Minimizing the Bag-of-Ngrams Difference for Non-Autoregressive Neural Machine Translation

Chenze Shao[1,2], **Jinchao Zhang**[3], **Yang Feng**[1,2*], **Fandong Meng**[3] and **Jie Zhou**[3]
[1] Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[2] University of Chinese Academy of Sciences
[3] Pattern Recognition Center, WeChat AI, Tencent Inc, China
{shaochenze18z,fengyang}@ict.ac.cn
{dayerzhang,fandongmeng,withtomzhou}@tencent.com

Abstract

Non-Autoregressive Neural Machine Translation (NAT) achieves significant decoding speedup through generating target words independently and simultaneously. However, in the context of non-autoregressive translation, the word-level cross-entropy loss cannot model the target-side sequential dependency properly, leading to its weak correlation with the translation quality. As a result, NAT tends to generate influent translations with over-translation and under-translation errors. In this paper, we propose to train NAT to minimize the Bag-of-Ngrams (BoN) difference between the model output and the reference sentence. The bag-of-ngrams training objective is differentiable and can be efficiently calculated, which encourages NAT to capture the target-side sequential dependency and correlates well with the translation quality. We validate our approach on three translation tasks and show that our approach largely outperforms the NAT baseline by about 5.0 BLEU scores on WMT14 En↔De and about 2.5 BLEU scores on WMT16 En↔Ro.

1 Introduction

Neural Machine Translation (NMT) has achieved impressive performance over the recent years (**?**; **?**; **?**; **?**; **?**). NMT models are typically built on the encoder-decoder framework where the encoder encodes the source sentence into distributed representations and the decoder generates the target sentence from the representations in an autoregressive manner: to predict the next word, previously predicted words have to be fed as inputs. The word-by-word generation manner of NMT determined by the autoregressive mechanism leads to high translation latency during inference and restricts application scenarios of NMT.

To reduce the translation latency for neural machine translation, non-autoregressive translation models (**?**) have been proposed. A basic NAT model takes the same encoder-decoder architecture as Transformer (**?**). Instead of using previously generated tokens as in autoregressive models, NAT takes other global signals as decoder inputs (**?**; **?**; **?**; **?**), which enables the simultaneous and independent generation of target words and achieves significant decoding speedup.

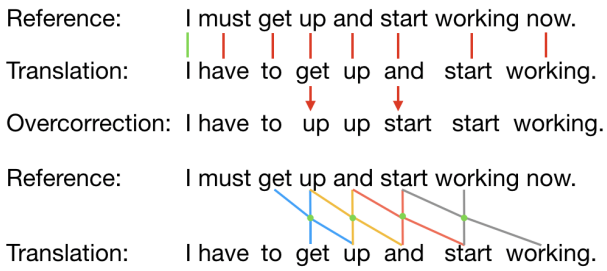

Figure 1: The cross-entropy loss gives a large penalty to the unaligned translation and may cause the overcorrection. The bag-of-2grams difference is robust to the unaligned translation and correlates well with the translation quality.

However, in the context of non-autoregressive translation, the word-level cross-entropy loss cannot model the target-side sequential dependency properly, leading to its weak correlation with the translation quality. The cross-entropy loss encourages NAT to generate the golden token in each position without considering the global correctness. For example, as Figure **??** illustrates, though the translation "I have to get up and start working" is semantically close to the reference, the cross-entropy loss will give it a large penalty due to its unalignment with the reference. Under the guidance of cross-entropy loss, the translation may be corrected to "I have to up up start start working" that is preferred by the cross-entropy loss but actually gets worse and contains repeated words, which is named as the overcorrection error (**?**). The limitation of the cross-entropy loss aggravates the weakness of NAT in modeling target sequential dependency, which often causes influent translation results with over-translation and under-translation errors (**?**; **?**). In addition, since the cross-entropy loss does not correlate well with the translation quality, it usually takes a long time for NAT to converge.

In this paper, we propose to model the target-side se-

*Corresponding author: Yang Feng.
Reproducible code: https://github.com/ictnlp/BoN-NAT.
Joint work with Pattern Recognition Center, WeChat AI, Tencent Inc, China.

quential dependency for NAT through minimizing the Bag-of-Ngrams (BoN) difference between the model output and the reference sentence. As the word-level cross-entropy loss cannot model the target-side sequential dependency properly, we propose to evaluate the NAT output on the n-gram level. Since the output of NAT may not be aligned with the reference, we do not require the strict alignment and instead optimize the bag-of-ngrams for NAT. As Figure **??** illustrates, the bag-of-ngrams difference is robust to unaligned translations and correlates well with the translation quality. Optimizing such a sequence-level objective usually faces the difficulty of exponential search space. Previous works (**?**; **?**) usually observe the objective as a reward and train the model under the reinforcement learning framework (**?**), which has relatively slow training speed and suffers from high estimation variance. We introduce a novel method to overcome the difficulty of exponential search space. Firstly, we define the BoN of NAT by the expectation of BoN on all possible translations. Then the BoN training objective is to minimize the BoN difference between NAT output and reference. For NAT, we give a simple and efficient method to calculate the BoN objective. The BoN training objective has the following advantages:

- It models the target-side sequential dependency for NAT and correlates well with the translation quality.
- It does not assume specific NAT model architectures and is easy to implement.
- It can be calculated accurately without making any approximation.
- It is differentiable and maintains fast training speed. Therefore it can not only be utilized for fine-tuning but also work together with cross-entropy loss to jointly train NAT from scratch.

