Retrieving Sequential Information for Non-Autoregressive Neural Machine Translation

Chenze Shao[123]**, Yang Feng**[12*]**, Jinchao Zhang**[3]**, Fandong Meng**[3]**, Xilin Chen**[12] **and Jie Zhou**[3]

[1] University of Chinese Academy of Sciences
[2] Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[3] Pattern Recognition Center, WeChat AI, Tencent Inc, China
{shaochenze18z, fengyang, xlchen}@ict.ac.cn
{dayerzhang, fandongmeng, withtomzhou}@tencent.com

Abstract

Non-Autoregressive Transformer (NAT) aims to accelerate the Transformer model through discarding the autoregressive mechanism and generating target words independently, which fails to exploit the target sequential information. Over-translation and under-translation errors often occur for the above reason, especially in the long sentence translation scenario. In this paper, we propose two approaches to retrieve the target sequential information for NAT to enhance its translation ability while preserving the fast-decoding property. Firstly, we propose a sequence-level training method based on a novel reinforcement algorithm for NAT (Reinforce-NAT) to reduce the variance and stabilize the training procedure. Secondly, we propose an innovative Transformer decoder named FS-decoder to fuse the target sequential information into the top layer of the decoder. Experimental results on three translation tasks show that the Reinforce-NAT surpasses the baseline NAT system by a significant margin on BLEU without decelerating the decoding speed and the FS-decoder achieves comparable translation performance to the autoregressive Transformer with considerable speedup.

1 Introduction

Neural machine translation (NMT) models (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014) solve the machine translation problem with the Encoder-Decoder framework and achieve impressive performance on translation quality. Recently, the Transformer model (Vaswani et al., 2017) further enhances the translation performance on multiple language pairs, while suffering from the slow decoding procedure, which re-

_____

Joint work with Pattern Recognition Center, WeChat AI, Tencent Inc, China.
⋆ Corresponding Author

Src	und noch tragischer ist , dass es Oxford war ⋯
Ref	even more tragic is that it was Oxford ⋯
NAT	and more more more more that it was Oxford ⋯
AR	and , more tragic , Oxford was ⋯

Table 1: A fragment of a long sentence translation. AR stands for the translation of the autoregressive Transformer. The output of the NAT model contains repeated translations of word 'more' and misses the word 'tragic'.

stricts its application scenarios. The slow decoding problem of the Transformer model is caused by its autoregressive nature, which means that the target sentence is generated word by word according to the source sentence representations and the target translation history.

Non-autoregressive Transformer model (Gu et al., 2017a) is proposed to accelerate the decoding process, which can simultaneously generate target words by discarding the autoregressive mechanism. Since the generation of target words is independent, NAT models utilize alternative information such as encoder inputs (Gu et al., 2017a), translation results from other systems (Lee et al., 2018; Guo et al., 2018) and latent variables (Kaiser et al., 2018) as decoder inputs. Without considering the target translation history, NAT models are weak to exploit the target words collocation knowledge and tend to generate repeated target words at adjacent time steps (Wang et al., 2019). Over-translation and under-translation problems are aggravated and often occur due to the above reasons. Table 1 shows an inferior translation example generated by a NAT model. Compared to the autoregressive Transformer, NAT models achieve significant speedup while suffering from a large gap in translation quality due to the lack of target sequential information.

In this paper, we present two approaches to retrieve the target sequential information for NAT models to enhance their translation ability and meanwhile preserve the fast-decoding property. Firstly, we propose a sequence-level training method based on a novel reinforcement algorithm for NAT (*Reinforce-NAT*) to reduce the variance and stabilize the training procedure. We leverage the sequence-level objectives (e.g., BLEU (Papineni et al., 2002), GLEU (Wu et al., 2017), TER (Snover et al., 2006)) instead of the cross-entropy objective to encourage NAT model to generate high quality sentences rather than the correct token for each position. Secondly, we propose an innovative Transformer decoder named *FS-decoder* to fuse the target sequential information into the top layer of the decoder. The bottom layers of the FS-decoder run in parallel to keep the decoding speed and the top layer of the FS-decoder can exploit target sequential information to guide the target words generation procedure.

We conduct experiments on three machine translation tasks (IWSLT16 En→De, WMT14 En↔De, WMT16 En→Ro) to validate our proposed approaches. Experimental results show that the Reinforce-NAT surpasses the baseline NAT system by a significant margin on the translation quality without decelerating the decoding speed, and the FS-decoder achieves comparable translation capacity to the autoregressive Transformer with considerable speedup.

2 Background

2.1 Autoregressive Neural Machine Translation

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

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

where θ is a set of model parameters and $\boldsymbol{y_{<t}} = \{y_1, \cdots, y_{t-1}\}$ is the translation history. Given the training set $D = \{\mathbf{X}^{\mathrm{M}}, \mathbf{Y}^{\mathrm{M}}\}$ with M sentence pairs, the training objective is to maximize the log-likelihood of the training data as:

$$\boldsymbol{\theta} = \arg\max_{\theta}\{\mathcal{L}(\theta)\}$$

$$\mathcal{L}(\theta) = \sum_{m=1}^{M} \sum_{t=1}^{T} \log(p(y_t^m|\boldsymbol{y_{<t}^m}, \boldsymbol{X}^m, \theta)), \qquad (2)$$

where the superscript m indicates the m-th sentence in the dataset. During training, golden target words are fed into the decoder as the translation history. During inference, the partial translation generated by decoding algorithms such as greedy search and beam search is fed into 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 one-by-one style. Due to the autoregressive property, the decoding speed is limited, which restricts the application of the autoregressive model.

