Rewriter-Evaluator Architecture for Neural Machine Translation

Yangming Li[♠] and **Kaisheng Yao**[♡]
[♠]Tencent AI Lab
[♡]Ant Group
newmanli@tencent.com,kaisheng.yao@antgroup.com

Abstract

A few approaches have been developed to improve neural machine translation (NMT) models with multiple passes of decoding. However, their performance gains are limited because of lacking proper policies to terminate the multi-pass process. To address this issue, we introduce a novel architecture of *Rewriter-Evaluator*. Translating a source sentence involves multiple rewriting passes. In every pass, a rewriter generates a new translation to improve the past translation. Termination of this multi-pass process is determined by a score of translation quality estimated by an evaluator. We also propose prioritized gradient descent (PGD) to jointly and efficiently train the rewriter and the evaluator. Extensive experiments on three machine translation tasks show that our architecture notably improves the performances of NMT models and significantly outperforms prior methods. An oracle experiment reveals that it can largely reduce performance gaps to the oracle policy. Experiments confirm that the evaluator trained with PGD is more accurate than prior methods in determining proper numbers of rewriting.

1 Introduction

Encoder-Decoder architecture (Sutskever et al., 2014) has been widely used in natural language generation, especially neural machine translation (NMT) (Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017; Zhang et al., 2019; Kitaev et al., 2020). Given a source sentence, an encoder firstly converts it into hidden representations, which are then conditioned by a decoder to produce a target sentence. In analogy to the development of statistical machine translation (SMT) (Och and Ney, 2002; Shen et al., 2004; Zhang and Gildea, 2008), some recent methods in NMT attempt to improve the encoder-decoder architecture with multi-pass decoding (Xia et al., 2017; Zhang et al., 2018; Geng et al., 2018; Niehues et al., 2016). In these models, more than one translation is generated for a source sentence. Except for the first translation, each of the later translations is conditioned on the previous one. While these methods have achieved promising results, they lack a proper termination poqlicy for this multi-turn process. For instance, Xia et al. (2017); Zhang et al. (2018) adopt a fixed number of decoding passes, which is inflexible and can be sub-optimal. Geng et al. (2018) utilize reinforcement learning (RL) (Sutton et al., 2000) to automatically decide the number of decoding passes. However, RL is known to be unstable due to the high variance in gradient estimation (Boyan and Moore, 1995).

To address this problem, we introduce a novel architecture, *Rewriter-Evaluator*. This architecture contains a rewriter and an evaluator. The translation process involves multiple passes. Given a source sentence, at every turn, the rewriter generates a new target sequence to improve the translation from the prior pass, and the evaluator measures the translation quality to determine whether to end the iterative rewriting process. Hence, the translation process is continued until a certain condition is met, such as no significant improvement in the measured translation quality. In implementations, the rewriter is a conditional language model (Sutskever et al., 2014) and the evaluator is a text matching model (Wang et al., 2017).

We also propose prioritized gradient descent (PGD) that facilitates training the rewriter and the evaluator both jointly and efficiently. PGD uses a priority queue to store previous translation cases. The queue stores translations with descending order of their scores, computed from the evaluator. The capacity of the queue is limited to be a few times of batch size. Due to its limited size, the queue pops those translations with high scores and only keeps the translations with lower scores. The samples in

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 5701–5710
August 1–6, 2021.