We evaluate the proposed method on three translation tasks (IWSLT16 En→De, WMT14 En↔De, WMT16 En↔Ro). Experimental results show that the fine-tuning method achieves large improvements over the pre-trained NAT baseline, and the joint training method further brings considerable improvements over the fine-tuning method, which outperforms the NAT baseline by about 5.0 BLEU scores on WMT14 En↔De and about 2.5 BLEU scores on WMT16 En↔Ro.

2 Background

2.1 Autoregressive Neural Machine Translation

Deep neural networks with autoregressive encoder-decoder framework have achieved great success on machine translation, with different choices of architectures such as RNN, CNN and Transformer. RNN-based models (**?**; **?**) have a sequential architecture that prevents them from being parallelized. CNN (**?**), and self-attention (**?**) based models have highly parallelized architecture, which solves the parallelization problem during training. However, during inference, the translation has to be generated word-by-word due to the autoregressive mechanism.

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

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

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

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

During decoding, the partial translation generated by decoding algorithms such as the greedy search and beam search is fed into the decoder to generate the next word.

2.2 Non-Autoregressive Neural Machine Translation

Non-autoregressive neural machine translation (**?**) is proposed to reduce the translation latency through parallelizing the decoding process. A basic NAT model takes the same encoder-decoder architecture as Transformer. The NAT encoder stays unchanged from the original Transformer encoder, and the NAT decoder has an additional positional attention block in each layer compared to the Transformer decoder. Besides, there is a length predictor that takes encoder states as input to predict the target length.

Given a source sentence $\boldsymbol{X} = \{x_1, ..., x_n\}$ and a target sentence $\boldsymbol{Y} = \{y_1, ..., y_T\}$, NAT models the translation probability from $\boldsymbol{X}$ to $\boldsymbol{Y}$ as:

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

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

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

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

$$\hat{y_t} = \arg\max_{y_t} p(y_t|\boldsymbol{X}, \theta). \tag{5}$$

3 Model

In this section, we will discuss the BoN objective for NAT in detail. We first formulize 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. The BoN objective is to minimize the difference between the BoN of NAT output and reference, which encourages NAT to capture the target-side sequential dependency and generate high-quality translations.

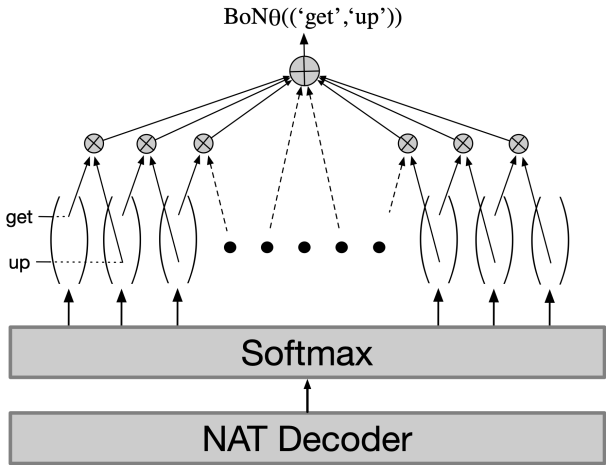

Figure 2: The calculation process of BoN_θ("get up"). First calaulate the probability of the bigram "get up" in each subarea and then accumulate the probabilities.

3.1 Bag-of-Ngrams

Bag-of-words (**?**) is the most commonly used text representation model which discards the word order and represent sentence as the multiset of its belonging words. Bag-of-ngrams (**?**; **?**) is proposed to enhance the text representation by takeing consecutive words (n-gram) into consideration. Assume that the vocabulary size is V, Bag-of-Ngrams (BoN) is a vector of size V^n, which is the sum of zero-one vectors where each vector is the one-hot representation of an n-gram. Formally, for a sentence $\boldsymbol{Y} = \{y_1, ..., y_T\}$, we use $\text{BoN}_{\boldsymbol{Y}}$ to denote the bag-of-ngrams of $\boldsymbol{Y}$. For an n-gram $\boldsymbol{g} = (g_1, \ldots, g_n)$, we use $\text{BoN}_{\boldsymbol{Y}}(\boldsymbol{g})$ to denote the value of $\boldsymbol{g}$ in $\text{BoN}_{\boldsymbol{Y}}$, which is the number of occurrences of n-gram $\boldsymbol{g}$ in sentence $\boldsymbol{Y}$ and can be formulized as follows:

$$\text{BoN}_{\boldsymbol{Y}}(\boldsymbol{g}) = \sum_{t=0}^{T-n} 1\{y_{t+1:t+n} = \boldsymbol{g}\}, \tag{6}$$

where $1\{\cdot\}$ is the indicator function that takes value from $\{0, 1\}$ whose value is 1 iff 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-ngrams. Formally, given a source sentence $\boldsymbol{X}$, the value of n-gram $\boldsymbol{g}$ in BoN_θ is defined as follows:

$$\text{BoN}_\theta(\boldsymbol{g}) = \sum_{\boldsymbol{Y}} P(\boldsymbol{Y}|\boldsymbol{X}, \theta) \cdot \text{BoN}_{\boldsymbol{Y}}(\boldsymbol{g}). \tag{7}$$

3.2 Efficient Calculation

It is unrealistic to directly calculate $\text{BoN}_{\boldsymbol{Y}}(\boldsymbol{g})$ according to Eq.(**??**) 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. **?** (**?**) only focuses on greedily chosen words and makes the search space very limited. **?** (**?**) sums up the distribution of all positions to generate the bag-of-words, which is biased by the teacher forcing.

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. Guiding by this unique property of NAT, we convert Eq.(**??**) to the following form:

$$\begin{aligned}
\text{BoN}_\theta(\boldsymbol{g}) &= \sum_{\boldsymbol{Y}} P(\boldsymbol{Y}|\boldsymbol{X}, \theta) \cdot \sum_{t=0}^{T-n} 1\{y_{t+1:t+n} = \boldsymbol{g}\} \\
&= \sum_{t=0}^{T-n} \sum_{\boldsymbol{Y}} P(\boldsymbol{Y}|\boldsymbol{X}, \theta) \cdot 1\{y_{t+1:t+n} = \boldsymbol{g}\} \\
&= \sum_{t=0}^{T-n} \sum_{\boldsymbol{Y}_{t+1:t+n}} P(\boldsymbol{Y}_{t+1:t+n}|\boldsymbol{X}, \theta) \cdot 1\{y_{t+1:t+n} = \boldsymbol{g}\} \\
&= \sum_{t=0}^{T-n} \prod_{i=1}^{n} p(y_{t+i} = g_i|\boldsymbol{X}, \theta).
\end{aligned} \tag{8}$$

Eq.(**??**) gives an efficient method to calculate $\text{BoN}_\theta(\boldsymbol{g})$: slide a window on NAT output distributions to obtain subareas of size n, and then accumulate the values of n-gram $\boldsymbol{g}$ in all subareas. Figure **??** illustrates the calculation process of a bigram "get up". It does not make any approximation and requires little computational effort.

3.3 Bag-of-Ngrams Objective

The training objective is to minimize the BoN difference between NAT output and reference. The difference can be measured by several metrics such as the L_1 distance, L_2 distance and cosine distance. In previous sections, we define the BoN for NAT and give an efficient calculation method for $\text{BoN}_\theta(\boldsymbol{g})$. However, since there are V^n different n-grams, calculating the complete BoN vector for NAT still consumes a huge amount of storage space and computing resources.

We use the sparsity of bag-of-ngrams to further simplify the calculation. As shown in Eq.(**??**), for NAT, its bag-of-ngrams BoN_θ is dense. On the contrary, assume that the reference sentence is $\hat{\boldsymbol{Y}}$, the vector $\text{BoN}_{\hat{\boldsymbol{Y}}}$ is very sparse where only a few entries of it have non-zero values. Using this property, we show that the L_1 distance between the two BoN vectors can be calculated efficiently. Unfortunately, some other distance metrics like L_2 distance and cosine distance cannot be simplified in this way.

Assume that the target length is T. Intutively, a sentence of length T has $T-n+1$ n-grams. We first verify the intution that the L_1-norm of $\text{BoN}_{\boldsymbol{Y}}$ and BoN_θ are both $T - n + 1$:

Algorithm 1 BoN-L_1

Input: model parameters θ, input sentence $\boldsymbol{X}$, reference sentence $\hat{\boldsymbol{Y}}$, prediction length T, n
Output: BoN precision BoN-p

construct the reference bag-of-ngrams $\text{BoN}_{\hat{\boldsymbol{Y}}}$
ref-ngrams=$\{\boldsymbol{g}|\text{BoN}_{\hat{\boldsymbol{Y}}}(\boldsymbol{g})$!= $0\}$
match = 0
for $\boldsymbol{g}$ in ref-ngrams **do**
    calculate $\text{BoN}_\theta(\boldsymbol{g})$ according to Eq.(**??**)
    match += $\min(\text{BoN}_\theta(\boldsymbol{g}), \text{BoN}_{\hat{\boldsymbol{Y}}}(\boldsymbol{g}))$
BoN-L_1 = 2(T-n+1-match)
return BoN-L_1