2.2 Sequence-Level Training for Autoregressive NMT

Reinforcement learning techniques (Sutton et al., 2000; Ng et al., 1999; Sutton, 1984) have been widely applied to improve the performance of the autoregressive NMT with sequence-level objectives (Shen et al., 2016; Ranzato et al., 2015; Bahdanau et al., 2016). As sequence-level objectives are usually non-differentiable, the loss function is defined as the negative expected reward:

$$\mathrm{L}_{\theta} = - \sum_{\mathbf{Y}=y_{1:T}} p(\mathbf{Y}|\mathbf{X}, \theta) \cdot r(\mathbf{Y}), \qquad (3)$$

where $\mathbf{Y} = y_{1:T}$ denotes possible sequences generated by the model, and $r(\mathbf{Y})$ is the corresponding reward such as BLEU, GLEU and TER for generating sequence $\mathbf{Y}$. Enumerating all the possible target sequences is impossible due to the exponential search space, and REINFORCE (Williams, 1992) gives an elegant way to estimate the gradient for Eq.(3) via sampling a sequence $\mathbf{Y}$ from the probability distribution and estimate the gradient with the gradient of log-probability weighted by the reward $r(\mathbf{Y})$:

$$\nabla_{\theta}\mathrm{L}_{\theta} =$$
$$- \mathop{\mathbb{E}}_{\mathbf{Y}}[\sum_{t=1}^{T} \nabla_{\theta} \log(p(y_t|\boldsymbol{y_{<t}}, \mathbf{X}, \theta)) \cdot r(\mathbf{Y})]. \qquad (4)$$

Current reinforcement learning (RL) methods are designed for autoregressive models. Moreover, previous investigations (Wu et al., 2018; Weaver and Tao, 2013) show that the RL-based training procedure is unstable due to its high variance of gradient estimation.

2.3 Non-Autoregressive Neural Machine Translation

Non-autoregressive neural machine translation (Gu et al., 2017a) is proposed to accelerate the decoding process, which can simultaneously generate target words by discarding the autoregressive mechanism.

The translation probability from $\boldsymbol{X}$ to $\boldsymbol{Y}$ is modeled as follows:

$$P(\boldsymbol{Y}|\boldsymbol{X},\theta) = \prod_{t=1}^{T} p(y_t|\boldsymbol{X},\theta). \quad (5)$$

Given the training set $D = \{X^M, Y^M\}$ with M sentence pairs, the training objective is to maximize the log-likelihood of the training data as:

$$\boldsymbol{\theta} = \arg\max_{\theta}\{\mathcal{L}(\theta)\}$$
$$\mathcal{L}(\theta) = \sum_{m=1}^{M}\sum_{t=1}^{T} \log(p(y_t^m|\boldsymbol{X}^m,\theta)). \quad (6)$$

During decoding, the translation with maximum likelihood can be easily obtained by taking the word with the maximum likelihood in every time step:

$$\hat{y}_t = \arg\max_{y_t} p(y_t|\boldsymbol{X},\theta) \quad (7)$$

NAT models do not utilize the target translation history, which results in its weakness in exploiting the target words collocation knowledge for generating correct target word sequence under the cross-entropy objective function. Compared to autoregressive models, NAT models achieve significant speedup while suffering from a large gap in the translation quality due to the lack of target sequential information.

3 Approaches

To retrieve the sequential information for NAT models for enhancing their translation ability and meanwhile preserving the fast-decoding property, we present two approaches: sequence-level training with a reinforcement algorithm for NAT models (Reinforce-NAT) to exploit the sequential information, and a novel Transformer decoder named FS-decoder to fuse sequential information into the top layer.

3.1 Sequence-Level Training for NAT Models

Word-level objective functions, such as the cross-entropy loss, focus on generating the correct token in each position, which will be inferior for NATs without the target sequential information. We propose to encourage NAT models to generate high-quality sentences rather that correct words with the sequence-level training algorithm (Reinforce-NAT).

Algorithm Derivation

In this section, we present the derivation of Reinforce-NAT and show its low variance and efficiency. We first introduce the REINFORCE algorithm (Williams, 1992) for NAT models.