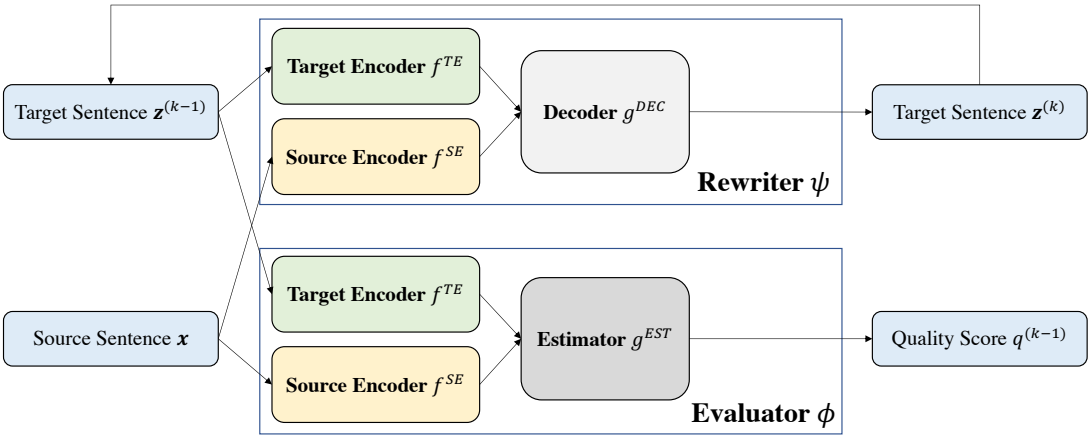

Figure 1: General architecture of *Rewriter-Evaluator*.

the queue are combined together with new cases from the training data to train the rewriter.

Rewriter-Evaluator has been applied to improve two mainstream NMT models, RNNSearch (Bahdanau et al., 2015) and Transformer (Vaswani et al., 2017). We have conducted extensive experiments on three translation tasks, NIST Chinese-to-English, WMT'18 Chinese-to-English, and WMT'14 English-to-German. The results show that our architecture notably improves the performance of NMT models and significantly outperforms related approaches. We conduct oracle experiment to understand the source of improvements. The oracle can pick the best translation from all the rewrites. Results indicate that the evaluator helps our models achieve the performances close to the oracle, outperforming the methods of fixing the number of rewriting turns. Compared against averaged performances using a fixed number of rewriting iterations, performance gaps to the oracle can be reduced by 80.7% in the case of RNNSearch and 75.8% in the case of Transformer. Quantitatively, we find the evaluator trained with PGD is significantly more accurate in determining the optimal number of rewriting turns. For example, whereas the method in Geng et al. (2018) has 50.2% accuracy in WMT'14, the evaluator achieves 72.5% accuracy on Transformer.

2 Rewriter-Evaluator

Rewriter-Evaluator consists of iterative processes involving a rewriting process ψ and an evaluation process ϕ. The process of translating an n-length source sentence $\mathbf{x} = [x_1, x_2, \cdots, x_n]$ is an application of the above processes. Assume we are at the k-th iteration ($k \geq 1$). The rewriter ψ generates a target sequence $\mathbf{z}^{(k)} = [z_1^{(k)}, z_2^{(k)}, \cdots, z_{l_k}^{(k)}]$ given the source sentence $\mathbf{x}$ and the past translation $\mathbf{z}^{(k-1)} = [z_1^{(k-1)}, z_2^{(k-1)}, \cdots, z_{l_{k-1}}^{(k-1)}]$ from the $(k-1)$-th turn. l_k and l_{k-1} are the sentence lengths. The evaluator ϕ estimates the translation quality score $q^{(k)}$ of the new translation $\mathbf{z}^{(k)}$, which is used for determining whether to end the multi-turn process. Formally, the k-th pass of a translation process is defined as

$$\begin{cases} \mathbf{z}^{(k)} = \psi(\mathbf{x}, \mathbf{z}^{(k-1)}) \\ q^{(k)} = \phi(\mathbf{x}, \mathbf{z}^{(k)}) \end{cases}. \quad (1)$$

Initially, $\mathbf{z}^{(0)}$ and $q^{(0)}$ are respectively set as an empty string and $-\infty$.

The above procedure is repeatedly carried out until not much improvement in the estimated quality score can be achieved, i.e.,

$$q^{(k)} + \epsilon < q^{(k-1)}, \epsilon > 0, \quad (2)$$

where ϵ is a small value tuned on the development set. Alternatively, the procedure is terminated if a certain number of iterations $K > 0$ is reached. In the former case, we adopt $\mathbf{z}^{(k-1)}$ as the final translation. In the latter case, the last translation $\mathbf{z}^{(K)}$ is accepted.

2.1 Architecture

A general architecture of *Rewriter-Evaluator* using Encoder-Decoder is illustrated in Fig. 1. The rewriter ψ consists of a source encoder f^{SE}, a target decoder f^{TE}, and a decoder g^{DEC}. The evaluator ϕ shares encoders with the rewriter and contains an estimator g^{EST}.