$$\sum_{\boldsymbol{g}} \text{BoN}_{\boldsymbol{Y}}(\boldsymbol{g}) = \sum_{t=0}^{T-n} \sum_{\boldsymbol{g}} 1\{y_{t+1:t+n} = \boldsymbol{g}\} = T - n + 1. \tag{9}$$

$$\begin{aligned} \sum_{\boldsymbol{g}} \text{BoN}_\theta(\boldsymbol{g}) &= \sum_{\boldsymbol{g}} \sum_{\boldsymbol{Y}} P(\boldsymbol{Y}|\boldsymbol{X}, \theta) \cdot \text{BoN}_{\boldsymbol{Y}}(\boldsymbol{g}) \\ &= \sum_{\boldsymbol{Y}} P(\boldsymbol{Y}|\boldsymbol{X}, \theta) \cdot \sum_{\boldsymbol{g}} \text{BoN}_{\boldsymbol{Y}}(\boldsymbol{g}) = T - n + 1. \end{aligned} \tag{10}$$

On this basis, we can derive the L_1 distance between BoN_θ and $\text{BoN}_{\hat{\boldsymbol{Y}}}$, where $\hat{\boldsymbol{Y}}$ is the reference sentence. We denote the L_1 distance as BoN-L_1 and convert it to the following form:

$$\begin{aligned} &\text{BoN-}L_1 = \sum_{\boldsymbol{g}} |\text{BoN}_\theta(\boldsymbol{g}) - \text{BoN}_{\hat{\boldsymbol{Y}}}(\boldsymbol{g})| \\ &= \sum_{\boldsymbol{g}} (\text{BoN}_\theta(\boldsymbol{g}) + \text{BoN}_{\hat{\boldsymbol{Y}}}(\boldsymbol{g}) - 2\min(\text{BoN}_\theta(\boldsymbol{g}), \text{BoN}_{\hat{\boldsymbol{Y}}}(\boldsymbol{g})) \\ &= 2(T - n + 1 - \sum_{\boldsymbol{g}} \min(\text{BoN}_\theta(\boldsymbol{g}), \text{BoN}_{\hat{\boldsymbol{Y}}}(\boldsymbol{g}))). \end{aligned} \tag{11}$$

The minimum between $\text{BoN}_\theta(\boldsymbol{g})$ and $\text{BoN}_{\hat{\boldsymbol{Y}}}(\boldsymbol{g})$ can be understood as the number of matches for the n-gram $\boldsymbol{g}$, 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 $\boldsymbol{g}$ appears in the reference sentence. Hence we can only focus on n-grams in the reference, which significantly reduce the computational effort and storage requirement. Algorithm **??** illustrates the calculation process of BoN-L_1.

We normalize the L_1 distance to range $[0, 1]$ by dividing the constant $2(T - n + 1)$. The BoN training objective is to minimize the normalized L_1 distance:

$$\mathcal{L}_{BoN}(\theta) = \frac{\text{BoN-}L_1}{2(T - n + 1)}. \tag{12}$$

As is usual practice in sequence-level training, we can use the BoN objective to fine-tune a pre-trained baseline model. We denote this method as **BoN-FT**. In addition, since the BoN objective is differentiable and can be calculated efficiently, it can also work together with cross-entropy loss to jointly train NAT from scratch. We use a hyper-parameter α to combine the two losses:

$$\mathcal{L}_{joint}(\theta) = \alpha \cdot \mathcal{L}_{MLE}(\theta) + (1 - \alpha) \cdot \mathcal{L}_{BoN}(\theta). \tag{13}$$

The joint training method is denoted as **BoN-Joint**. After the joint training, we can fine-tune the model using the BoN objective only. We denote this method as **BoN-Joint+FT**.

4 Related Work

? (**?**) introduced the non-autoregressive Transformer to reduce the translation latency of NMT, which comes at the cost of translation quality. Instead of using previously generated tokens as in autoregressive models, NAT take other global signals as decoder inputs. **?** (**?**) introduced the uniform copy and fertility to copy from source inputs. **?**; **?**; **?** (**?**; **?**; **?**) proposed to use latent variables to improve the performance of non-autoregressive Transformer. **?** (**?**) introduced syntactically supervised Transformers, which first autoregressively predicts a chunked parse tree and then generate all target tokens conditioned on it. **?** (**?**) proposed to iteratively refine the translation where the outputs of decoder are fed back as inputs in the next iteration. **?** (**?**) proposed to improve non-autoregressive models through distilling knowledge from autoregressive models. **?** (**?**) introduced Levenshtein Transformer for more flexible and amenable sequence generation. **?** (**?**) introduced the semi-autoregressive Transformer that generates a group of words each time. **?** (**?**) proposed to enhance decoder inputs with phrase-table lookup and embedding mapping. **?** (**?**) proposed an end-to-end non-autoregressive model using connectionist temporal classification. **?** (**?**) proposed an imitation learning framework where the non-autoregressive learner learns from the autoregressive demonstrator through an imitation module. **?** (**?**) pointed out that NAT is weak in capturing sequential dependency and proposed the similarity regularization and reconstruction regularization to reduce errors of repeated and incomplete translations. **?**; **?** (**?**; **?**) proposed to model the reordering information for non-autoregressive Transformer. **?** (**?**) proposed Reinforce-NAT to train NAT with sequence-level objectives. As the techniques for variance reduction makes the training speed relatively slow, this method is restricted in fine-tuning.