In NAT models, with the non-autoregressive translation probability defined in Eq.(5), the gradient of the expected loss is:

$$\nabla_{\theta}L_{\theta} = -\sum_{\mathbf{Y}}\nabla_{\theta}\prod_{t=1}^{T} p(y_t|\boldsymbol{X},\theta) \cdot r(\mathbf{Y}). \quad (8)$$

Directly applying the REINFORCE algorithm to Eq.(8) will make the gradient update in every postion guided by the same sentence reward $r(\mathbf{Y})$, which is similar to the method for autoregressive models and is unstable during training. Instead, for NAT models, Eq.(8) can be further reduced to the following form, which is the gradient of target words probability weighted by their corresponding expected rewards[1]:

$$\nabla_{\theta}L_{\theta} = -\sum_{t=1}^{T}\sum_{y_t}\nabla_{\theta}p(y_t|\mathbf{X},\theta) \cdot r(y_t), \quad (9)$$

where $r(y_t)$ is the expected reward when y_t is fixed:

$$r(y_t) = \mathbb{E}_{y_{1:t-1}}\mathbb{E}_{y_{t+1:T}} r(\mathbf{Y}). \quad (10)$$

In Eq.(9), the predicted word y_t in position t is evluated by its corresponding expected reward $r(y_t)$, which is more accurate than the sentence reward $r(\mathbf{Y})$. The $r(y_t)$ can be estimated by Monte Carlo sampling, as illustrated in algorithm 1. Specifically, we fix y_t in position t and sample other words from the probability distribution $p(\cdot|\mathbf{X},\theta))$ for n times. The estimated value of $r(y_t)$ is the average reward of the n sampled sentences. Notice that the expected reward $r(y_t)$ can be estimated without running the decoder for multiple times, which is a major advantage of NAT models in sequence-level training.

---

[1]The proof is provided in the appendix

Algorithm 1 Estimation of $r(y_t)$

Input: the output probability distribution $p(\cdot|\mathbf{X}, \theta))$, t, y_t, T, sampling times n
Output: estimate of $r(y_t)$

1: $r = 0$, $i = 0$
2: **for** $i < n$ **do**
3:     sample $\tilde{\boldsymbol{y}}_{1:t-1}$, $\tilde{\boldsymbol{y}}_{t+1:T}$ from $p(\cdot|\mathbf{X}, \theta))$
4:     $\tilde{\mathbf{Y}} = \{\tilde{\boldsymbol{y}}_{1:t-1}, y_t, \tilde{\boldsymbol{y}}_{t+1:T}\}$
5:     $r \mathrel{+}= r(\tilde{\mathbf{Y}})$
6:     $i \mathrel{+}= 1$
7: $r = r/n$
8: **return** r

Algorithm 2 Reinforce-NAT

Input: the output probability distribution $p(\cdot|\mathbf{X}, \theta))$, traversing count k, sample times n
Output: estimate of $\nabla_\theta \mathrm{L}_\theta$ in position t according to Eq.(12)

1: $T_K = \{\text{words ranking top-}k \text{ in } p(\cdot|\mathbf{X}, \theta))\}$
2: $\nabla_\theta \mathrm{L}_\theta = 0$, $\tilde{p} = p$, $P_k = 0$
3: **for** y_t in T_K **do**
4:     estimate $r(y_t)$ by algorithm 1 with sample times n
5:     $\nabla_\theta \mathrm{L}_\theta \mathrel{-}= \nabla_\theta p(y_t|\mathbf{X}, \theta) \cdot r(y_t)$
6:     $\tilde{p}(y_t|\mathbf{X}, \theta) = 0$
7:     $P_k \mathrel{+}= p(y_t|\mathbf{X}, \theta)$
8: normalize $\tilde{p}(\cdot|\mathbf{X}, \theta)$
9: sample y_t from $\tilde{p}(\cdot|\mathbf{X}, \theta)$
10: estimate $r(y_t)$ by algorithm 1 with sample times n
11: $\nabla_\theta \mathrm{L}_\theta \mathrel{-}= (1 - P_k) \cdot \nabla_\theta \log(p(y_t|\mathbf{X}, \theta)) \cdot r(y_t)$
12: **return** $\nabla_\theta \mathrm{L}_\theta$

The gradient in Eq.(9) can be estimated with REINFORCE (Williams, 1992):

$$\nabla_\theta \mathrm{L}_\theta = -\sum_{t=1}^{T} \mathbb{E}_{y_t}[\nabla_\theta \log(p(y_t|\mathbf{X}, \theta)) \cdot r(y_t)]. \tag{11}$$

Eq.(11) corresponds to a gradient estimation method through sampling a target word y_t and the gradient of the log-probability of y_t weighted by reward $r(y_t)$ is utilized to estimate the expected gradient over the vocabulary. Though the estimation is unbiased, the gradient estimator still suffers from high variance. The variance can be eliminated by traversing the whole vocabulary, but it is unaffordable due to the huge vocabulary size.

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 other words with small probabilities have small gradients[2]. Hence the variance will be effectively reduced if we can eliminate the variance from top-ranking words. This motivates us to compute gradients of the top-ranking words accurately and estimate the rest via the REINFORCE algorithm.

We can build an unbiased estimation of Eq.(9) by traversing top-k words and estimating the rest via one sampling:

$$\nabla_\theta \mathrm{L}_\theta = -\sum_{t=1}^{T} \Big(\sum_{y_t \in T_K} \nabla_\theta p(y_t|\mathbf{X}, \theta) \cdot r(y_t)$$
$$+ (1 - P_k) \cdot \mathbb{E}_{y_t \sim \tilde{p}}[\nabla_\theta \log(p(y_t|\mathbf{X}, \theta)) \cdot r(y_t)]). \tag{12}$$

Algorithm 2 illustrates the proposed method. Although this algorithm will lead to multiple es-

---

[2]In the softmax layer, the gradient is proportional to the output probability

timations of the expected reward $r(y_t)$, the training cost is relatively low for the reason that the independent generation of target words makes NAT models efficient in estimating the expected reward, which will be either very expensive (Yu et al., 2017) or biased (Bahdanau et al., 2016) for autoregressive models.