Assume it is at the k-th pass. Firstly, the source encoder f^{SE} casts the source sentence $\mathbf{x}$ into word

Algorithm 1: Prioritized Gradient Descent (PGD)

Input: rewriter ψ, evaluator ϕ, training set T, batch size B, and expected iteration number E.
Output: well-trained rewriter ψ and well-trained evaluator ϕ.

Initialize an empty priority queue A with the capacity $C \leftarrow B \times E$.
while *Models are not converged* **do**
  Pop B cases with high quality scores from priority queue A and discard them.
  Randomly sample a B-sized batch of training cases S from T.
  for $(\mathbf{x}, \mathbf{y}) \in S$ **do**
    Push the quadruple $(\mathbf{x}, \mathbf{y}, [\text{“SOS”}, \text{“EOS”}], -\infty)$ into queue A.
  Initialize an empty priority queue D of limited size C.
  Initialize an empty list F to collect samples for training.
  for $(\mathbf{x}, \mathbf{y}, \mathbf{z}^{(k-1)}, r^{(k-1)}) \in A$ **do**
    Obtain translation $\mathbf{z}^{(k)}$ and quality score $q^{(k)}$, respectively, using Eq. (5) and Eq. (6).
    Push sample $(\mathbf{x}, \mathbf{y}, \mathbf{z}^{(k)}, q^{(k)})$ into list F.
    Compute quality rate $r^{(k)}$ using Eq. (9).
    Push quadruple $(\mathbf{x}, \mathbf{y}, \mathbf{z}^{(k)}, r^{(k)})$ into queue D.
  Optimize rewriter ψ with the samples in list F to reduce loss in Eq. (7).
  Optimize evaluator ϕ with the samples in list F to reduce loss in Eq. (8).
  Update priority queue A: $A \leftarrow D$.

representations $\mathbf{h}_i, 1 \leq i \leq n$:

$$\mathbf{H} = [\mathbf{h}_1; \mathbf{h}_2; \cdots; \mathbf{h}_n] = f^{SE}(\mathbf{x}), \tag{3}$$

where operation $[;]$ is row-wise vector concatenation. Similarly, the translation $\mathbf{z}^{(k-1)}$ from the previous turn $k-1$ is encoded as

$$\begin{aligned} \mathbf{P}^{(k-1)} &= [\mathbf{p}_1^{(k-1)}; \mathbf{p}_2^{(k-1)}; \cdots; \mathbf{p}_{l_{k-1}}^{(k-1)}] \\ &= f^{TE}(\mathbf{z}^{(k-1)}) \end{aligned}. \tag{4}$$

Then, the decoder g^{DEC} of the rewriter ψ produces a new translation $\mathbf{z}^{(k)}$ as

$$\mathbf{z}^{(k)} = g^{DEC}(\mathbf{H}, \mathbf{P}^{(k-1)}). \tag{5}$$

Ultimately, the evaluator ϕ scores the new translation $\mathbf{z}^{(k)}$ with the estimator g^{EST}:

$$\begin{cases} \mathbf{P}^{(k)} = f^{TE}(\mathbf{z}^{(k)}) \\ q^{(k)} = g^{EST}(\mathbf{H}, \mathbf{P}^{(k)}) \end{cases}. \tag{6}$$

The implementation can be applied to a variety of architectures. The encoders, f^{SE} and f^{TE}, can be any sequence model, such as CNN (Kim, 2014). The decoder g^{DEC} is compatible with any language model (e.g., Transformer). The estimator g^{EST} is a text matching model, e.g., ESIM (Chen et al., 2017). In Sec. 4, we apply this implementation to improve generic NMT models.

2.2 Training Criteria

We represent the ground truth target sentence as a $(m+1)$-length sequence $\mathbf{y} = [y_0, y_1, \cdots, y_m]$. The rewriter ψ is trained via teacher forcing. We use $\mathbf{o}_i$ to denote the probability of the i-th target word, which is the prediction of feeding its prior words $[y_0, y_1, \cdots, y_{i-1}]$ into the decoder g^{DEC}. The training loss for the rewriter is

$$\mathcal{J}^{\psi} = \sum_{1 \leq i \leq m} -\log(\mathbf{o}_i[y_i]). \tag{7}$$

where $y_0 =$ “[SOS]” and $y_m =$ “[EOS]”, marking the ends of a target sentence.