Training NMT with discrete sequence-level objectives has been widely investigated under the reinforcement learning framework (**?**; **?**; **?**; **?**; **?**). Recently, differentiable sequence-level training objectives for autoregressive NMT have attracted much attention. **?** (**?**) applied the deterministic policy gradient algorithm to train NMT actor with the differentiable critic. **?** (**?**) proposed to use bag-of-words as target during training. **?** (**?**) proposed to evaluate NMT outputs with probabilistic n-grams.

5 Experimental Setting

Datasets We use several widely adopted benchmark datasets to evaluate the effectiveness of our proposed method: IWSLT16 En→De (196k pairs), WMT14 En↔De (4.5M pairs) and WMT16 En↔Ro (610k pairs). For WMT14 En↔De, we employ `newstest-2013`

and newstest-2014 as development and test sets. For WMT16 En↔Ro, we take newsdev-2016 and newstest-2016 as development and test sets. For IWSLT16 En→De, we use the test2013 for validation. We use the preprocessed datasets released by **?** (**?**), where all sentences are tokenized and segmented into subwords units (**?**). The vocabulary size is 40k and is shared for source and target languages. We use BLEU (**?**) to evaluate the translation quality.

Baselines We take the Transformer model (**?**) as our autoregressive baseline as well as the teacher model. The non-autoregressive model with 2 iterative refinement iterations (IRNAT) (**?**) is our non-autoregressive baseline, and the methods we propose are all experimented on this baseline model. We also include three relevant NAT works for comparison, NAT with fertility (NAT-FT) (**?**), NAT with auxiliary regularization (NAT-REG) (**?**) and reinforcement learning for NAT (Reinforce-NAT) (**?**).

Model Configurations We closely follow the settings of **?** (**?**), **?** (**?**) and **?** (**?**). For IWSLT16 En→De, we use the small Transformer (d_{model}=278, d_{hidden}=507, n_{layer}=5, n_{head}=2, p_{dropout}=0.1, t_{warmup}=746). For experiments on WMT datasets, we use the base Transformer (**?**) (d_{model}=512, d_{hidden}=512, n_{layer}=6, n_{head}=8, p_{dropout}=0.1, t_{warmup}=16000). We use Adam (**?**) for the optimization. In the main experiment, the hyper-parameter α to combine the BoN objective and cross-entropy loss set to be 0.1. We set n=2, that is, we use the bag-of-2grams objective to train the model. The effects of α and n will be analyzed in detail later.

Training and Inference During training, we apply the sequence-level knowledge distillation (**?**), which constructs the distillation corpus where the target side of the training corpus is replaced by the output of the autoregressive Transformer. We directly use the distillation corpus released by **?** (**?**). We use the original corpus to train the autoregressive Transformer and distillation corpus to train non-autoregressive models. For NAT baseline, we stop training when the number of training steps exceeds 300k and there is no further improvements in the last 100k steps. For BoN-Joint, we stop training when the number of training steps exceeds 150k and there is no further improvements in the last 100k steps. For BoN-FT, the number of fine-tuning steps is fixed to 5k.

After training NAT, we train a target length predictor to predict the length difference between the source and target sentences. The target length predictor takes the sum of encoder hidden states as input and feeds it to a softmax classifier after an affine transformation. During inference, we first run the encoder and apply the length predictor to predict the target length. Then we construct decoder inputs through uniform copy (**?**) and run the decoder to generate the translation. During postprocessing, we remove any token that is generated repeatedly. The training and decoding speed are measured on a single Geforce GTX TITAN X.

6 Results and Analysis

6.1 Main Results

We report our main results in Table **??**, from which we have the following observations:

1. **BoN-FT achieves significant training speedup over Reinforce-NAT and outperforms Reinforce-NAT on BLEU scores**. BoN-FT and Reinforce-NAT are both fine-tuning approaches. BoN-FT slightly outperforms Reinforce-NAT on four translation directions and achieves at most 3.74 BLEU improvements over the NAT-Base on WMT14 De→En. More importantly, since the BoN objective is differentiable and can be efficiently calculated, the training speed of BoN-FT is nearly 10 times faster than Reinforce-NAT, which makes the fine-tuning process very efficient.