Reinforce-NAT

To give the clear description, we firstly define symbols in Algorithm 2:

1) $p(\cdot|\mathbf{X}, \theta))$ is the output probability distribution generated by the decoder on the target vocabulary at time t. 2) T_K is the set of target words with top-k probabilities. 3) P_k is the sum of probabilities in T_K, 4) $\tilde{p}$ is the normalized probability distribution after removing probabilities of words in T_K.

The algorithm takes the output probability distribution p, the traversing count k and the sampling times n as input and output the gradient estimation at step t. We divide the gradient estimation procedure at step t into two parts: *traversing* and *sampling*.

The algorithm firstly builds the set T_K with words ranking top-k in probability (line 1), then estimates expected rewards for words in T_K by algorithm 1 (line 3, line 4). The accumulated gradient in T_K are obtained by traversing the words in T_K and accumulating gradients of their probability functions, which are weighted by correspond-

ing rewards (line 5).

After the traversing procedure for accumulating gradients for words in T_K, the algorithm estimates the expected gradient for words that are not in T_K in the *sampling* procedure. The algorithm obtains the probability distribution $\tilde{p}$ over the rest of words through masking probabilities of words in the T_k (line 6, line8). A word y_t from the distribution $\tilde{p}$ (line 9) is sampled to compute the gradient of the log-probability of y_t and then estimate the reward of $r(y_t)$. The weight for this estimation is $1 - P_k$, where P_k is the sum of probabilities in T_K. Finally, the estimated gradient is the sum of gradients from Top-k words and the sampled word. (line 11).

In a word, the algorithm aims to traverse gradients of important words since they can dominate the gradient estimation, and estimate the gradient of less important words via one sampling.

3.2 Fuse Sequential Information

We propose an innovative Transformer decoder named FS-decoder to fuse the target sequential information into the top layer of the decoder. The FS-decoder consists of four parts: bottom layers, the fusion layer, the top layer and the softmax layer. In the decoder, we parallelize bottom layers in an non-autoregressive way to accelerate the model but serialize the top layer in an autoregressive way to enhance the translation quality. The teacher forcing algorithm (Williams and Zipser, 1989) is applied in the training where target embeddings are directly fed to the fusion layer. During decoding, FS-decoder only needs to run the top layer autoregressively.

We illustrate the model in figure 1 and describe the detailed architecture of the FS-decoder in the following. Assume that the original Transformer has n decoder layers, the source sentence has length T_s, the target sentence has length T, and the predicted target length is T'. Here we directly look up the source-target length dictionary to predict the target length.

Bottom Layers. The decoder of FS-decoder contains n-1 bottom layers, which are identical to the decoder layers of NAT models (Gu et al., 2017a). Each layer consists of four sub-layers: the self-attention layer, the positional attention layer, the source side attention layer and the position-wise feed-forward layer. The inputs for bottom decoders $\mathbf{X}'$ are uniformly copied (Gu et al., 2017a)
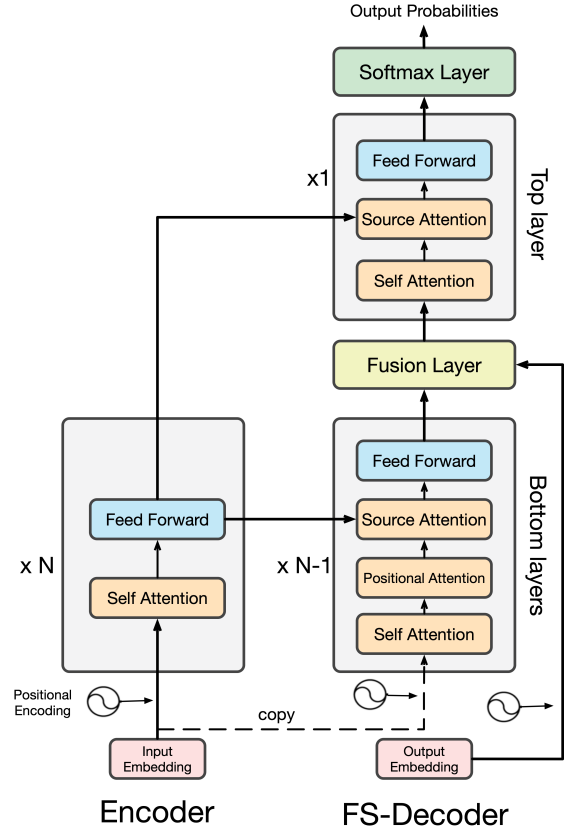

Figure 1: The architecture of FS-decoder. The decoder consists of $n-1$ bottom layers, the fusion layer, the top layer and the softmax layer.

from the source input $\mathbf{X}$ where each decoder input in position t is a copy of the source input in position $\text{Round}(T't/T_s)$:

$$\mathbf{X}' = \text{Uniform}(\mathbf{X}). \qquad (13)$$

The bottom layers take the inputs $\mathbf{X}'$ and output the hidden states $\mathbf{H}'$ with the same length T'.