For the evaluator ϕ, we incur a hinge loss between the translation score of the ground truth $\mathbf{y}$ and that of the current translation $\mathbf{z}^{(k)}$ as

$$\begin{cases} q^* = \phi(\mathbf{x}, \mathbf{y}) \\ \mathcal{J}^{\phi} = \max(0, 1 - q^* + q^{(k)}) \end{cases}. \tag{8}$$

At training time, translation $\mathbf{z}^{(k)}$ is generated via greedy search, instead of beam search, to reduce training time.

3 Prioritized Gradient Descent

We present prioritized gradient descent (PGD) to train the proposed architecture. Instead of the random sampling used in stochastic gradient descent

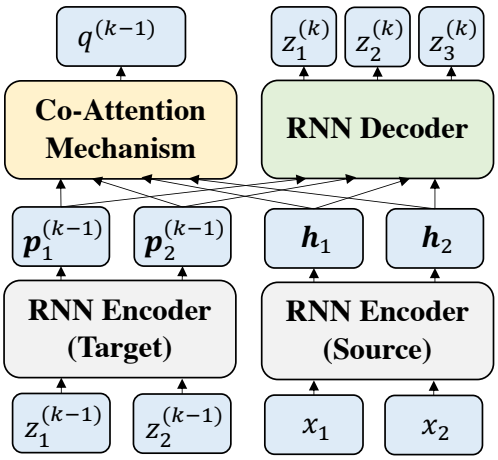

Figure 2: RNNSearch with *Rewriter-Evaluator*.

(SGD) (Bottou and Bousquet, 2008), PGD uses a priority queue to store previous training cases that receive low scores from the evaluator. Randomly sampled training cases together with those from the priority queue are used for training.

Details of PGD are illustrated in Algorithm 1. Initially, we set a priority queue A (1-st line) with a limited size $C = B \times E$. B is the batch size. E, the expected number of rewriting iterations, is set as $\frac{K}{2}$. The queue A is ordered with a quality rate in descending order, where the top one corresponds to the highest rate. The quality rate of a certain sample $(\mathbf{x}, \mathbf{y}, \mathbf{z}^{(k)})$ is computed as

$$r^{(k)} = (1-\rho) * \text{BLEU}(\mathbf{z}^{(k)}, \mathbf{y}) + \rho * q^{(k)}, \quad (9)$$

where the weight ρ is controlled by an annealing schedule $\frac{j}{j+1}$ with j being the current training epoch and BLEU (Papineni et al., 2002). The rate $r^{(k)}$ is dominated by BLEU in the first few epochs, and is later dominated by the evaluation score $q^{(k)}$ with an increasing number of epochs. This design is to mitigate the cold start problem when training an evaluator ϕ. At every training epoch, PGD firstly discards a certain number of previous training samples with high quality rates (3-rd line) from queue A. It then replaces them with newly sampled samples S (4-th to 6-th lines). Every sample $(\mathbf{x}, \mathbf{y}, \mathbf{z}^{(k-1)}, r^{(k-1)})$ in queue A is then rewritten into a new translation $\mathbf{z}^{(k)}$ by the rewriter. These are scored by the evaluator ϕ (10-th lines). These new samples are used to respectively train the rewriter ψ and the evaluator ϕ (14-th to 15-th lines) with Eq. (7) and Eq. (8).

PGD keeps low-quality translations in the queue A for multi-pass rewriting until they are popped out from queue A with high scores from the evaluator ϕ. Hence, the evaluator ϕ is jointly trained with the rewriter to learn discerning the quality of translations from the rewriter ψ, in order to help the rewriter reduce loss in Eq. (7).

PGD uses a large queue ($B \times E$) to aggregate the past translations and newly sampled cases. Computationally, this is more efficient than explicit B times of rewriting to obtain samples. This requires extra memory space in exchange for lowing training time. In Sec. 5.7, we will show that the additional increase of training time by PGD is less than 20%, which is tolerable.

4 Applications

Following Sec. 2.1, we use *Rewriter-Evaluator* to improve RNNSearch and Transformer.