2. **BoN-Joint+FT achieves considerable improvements over BoN-FT**. By combining the BoN objective and cross-entropy loss, we obtain BoN-Joint, which achieves considerable improvements over BoN-FT. Though the training speed for BoN-Joint is about two times slower, BoN-Joint only needs about half of training steps to converge, so the training time is almost the same as NAT-Base. BoN-Joint+FT further improves the performance through fine-tuning BoN-Joint with the BoN objective, which achieves about 5.0 BLEU improvements on WMT14 En↔De, about 2.5 BLEU improvements on WMT16 En↔Ro and 1.59 BLEU improvements on IWSLT16 En→De.

3. **Our methods achieve higher improvements on WMT datasets than the IWSLT dataset**. One interesting phenomenon is that our methods achieve higher improvements on WMT datasets. The BLEU improvement is 4.85 on WMT14 En→De but only 1.59 on IWSLT16 En→De. As we will analyze later, this has a lot to do with the sentence length of datasets since the cross-entropy loss does not perform well on long sentences.

6.2 Correlation with Translation Quality

In this section, we experiment with the correlation between the loss function and translation quality. We are interested in how the cross-entropy loss and BoN objective correlate with the translation quality. As a common practice, the BLEU score is used to represent the translation quality. We experiment on the WMT14 En→De development set, which contains 3000 sentences. Firstly we randomly divide the dataset into 100 subsets of size 30. Then we use the NAT model to decode the 100 subsets and compute the BLEU score and loss function on every subset. Finally we calculate the pearson correlation between the 100 BLEU 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 BoN-L_1 normalized by $2(T-n+1)$. We respectively set n to 1, 2, 3 and 4 to test different n-gram size. Table **??** lists the correlation results.

We can see that between different BoN objectives, $n = 1$, which represents the bag-of-words objective, underperforms other choices of n, indicating the necessity of modeling sequential dependency for NAT. Another observation is that all the four BoN objectives outperform the cross-entropy loss by large margins. To find out where the performance gap

		IWSLT'16 En-De			WMT'16 En-Ro			WMT'14 En-De		
		En→	speedup	secs/b	En→	Ro→	speedup	En→	De→	speedup
AR	b=1	28.64	1.09×	0.20	31.93	31.55	1.23×	23.77	28.15	1.13×
	b=4	28.98	1.00×	0.20	32.40	32.06	1.00×	24.57	28.47	1.00×
NAT Models	NAT-FT (**?**)	26.52	15.6×	–	27.29	29.06	–	17.69	21.47	–
	IRNAT(iter=2) (**?**)	24.82	6.64×	–	27.10	28.15	7.68×	16.95	20.39	8.77×
	IRNAT(adaptive) (**?**)	27.01	1.97×	–	29.66	30.30	2.73×	21.54	25.43	2.38×
	NAT-REG (**?**)	–	–	–	–	–	–	20.65	24.77	27.6×
	Reinforce-NAT (**?**)	25.18	8.43×	13.40	27.09	27.93	9.44×	19.15	22.52	10.73×
Our Models	NAT-Base	24.13	8.42×	0.62	25.96	26.49	9.41×	16.05	19.46	10.76×
	BoN-FT (n=2)	25.03	8.44×	1.41	27.21	27.95	9.50×	19.27	23.20	10.72×
	BoN-Joint (n=2, α=0.1)	25.63	8.39×	1.49	28.12	29.03	9.44×	20.75	24.47	10.79×
	BoN-Joint+FT (n=2, α=0.1)	25.72	8.40×	1.41	28.31	29.29	9.51×	20.90	24.61	10.77×

Table 1: Generation quality (4-gram BLEU), speedup and training speed (seconds/batch). Decoding speed is measured sentence-by-sentence from the En→ direction. AR: the autoregressive Transformer baseline and the teacher model. b: beam size. NAT-Base: the non-autoregressive Transformer baseline trained by the cross-entropy loss. BoN-FT: fine-tune the NAT-Base with the BoN objective. BoN-Joint: combine the BoN objective and cross-entropy loss to jointly train NAT from scratch. BoN-Joint+FT: fine-tune the BoN-Joint with the BoN objective.

Loss function	CE	$n=1$	$n=2$	$n=3$	$n=4$
Correlation	0.37	0.56	0.70	0.67	0.61

Table 2: The pearson correlation bewteen loss functions and translation quality. $n = k$ represents the bag-of-kgrams training objective. CE represents the cross-entropy loss.

comes from, we analyze the effect of sentence length in the following experiment. We evenly divide the dataset into two parts according to the source length, where the first part consists of 1500 short sentences and the second part consists of 1500 long sentences. We respectively measure the pearson correlation on the two parts and list the results in Table **??**:

	all	short	long
Cross-Entropy	0.37	0.52	0.21
BoN (n=2)	0.70	0.79	0.81

Table 3: The pearson correlation bewteen loss functions and translation quality on short sentences and long sentences.

