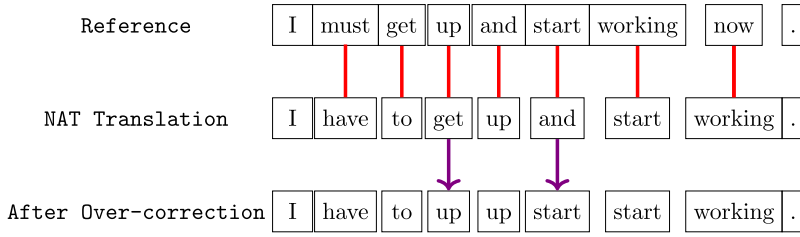
Figure 1
One example of NAT that is not aligned verbatim with the reference sentence. The red line indicates the misalignment that will receive a large penalty from the word-level cross-entropy loss. The purple arrow indicates the possible overcorrection error.

Table 1
A translation case on the validation set of WMT14 De-En. Source and Target are the source sentence and reference sentence, respectively. AT and NAT are the output of the autoregressive Transformer and non-autoregressive Transformer, respectively.

Source	Es gibt Krebsarten, die aggressiv und andere, die indolent sind.
Reference	There are aggressive cancers and others that are indolen.
AT	There are cancers that are aggressive and others that are indolent.
NAT	There are cancers cancer aggressive aggressive others are indindent.

As shown in Figure 1, though the translation "I have to get up and start working." has similar semantics to the reference sentence, the word-level cross-entropy loss will give it a large penalty because it is not aligned verbatim with the reference sentence. Under the guidance of cross-entropy loss, the translation may be further corrected to "I have to up up start start working.". This is preferred by the cross-entropy loss but the translation quality will actually get worse, which is called the **overcorrection error** (Zhang et al. 2019). The essential reason for the overcorrection error is that the loss function evaluates the generation quality of each position independently and does not model the sequential dependency. As a result, NAT tends to focus on local correctness while ignoring the overall translation quality, and therefore generates influent translations with many over-translation and under-translation errors. As shown in Table 1, the output of NAT is incomplete and contains repeated words like "cancer" and "aggressive."

In this article, we propose using sequence-level training objectives to train NAT models, which evaluate the NAT outputs as a whole and correlate well with the real translation quality. First, we propose training NAT models to optimize sequence-level evaluation metrics (e.g., BLEU [Papineni et al. 2002], GLEU [Wu et al. 2016], and ROUGE [Lin 2004]). These metrics are usually non-differentiable, and reinforcement learning techniques (Sutton 1984; Williams 1992; Sutton et al. 1999) are widely applied to train autoregressive NMT to optimize these discrete objectives (Ranzato et al. 2016; Bahdanau et al. 2017). However, the training procedure is usually unstable due to the high variance of the gradient estimation. Using the appealing characteristics of non-autoregressive generation, we propose several novel reinforcement algorithms customized for NAT, which outperform the conventional method by reducing the variance of gradient estimation. Second, we introduce a novel training objective for NAT models, which

aims to minimize the Bag-of-*N*-grams (BoN) difference between the model output and the reference sentence. As the word-level loss cannot properly model the sequential dependency, we propose to evaluate the NAT output at the *n*-gram level. Since the output of NAT may not be aligned verbatim with the reference, we do not require the strict alignment and optimize the BoN for NAT. Optimizing such an objective usually faces the difficulty of the exponential search space, and we find that the difficulty can be overcome through using the characteristics of non-autoregressive generation. In summary, the BoN training objective has many appealing properties. It is differentiable and can be calculated efficiently without doing any approximations. Most importantly, the BoN objective correlates well with the overall translation quality, as we demonstrate in the experiments.

The reinforcement learning method can train NAT with any sequence-level objective, but it requires a lot of calculations on the CPU to reduce the variance of gradient estimation. The bag-of-*n*-grams method can efficiently calculate the BoN objective without doing any approximations, but the choice of training objectives is very limited. The cross-entropy loss also has strengths such as high-speed training and is suitable for model warmup. Therefore, we apply a three-stage training strategy to combine the two sequence-level training methods and the word-level training to train the NAT model. We validate our approach on four translation tasks (WMT14 En↔De, WMT16 En↔Ro), which shows that our approach largely outperforms NAT baselines and achieves remarkable performance on all translation tasks.

This article extends our conference papers on non-autoregressive translation (Shao et al. 2019, 2020) in three major directions. First, we propose several novel sequence-level training algorithms in this article. In the context of reinforcement learning, we propose the traverse-based method Traverse-Ref, which practically eliminates the variance of gradient estimation and largely outperforms the best method Reinforce-Top-*k* proposed in Shao et al. (2019). We also propose to use bag-of-words as the training objective of NAT. The bag-of-words vector can be explicitly calculated, so it supports a variety of distance metrics such as BoW-L_1, BoW-L_2, and BoW-*Cos* as loss functions, which enables us to analyze the performance of different distance metrics on NAT. Second, we explore the combination of the reinforcement learning based method and the bag-of-*n*-grams method and propose a three-stage training strategy to better combine their advantages. Finally, we conduct experiments on a stronger baseline model (Ghazvininejad et al. 2019) and a larger batch size setting to show the effectiveness of our approach, and we also provide a more detailed analysis. The article is structured as follows. We explain the vanilla non-autoregressive translation and sequence-level training in Section 2. We introduce our sequence-level training methods in Section 3. We review the related works on non-autoregressive translation and sequence-level training in Section 4. In Section 5, we introduce the experimental design, conduct experiments to evaluate the performance of our methods and conduct a series of analyses to understand the underlying key components in them. Finally, we conclude in Section 6 by summarizing the contributions of our work.

2. Background

2.1 Autoregressive Neural Machine Translation

Deep neural networks with the autoregressive encoder-decoder framework have achieved state-of-the-art results on machine translation, with different choices of architectures such as Recurrent Neural Network (RNN), Convolutional Neural Network

(CNN), and Transformer. RNN-based models (Bahdanau, Cho, and Bengio 2015; Cho et al. 2014) have a sequential architecture that makes them difficult to be parallelized. CNN (Gehring et al. 2017) and self-attention (Vaswani et al. 2017) based models have highly parallelized architectures, which solves the parallelization problem during the training. However, during the inference, the translation has to be generated word-by-word due to the autoregressive mechanism.

Given a source sentence $X = \{x_1, \dots, x_n\}$ and a target sentence $Y = \{y_1, \dots, y_T\}$, the autoregressive NMT models the translation probability from X to Y sequentially as:

$$P(Y|X, \theta) = \prod_{t=1}^{T} p(y_t|y_{<t}, X, \theta) \tag{1}$$

where θ is a set of model parameters and $y_{<t} = \{y_1, \cdots, y_{t-1}\}$ is the translation history. The standard training objective is the cross-entropy loss, which minimizes the negative log-likelihood as:

$$\mathcal{L}_{MLE}(\theta) = -\sum_{t=1}^{T} \log(p(y_t|y_{<t}, X, \theta)) \tag{2}$$

During the training, the teacher forcing algorithm (Williams and Zipser 1989) is applied, where golden target words are fed into the decoder as the translation history. During the inference, because there is no polynomial time algorithm to find the translation with the maximum likelihood, autoregressive models have to rely on decoding algorithms such as greedy search and beam search to generate the translation. The partial translation generated by the decoding algorithm is fed back to the decoder to guide the generation of the next word. The prominent feature of the autoregressive model is that it requires the target-side historical information in the decoding procedure. Therefore, target words are generated in the word-by-word style, which leads to high translation latency and restricts the application of the autoregressive model.

2.2 Non-Autoregressive Neural Machine Translation

Non-autoregressive neural machine translation (Gu et al. 2018) is proposed to reduce the translation latency through parallelizing the decoding process. A basic NAT model takes the same encoder-decoder architecture as the Transformer, except that there is a length predictor that takes encoder states as input to predict the target length. Given a source sentence $X = \{x_1, \dots, x_n\}$ and a target sentence $Y = \{y_1, \dots, y_T\}$, NAT models the translation probability from X to Y as:

$$P(Y|X, \theta) = \prod_{t=1}^{T} p_t(y_t|X, \theta) \tag{3}$$

where θ is a set of model parameters and $p_t(y_t|X, \theta)$ indicates the translation probability of word y_t in position t. The cross-entropy loss is applied to minimize the negative log-likelihood as:

$$\mathcal{L}_{MLE}(\theta) = -\sum_{t=1}^{T} \log(p_t(y_t|X, \theta)) \tag{4}$$

During the training, the target length T is usually set as the reference length. During the inference, the target length T is obtained from the length predictor, and then the translation of length T with the maximum likelihood can be easily obtained by taking the word with the maximum likelihood at each time step:

$$\hat{y}_t = \arg\max_{y_t} p_t(y_t|\boldsymbol{X}, \theta) \tag{5}$$

2.3 Sequence-Level Training for Autoregressive NMT

Reinforcement learning techniques (Sutton 1984; Ng, Harada, and Russell 1999; Sutton et al. 1999) have been widely applied to improve the performance of the autoregressive NMT with sequence-level objectives (Ranzato et al. 2016; Bahdanau et al. 2017; Wu et al. 2016; Yang et al. 2018). As sequence-level objectives are usually non-differentiable, the loss function is defined as the negative expected reward:

$$L_\theta = - \sum_{\boldsymbol{Y}=y_{1:T}} p(\boldsymbol{Y}|\boldsymbol{X}, \theta) \cdot r(\boldsymbol{Y}) \tag{6}$$

where $\boldsymbol{Y} = y_{1:T}$ denotes a possible sequence generated by the model, and $r(\boldsymbol{Y})$ is the corresponding reward (e.g., BLEU) for generating the sentence $\boldsymbol{Y}$. Enumerating all possible target sentences is impossible due to the exponential search space, and REINFORCE (Williams 1992) gives an elegant way to estimate the gradient:

$$
\begin{aligned}
\nabla_\theta L_\theta &= - \sum_{\boldsymbol{Y}} p(\boldsymbol{Y}|\boldsymbol{X}, \theta) \cdot \frac{\nabla_\theta p(\boldsymbol{Y}|\boldsymbol{X}, \theta)}{p(\boldsymbol{Y}|\boldsymbol{X}, \theta)} \cdot r(\boldsymbol{Y}) \\
&= - \mathop{\mathbb{E}}_{\boldsymbol{Y}} [\nabla_\theta \log(p(\boldsymbol{Y}|\boldsymbol{X}, \theta)) \cdot r(\boldsymbol{Y})] \\
&= - \mathop{\mathbb{E}}_{\boldsymbol{Y}} [\sum_{t=1}^{T} \nabla_\theta \log(p(y_t|\boldsymbol{y}_{<t}, \boldsymbol{X}, \theta)) \cdot r(\boldsymbol{Y})]
\end{aligned}
\tag{7}
$$

Equation (7) indicates that we can obtain an unbiased estimation of the gradient through the training process summarized as follows.