RNNSearch w/ Rewriter-Evaluator. The improved RNNSearch is illustrated in Fig. 2. The two encoders (i.e., f^{SE} and f^{TE}) and the decoder g^{DEC} are GRU (Chung et al., 2014). We omit computation details of these modules and follow their settings in Bahdanau et al. (2015). Note that, at every decoding step, the hidden state of decoder is attended to not only $\mathbf{h}_i, 1 \le i \le n$ but also $\mathbf{p}_j^{(k-1)}, 1 \le j \le l_{k-1}$.

We apply co-attention mechanism (Parikh et al., 2016) to model the estimator f^{EST}. Firstly, we capture the semantic alignment between the source sentence $\mathbf{x}$ and the translation $\mathbf{z}^{(k-1)}$ as

$$\begin{cases} \alpha_{i,j} = \mathbf{h}_i^T \mathbf{W} \mathbf{p}_j^{(k-1)} \\ \widetilde{\mathbf{h}}_i = \sum_j \frac{\exp(\alpha_{i,j})}{\sum_{j'} \exp(\alpha_{i,j'})} \mathbf{p}_j^{(k-1)} \\ \widetilde{\mathbf{p}}_j^{(k-1)} = \sum_i \frac{\exp(\alpha_{i,j})}{\sum_{i'} \exp(\alpha_{i',j})} \mathbf{h}_i \end{cases}. \quad (10)$$

Then, we use average pooling to extract features and compute the quality score:

$$q^{(k-1)} = \mathbf{v}^T \Big(\frac{\sum_i \widetilde{\mathbf{h}}_i}{n} \oplus \frac{\sum_j \widetilde{\mathbf{p}}_j^{(k-1)}}{l_{k-1}} \Big), \quad (11)$$

where $\oplus$ is column-wise vector concatenation.

Transformer w/ Rewriter-Evaluator. The Transformer (Vaswani et al., 2017) is modified to an architecture in Fig. 3. The input to the encoder contains a source sentence $\mathbf{x}$, a special symbol "ALIGN", and the past translation $\mathbf{z}^{(k-1)}$:

$$\mathbf{x}' = \mathbf{x} \odot [\text{"ALIGN"}] \odot \mathbf{z}^{(k-1)}, \quad (12)$$

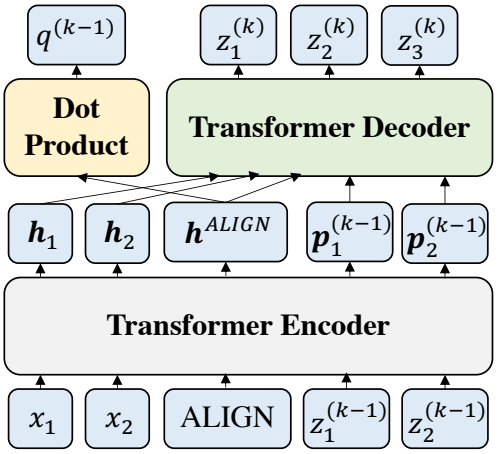

Figure 3: Transformer with *Rewriter-Evaluator*.

where operation $\odot$ denotes the concatenation of two sequences.

The following mask matrix is applied to every layer in the encoder:

$$\begin{bmatrix} \mathbf{1}_{n\times n} & \mathbf{0}_{1\times n}^T & \mathbf{0}_{n\times l_{k-1}} \\ \mathbf{1}_{1\times n} & 1 & \mathbf{1}_{1\times l_{k-1}} \\ \mathbf{0}_{l_{k-1}\times n} & \mathbf{0}_{1\times l_{k-1}}^T & \mathbf{1}_{l_{k-1}\times l_{k-1}} \end{bmatrix}. \quad (13)$$

In this way, the words in $\mathbf{x}$ can't attend to those in $\mathbf{z}^{(k-1)}$ and vice versa. "ALIGN" can attend to the words both in $\mathbf{x}$ and $\mathbf{z}^{(k-1)}$. This design is to avoid cross-sentence attention in encoder layers. In earlier studies, we find it slightly improves the performances of models.

We denote the representation for "ALIGN" in the final encoder layer as $\mathbf{h}^{ALIGN}$. The estimator f^{EST} obtains the quality score as

$$q^{(k-1)} = \mathbf{v}^T \mathbf{h}^{ALIGN}, \quad (14)$$

in which $\mathbf{v}$ is a learnable vector.