Fusion Layer. The fusion layer is a linear transformation layer with a ReLU activation, which fuses the outputs from bottom layers $\mathbf{H}'$ and target embeddings $\mathbf{Y}$ in each position t as:

$$\mathbf{H_t} = \mathbf{ReLu}(\mathbf{WH'_t} + \mathbf{UY_t}), \qquad (14)$$

where $\mathbf{W}$ and $\mathbf{U}$ are weight matrices, $t = 1, 2, \cdots, T$. $\mathbf{H}'$ will be padded to length T when T' is smaller than T. Outputs of the fusion layer are then fed to the top layer.

Top Layer. The top layer of the decoder is identical to the original Transformer decoder layer, which does not contain the positional attention layer compared to bottom layers. The outputs are fed to the softmax layer.

3017

Like other autoregressive models, FS-decoder has to generate translations through decoding algorithms such as greedy search and beam search. During decoding, bottom layers run in advance to prepare the inputs for the fusion layer, and then the fusion layer and top layer run autoregressively with the embedding of predicted token fed to the fusion layer.

4 Related Work

Gu et al. (2017a) introduced the non-autoregressive Transformer model to accelerate the translation. Lee et al. (2018) proposed a non-autoregressive sequence model based on iterative refinement, where the outputs of the decoder are fed back as inputs in the next iteration. Guo et al. (2018) proposed to enhance the decoder inputs with phrase-table lookup and embedding mapping. Kaiser et al. (2018) used a sequence of autoregressively generated discrete latent variables as inputs of the decoder. Knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016) is a method for training a smaller and faster student network to perform better by learning from a teacher network, which is crucial in NAT models. Gu et al. (2017a) applied Sequence-level knowledge distillation to eliminate the multi-modality in the training corpus. Li et al. (2018) further proposed to improve non-autoregressive models through distilling knowledge from intermediary hidden states and attention weights of autoregressive models.

Apart from non-autoregressive translation, there are works toward speeding up the translation from other perspectives. Wang et al. (2018) proposed the semi-autoregressive Transformer that generates a group of words in parallel at each time step. Press and Smith (2018) proposed the eager translation model that does not use the attention mechanism and has low latency. Zhang et al. (2018a) proposed the average attention network to accelerate decoding, which achieves significant speedup over the uncached Transformer. Zhang et al. (2018b) proposed cube pruning to speedup the beam search for neural machine translation without damaging the translation quality.

Sequence-level training techniques have been widely explored in autoregressive neural machine translation, where most works (Ranzato et al., 2015; Shen et al., 2016; Wu et al., 2016; He et al., 2016; Wu et al., 2017; Yang et al., 2017)

relied on reinforcement learning (Williams, 1992; Sutton et al., 2000) to build the gradient estimator. Recently, techniques for sequence-level training with continuous objectives have been explored, including deterministic policy gradient algorithms (Gu et al., 2017b), bag-of-words objective (Ma et al., 2018) and probabilistic n-gram matching (Shao et al., 2018). However, to the best of our knowledge, sequence-level training has not been applied to non-autoregressive models yet.

The methods of variance reduction through focusing on the important parts of the distribution include importance sampling (Bengio et al., 2003; Glynn and Iglehart, 1989) and complementary sum sampling (Botev et al., 2017). Importance sampling estimates the properties of a particular distribution through sampling on a different proposal distribution. Complementary sum sampling reducdes the variance through suming over the important subset and estimating the rest via sampling.

5 Experiments

5.1 Settings

Dataset. We conduct experiments on three translation tasks[3]: IWSLT16 En→De (196k pairs), WMT14 En↔De (4.5M pairs) and WMT16 En↔Ro (610k pairs). We use the preprocessed datasets released by Lee et al. (2018), where all sentences are tokenized and segmented into sub-word units using the BPE algorithm (Sennrich et al., 2016). For all tasks, source and target languages share the vocabulary with size 40k. 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.

Baselines. We take the Transformer model (Vaswani et al., 2017) as the autoregressive baseline. The non-autoregressive model based on iterative refinement (Lee et al., 2018) is the non-autoregressive baseline, and we set the number of iterations to 2.

Pre-train. To evaluate the sequence-level training methods, we pre-train the NAT baseline first and then fine-tune the baseline model with GLEU