- Given a source sentence X, sample a translation Y from $p(\boldsymbol{Y}|\boldsymbol{X}, \theta)$.
- Calculate the reward $r(\boldsymbol{Y})$ for the sampled sentence.
- Update the model by the estimated gradient $\nabla_\theta \log(p(\boldsymbol{Y}|\boldsymbol{X}, \theta)) \cdot r(\boldsymbol{Y})$

Although it has the advantage of unbiased estimation, previous investigations show that the reinforcement learning based training procedure is unstable due to its high variance of gradient estimation, which mainly comes from its two drawbacks. First, word predictions in the sentence are treated equally and receive the same reward, which ignores the fact that a bad translation is often due to translation errors in only

a few positions, and predictions in other positions should not be held responsible. Second, as the reward is usually defined to be positive, the algorithm will always tend to raise the probability of the sampled sentence, leading to many inefficient parameter updates. There are solutions like reward shaping (Ng, Harada, and Russell 1999) or baseline reward (Weaver and Tao 2001) to reduce the estimation variance. However, as the sampling cost of autoregressive models is very expensive, they will either lead to biased estimation (Ranzato et al. 2016; Bahdanau et al. 2017) or be time-consuming (Shen et al. 2016; Yu et al. 2017).

3. Sequence-Level Training for NAT

In this article, we will present our sequence-level training methods in detail. We first discuss training methods based on the reinforcement learning, which can train NAT with any sequence-level objective that correlates well with the translation quality. We then introduce the differentiable bag-of-n-grams training objective to train NAT, which can be efficiently calculated without doing any approximations. Finally, we use a three-stage training strategy to combine the strengths of the two training methods and the word-level training.

3.1 Reinforcement Learning

The word-level cross-entropy loss cannot evaluate the output of NAT properly, so it is necessary to train NAT with sequence-level objectives (e.g., BLEU), which is more suitable for NAT models and correlates well with the translation quality. However, these objectives are usually discrete and therefore non-differentiable, so we propose to optimize the training objective under the reinforcement learning framework (Williams 1992). The basic idea is to define the loss function as the negative expected reward, and find an unbiased estimation of its gradient:

$$\nabla_\theta L_\theta = - \sum_{\mathbf{Y}=y_{1:T}} \nabla_\theta p(\mathbf{Y}|\mathbf{X},\theta) \cdot r(\mathbf{Y})$$

$$= - \sum_{\mathbf{Y}} \nabla_\theta \prod_{t=1}^{T} p_t(y_t|\mathbf{X},\theta) \cdot r(\mathbf{Y}) \tag{8}$$

In the following, we first present the basic method **Reinforce-Base** in Section 3.1.1, which directly applies REINFORCE (Williams 1992) to estimate Equation (8). Then we develop its improved version **Reinforce-Step** in Section 3.1.2, where the prediction in each time step has a unique step reward instead of sharing a sentence reward. **Reinforce-Top-k** is further proposed in Section 3.1.3 to reduce the estimation variance by paying more attention to the top-ranking words, which are more important than others in the gradient estimation. Taking advantage of the fact that many reward functions are based on the comparison with the reference sentence, we finally propose **Traverse-Ref** in Section 3.1.4 to calculate the gradient accurately by only traversing words in the reference sentence. The time complexity of these methods is analyzed in Section 3.1.5.

Algorithm 1 Reinforce-Base

Input: probability distribution $p(\cdot|\mathbf{X}, \theta))$, length T
Output: the estimate of $\nabla_\theta L_\theta$
  1: $\nabla_\theta L_\theta = 0$
  2: **for** $t = 1$ **to** T **do**
  3:     sample y_t from $p_t(\cdot|\mathbf{X}, \theta))$
  4:     $\nabla_\theta L_\theta \mathrel{+}= -\nabla_\theta \log(p_t(y_t|\mathbf{X}, \theta))$
  5: **end for**
  6: construct $\mathbf{Y} = \{y_1, \ldots, y_T\}$, calculate $r(\mathbf{Y})$
  7: $\nabla_\theta L_\theta = r(\mathbf{Y}) \cdot \nabla_\theta L_\theta$
  8: **return** $\nabla_\theta L_\theta$

3.1.1 Reinforce-Base. We can put aside the characteristics of the NAT and follow autoregressive models to directly apply the REINFORCE algorithm to estimate the gradient:

$$\nabla_\theta L_\theta = - \mathop{\mathbb{E}}_{\mathbf{Y}}[\sum_{t=1}^{T} \nabla_\theta \log(p_t(y_t|\mathbf{X}, \theta)) \cdot r(\mathbf{Y})] \tag{9}$$

According to Equation (9), the NAT model first generates the probability distribution and samples a sentence from the distribution. Then the reward of the sampled sentence is calculated to evaluate the translation quality. The loss function becomes the log-probability of the sampled sentence weighted by the reward, where the reward acts as the learning rate to enhance the learning of high-quality samples. Algorithm 1 describes the estimation process.

3.1.2 Reinforce-Step. In Reinforce-Base, the word prediction y_t in every time step t receives the same sentence reward $r(\mathbf{Y})$, ignoring the characteristic of independent generation in NAT models. Intuitively, because the word prediction in every step t is independent of other steps, its reward should also only be related to the word y_t. Therefore, we derive the Equation (8) to the following form, where the gradient in each time step t is weighted by the step reward $r(y_t)$, which is defined as the expectation of reward when the prediction in step t is y_t:

Theorem 1

$$\nabla_\theta L_\theta = - \sum_{t=1}^{T} \sum_{y_t} \nabla_\theta p_t(y_t|\mathbf{X}, \theta) \cdot r(y_t) \tag{10}$$

$$\text{where} \quad r(y_t) = \mathop{\mathbb{E}}_{y_{1:t-1}} \mathop{\mathbb{E}}_{y_{t+1:T}} r(\mathbf{Y}) \tag{11}$$

Proof.

$$\nabla_\theta L_\theta = -\sum_Y \nabla_\theta \prod_{t=1}^T p_t(y_t|X,\theta) \cdot r(Y)$$

$$= -\sum_Y \sum_{t=1}^T \nabla_\theta p_t(y_t|X,\theta) \cdot \prod_{i=1}^{t-1} p_i(y_i|X,\theta) \cdot \prod_{j=t+1}^T p_j(y_j|X,\theta) \cdot r(Y)$$

$$= -\sum_{t=1}^T \sum_Y \nabla_\theta p_t(y_t|X,\theta) \cdot \prod_{i=1}^{t-1} p_i(y_i|X,\theta) \cdot \prod_{j=t+1}^T p_j(y_j|X,\theta) \cdot r(Y)$$

$$= -\sum_{t=1}^T \sum_{y_t} \nabla_\theta p_t(y_t|X,\theta) \cdot \sum_{y_{1:t-1}} \sum_{y_{t+1:T}} \prod_{i=1}^{t-1} p_i(y_i|X,\theta) \cdot \prod_{j=t+1}^T p_j(y_j|X,\theta) \cdot r(Y)$$

$$= -\sum_{t=1}^T \sum_{y_t} \nabla_\theta p_t(y_t|X,\theta) \cdot \mathop{\mathbb{E}}_{y_{1:t-1}} \mathop{\mathbb{E}}_{y_{t+1:T}} r(Y).$$

$$= -\sum_{t=1}^T \sum_{y_t} \nabla_\theta p_t(y_t|X,\theta) \cdot r(y_t) \tag{12}$$

□

In Equation (10), each time step receives an accurate step reward $r(y_t)$ instead of sharing a sentence reward $r(Y)$. We can still apply the REINFORCE algorithm to estimate the gradient:

$$\nabla_\theta L_\theta = -\sum_{t=1}^T \mathop{\mathbb{E}}_{y_t}[\nabla_\theta \log(p_t(y_t|X,\theta)) \cdot r(y_t)] \tag{13}$$

The estimation process of Reinforce-Step is basically the same as Reinforce-Base, except that each word prediction receives a step reward rather than the sentence reward. The step reward is defined as the expectation of reward when the prediction in step t is specified, which can be estimated by Monte Carlo sampling, as illustrated in algorithm 2. Specifically, we fix the prediction y_t in step t and sample words of other time steps from the probability distribution $p(\cdot|X,\theta))$. After obtaining the sentence, we calculate the sentence reward $r(Y)$. We repeat this process for n times and use the average reward to estimate the step reward $r(y_t)$. Algorithm 3 describes the process of Reinforce-Step.

The idea of assigning a unique reward to each time step has similarities with the actor-critic approach in NMT (Ranzato et al. 2016; Bahdanau et al. 2017), which uses a critic network to predict the expected reward $r(y_{1:t})$ after generating t words. In comparison, the step reward $r(y_t)$ in Reinforce-Step is more accurate since it does not depend on the previously generated words. Besides, due to the bias of the neural network prediction, the gradient estimation of the actor-critic approach is biased. In comparison, Reinforce-Step can obtain an unbiased estimation of the gradient.

3.1.3 Reinforce-Top-k. The estimation variance of Reinforce-Step can be further reduced if we can traverse the vocabulary to directly calculate the Equation (10) instead of

---

Algorithm 2 Estimation of step reward $r(y_t)$

Input: probability distribution $p(\cdot|\mathbf{X}, \theta))$, step t, word y_t, length T, sampling times n
Output: the estimate of $r(y_t)$
  1: $r = 0$
  2: **for** $i = 1$ **to** n **do**
  3:     sample $y_{1:t-1}, y_{t+1:T}$ from $p(\cdot|\mathbf{X}, \theta))$
  4:     construct $\mathbf{Y} = \{y_{1:t-1}, y_t, y_{t+1:T}\}$, calculate $r(\mathbf{Y})$
  5:     $r \mathrel{+}= r(\mathbf{Y})$
  6: **end for**
  7: $r = r/n$
  8: **return** r

Algorithm 3 Reinforce-Step

Input: probability distribution $p(\cdot|\mathbf{X}, \theta))$, length T, sampling times n
Output: the estimate of $\nabla_\theta L_\theta$
  1: $\nabla_\theta L_\theta = 0$
  2: **for** $t = 1$ **to** T **do**
  3:     sample y_t from $p_t(\cdot|\mathbf{X}, \theta))$
  4:     estimate $r(y_t)$ by algorithm 2 with sampling times n
  5:     $\nabla_\theta L_\theta \mathrel{+}= -r(y_t) \cdot \nabla_\theta \log(p_t(y_t|\mathbf{X}, \theta))$
  6: **end for**
  7: **return** $\nabla_\theta L_\theta$

---

applying REINFORCE for estimation. However, this will bring a high computational cost due to the large vocabulary size. Therefore, we take a step back and only traverse a subset of the vocabulary. The subset contains important words for the gradient estimation and filters out unimportant words, so we can effectively reduce the estimation variance and meantime maintain the acceptable training speed.

The probability distribution over the target vocabulary is usually a centered distribution where the top-ranking words occupy the central part of the distribution, and the softmax layer ensures that the other words with small probabilities have small gradients. Hence the variance will be effectively reduced if we can eliminate the variance of top-ranking words. This motivates us to compute gradients of the top-ranking words accurately and estimate the rest via the REINFORCE algorithm.