From Table **??**, we can see that the correlation of cross-entropy loss drops sharply as the sentence length increases, where the BoN objective still has 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 sentence length grows, it becomes harder for NAT to align the translation with reference, which leads to a decrease of correlation between cross-entropy loss and translation quality. In contrast, the BoN objective is robust to unaligned translations, so its correlation with translation quality stays strong when translating long sentences.

6.3 Number of Removed Tokens

In this section, we study the number of tokens removed during postprocessing to see how our method improves translation quality. We sort the WMT14 En→De development set by sentence length and split it evenly into the short part and long part accordingly. Then we translate them using NAT-Base and BoN-Joint respectively and record in table **??** the number of removed repeated tokens during postprocessing.

	short	long	all
Total	22023	56142	78165
NAT-Base	1232(5.6%)	7317(13.0%)	9643(12.3%)
BoN-Joint	515(2.3%)	1726(3.1%)	2241(2.9%)

Table 4: The number of removed repeated tokens for NAT-Base and BoN-Joint. Total represents the total length of reference sentences.

As shown in table **??**, NAT-Base suffers from the repeated translation when translating long sentences, which is efficiently alleviated in BoN-Joint. This is in line with our analysis. As sentence length grows, it becomes harder for NAT to align the translation with reference, which results in more overcorrection errors and makes more repeated tokens be translated.

6.4 Effect of Sentence Length

Previously, we give the correlation between loss functions and the translation quality under different sentence lengths. In this section, we will experiment on the BLEU performance of NAT baseline and our models on different sentence lengths and see whether the better correlation contributes to better BLEU performance. We conduct the analysis on the WMT14 En→De development set and divide the sentence

pairs into different length buckets according to the length of the reference sentence. Figure **??** shows our results.

We can see that the performance of NAT-Base drops quickly as sentence length increases, where the autoregressive Transformer and BoN models have stable performance over different sentence lengths. This can be well explained by the correlation results. As sentence length grows, the correlation between the cross-entropy loss and the translation quality drops sharply, which leads to the weakness of NAT in translating long sentences.

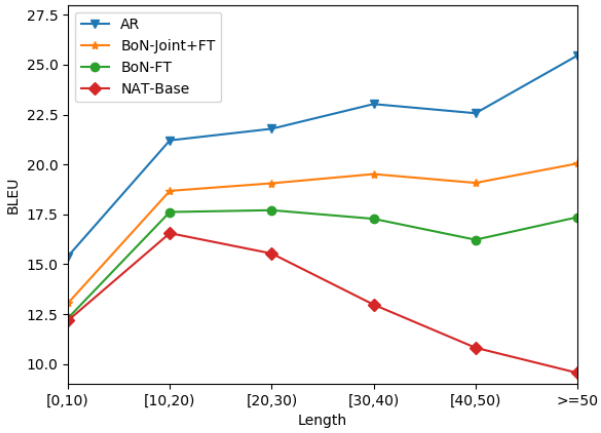

Figure 3: BLEU performance on different length buckets.

6.5 Effect of α

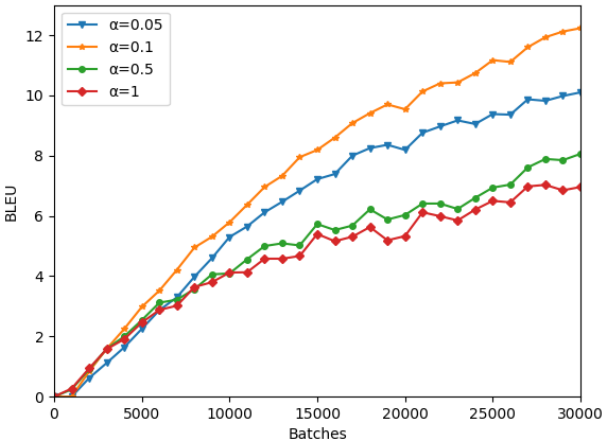

Figure 4: Training curves for BoN-Joint with different hyper-parameter α.

In this section, we study the effect of the hyper-parameter α, which is defined in Eq.(**??**) and is used to control the proportion of the cross-entropy loss and BoN loss. We set $n = 2$ and use different α to train BoN-Joint on WMT14 En→De. Figure **??** shows the BLEU scores on the development set.

We can see from Figure **??** that $\alpha = 0.1$ has a stable and fast BLEU improvement over time. When we set $\alpha = 0.1$, the time required for model convergence is the shortest, and the BLEU score after convergence is the best over other choices of α. In addition, we find that when we set $n = 2$, $\alpha = 0.1$ performs well in all the three datasets, which frees us from the hyper-parameter tuning.

6.6 Effect of N-gram Size

In this section, we study the effect of n-gram size. We respectively set n to 1, 2, 3, and 4 and evaluate the performance of BoN-FT and BoN-Joint on the WMT14 En→De development set. When training BoN-Joint, we tune the hyper-parameter α for every n to obtain the optimal performance. The results are listed in Table **??**. For BoN-FT, the BoN objective is only used to fine-tune the NAT baseline, and different choices for n do not have a large impact on the BLEU performance. For BoN-Joint, the choice of n becomes more important, and $n = 2$ significantly outperforms other choices of n, which is consistent with the correlation result in Table **??**.