---

[3]We release the source code in https://github.com/ictnlp/RSI-NAT

		IWSLT'16 En-De			WMT'16 En-Ro				WMT'14 En-De			
		En→	toks/s	speedup	secs/b	En→	Ro→	toks/s	speedup	En→	De→	toks/s	speedup
AR	b=1	28.13	45.3	1.09×	0.20	31.53	31.35	45.6	1.23×	23.67	28.04	33.7	1.13×
	b=4	28.25	41.6	1.00×	0.20	31.85	31.60	37.1	1.00×	24.29	28.86	29.9	1.00×
NAT	FT	26.52	–	15.6×	–	27.29	29.06	–	–	17.69	21.47	–	–
	FT+NPD	28.16	–	2.36×	–	29.79	31.44	–	–	19.17	23.20	–	–
IRNAT	iter=2	24.82	423.8	6.64×	–	27.10	28.15	332.7	7.68×	16.95	20.39	393.6	8.77×
	adaptive	27.01	125.9	1.97×	–	29.66	30.30	118.3	2.73×	21.54	25.43	107.2	2.39×
Our Models	NAT-base	24.13	350.2	8.42×	0.62	25.96	26.49	349.0	9.41×	16.05	19.46	321.7	10.76×
	+REINFORCE	24.30	354.1	8.51×	2.51	26.49	27.20	346.7	9.35×	18.47	21.89	323.2	10.81×
	+Reinforce-NAT	25.18	350.6	8.43×	13.40	27.09	27.93	350.3	9.44×	19.15	22.52	320.9	10.73×
	FS-decoder(b=1)	27.58	168.7	4.06×	0.241	30.53	30.68	170.5	4.60×	21.53	27.20	143.3	4.79×
	FS-decoder(b=4)	27.78	140.8	3.38×	0.241	30.57	30.83	137.1	3.70×	22.27	27.25	112.2	3.75×

Table 2: Generation quality (4-gram BLEU), decoding efficiency (tokens/sec), speedup and training speed (seconds/batch). Decoding efficiency is measured sentence-by-sentence from the En→ direction. Speedup is calculated over the autoregressive Transformer with beam size 4. NAT: non-autoregressive transformer models (Gu et al., 2017a). IRNAT: iterative refinement for NAT (Lee et al., 2018). AR: the autoregressive Transformer model. b: beam size. FS-decoder: fuse the sequential information into the top layer. NAT-base: our non-autoregressive baseline. +REINFORCE: finetune the NAT-base with REINFORCE according to Eq.(11). +Reinforce-NAT: finetune the NAT-base with Reinforce-NAT according to Eq.(12).

(Wu et al., 2016), which outperforms other metrics in our experiments. We stop the pre-train procedure, when training steps are more than 300k and no further improvements on the validation set are observed in last 100k steps.

Hyperparameters. We closely follow the setting of Gu et al. (2017a) and Lee et al. (2018). In IWSLT16 En-De, we use the small model (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 Vaswani et al. (2017) (d_{model}=512, d_{hidden}=512, n_{layer}=6, n_{head}=8, $p_{dropout}$=0.1, t_{warmup}=16000). The traversing count k and the sampling times n in algorithm 2 are respectively set to 5 and 20. We use Adam (Kingma and Ba, 2014) for the optimization. During decoding, we remove any token that is generated repeatly. The decoding speed is measured on a single Geforce GTX TITAN X.

Knowledge Distillation. Knowledge distillation (Kim and Rush, 2016; Hinton et al., 2015) is proved to be crucial for successfully training NAT models (Gu et al., 2017a; Li et al., 2018). For all the translation tasks, we apply sequence-level knowledge distillation to construct the distillation corpus where the target side of the training corpus is replaced by the output of an autoregressive Transformer model. We use original corpora to train the autoregressive baseline and distillation corpora to train other models.

5.2 Main Results

We compare our models with the NAT (Gu et al., 2017a) and the IRNAT (Lee et al., 2018). Table 2 shows the experiment results. We observe that models based on sequence-level training approaches, including REINFORCE and Reinforce-NAT, significantly surpass the NAT baseline on BLEU without damaging the decoding speed. The Reinforce-NAT model outperforms the RE-INFORCE model in terms of BLEU points. On WMT14 En↔De, the Reinforce-NAT model achieves significant improvements by more than 3 BLEU points and outperforms NAT(FT) (Gu et al., 2017a) and IRNAT(iteration=2) (Lee et al., 2018). The above results demonstrate the effectiveness of sequence-level training and prove the strong ability of Reinforce-NAT. The experiment on the FS-decoder show that it brings huge BLEU improvements over the NAT baseline and even achieves comparable performance to the autoregressive Transformer with considerable speedup, which proves the capacity of the FS-decoder.

5.3 Training Speed

Table 2 shows the training time per batch of our methods. Sequence-level training methods (i.e., REINFORCE and Reinforce-NAT) are slower than the word-level training. The bottleneck lies in the calculation of the reward (i.e., GLEU), which takes place in CPU and can be accelerated by multi-processing. Besides, these methods are only

utilized to fine-tune the baseline model and take less than 10,000 batches to converge, which make the relatively low training speed affordable.