First, we denote the subset of target words with top-k probabilities in step t as T_k^t. As defined in Equation (14), P_k^t is the sum of probabilities in T_k^t, and $\tilde{p}$ is the normalized probability distribution after removing the words in T_k^t:

$$P_k^t = \sum_{y_t \in T_k^t} p_t(y_t|\mathbf{X}, \theta), \quad \tilde{p}_t(y_t|\mathbf{X}, \theta) = \begin{cases} 0, & y \in T_k^t \\ \frac{p_t(y_t|\mathbf{X}, \theta)}{1 - P_k^t}, & y \notin T_k^t \end{cases} \tag{14}$$

We can divide the gradient into two parts, and process them in different ways. For the important words in T_k^t, we traverse them and accumulate their gradients. For other words, we estimate their gradients by one sampling from the normalized distribution $\tilde{p}$, and weight the estimation by $1 - P_k^t$. The following equation shows that an unbiased

---

Algorithm 4 Reinforce-Top-k

Input: probability distribution $p(\cdot|\mathbf{X},\theta))$, top-$k$ size k, length T, sampling times n
Output: the estimate of $\nabla_\theta L_\theta$
1: $\nabla_\theta L_\theta = 0$
2: **for** $t = 1$ **to** T **do**
3:      $T_k^t = \{\text{words ranking top-}k \text{ in } p_t(\cdot|\mathbf{X},\theta))\}$
4:      $\tilde{p}_t = p_t,\ P_k^t = 0$
5:      **for** y_t in T_k^t **do**
6:          estimate $r(y_t)$ by algorithm 2 with sampling times n
7:          $\nabla_\theta L_\theta \mathrel{+}= -\nabla_\theta p_t(y_t|\mathbf{X},\theta) \cdot r(y_t)$
8:          $\tilde{p}_t(y_t|\mathbf{X},\theta) = 0,\ P_k \mathrel{+}= p_t(y_t|\mathbf{X},\theta)$
9:      **end for**
10:      $\tilde{p}_t = \tilde{p}_t/(1 - P_k)$, sample y_t from $\tilde{p}_t(\cdot|\mathbf{X},\theta)$
11:      estimate $r(y_t)$ by algorithm 2 with sampling times n
12:      $\nabla_\theta L_\theta \mathrel{+}= -(1 - P_k) \cdot \nabla_\theta \log(p_t(y_t|\mathbf{X},\theta)) \cdot r(y_t)$
13: **end for**
14: **return** $\nabla_\theta L_\theta$

---

estimate of the gradient can be obtained in this way, which effectively reduces the estimation variance with the acceptable training cost. Algorithm 4 describes the process of Reinforce-Top-k.

Theorem 2

$$\nabla_\theta L_\theta = -\sum_{t=1}^{T}(\sum_{y_t \in T_k^t} \nabla_\theta p_t(y_t|\mathbf{X},\theta) \cdot r(y_t) + (1 - P_k^t) \cdot \mathop{\mathbb{E}}_{y_t \sim \tilde{p}}[\nabla_\theta \log(p_t(y_t|\mathbf{X},\theta)) \cdot r(y_t)])$$

(15)

Proof.

$$\nabla_\theta L_\theta = -\sum_{t=1}^{T}\sum_{y_t} \nabla_\theta p_t(y_t|\mathbf{X},\theta) \cdot r(y_t)$$

$$= -\sum_{t=1}^{T}(\sum_{y_t \in T_k^t} \nabla_\theta p_t(y_t|\mathbf{X},\theta) \cdot r(y_t) + \sum_{y_t \notin T_k^t} \nabla_\theta p_t(y_t|\mathbf{X},\theta) \cdot r(y_t))$$

$$= -\sum_{t=1}^{T}(\sum_{y_t \in T_k^t} \nabla_\theta p_t(y_t|\mathbf{X},\theta) \cdot r(y_t) + (1 - P_k^t) \cdot \sum_{y_t \notin T_k^t} \frac{p_t(y_t|\mathbf{X},\theta)}{1 - P_k^t} \nabla_\theta \log(p_t(y_t|\mathbf{X},\theta)) \cdot r(y_t))$$

$$= -\sum_{t=1}^{T}(\sum_{y_t \in T_k^t} \nabla_\theta p_t(y_t|\mathbf{X},\theta) \cdot r(y_t) + (1 - P_k^t) \cdot \mathop{\mathbb{E}}_{y_t \sim \tilde{p}}[\nabla_\theta \log(p_t(y_t|\mathbf{X},\theta)) \cdot r(y_t)])$$

(16)

$\square$

3.1.4 Traverse-Ref. Due to the large vocabulary size, it is costly to directly calculate the gradient according to Equation (10), so we have to rely on reinforcement learning algorithms for gradient estimation. However, we find that when the reward function is *Reference-Based*, it becomes possible to traverse words in the whole vocabulary and calculate their rewards in a short time, enabling us to directly calculate the gradient according to Equation (10). First, we call a word out-of-$\mathbf{X}$ if it does not appear in sentence $\mathbf{X}$, and define the Reference-Based reward as follows:

Definition 1
A reward function $r(\mathbf{Y})$ is Reference-Based if it evaluates $\mathbf{Y}$ by comparing it with a reference sentence $\hat{\mathbf{Y}}$ and the reward does not change when we replace any out-of-$\hat{\mathbf{Y}}$ word in $\mathbf{Y}$ by other out-of-$\hat{\mathbf{Y}}$ words.

From the definition, we can see that many widely used reward functions are Reference-Based. For example, since words that do not appear in the reference sentence can never match the reference, the rewards based on *n*-gram matching (e.g., BLEU, GLEU, and ROUGE) will not be changed by replacing any out-of-reference word with other out-of-reference words, so these rewards are Reference-Based. Recall that the step reward $r(y_t)$ is defined by the expectation of reward when the prediction in step t is y_t. According to the definition of Reference-Based, the step reward $r(y_t)$ will take the same value as long as the reward function is Reference-Based and y_t does not appear in the reference sentence. Therefore, we can divide the vocabulary into two parts. For words in the reference sentence, we traverse these words and estimate their step rewards. For out-of-reference words, we only need to estimate the step reward once because they take the same value. Finally, we calculate the gradient according to Equation (10). In this way, the variance caused by the reinforcement learning method is completely eliminated, so the estimation variance only comes from the estimation of the step reward. Algorithm 5 describes the process of Traverse-Ref.

Notice that Reinforce-Top-*k* and Traverse-Ref are not applicable to sentence-level reward. Reinforce-Top-*k* works for word-level reward since the top-*k* words can usually occupy a large part of probability distribution. However, considering the exponential search space, traversing top-*k* sentences generally does not have a great impact on gradient estimation. Traverse-Ref works for word-level reward by reducing the search space of vocabulary size V to T reference words and one out-of-reference word. However, for sentence reward, the search space is only reduced from V^T to $(T+1)^T$, which is still intractable.

3.1.5 Time Complexity. The proposed methods will not affect the time complexity on GPU, but the computational cost on CPU becomes non-negligible since we have to calculate the reward for many times. We take the calculation of the reward as a time unit and give the time complexity of the proposed methods in Table 2. Generally, the time complexity increases as the algorithm evolves from Reinforce-Base to Traverse-Ref.

3.2 Bag-of-*N*-grams

Reinforcement learning based methods optimize the sequence-level objective through the gradient estimation. To stabilize the training process, we make many efforts to

---

Algorithm 5 Traverse-Ref

Input: probability distribution $p(\cdot|\mathbf{X}, \theta))$, length T, sampling times n, target vocabulary V, reference sentence $\hat{\mathbf{Y}}$
Output: the estimate of $\nabla_\theta L_\theta$
1: $\nabla_\theta L_\theta = 0$
2: $V_{\hat{\mathbf{Y}}} = \{$words in the reference sentence $\hat{\mathbf{Y}}\}$
3: randomly choose a word w from $V \setminus V_{\hat{\mathbf{Y}}}$
4: **for** $t = 1$ **to** T **do**
5:     **for** y_t in $V_{\hat{\mathbf{Y}}}$ **do**
6:         estimate $r(y_t)$ by algorithm 2 with sampling times n
7:     **end for**
8:     estimate $r(w)$ by algorithm 2 with sampling times n
9:     **for** y_t in $V \setminus V_{\hat{\mathbf{Y}}}$ **do**
10:         $r(y_t) = r(w)$
11:     **end for**
12:     **for** y_t in V **do**
13:         $\nabla_\theta L_\theta$ += $-\nabla_\theta p_t(y_t|\mathbf{X}, \theta) \cdot r(y_t)$
14:     **end for**
15: **end for**
16: **return** $\nabla_\theta L_\theta$

Table 2
Time complexity of the proposed methods on CPU. T is the length of the target sentence, n is the sampling times when estimating the step reward, and k is the top-k size in Reinforce-Top-k.

	Reinforce-Base	Reinforce-Step	Reinforce-Top-k	Traverse-Ref
Time Complexity	$\mathcal{O}(1)$	$\mathcal{O}(nT)$	$\mathcal{O}(nkT)$	$\mathcal{O}(nT^2)$

reduce the estimation variance. However, this requires a lot of reward calculation and hence the computational cost on CPU becomes large. In this section, we introduce a novel training objective based on bag-of-n-grams for NAT, which is differentiable and can be efficiently calculated without doing any approximations.

Bag-of-Words (BoW) (Joachims 1998) is a widely used text representation model that discards the word order and represents a sentence as the multiset of its belonging words. Bag-of-n-grams (Pang, Lee, and Vaithyanathan 2002; Li et al. 2016) is proposed to enhance the text representation by taking consecutive words (n-gram) into consideration. Besides, bag-of-n-grams also plays an important role in the evaluation of translation quality. Recall those evaluation metrics that can evaluate the translation quality well (e.g., BLEU, GLEU, and ROUGE); many of them are based on the accuracy or recall of n-grams, which basically depends on the intersection size of bag-of-n-grams. Therefore, we propose to directly train NAT to minimize the Bag-of-N-grams (BoN) difference between the NAT output and reference. We first define the BoN of a discrete sentence by the sum of n-gram vectors with one-hot representation. Then we define the BoN of NMT by the expectation of BoN on all possible translations and give an efficient method to calculate the BoN of NAT. Finally, we give methods to calculate the BoN distance between the NAT output and reference.

3.2.1 Definition. BoN is the sum of vectors where each vector is the one-hot representation of an n-gram, which has the size $|V|^n$ when the vocabulary size is $|V|$. Formally, for a sentence $Y = \{y_1, \ldots, y_T\}$, we use BoN_Y to denote the bag-of-n-grams of Y. For an n-gram $\mathbf{g} = (g_1, \ldots, g_n)$, we use $\text{BoN}_Y(\mathbf{g})$ to denote the value of entry $\mathbf{g}$ in BoN_Y, which is the number of occurrences of n-gram $\mathbf{g}$ in sentence Y and is formulized as follows:

$$\text{BoN}_Y(\mathbf{g}) = \sum_{t=0}^{T-n} 1\{y_{t+1:t+n} = \mathbf{g}\} \tag{17}$$

where $1\{\cdot\}$ is the indicator function that takes value from $\{0, 1\}$ whose value is 1 if and only if the inside condition holds.