	$n=1$	$n=2$	$n=3$	$n=4$
BoN-FT	18.06	18.13	18.17	18.08
BoN-Joint	17.73	19.28	18.45	17.97

Table 5: The BLEU performance of BoN-FT and BoN-Joint under different choices of n.

7 Conclusion

In the context of non-autoregressive translation, the cross-entropy loss cannot model the target-side sequential dependency properly and suffers from the weak correlation with the translation quality. In this paper, we propose a novel Bag-of-Ngrams objective to model the target-side sequential dependency for NAT. The BoN objective is differentiable and can be calculated efficiently. Experiments show that the BoN objective correlates well with the translation quality and achieves large improvements over the NAT baseline.

8 Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work was supported by National Natural Science Foundation of China (NO.61876174) and National Key R&D Program of China (NO.2017YFE9132900).

References

Akoury, N.; Krishna, K.; and Iyyer, M. 2019. Syntactically supervised transformers for faster neural machine translation. In *ACL*.

Anonymous. 2020. {PNAT}: Non-autoregressive transformer by position learning. In *Submitted to International Conference on Learning Representations*. under review.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Gu, J.; Bradbury, J.; Xiong, C.; Li, V. O.; and Socher, R. 2017. Non-autoregressive neural machine translation. In *ICLR*.

Gu, J.; Cho, K.; and Li, V. O. 2017. Trainable greedy decoding for neural machine translation. In *EMNLP*.

Gu, J.; Wang, C.; and Zhao, J. 2019. Levenshtein transformer. In *Advances in NeurIPS*.

Guo, J.; Tan, X.; He, D.; Qin, T.; Xu, L.; and Liu, T.-Y. 2019. Non-autoregressive neural machine translation with enhanced decoder input. In *AAAI*, volume 33.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*. Springer.

Kaiser, L.; Bengio, S.; Roy, A.; Vaswani, A.; Parmar, N.; Uszkoreit, J.; and Shazeer, N. 2018. Fast decoding in sequence models using discrete latent variables. In *ICML*.

Kim, Y., and Rush, A. M. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Lee, J.; Mansimov, E.; and Cho, K. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *EMNLP*.

Li, B.; Liu, T.; Zhao, Z.; Wang, P.; and Du, X. 2017. Neural bag-of-ngrams. In *AAAI*.

Li, Z.; Lin, Z.; He, D.; Tian, F.; QIN, T.; WANG, L.; and Liu, T.-Y. 2019. Hint-based training for non-autoregressive machine translation. In *EMNLP-IJCNLP*.

Libovickỳ, J., and Helcl, J. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *EMNLP*.

Ma, S.; Xu, S.; Wang, Y.; and Lin, J. 2018. Bag-of-words as target for neural machine translation. In *ACL*.

Ma, X.; Zhou, C.; Li, X.; Neubig, G.; and Hovy, E. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *EMNLP-IJCNLP*.

Pang, B.; Lee, L.; and Vaithyanathan, S. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *ACL*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Ran, Q.; Lin, Y.; Li, P.; and Zhou, J. 2019. Guiding non-autoregressive neural machine translation decoding with reordering information. *arXiv preprint arXiv:1911.02215*.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Roy, A.; Vaswani, A.; Neelakantan, A.; and Parmar, N. 2018. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Shao, C.; Feng, Y.; Zhang, J.; Meng, F.; Chen, X.; and Zhou, J. 2019. Retrieving sequential information for non-autoregressive neural machine translation. In *ACL*.

Shao, C.; Chen, X.; and Feng, Y. 2018. Greedy search with probabilistic n-gram matching for neural machine translation. In *EMNLP*.

Shen, S.; Cheng, Y.; He, Z.; He, W.; Wu, H.; Sun, M.; and Liu, Y. 2015. Minimum risk training for neural machine translation. In *ACL*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in NIPS*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in NIPS*.

Wang, Y.; Tian, F.; He, D.; Qin, T.; Zhai, C.; and Liu, T.-Y. 2019. Non-autoregressive machine translation with auxiliary regularization. In *AAAI*.

Wang, C.; Zhang, J.; and Chen, H. 2018. Semi-autoregressive neural machine translation. In *EMNLP*.

Wei, B.; Wang, M.; Zhou, H.; Lin, J.; and Sun, X. 2019. Imitation learning for non-autoregressive neural machine translation. In *ACL*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*. Springer.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wu, L.; Tian, F.; Qin, T.; Lai, J.; and Liu, T.-Y. 2018. A study of reinforcement learning for neural machine translation. In *EMNLP*.

Zhang, W.; Feng, Y.; Meng, F.; You, D.; and Liu, Q. 2019. Bridging the gap between training and inference for neural machine translation. In *ACL*.