5.4 Effect of top-k size in Reinforce-NAT

The Reinforce-NAT is proposed on the basis that the top-k words can occupy the central part of the probability distribution. However, it remains unknown which k is appropriate for us. A large k will slow down the training, and a small k will be not enough to dominate the probability distribution. We statistically and experimentally analyze the choice of k in Reinforce-NAT. We respectively set k to 1, 5 and 10 and record the top-k probabilities in 10,000 target word predictions. Figure 2 and Table 3 illustrate the statistical properties of top-k probabilities. In figure 2, the x-axis divides the probability distribution into 5 intervals, and the y-axis indicates the number of times that the top-k probabilities are within this interval. In Table 3, we estimate the expection of top-k probabilities for different k. We find that $k = 5$ is a desirable choice that can cover a large portion of the probability distribution, and the marginal utility for a larger k is limitted.
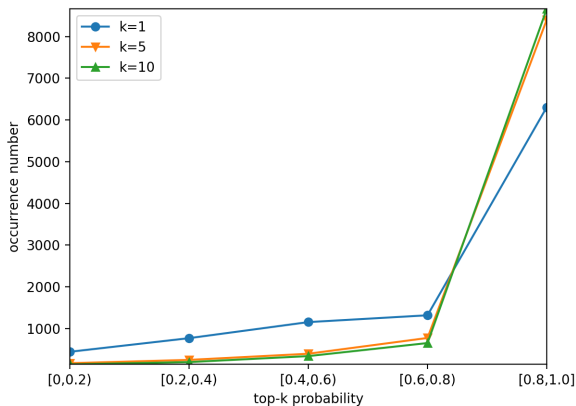

Figure 2: top-k probability distributions for k=1, 5 and 10

k	1	5	10	100	1000
$E[P_k]$	0.818	0.916	0.929	0.948	0.968

Table 3: top-k probability expection for k=1, 5, 10, 100, 1000

We further conduct experiments on IWSLT16 En→De to confirm the conclusion. We respectively set k to 0, 1, 5 and 10 in Reinforce-NAT and draw training curves. Figure 3 shows that

REINFORCE($k = 0$) is very unstable in the training, and greater k in Reinforce-NAT generally leads to better performance. In line with our previous conclusion, $k = 5$ is an ideal choice since it does not have a large performance gap between larger k.
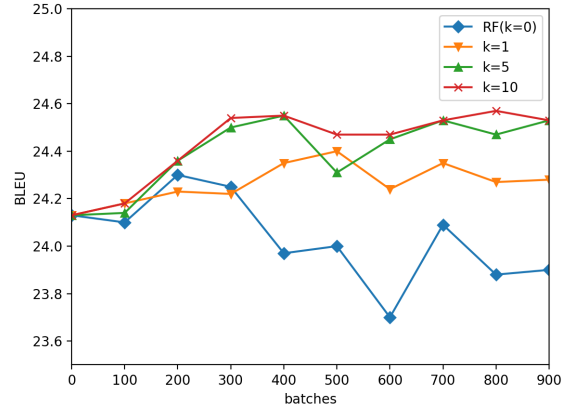

Figure 3: training curves for $k = 0, 1, 5$ and 10.

5.5 Performance over Different Lengths

Table 2 shows that the performance of Reinforce-NAT varies with datasets. Though IWSLT16 En→De and WMT14 En→De have the same language pair, Reinforce-NAT achieves an improvement of more than 3 BLEU points on WMT14 but only have about 1.0 BLEU points improvement on IWSLT16. We attribute this phenomenon to the length difference between two datasets. The WMT14 En→De dataset is in the news-domain, whose sentences are statistically longer than the spoken-domain IWSLT16 En→De dataset.

Figure 4 shows BLEU scores over sentences in different length buckets. The BLEU scores of NAT-Base have a distinct decrease when the sentence length is over 40, while other models perform well on long sentences. It confirms that NAT models are weak in translating long sentences and our solutions can effectively improve the performance of NAT models on long sentences through leveraging sequential information.

5.6 Case Study

In Table 4, we present a translation case from the validation set of WMT14 De→En. The case shows that the translation quality rise in the order of NAT-Base, +Reinforce-NAT, FS-decoder to AR-Base and the performance gap is large between NAT-Base and other models. Particularly, NAT models suffer from over-translation and

Source	und noch tragischer ist , dass es Oxford war - eine Universitt , 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 .
NAT-Base	and more more more more that it was Oxford - a university that not not only only TTory Prime Minister , but has has to hidden hidden behind an unfounded reputation of equality and freedom of thought .
Reinforce-NAT	and more more tragic is that it was Oxford - a university that did not only produce 14 Tory Prime Minister , but has still to be hidden behind an unfied reputation of equality and freedom of thought .
FS-decoder	and even more tragic , it was Oxford - a university that produced not only 14 Tory Prime Minister , but still hidden behind an unbridled reputation of equality and freedom of thought .
AR-Base	and , more tragic , Oxford was - a university that not only produced 14 Tory Prime Minister , but still hidden behind an unprecedented reputation for equality and freedom of thought .

Table 4: A translation case on WMT14 De→En task. Over-translation and under-translation errors occur in the translation of NAT-Base.
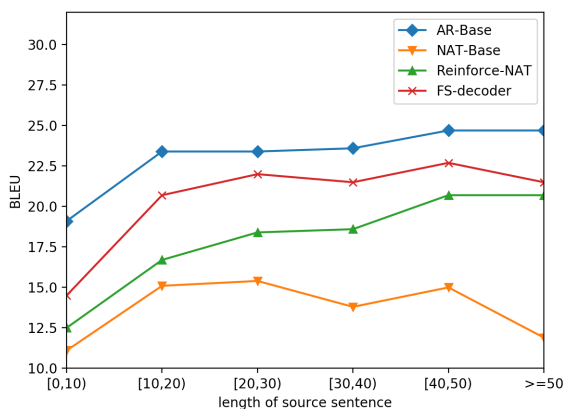

Figure 4: The BLEU scores on the validation set of WMT14 En→De over sentences in different length buckets. The beam size of FS-decoder and AR-Base is 1.

under-translation when translating long sentences, which is efficiently alleviated by Reinforce-NAT and RF-Decoder.