For a discrete sentence, our definition of BoN is consistent with previous work. However, there is no clear definition of BoN for sequence models like NMT, which model the probability distribution on the whole target space. A natural approach is to consider all possible translations and use the expected BoN to define the BoN for sequence models. For NMT with parameter θ, we use BoN_θ to denote its bag-of-n-grams. Formally, given a source sentence X, the value of entry $\mathbf{g}$ in BoN_θ is defined as follows:

$$\text{BoN}_\theta(\mathbf{g}) = \sum_Y P(Y|X, \theta) \cdot \text{BoN}_Y(\mathbf{g}) \tag{18}$$

3.2.2 Efficient Calculation. It is unrealistic to directly calculate $\text{BoN}_Y(\mathbf{g})$ according to Equation (18) due to the exponential search space. For autoregressive NMT, because of the conditional dependency in modeling translation probability, it is difficult to simplify the calculation without loss of accuracy. Fortunately, NAT models the translation probability in different positions independently, which enables us to divide the target sequence into subareas and analyze the BoN in each subarea without being influenced by other positions. Using this unique property of NAT, we can convert Equation (18) to the following form:

Theorem 3

$$\text{BoN}_\theta(\mathbf{g}) = \sum_{t=0}^{T-n} \prod_{i=1}^{n} p_{t+i}(y_{t+i} = g_i | X, \theta) \tag{19}$$

Proof.

$$\text{BoN}_\theta(\mathbf{g}) = \sum_Y P(Y|X, \theta) \cdot \sum_{t=0}^{T-n} 1\{y_{t+1:t+n} = \mathbf{g}\}$$

$$= \sum_{t=0}^{T-n} \sum_Y P(Y|X, \theta) \cdot 1\{y_{t+1:t+n} = \mathbf{g}\}$$

$$= \sum_{t=0}^{T-n} \sum_{Y_{1:t}} P(Y_{1:t}|X, \theta) \sum_{Y_{t+n+1:T}} P(Y_{t+n+1:T}|X, \theta) \sum_{Y_{t+1:t+n}} P(Y_{t+1:t+n}|X, \theta) \cdot 1\{y_{t+1:t+n} = \mathbf{g}\}$$

$$= \sum_{t=0}^{T-n} \sum_{Y_{t+1:t+n}} P(Y_{t+1:t+n}|X,\theta) \cdot 1\{y_{t+1:t+n} = g\}$$

$$= \sum_{t=0}^{T-n} \prod_{i=1}^{n} p_{t+i}(y_{t+i} = g_i|X,\theta) \tag{20}$$

$\square$

Equation (19) gives an efficient method to calculate $\text{BoN}_\theta(g)$, where we slide a window on NAT output distributions to obtain all continuous subareas of size n, and then accumulate the counts of n-gram g in all subareas. This process does not make any approximation and requires little computational effort.

3.2.3 Bag-of-Words Objective. Our objective is to minimize the BoN difference between NAT output and reference. The difference can be measured by many metrics such as the L_1 distance, L_2 distance, and cosine distance. BoN is defined to be a vector of size $|V|^n$ where $|V|$ is the vocabulary size. Though we have an efficient calculation method for $\text{BoN}_\theta(g)$, computing the complete BoN vector for NAT is still unaffordable due to the large BoN size. The only exception is the case of $n = 1$, where the bag-of-n-grams degenerates into BoW. In this situation, we only need to sum the probability distributions in all time steps to obtain BoW_θ and apply distance metrics to calculate BoW distances like BoW-L_1, BoW-L_2, and BoW-*Cos*.

3.2.4 Bag-of-N-grams Objective. For $n > 1$, the complete BoN vector is unavailable, so many distance metrics like L_2 distance and cosine distance cannot be calculated. Fortunately, we find that the L_1 distance between the two BoN vectors, denoted as BoN-L_1, can be simplified using the sparsity of bag-of-n-grams. As shown in Equation (19), for NAT, its bag-of-n-grams vector BoN_θ is dense. On the contrary, assume that the reference sentence is $\hat{Y}$, the vector $\text{BoN}_{\hat{Y}}$ is very sparse where only a few entries of it have non-zero values. Using this property, we can write BoN-L_1 as follows:

Theorem 4

$$\text{BoN-}L_1 = 2(T - n + 1 - \sum_g \min(\text{BoN}_\theta(g), \text{BoN}_{\hat{Y}}(g))) \tag{21}$$

Proof. First, we show that the L_1-norm of BoN_Y and BoN_θ are both $T - n + 1$:

$$\sum_g \text{BoN}_Y(g) = \sum_{t=0}^{T-n} \sum_g 1\{y_{t+1:t+n} = g\} = T - n + 1$$

$$\sum_g \text{BoN}_\theta(g) = \sum_g \sum_Y P(Y|X,\theta) \cdot \text{BoN}_Y(g) \tag{22}$$

$$= \sum_Y P(Y|X,\theta) \cdot \sum_g \text{BoN}_Y(g) = T - n + 1$$

Algorithm 6 BoN-L_1

Input: probability distribution $p(\cdot|\mathbf{X}, \theta))$, reference sentence $\hat{Y}$, prediction length T, n
Output: BoN distance BoN-L_1
 1: construct the bag-of-n-grams BoN$_{\hat{Y}}$ for the reference sentence
 2: ref-n-grams = $\{g|\text{BoN}_{\hat{Y}}(g) \ != 0\}$
 3: match = 0
 4: **for** g in ref-n-grams **do**
 5:     calculate BoN$_\theta(g)$ according to Equation (19)
 6:     match += min(BoN$_\theta(g)$, BoN$_{\hat{Y}}(g)$)
 7: **end for**
 8: BoN-L_1 = $2 \cdot (T\text{-}n\text{+}1\text{-match})$
 9: **return** BoN-L_1

On this basis, we can convert BoN-L_1 to the following form:

$$\text{BoN-}L_1 = \sum_g |\text{BoN}_\theta(g) - \text{BoN}_{\hat{Y}}(g)|$$

$$= \sum_g (\text{BoN}_\theta(g) + \text{BoN}_{\hat{Y}}(g) - 2\min(\text{BoN}_\theta(g), \text{BoN}_{\hat{Y}}(g))) \qquad (23)$$

$$= 2(T - n + 1 - \sum_g \min(\text{BoN}_\theta(g), \text{BoN}_{\hat{Y}}(g)))$$

<div align="right">□</div>

The minimum between BoN$_\theta(g)$ and BoN$_{\hat{Y}}(g)$ can be understood as the number of matches for the n-gram g, and the L_1 distance measures the number of n-grams predicted by NAT that fails to match the reference sentence. Notice that the minimum will be nonzero only if the n-gram g appears in the reference sentence. Hence we can only focus on n-grams in the reference, which significantly reduces the computational effort and storage requirement. Algorithm 6 illustrates the calculation process of BoN-L_1.

We normalize distances to range $[0, 1]$ as training objectives. For BoW distances, we keep BoW-Cos unchanged and divide BoW-L_1 and BoW-L_2 by the constant $2T$:

$$\mathcal{L}_{\text{BoW-}L_1}(\theta) = \frac{\text{BoW-}L_1}{2T} \qquad \mathcal{L}_{\text{BoW-}L_2}(\theta) = \frac{\text{BoW-}L_2}{2T} \qquad (24)$$

For BoN, we divide BoN-L_1 by the constant $2(T - n + 1)$:

$$\mathcal{L}_{\text{BoN-}L_1}(\theta) = \frac{\text{BoN-}L_1}{2(T - n + 1)} \qquad (25)$$

3.3 Training Strategy

The reinforcement learning method can train NAT with any sequence-level objective that correlates well with the translation quality, but it requires a lot of calculations on CPU to reduce the variance of gradient estimation. The bag-of-n-grams method can efficiently calculate the BoN objective without doing any approximations, but the training objective is limited to the L_1 distance. The word-level cross-entropy loss cannot

evaluate the output of NAT properly, but it also has strengths like high-speed training and it is suitable for model warmup.

Therefore, we propose to use a three-stage training strategy to combine the strengths of the two training methods and the cross-entropy loss. First, we use the cross-entropy loss to pretrain the NAT model, and this process takes the most training steps. Then we use the bag-of-n-grams objective to finetune the pretrained model for a few training steps. Finally, we apply the reinforcement learning method to finetune the model to optimize the sequence-level objective, where this process takes the least training steps. There are also other training strategies like two-stage training and joint training, and we will show the efficiency of three-stage training in the experiment. The loss based on reinforcement learning or bag-of-n-grams can also be used alone to finetune the model pretrained by the cross-entropy loss. We will adopt this strategy when analyzing these methods separately.

4. Related Work

4.1 Non-Autoregressive Translation

Gu et al. (2018) proposed non-autoregressive translation to reduce the translation latency, which generates all target tokens simultaneously. Although accelerating the decoding process significantly, the acceleration comes at the cost of translation quality. Therefore, intensive efforts have been devoted to improving the performance of NAT, which can be roughly divided into the following categories.

Latent Variables. NAT suffers from the multimodality problem, which can be mitigated by introducing a latent variable to directly model the nondeterminism in the translation process. Gu et al. (2018) proposed to use fertility scores specifying the number of output words each input word generates to model the latent variable. Kaiser et al. (2018) autoencoded the target sequence into a sequence of discrete latent variables and decoded the output sequence from the latent sequence in parallel. Based on variational inference, Ma et al. (2019) proposed FlowSeq to model sequence-to-sequence generation using generative flow, and Shu et al. (2020) introduced LaNMT with continuous latent variables and deterministic inference. Bao et al. (2019); Ran et al. (2021) used the position information as latent variables to explicitly model the reordering information in the decoding procedure.

Decoding Methods. The fully non-autoregressive transformer generates all target words in one run, which suffers from the large performance degradation. Therefore, researchers were interested in alternative decoding methods that are slightly slower but can significantly improve the translation quality. Lee, Mansimov, and Cho (2018) proposed the iterative decoding based NAT model IRNAT to iteratively refine the translation where the outputs of the decoder are fed back as the decoder inputs in the next iteration. The pattern of iterative decoding was adopted by many non-autoregressive models. Ghazvininejad et al. (2019) and Kasai et al. (2020) refine model output iteratively by masking part of the translation and predicting the masks in each iteration. Gu, Wang, and Zhao (2019) introduced the Levenshtein Transformer to iteratively refine the translation with insertion and deletion operations. In addition to iterative decoding, Sun et al. (2019) proposed to incorporate the Conditional Random Fields in the top of NAT decoder to help the NAT decoding. Wang, Zhang, and Chen (2018) introduced the semi-autoregressive decoding mechanism that generates a group

of words each time. Ran et al. (2020) proposed another semi-autoregressive model named RecoverSAT, which generates a translation as a sequence of simultaneously generated segments.

Training Objectives. As the cross-entropy loss cannot evaluate NAT outputs properly, researchers attempt to improve the model performance by introducing better training objectives. In addition to sequence-level training (Shao et al. 2019, 2020), Wang et al. (2019) proposed the similarity regularization and reconstruction regularization to reduce errors of repeated and incomplete translations. Libovický and Helcl (2018); Saharia et al. (2020) applied the Connectionist Temporal Classification loss to marginalize out latent alignments using dynamic programming. Ghazvininejad et al. (2020) proposed the aligned cross-entropy loss, which uses a differentiable dynamic program based on the best monotonic alignment between target tokens and model predictions.

Other Improvements. Besides the above mentioned categories, some researchers improve the NAT performance from other perspectives. Guo et al. (2019) proposed to enhance the inputs of NAT decoder with phrase-table lookup and embedding mapping. Akoury, Krishna, and Iyyer (2019) introduced syntactically supervised Transformers, which first autoregressively predict a chunked parse tree and then generate all target tokens conditioned on it. Zhou and Keung (2020) proposed to improve NAT performance with source-side monolingual data. Shan, Feng, and Shao (2021) proposed to model the coverage information for NAT. Li et al. (2019); Wei et al. (2019) improve the performance of NAT by exploring better methods to learn from autoregressive models. Zhou, Gu, and Neubig (2020) investigated the knowledge distillation technique in NAT. Tu et al. (2020) introduced the energy-based inference networks as an alternative to knowledge distillation.

4.2 Sequence-Level Training for Autoregressive NMT

Neural machine translation models are usually trained with the word-level loss under the teacher forcing algorithm (Williams and Zipser 1989), which forces the model to generate the next word based on the previous ground-truth words other than the model outputs during the training. However, this training method suffers from the *exposure bias* problem (Ranzato et al. 2016) because the model is exposed to different data distributions during training and inference. To alleviate the exposure bias problem, some researchers improve the teacher forcing algorithm to professor forcing (Goyal et al. 2016) or seer forcing (Feng et al. 2021). Scheduled sampling (Bengio et al. 2015; Venkatraman, Hebert, and Bagnell 2015) is the direct solution for exposure bias, which attempts to alleviate the exposure bias problem through mixing ground-truth words and previously predicted words as inputs during the training. However, the generated sequence may not be aligned with the target sequence, which is inconsistent with the word-level loss. Therefore, it is a natural solution to apply sequence-level training to eliminate the exposure bias in the autoregressive NMT.

Sequence-level training objectives are usually non-differentiable, and reinforcement learning techniques (Williams 1992; Sutton et al. 1999) are widely applied to train autoregressive NMT to optimize discrete objectives. Ranzato et al. (2016) first pointed out the exposure bias problem and proposed the MIXER algorithm to alleviate the exposure bias, which is a combination of the word-level cross-entropy loss and the sequence-level loss optimized by the REINFORCE algorithm. Bahdanau et al. (2017) presented an approach to training neural networks to generate sequences using actor-critic methods

from reinforcement learning. He et al. (2016) proposed a dual learning approach to train the forward NMT model using reward signals provided by the backward model. Wu et al. (2016) introduced a new sequence evaluation metrics GLEU for the sequence-level training of Google's NMT System. Yu et al. (2017) proposed a sequence generation framework called SeqGAN to overcome the differentiable difficulty of GAN through reinforcement learning, which is then applied by Wu et al. (2018b), Yang et al. (2018) to train NMT under the generator-discriminator framework. Wu et al. (2018a) conducted a systematic study on the reinforcement learning based training method for NMT.

In addition to reinforcement learning based methods, there are also some approaches that can train NMT with sequence-level objectives. Shen et al. (2016) introduced Minimum Risk Training for NMT to minimize the expected risk on training data. Norouzi et al. (2016) proposed Reward Augmented Maximum Likelihood to incorporate sequence-level reward into a maximum likelihood framework. Edunov et al. (2018) surveyed a range of classical objective functions and applied them to neural sequence to sequence models. Ma et al. (2018) proposed to optimize NMT by the bag-of-words training objective. Shao, Chen, and Feng (2018) introduced probabilistic *n*-gram matching to transform the discrete sequence-level objective into the differentiable form.

As shown above, sequence-level training has attracted much attention of researchers and has been deeply studied on autoregressive models. However, though sequence-level training is more essential on non-autoregressive models, its application on NAT has not been well studied before.

5. Experiments

5.1 Setup

Data Sets. We evaluate our proposed methods on four translation tasks: WMT14 English↔German (WMT14 En↔De) and WMT16 English↔Romanian (WMT16 En↔Ro). We use the standard tokenized BLEU (Papineni et al. 2002) to evaluate the translation quality. For WMT14 En↔De, we use the WMT 2016 corpus consisting of 4.5M sentence pairs for the training. The validation set is `newstest2013` and the test set is `newstest2014`. We learn a joint BPE model (Sennrich, Haddow, and Birch 2016) with 32K operations to process the data and share the vocabulary for source and target languages. For WMT16 En↔Ro, we use the WMT 2016 corpus consisting of 610K sentence pairs for the training. We take `newsdev-2016` and `newstest-2016` as development and test sets. We learn a joint BPE model with 32K operations to process the data and share the vocabulary for source and target languages.

Knowledge Distillation. Knowledge distillation (Hinton, Vinyals, and Dean 2015; Kim and Rush 2016) is proved to be crucial for successfully training NAT models. For all the translation tasks, we first train an autoregressive model as the teacher and apply sequence-level knowledge distillation to construct the distillation corpus where the target side of the training corpus is replaced by the output of the autoregressive Transformer model. We use the distillation corpora to train NAT models.

Baselines. We take the base version of Transformer (Vaswani et al. 2017) as our autoregressive baseline as well as the teacher model. The NAT baseline takes the same structure as the base Transformer except that we modify the attention mask of the decoder, so it does not mask out the future tokens. We perform uniform copy from source embeddings (Gu et al. 2018) to construct decoder inputs. We use a target length

predictor to predict the length of the target sentence, which takes the encoder hidden states as inputs and feeds it to a softmax classifier after an affine transformation. We use the golden length during the training and the predicted length during the inference.

Rescoring. For inference, we follow the common practice of noisy parallel decoding (Gu et al. 2018), which generates a number of decoding candidates in parallel and selects the best translation via rescoring with the autoregressive teacher. We generate multiple translation candidates by predicting the target length T and generate translations with lengths ranging from $[T - B, T + B]$, where B is the beam size. The autoregressive teacher calculates the cross-entropy loss of the $n = 2B + 1$ translations and selects the translation with the lowest loss.

Hyperparameters. In the main experiments, we set the top-k size k to 5, the sampling times n to 10, and the N in BoN to 2. We use the ROUGE-2 score as the reward in reinforcement learning. In the pretraining stage of the three-stage training, the number of training steps is 300k for WMT14 En↔De and 150k for WMT16 En↔Ro. In the second stage, we use the BoN objective to finetune the model for 3k steps. In the final stage, we use sequence-level evaluation metrics to finetune the model for 300 steps. The batch size is 128k for pretraining and 512k for finetuning. For WMT14 En↔De, we use a dropout rate of 0.3 during the pretraining and a dropout rate of 0.1 during the finetuneing. For WMT16 En↔Ro, we use a dropout rate of 0.3 during the pretraining and finetuneing. We also use 0.01 L_2 weight decay and label smoothing with $\epsilon = 0.1$ for regularization. We follow the weight initialization schema from BERT (Devlin et al. 2019). All models are optimized with Adam (Kingma and Ba 2014) with $\beta = (0.9, 0.98)$ and $\epsilon = 10^{-8}$. The learning rate warms up to $5 \cdot 10^{-4}$ within 10k steps, and then decays with the inverse square-root schedule. We use 8 GeForce RTX 3090 GPUs for the training.

5.2 Main Results

We first compare the performance of our proposed methods, including the Reinforcement Learning (RL) based methods (i.e., Reinforce-Base, Reinforce-Step, Reinforce-Top-k, and Traverse-Ref) and the Bag-of-N-grams (BoN) based methods (i.e., BoW-*Cos*, BoW-L_2, BoW-L_1, and BoN-L_1 (N = 2)). We also adopt the three-stage training strategy to combine the best performing methods of the above two categories (i.e., Traverse-Ref and BoN-L_1 (N = 2)), which is denoted as BoN+RL. Table 3 reports the experiment results of our methods, from which we have the following observations.

1. Sequence-level training can effectively improve the performance of non-autoregressive models. All the methods listed in Table 3 can greatly improve the translation quality of non-autoregressive models. Even the simplest method Reinforce-Base achieves more than 3 BLEU improvements on the WMT14 data set, indicating that sequence-level training is very suitable for non-autoregressive models.

2. The methods we propose for variance reduction are helpful to enhance the performance of the reinforcement learning. Comparing the reinforcement learning based methods, Reinforce-Step reduces the estimation variance by replacing the sentence reward with step reward, which improves Reinforce-Base by about 0.5 BLEU. Reinforce-Top-k further improves Reinforce-Step by about 0.8 BLEU by eliminating the variance of important words. Finally, Traverse-Ref gives a method to traverse the whole search space for Reference-Based rewards, which improves Reinforce-Top-k by about

0.4 BLEU. In summary, the methods we propose for variance reduction are helpful to enhance the performance of reinforcement learning.

3. Among the three BoW training objectives, the L_1 distance is very suitable for the training of non-autoregressive models. Comparing the three Bag-of-Words objectives, BoW-L_1 achieves the best performance and largely outperforms the other two objectives, indicating that the L_1 distance of BoW is very suitable for the training of non-autoregressive models. Regarding the bag-of-n-grams objective, the main limitation is that many distance metrics like L_2 distance and cosine distance cannot be calculated, and the observation on BoW can alleviate this concern to some extent.

4. Three-stage training can effectively combine reinforcement learning and bag-of-n-grams. Three-stage training achieves the best performance by combining the best methods of the two categories (i.e., Traverse-Ref and BoN-L_1 (N = 2)), which improves the NAT baseline by more than 5 BLEU scores on the WMT14 data set and more than 2 BLEU scores on the WMT16 data set. We use Seq-NAT to represent this method.

5.3 Sequence-Level Training for Iterative NAT

In the previous section, we have verified the effect of sequence-level training on the vanilla NAT, which is non-iterative and uses a single forward pass during the decoding. In this section, we conduct experiments to evaluate the effect of sequence-level training on iterative NAT, which is an important class of NAT models. We use the Conditional Masked Language Model (CMLM) with mask-predict decoding (Ghazvininejad et al. 2019) as our baseline model, which is a strong iterative NAT model. We apply sequence-level training to finetune the CMLM baseline and call this method Seq-CMLM. Figure 2 shows the BLEU scores of CMLM and Seq-CMLM under a different number of iterations.

Table 3
The performance (test set BLEU) of our methods on all of our benchmarks. All models except Transformer are purely non-autoregressive, using a single forward pass during the argmax decoding. NAT-Base is the baseline NAT model. All other NAT models are finetuned from the NAT-Base.

|	Model	Speedup	WMT14		WMT16	
			EN-DE	DE-EN	EN-RO	RO-EN
Base	Transformer	1.0×	27.42	31.63	34.18	33.72
	NAT-Base	15.6×	19.51	24.47	28.89	29.35
RL	Reinforce-Base	15.6×	23.23	27.59	29.67	29.85
	Reinforce-Step	15.6×	23.76	28.08	30.14	30.31
	Reinforce-Top-k	15.6×	24.68	29.05	30.73	30.88
	Traverse-Ref	15.6×	25.15	29.50	31.12	31.34
BoN	BoW-Cos	15.6×	23.90	28.11	29.72	29.69
	BoW-L_2	15.6×	24.22	29.03	30.08	29.93
	BoW-L_1	15.6×	24.75	29.41	31.01	31.19
	BoN-L_1 (N = 2)	15.6×	25.28	29.66	31.37	31.51
3-Stage	RL+BoN	15.6×	**25.54**	**29.91**	**31.69**	**31.78**
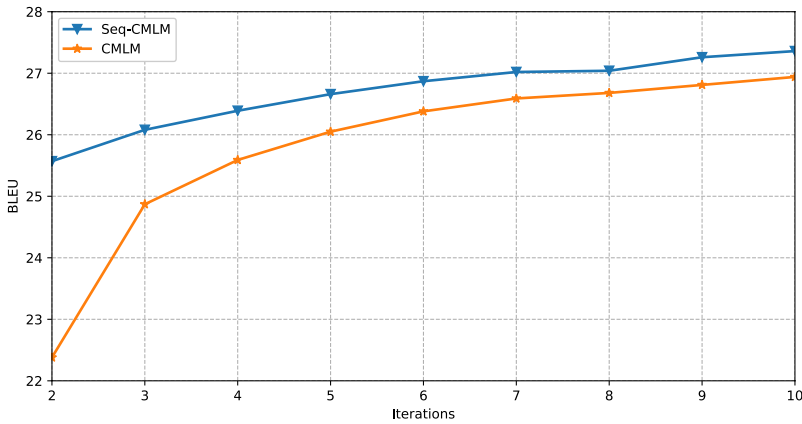
Figure 2
BLEU scores of CMLM and Seq-CMLM on the test set of WMT14 En-De under a different number of iterations. For the elegance of the figure, we removed the 1 iteration result, which is 17.47 for CMLM and 24.90 for Seq-CMLM.

From Figure 2, we can see that Seq-CMLM consistently outperforms CMLM on all numbers of iterations. Even with 10 iterations, Seq-CMLM can achieve an improvement of 0.42 BLEU to CMLM, reaching a BLEU score of 27.36, showing that sequence-level training is also very effective on Iterative NAT.