6 Conclusion

In this paper, we aim to retrieve the sequential information for NAT models to enhance their translation ability while preserving fast-decoding property. Firstly, we propose a sequence-level training method based on a novel reinforcement algorithm for NAT (Reinforce-NAT), which significantly improves the performance of NAT models without decelerating the decoding speed. Secondly, we propose an innovative Transformer decoder named FS-decoder to fuse the target sequential information into the top layer of the decoder, which achieves comparable performance to the Transformer and still maintains substantial speedup.

In the future, we plan to investigate better methods to leverage the sequential information. We believe that the following two directions are worth study. First, exploiting other sequence-level training objectives like bag-of-words (Ma et al., 2018). Second, using sequential information distilled from the autoregressive teacher model to guide the training of the student non-autoregressive model.

7 Acknowledgments

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Jean-Sébastien Senécal, et al. 2003. Quick training of probabilistic neural nets by importance sampling. In *AISTATS*, pages 1–9.

Aleksandar Botev, Bowen Zheng, and David Barber. 2017. Complementary sum sampling for likelihood approximation in large scale classification. In *Artificial Intelligence and Statistics*, pages 1030–1038.

Kyunghyun Cho, 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. *arXiv preprint arXiv:1406.1078*.

Peter W Glynn and Donald L Iglehart. 1989. Importance sampling for stochastic simulations. *Management Science*, 35(11):1367–1392.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017a. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Jiatao Gu, Kyunghyun Cho, and Victor OK Li. 2017b. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978.

Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. 2018. Non-autoregressive neural machine translation with enhanced decoder input. *arXiv preprint arXiv:1812.09664*.

Di He, 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*, pages 820–828.

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

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Pamar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*.

Yoon Kim 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.

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

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*.

Zhuohan Li, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Hint-based training for non-autoregressive translation.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. *arXiv preprint arXiv:1805.04871*.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ofir Press and Noah A. Smith. 2018. You may not need attention. *CoRR*, abs/1810.13409.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Chenze Shao, 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.

Shiqi Shen, 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)*, volume 1, pages 1683–1692.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Richard Stuart Sutton. 1984. Temporal credit assignment in reinforcement learning.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Chunqi Wang, Ji Zhang, and Haiqing Chen. 2018. Semi-autoregressive neural machine translation. *arXiv preprint arXiv:1808.08583*.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. *arXiv preprint arXiv:1902.10245*.

Lex Weaver and Nigel Tao. 2013. The optimal reward baseline for gradient-based reinforcement learning. *Processings of the Seventeeth Conference on Uncertainty in Artificial Intelligence*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1808.08866*.

Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.

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

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2017. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018a. Accelerating neural transformer via an average attention network. *arXiv preprint arXiv:1805.00631*.

Wen Zhang, Liang Huang, Yang Feng, Lei Shen, and Qun Liu. 2018b. Speeding up neural machine translation decoding by cube pruning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4284–4294.

A   Supplemental Material

Proof for Eq.(9):

$$
\begin{aligned}
-\nabla_\theta \mathrm{L}_\theta &= \sum_{\mathbf{Y}} \nabla_\theta \prod_{t=1}^{T} p(y_t|\boldsymbol{X},\theta) \cdot r(\mathbf{Y}) \\
&= \sum_{\mathbf{Y}} \sum_{t=1}^{T} \nabla_\theta p(y_t|\boldsymbol{X},\theta) \cdot \prod_{i=1}^{t-1} p(y_i|\boldsymbol{X},\theta) \cdot \prod_{j=t+1}^{T} p(y_j|\boldsymbol{X},\theta) \cdot r(\mathbf{Y}) \\
&= \sum_{t=1}^{T} \sum_{\mathbf{Y}} \nabla_\theta p(y_t|\boldsymbol{X},\theta) \cdot \prod_{i=1}^{t-1} p(y_i|\boldsymbol{X},\theta) \cdot \prod_{j=t+1}^{T} p(y_j|\boldsymbol{X},\theta) \cdot r(\mathbf{Y}) \\
&= \sum_{t=1}^{T} \sum_{y_t} \nabla_\theta p(y_t|\boldsymbol{X},\theta) \cdot \sum_{\mathbf{y_{1:t-1}}} \sum_{\mathbf{y_{t+1:T}}} \prod_{i=1}^{t-1} p(y_i|\boldsymbol{X},\theta) \cdot \prod_{j=t+1}^{T} p(y_j|\boldsymbol{X},\theta) \cdot r(\mathbf{Y}) \\
&= \sum_{t=1}^{T} \sum_{y_t} \nabla_\theta p(y_t|\boldsymbol{X},\theta) \cdot \mathop{\mathbb{E}}_{y_{1:t-1}} \mathop{\mathbb{E}}_{y_{t+1:T}} r(\mathbf{Y}). \\
&= \sum_{t=1}^{T} \sum_{y_t} \nabla_\theta p(y_t|\mathbf{X},\theta) \cdot r(y_t)
\end{aligned}
\tag{15}
$$