5.4 Speedup in Batch Decoding

Non-autoregressive models have high speedup in sentence by sentence translation, but this advantage will gradually decrease when we increase the size of decoding batch, making the advantage of NAT in practical application questioned. We resolve this concern by measuring the translation latency of NAT and AT models under different sizes of decoding batches. We conduct experiments on the test set of WMT14 En→De and show the results in Figure 3.
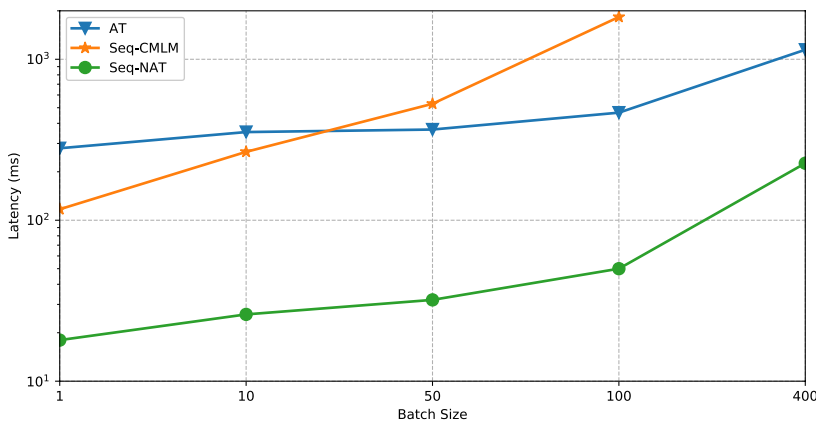

Figure 3
The translation latency of AT, Seq-NAT, and Seq-CMLM measured on the test set of WMT14 En→De. The decoding batch of 400 sentences is not applicable to Seq-CMLM due to the memory limit.

Table 4
The Pearson correlation between loss functions and translation quality. $n = k$ represents the
bag-of-k-grams training objective. CE represents the cross-entropy loss.

Loss function	CE	$n = 2$	$n = 3$	$n = 4$
Correlation	0.56	0.87	0.84	0.79

From Figure 3, as the size of decoding batch increases, both NAT models have
higher translation latency. Notably, the iterative model Seq-CMLM becomes even much
slower than the autoregressive model when using large batch size. On the contrary, the
one-iteration model Seq-NAT still maintains more than $5\times$ speedup during the batch
decoding, demonstrating the efficiency of non-autoregressive generation.

5.5 Correlation with Translation Quality

In this section, we conduct experiments to analyze the correlation between loss func-
tions and the translation quality. We are interested in how the cross-entropy loss and
BoN objective correlate with the translation quality. We do not analyze the reinforce-
ment learning based methods because they do not calculate the loss function, but
directly estimate the gradient of the loss. We use the GLEU score to represent the
translation quality, which is more accurate than BLEU in sentence-level evaluation (Wu
et al. 2016). We conduct experiments on the validation set of WMT14 En→De, which
contains 3,000 sentences. First, we load the NAT-Base model and calculate the loss of
every sentence in the validation set. Then we use the NAT-Base model to decode the
validation set and calculate the GLEU score of every sentence. Finally, we calculate the
Pearson correlation between the 3,000 GLEU scores and losses.

For the cross-entropy loss, we normalize it by the target sentence length. The BoN
training objective is the L_1 distance normalized by $2(T - n + 1)$. We respectively set n to
2, 3, and 4 to test different n-gram sizes. Table 4 lists the correlation results.

From Table 4, we can see that all three BoN objectives outperform the cross-entropy
loss by large margins, and the $n = 2$ setting achieves the highest correlation 0.87. To
find out where the improvements come from, we analyze the effect of sentence length
in the following experiment. We evenly divide the data set into two parts according to
the source length. The first part consists of 1,500 short sentences and the second part
consists of 1,500 long sentences. We respectively measure the Pearson correlation on the
two parts and report the results in Table 5:

From Table 5, we can see that the correlation of the cross-entropy loss drops as
the sentence length increases, where the BoN objective still has a strong correlation on
long sentences. The reason is not difficult to explain. The cross-entropy loss requires the
strict alignment between the translation and reference. As the sentence length grows,
it becomes harder for NAT to align the translation with the reference, which leads to a
decrease of correlation between cross-entropy loss and translation quality. In contrast,
the BoN objective is robust to unaligned situations, so its correlation with translation
quality stays strong when translating long sentences.

5.6 Effect of Training Strategy

In this section, we analyze the effect of training strategy that combines the word-level
loss and the two methods based on reinforcement learning and bag-of-n-grams. Before

Table 5
The Pearson correlation between loss functions and translation quality on short sentences and long sentences.

	all	short	long
Cross-Entropy	0.56	0.68	0.44
BoN ($n = 2$)	0.87	0.89	0.86

Table 6
The training time required for different methods to process 64k tokens. The time is measured on the training set of WMT14 En-De with a single GeForce RTX 3090 GPU. CE is the cross-entropy loss. RF is the abbreviation of Reinforce.

Method	CE	BoW	BoN ($n = 2$)	RF-Base	RF-Step	RF-Top-k	Traverse-Ref
Time	1.2s	1.5s	7.1s	2.2s	16.9s	31.3s	63.2s

Table 7
Validation BLEU scores and training time of different training strategies on WMT14 En-De. CE represents the cross-entropy loss and TR represents the Traverse-Ref loss. The training time is measured on 8 GeForce RTX 3090 GPUs.

Strategy	BLEU	Time
1. CE—BoN+TR	24.56	65.6h
2. CE+BoN—BoN—TR	24.82	93.1h
3. CE—TR—BoN	24.63	63.3h
4. CE—BoN—TR	24.69	37.5h

discussing the training strategy, we first give the training speed of each method in Table 6. As we can see, Traverse-Ref is the slowest method, which is nearly 10 times slower than BoN. Therefore, when choosing a training strategy, it is necessary to avoid a large number of calculations of Traverse-Ref.

We consider four training strategies that involve the word-level cross-entropy loss, the Traverse-Ref loss and the BoN loss. First, we consider the two-stage strategy that uses the cross-entropy loss for pretraining and finetunes the model by the weighted summation of the Traverse-Ref and BoN losses. The second strategy follows the joint training strategy in Shao et al. (2020) to combine the BoN and cross-entropy loss for pretraining, and then finetunes the model sequentially by BoN and Traverse-Ref. The latter two strategies adopt the three-stage strategy that uses the cross-entropy loss for pretraining and sequentially uses Traverse-Ref and BoN for finetuning. We report the BLEU scores of the four strategies together with the training time in Table 7.

Table 7 shows that the second strategy achieves the best performance but suffers from high training cost. The fourth strategy is more economical; it achieves a slightly lower BLEU but greatly shortens the training time. Compared with the other two strategies, it outperforms them on both BLEU and training time. Therefore, we finally adopt the fourth strategy to combine the word-level training and sequence-level training methods.
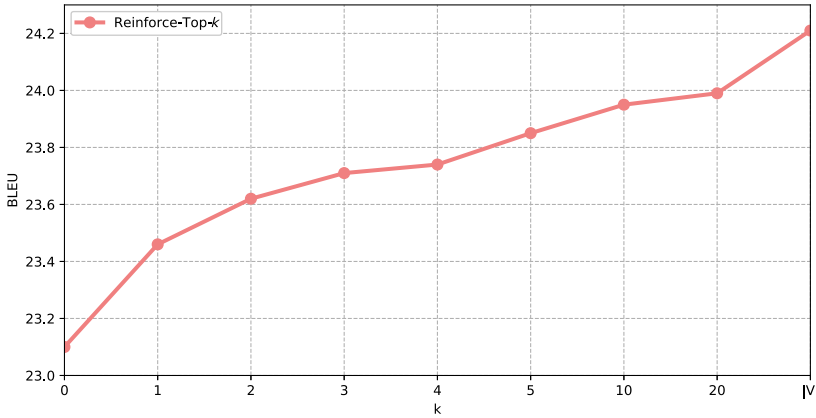
Figure 4
BLEU scores of Reinforce-Top-k with different k on the validation set of WMT14 En-De.

5.7 Effect of Hyperparameters

In this section, we analyze the effect of some hyperparameters in our method that will affect the model performance, including the top-k size k and the reward function in reinforcement learning, the n-gram size n in bag-of-n-grams training, and the batch size for finetuning.

Top-k Size. Reinforce-Top-k is proposed to reduce the estimation variance by traversing the top-k words, which is important in the gradient estimation. Intuitively, a larger l will make the model stronger. When k is 0, Reinforce-Top-k degenerates to Reinforce-Step. When k equals the vocabulary size $|V|$, Reinforce-Top-k has the same performance with Traverse-Ref. However, using such a large k will make the training very slow. Therefore, we need to find an appropriate k to balance the performance and training cost. We conduct experiments on the validation set of WMT14 En-De to see the effect of top-k size k and illustrate our results in Figure 4.

From Figure 4, we can see that the model performance steadily improves as k rises from 0 to 5. When k rises from 5 to 10, the model performance is also slightly improved. However, we can barely see improvements from $k = 10$ to $k = 20$, showing that the appropriate k is between 5 to 10. In addition, we use Traverse-Ref to show the $k = |V|$ result, which achieves considerable improvements to Reinforce-Top-k.

Reward Function. The performance of reinforcement learning based methods is influenced by the reward function it uses. Our methods have almost no restriction on the reward function, where only Traverse-Ref requires the reward function to Reference-Based. Therefore, we choose three widely used Reference-Based rewards BLEU (Papineni et al. 2002), GLEU (Wu et al. 2016), and ROUGE-2 (Lin 2004) as candidates. We use the three rewards to finetune the NAT baseline and report their results in Table 8. We also directly evaluate the three rewards by the Pearson correlation coefficient with translation quality. We use the WMT16 DAseg De-En data set for evaluation, which consists of 560 source sentences, model translations, reference sentences, and human scores. We obtain the rewards of the model translations and calculate the Pearson correlation coefficient between rewards and human scores, as shown in Table 8. We can see that there is no significant difference in the BLEU performance of these three

Table 8
BLEU scores on the validation set of WMT14 En-De when using BLEU, GLEU, or ROUGE-2 as reward functions, and the Pearson correlation coefficient of these rewards.

	Correlation	Reinforce-Base	Reinforce-Step	Traverse-Ref	Average
BLEU	0.389	22.58	23.01	24.12	23.24
GLEU	0.482	22.56	23.17	24.16	23.30
ROUGE-2	0.483	22.39	23.14	24.25	23.26

Table 9
Validation BLEU scores of BoN-L_1 with different n on WMT14 En-De and the time required to process 64k tokens during the training. The time is measured with a single GeForce RTX 3090 GPU.

n	$n = 2$	$n = 3$	$n = 4$
BLEU	24.37	24.29	24.07
Time	7.1s	9.7s	12.3s

rewards. In terms of the correlation, BLEU underperforms ROUGE-2 and GLEU by a large margin, which is possibly due to instability of BLEU as there are usually little matches of 3-gram or 4-gram in sentence-level evaluation. We finally use the ROUGE-2 as the reward function because of its overall performance and fast calculation in our implementation.

N-gram Size. Table 3 has shown that the bag-of-n-grams ($N = 2$) objective outperforms the bag-of-words objective, but the effect of different n-gram sizes n has not been analyzed. Therefore, we conduct experiments on the validation set of WMT14 En-De to see the performance of bag-of-n-grams objectives with different choices of n, and we also provide the training speed of BoN-L_1 with different n. Results are listed in Table 9. We can see that $n = 2$ slightly outperforms other choices of n, which is consistent with the correlation result in Table 4. Furthermore, BoN-L_1 with $n = 2$ is much faster than other choices of n during the training, so we set $n = 2$ in the main experiment.

Batch Size for Finetuning. In the training of deep neural models, a larger batch size usually leads to stronger performance, which comes with the cost of greater training costs. In the sequence-level training scenario, since we only need to finetune the model for a few steps, we can increase the batch size within a reasonable range, which only slightly increases the training cost but brings considerable improvements on the model performance. To show the effect of the batch size, we use different batch sizes during the BoN finetuning and report the corresponding BLEU scores and total training time in Table 10. We can see that the BLEU score steadily increases as the batch size for finetuning increases. In terms of training time, even when we use a batch size of 512k, which is 4 times the size of the pretraining, the training time is only 1.25 times the NAT baseline.

5.8 Effect of Sentence Length

In Section 5.5, we analyze the correlation between loss functions and the translation quality under different sentence lengths, which shows that sequence-level losses greatly

Table 10
Validation BLEU scores on WMT14 En-De and training costs when using different batch size during the finetuning. NoFT represents the NAT baseline without finetuning. The training time is measured on 8 GeForce RTX 3090 GPUs.

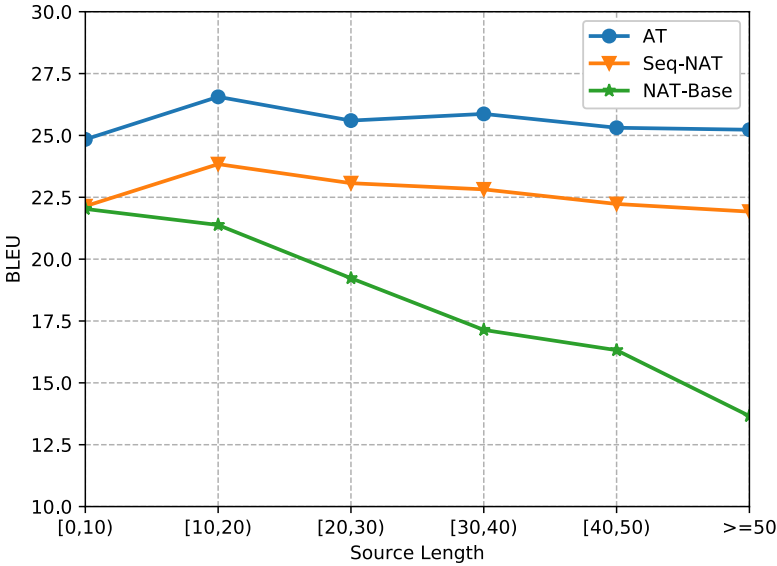| Batch Size	NoFT	32k	64k	128k	256k	512k
BLEU	19.08	23.77	23.93	24.15	24.27	24.37
Time		24.8h	25.2h	25.6h	26.3h	27.9h	31.0h

Figure 5
Validation BLEU scores of baseline methods and Seq-NAT on WMT14 En-De under different length buckets.

outperform the word-level loss in terms of correlation when evaluating long sentences. In this section, we calculate the BLEU performance of baseline methods and our model on different sentence lengths and see whether the better correlation contributes to better BLEU performance. We conduct the experiment on the validation of WMT14 En→De and divide the sentence pairs into different length buckets according to the length of the source sentence. We use Seq-NAT to represent our best performing method, and calculate the BLEU scores of baseline models and Seq-NAT under different length buckets. The results are shown in Figure 5.

From Figure 5, we can see that NAT-Base and Seq-NAT have similar performance when translating short sentences. However, the translation quality of NAT-Base drops quickly as sentence length increases, where the autoregressive Transformer and Seq-NAT have stable performance over different sentence lengths, which is in good agreement with the correlation results. As the sentence length grows, the correlation between the cross-entropy loss and the translation quality drops, which leads to the weakness of NAT in translating long sentences. On the contrary, sequence-level losses evaluate the translation quality of long sentences with high correlations, so Seq-NAT has stable performance on long sentences.

Table 11
Performance comparison between our method Seq-NAT and existing methods. The speedup is measured on the WMT14 En-De test set with batch size 1. "—" indicates that the result is not reported. n is the number of candidates rescored by the autoregressive teacher.

Model	WMT14		WMT16		Speedup
	EN-DE	DE-EN	EN-RO	RO-EN	
Autoregressive					
Transformer	27.42	31.63	34.18	33.72	1.0×
Non-Autoregressive w/o Rescoring					
NAT-FT (Gu et al. 2018)	17.69	21.47	27.29	29.06	15.6×
LT (Kaiser et al. 2018)	19.80	—	—	—	3.8×
CTC (Libovický and Helcl 2018)	17.68	19.80	19.93	24.71	—
ENAT (Guo et al. 2019)	20.65	23.02	30.08	—	25.3×
NAT-REG (Wang et al. 2019)	20.65	24.77	—	—	27.6×
NAT-Hints (Li et al. 2019)	21.11	25.24	—	—	30.8×
Reinforce-NAT (Shao et al. 2019)	19.15	22.52	27.09	27.93	10.77×
BoN-Joint+FT (Shao et al. 2020)	20.90	24.61	28.31	29.29	10.73×
imitate-NAT (Wei et al. 2019)	22.44	25.67	28.61	28.90	18.6×
FlowSeq (Ma et al. 2019)	23.72	28.39	29.73	30.72	—
NART-DCRF (Sun et al. 2019)	23.44	27.22	—	—	10.4×
ReorderNAT (Ran et al. 2021)	22.79	27.28	29.30	29.50	16.1×
PNAT (Bao et al. 2019)	23.05	27.18	—	—	7.3×
FCL-NAT (Junliang et al. 2020)	21.70	25.32	—	—	28.9×
AXE (Ghazvininejad et al. 2020)	23.53	27.90	30.75	31.54	—
EM (Sun and Yang 2020)	24.54	27.93	—	—	16.4×
Imputer (Saharia et al. 2020)	**25.80**	28.40	**32.30**	31.70	—
Seq-NAT (ours)	25.54	**29.91**	31.69	**31.78**	15.6×
Non-Autoregressive w/ Rescoring					
NAT-FT (n = 10) (Gu et al. 2018)	18.66	22.41	29.02	30.76	7.68×
NAT-FT (n = 100) (Gu et al. 2018)	19.17	23.20	29.79	31.44	2.36×
LT (n = 10) (Kaiser et al. 2018)	21.0	—	—	—	—
ENAT (n = 9) (Guo et al. 2019)	24.28	26.10	**34.51**	—	12.4×
NAT-REG (n = 9) (Wang et al. 2019)	24.61	28.90	—	—	15.1×
NAT-Hints (n = 9) (Li et al. 2019)	25.20	29.52	—	—	17.8×
imitate-NAT (n = 7) (Wei et al. 2019)	24.15	27.28	31.45	31.81	9.70×
FlowSeq (n = 15) (Ma et al. 2019)	25.03	30.48	31.89	32.43	—
NART-DCRF (n = 9) (Sun et al. 2019)	26.07	29.68	—	—	6.14×
PNAT (n = 7) (Bao et al. 2019)	—	27.90	—	—	3.7×
FCL-NAT (n = 9) (Junliang et al. 2020)	25.75	29.50	—	—	16.0×
EM (n = 9) (Sun and Yang 2020)	25.75	29.29	—	—	9.14×
Seq-NAT (n = 9, ours)	**26.35**	**30.70**	33.21	**33.28**	9.0×

5.9 Performance Comparision

We use Seq-NAT to represent our best performing method. In Table 11, we compare the performance of Seq-NAT against the autoregressive Transformer and strong non-iterative NAT baseline models. Table 11 shows that Seq-NAT outperforms most existing NAT systems, and the performance gap between Seq-NAT and the autoregressive teacher is about 2 BLEU on average. Rescoring 9 candidates further improves the translation quality and narrows the performance gap to about 0.8 BLEU on average. It

Table 12
Three translation cases in the validation set of WMT14 De-En. Source and Target are, respectively, the source sentence and reference sentence. AT is the output of the autoregressive Transformer. NAT-Base is the output of the NAT baseline. Seq-NAT is the output of our model.

Source	Es gibt Krebsarten, die aggressiv und andere, die indolent sind .
Target	There are aggressive cancers and others that are indolent .
AT	There are cancers that are aggressive and others that are indolent .
NAT-Base	There are cancers cancer aggressive aggressive others are indindent .
Seq-NAT	There are cancers that are aggressive and others that indolent .

Source	Wir wissen ohne den Schatten eines Zweifels, dass wir ein echtes neues Teilchen haben, und dass es dem vom Standardmodell vorausgesagten Higgs-Boson stark Ãd'hnelt .
Target	We know without a shadow of a doubt that it is a new authentic particle, and greatly resembles the Higgs boson predicted by the Standard Model .
AT	We know without the shadow of a doubt that we have a real new particle, and that it is very similar to the Higgs Boson predicted by the standard model .
NAT-Base	We know without without shadow shadow of doubt doubt that we have a new particle le that it is very similar similar to HiggsgsBoson predicted by the standard model .
Seq-NAT	We know without the shadow of a doubt that we have a real new particle and that it is very similar to the Higgs-Boson predicted by the standard model .

Source	und noch tragischer ist, dass es Oxford war - eine UniversitÃd't, die nicht nur 14 Tory-Premierminister hervorbrachte, sondern sich bis heute hinter einem unverdienten Ruf von Gleichberechtigung und Gedankenfreiheit versteckt .
Target	even more tragic is that it was Oxford, which not only produced 14 Tory prime ministers, but, to this day, hides behind an ill-deserved reputation for equality and freedom of thought .
AT	And more tragically, it was Oxford - a university that not only produced 14 Tory prime ministers but hides to this day behind an undeserved reputation of equality and freedom of thought .
NAT-Base	More more tragic tragic it was Oxford Oxford Oxford university university university not only 14 14 14 Torprime prime ministers but but hihihihiday behind an unved call for equality and freedom of thought .
Seq-NAT	More is even more tragic that it was Oxford - a university that not only produced 14 Tory prime ministers, but continues continues far hidden behind an unved call for equality and freedom of thought .

is also worth noting that Seq-NAT does not affect the translation speed, which has the same speedup 15.6× as NAT-Base. After rescoring 9 candidates, Seq-NAT still maintains 9.0× speedup.

5.10 Case Study

In Table 12, we present three translation cases from the validation set of WMT14 De-En to analyze how sequence-level training improves the translation quality of NAT. We can see from the three cases that the NAT baseline suffers from over-translation and under-translation errors especially when translating long sentences. The output of NAT-Base contains many repeated translations like "aggressive," "shadow," and "14." Additionally, the translation is incomplete since much information is missing. As we mentioned before, this is due to the limitation of the word-level cross-entropy loss we use, which evaluates the generation quality of each position independently and does not model the target-side sequential dependency, making NAT only focus on local correctness and ignore the overall translation quality.

When we look at the translation results of Seq-NAT, we can see that the errors of over-translation and under-translation are significantly reduced. Although there are still a few repeated translations when translating long sentences, the translation results are basically accurate and comparable to the autoregressive Transformer. Compared with the NAT baseline, Seq-NAT focuses more on the overall accuracy after the sequence-level training, which greatly improves the translation quality.

6. Conclusion

Non-autoregressive translation achieves significant decoding speedup through generating target words independently and simultaneously. However, the word-level cross-entropy loss cannot evaluate the output of NAT properly. As a result, NAT has a relatively low translation quality and tends to generate translations with over-translation and under-translation errors. In this article, we propose to train NAT with sequence-level training objectives. First, we propose to train NAT to optimize the sequence-level evaluation metric based on novel reinforcement algorithms customized for NAT. Then we introduce a novel bag-of-n-grams objective for NAT, which is differentiable and can be calculated efficiently. Finally, we use a three-stage training strategy to combine the strengths of the two training methods and the word-level loss. Experimental results show that our method achieves remarkable performance on all translation tasks.

References

Akoury, Nader, Kalpesh Krishna, and Mohit Iyyer. 2019. Syntactically supervised transformers for faster neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1269–1281, Florence. https://doi.org/10.18653/v1/P19-1122

Bahdanau, Dzmitry, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR* 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Bao, Yu, Hao Zhou, Jiangtao Feng, Mingxuan Wang, Shujian Huang, Jiajun Chen, and LI

Lei. 2019. Non-autoregressive transformer by position learning. *arXiv preprint arXiv:1911.10677*.

Bengio, Samy, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1171–1179, Cambridge, MA.

Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha. `https://doi.org/10.3115/v1/D14-1179`

Devlin, Jacob, 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, MN.

Edunov, Sergey, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364, New Orleans, LA. `https://doi.org/10.18653/v1/N18-1033`

Feng, Yang, Shuhao Gu, Dengji Guo, Zhengxin Yang, and Chenze Shao. 2021. Guiding teacher forcing with seer forcing for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2862–2872, Online. `https://doi.org/10.18653/v1/2021.acl-long.223`

Gehring, Jonas, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning -*

Volume 70, ICML'17, pages 1243–1252, JMLR.org.

Ghazvininejad, Marjan, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020. Aligned cross entropy for non-autoregressive machine translation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119, of *Proceedings of Machine Learning Research*, pages 3515–3523.

Ghazvininejad, Marjan, 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. `https://doi.org/10.18653/v1/D19-1633`

Goyal, Anirudh, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4601–4609.

Gu, Jiatao, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Gu, Jiatao, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11179–11189.

Guo, Junliang, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. 2019. Non-autoregressive neural machine translation with enhanced decoder input. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3723–3730. `https://doi.org/10.1609/aaai.v33i01.33013723`

He, Di, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems 29*, pages 820–828.

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Joachims, Thorsten. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML'98, pages 137–142. https://doi.org/10.1007/BFb0026683

Junliang, Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2020. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:7839–7846. https://doi.org/10.1609/aaai.v34i05.6289

Kaiser, Lukasz, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2390–2399, Stockholm.

Kasai, Jungo, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Non-autoregressive machine translation with disentangled context transformer. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5144–5155.

Kim, Yoon and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, TX. https://doi.org/10.18653/v1/D16-1139

Kingma, Diederik P. and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lee, Jason, 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. https://doi.org/10.18653/v1/D18-1149

Li, Bofang, Zhe Zhao, Tao Liu, Puwei Wang, and Xiaoyong Du. 2016. Weighted neural bag-of-n-grams model: New baselines for text classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1591–1600, Osaka.

Li, Zhuohan, Zi Lin, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Hint-based training for non-autoregressive machine 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 5708–5713, Hong Kong. https://doi.org/10.18653/v1/D19-1573

Libovický, Jindřich and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels. https://doi.org/10.18653/v1/D18-1336

Lin, Chin Yew. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona.

Ma, Shuming, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 332–338, Melbourne. https://doi.org/10.18653/v1/P18-2053

Ma, Xuezhe, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. 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 4282–4292, Hong Kong. https://doi.org/10.18653/v1/D19-1437

Ng, Andrew Y., Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 278–287.

Norouzi, Mohammad, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1731–1739.

Pang, Bo, Lillian Lee, and Shivakumar
   Vaithyanathan. 2002. Thumbs up?
   Sentiment classification using machine
   learning techniques. In *Proceedings of the
   2002 Conference on Empirical Methods in
   Natural Language Processing (EMNLP 2002)*,
   pages 79–86. https://doi.org/10.3115
   /1118693.1118704

Papineni, Kishore, 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, PA. https://doi.org/10
   .3115/1073083.1073135

Ran, Qiu, Yankai Lin, Peng Li, and Jie Zhou.
   2020. Learning to recover from
   multi-modality errors for
   non-autoregressive neural machine
   translation. In *Proceedings of the 58th
   Annual Meeting of the Association for
   Computational Linguistics*, pages 3059–3069,
   Online. https://doi.org/10.18653/v1
   /2020.acl-main.277

Ran, Qiu, Yankai Lin, Peng Li, and Jie Zhou.
   2021. Guiding non-autoregressive neural
   machine translation decoding with
   reordering information. In *Thirty-Fifth
   AAAI Conference on Artificial Intelligence,
   AAAI 2021, Virtual Event, February 2-9,
   2021*, pages 13727–13735.

Ranzato, Marc'Aurelio, Sumit Chopra,
   Michael Auli, and Wojciech Zaremba. 2016.
   Sequence level training with recurrent
   neural networks. In *4th International
   Conference on Learning Representations, ICLR
   2016, San Juan, Puerto Rico, May 2-4, 2016,
   Conference Track Proceedings*.

Saharia, Chitwan, William Chan, Saurabh
   Saxena, and Mohammad Norouzi. 2020.
   Non-autoregressive machine translation
   with latent alignments. In *Proceedings of the
   2020 Conference on Empirical Methods in
   Natural Language Processing (EMNLP)*,
   pages 1098–1108. Online. https://doi
   .org/10.18653/v1/2020.emnlp-main.83

Sennrich, Rico, 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. https://doi.org
   /10.18653/v1/P16-1162

Shan, Yong, Yang Feng, and Chenze Shao.
   2021. Modeling coverage for
   non-autoregressive neural machine
   translation. *arXiv preprint arXiv:2104.11897.*

https://doi.org/10.18653/v1/D18
   -1510

Shao, Chenze, Xilin Chen, and Yang Feng.
   2018. Greedy search with probabilistic
   n-gram matching for neural machine
   translation. In *Proceedings of the 2018
   Conference on Empirical Methods in Natural
   Language Processing*, pages 4778–4784,
   Brussels. https://doi.org/10.18653
   /v1/D18-1510

Shao, Chenze, Yang Feng, Jinchao Zhang,
   Fandong Meng, Xilin Chen, and Jie Zhou.
   2019. Retrieving sequential information for
   non-autoregressive neural machine
   translation. In *Proceedings of the 57th
   Annual Meeting of the Association for
   Computational Linguistics*, pages 3013–3024,
   Florence. https://doi.org/10.18653
   /v1/P19-1288

Shao, Chenze, Jinchao Zhang, Yang Feng,
   Fandong Meng, and Jie Zhou. 2020.
   Minimizing the bag-of-ngrams difference
   for non-autoregressive neural machine
   translation. In *The Thirty-Fourth AAAI
   Conference on Artificial Intelligence, AAAI
   2020, New York, NY, USA, February 7-12,
   2020*, pages 198–205. https://doi.org
   /10.1609/aaai.v34i01.5351

Shen, Shiqi, Yong Cheng, Zhongjun He, Wei
   He, Hua Wu, Maosong Sun, and Yang Liu.
   2016. Minimum risk training for neural
   machine translation. In *Proceedings of the
   54th Annual Meeting of the Association for
   Computational Linguistics (Volume 1: Long
   Papers)*, pages 1683–1692, Berlin. https://
   doi.org/10.18653/v1/P16-1159

Shu, Raphael, Jason Lee, Hideki Nakayama,
   and Kyunghyun Cho. 2020.
   Latent-variable non-autoregressive neural
   machine translation with deterministic
   inference using a delta posterior. In *The
   Thirty-Fourth AAAI Conference on Artificial
   Intelligence, AAAI 2020, New York, NY,
   USA, February 7-12, 2020*, pages 8846–8853.
   https://doi.org/10.1609/aaai.v34i05

Sun, Zhiqing, Zhuohan Li, Haoqing Wang,
   Di He, Zi Lin, and Zhihong Deng. 2019,
   Fast structured decoding for sequence
   models. In *Advances in Neural Information
   Processing Systems 32*, pages 3016–3026.

Sun, Zhiqing and Yiming Yang. 2020. An
   EM approach to non-autoregressive
   conditional sequence generation. In
   *Proceedings of the 37th International
   Conference on Machine Learning*, volume 119
   of *Proceedings of Machine Learning Research*,
   pages 9249–9258.

Sutskever, Ilya, Oriol Vinyals, and Quoc V.
   Le. 2014. Sequence to sequence learning

with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112.

Sutton, Richard S., David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063.

Sutton, Richard Stuart. 1984. *Temporal Credit Assignment in Reinforcement Learning*. Ph.D. thesis. AAI8410337.

Tu, Lifu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. 2020. ENGINE: Energy-based inference networks for non-autoregressive machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2826, Online.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010.

Venkatraman, Arun, Martial Hebert, and J. Andrew Bagnell. 2015. Improving multi-step prediction of learned time series models. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 3024–3030.

Wang, Chunqi, Ji Zhang, and Haiqing Chen. 2018. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 479–488, Brussels. `https://doi.org/10.18653/v1/D18-1044`

Wang, Yiren, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 5377–5384. `https://doi.org/10.1609/aaai.v33i01.33015377`

Weaver, Lex and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 538–545.

Wei, Bingzhen, Mingxuan Wang, Hao Zhou, Junyang Lin, and Xu Sun. 2019. Imitation learning for non-autoregressive neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1304–1312, Florence. `https://doi.org/10.18653/v1/P19-1125`

Williams, Ronald J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256. `https://doi.org/10.1007/BF00992696`

Williams, Ronald J. and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280. `https://doi.org/10.1162/neco.1989.1.2.270`

Wu, Lijun, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018a. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621, Brussels. `https://doi.org/10.18653/v1/D18-1397`

Wu, Lijun, Yingce Xia, Fei Tian, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018b. Adversarial neural machine translation. In *Proceedings of the 10th Asian Conference on Machine Learning*, volume 95 of *Proceedings of Machine Learning Research*, pages 534–549.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and others. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yang, Zhen, Wei Chen, Feng Wang, and Bo Xu. 2018. Improving neural machine translation with conditional sequence generative adversarial nets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1346–1355, New Orleans, LA. `https://doi.org/10.18653/v1/N18-1122`

Yu, Lantao, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2852–2858.

Zhang, Wen, Yang Feng, Fandong Meng,
    Di You, and Qun Liu. 2019. Bridging the
    gap between training and inference for
    neural machine translation. In *Proceedings
    of the 57th Annual Meeting of the Association
    for Computational Linguistics*,
    pages 4334–4343, Florence. `https://doi`
    `.org/10.18653/v1/P19-1426`
Zhou, Chunting, Jiatao Gu, and Graham
    Neubig. 2020. Understanding knowledge
    distillation in non-autoregressive machine

translation. In *8th International Conference
    on Learning Representations, ICLR 2020,
    Addis Ababa, Ethiopia, April 26-30, 2020*.
Zhou, Jiawei and Phillip Keung. 2020.
    Improving non-autoregressive neural
    machine translation with monolingual
    data. In *Proceedings of the 58th Annual
    Meeting of the Association for Computational
    Linguistics*, pages 1893–1898, Online.
    `https://doi.org/10.18653/v1/2020`
    `.acl-main.171`