5 Experiments

We have conducted extensive experiments on three machine translation tasks: NIST Chinese-to-English (Zh→En), WMT'18 Chinese-to-English, and WMT'14 English-to-German (En→De). The results show that *Rewriter-Evaluator* significantly improves the performances of NMT models and notably outperforms prior post-editing methods. Oracle experiment verifies the effectiveness of the evaluator. Termination accuracy analysis shows our evaluator is much more accurate than prior methods in determining the optimal number of rewriting turns. We also perform ablation studies to explore the effects of some components.

5.1 Experimental Setup

For NIST Zh→En, the training set contains 1.25M sentence pairs extracted from LDC corpora, including LDC2002E18, LDC2003E07, LDC2003E14, a portion of LDC2004T07, LDC2004T08, and LDC2005T06. We adopt NIST 2002 (MT02) as the validation set. We use NIST 2003 (MT03), NIST 2004 (MT04), NIST 2005 (MT05), and NIST 2006 (MT06) for tests. For WMT'18 Zh→En[1], we use 18.4M preprocessed data, with byte pair encoding (BPE) tokenization (Sennrich et al., 2016). We use *newstest2017* for validation and *newstest2018* for test. For WMT'14 En→De[2], following the same setting as in Vaswani et al. (2017), we use 4.5M preprocessed data that is tokenized via BPE with 32k merge operations and a shared vocabulary for English and German. We use *newstest2013* for development and *newstest2014* for test.

We train all the models with 150k steps for NIST Zh→En, 300k steps for WMT'18 Zh→En, and 300k steps for WMT'14 En→De. We select the model that performs the best on validations and report their performances on test sets. Using *multi-bleu.perl*[3], we measure case-insensitive BLEU scores and case-sensitive ones for NIST Zh→En and WMT'14 En→De, respectively. For WMT'18 Zh→En, we use the case-sensitive BLEU scores calculated by *mteval-v13a.pl*[4]. The improvements of the proposed models over the baselines are statistically significant with a reject probability smaller than 0.05 (Koehn, 2004).

For RNNSearch, the dimensions of word embeddings and hidden layers are both 600. Encoder has 3 layers and decoder has 2 layers. Dropout rate is set to 0.2. For Transformer, we follow the setting of *Transformer-Base* in Vaswani et al. (2017). Both models use beam size of 4 and the maximum number of training tokens at every step is 4096. We use Adam (Kingma and Ba, 2014) for optimization. In all the experiments, the proposed models run on NVIDIA Tesla V100 GPUs. For *Rewriter-Evaluator*, the maximum number of rewriting iterations K is 6 and termination threshold ϵ is 0.05. Hyper-parameters are obtained by grid search, except for the Transformer backbone.

[1] http://www.statmt.org/wmt18/translation-task.html.
[2] http://www.statmt.org/wmt14/translation-task.html.
[3] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl.
[4] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/mteval-v13a.pl.

Method		NIST Zh→En				
		MT03	MT04	MT05	MT06	Avg.
Deliberation Networks (Xia et al., 2017)		37.82	40.56	37.67	37.20	38.31
ABD-NMT (Zhang et al., 2018)		38.01	41.20	38.07	37.59	38.71
Adaptive Multi-pass Decoder (Geng et al., 2018)		38.39	41.43	38.54	37.86	39.05
Our Work	RNNsearch	37.20	40.42	36.75	36.29	37.67
	w/ *Rewriter-Evaluator*	**40.01**	**43.25**	**39.97**	**39.83**	**40.77**
	Transformer	46.75	47.93	47.61	46.58	47.22
	w/ *Rewriter-Evaluator*	**47.88**	**48.71**	**48.56**	**47.92**	**48.27**

Table 1: Experiment results of the proposed models and all the baselines on NIST Zh→En.

Method		WMT'14 En→De	WMT'18 Zh→En
Adaptive Multi-pass Decoder (Geng et al., 2018)		26.55	22.39
Our Work	RNNsearch	25.79	21.47
	w/ *Rewriter-Evaluator*	**27.86**	**23.71**
	Transformer	27.53	23.65
	w/ *Rewriter-Evaluator*	**28.91**	**25.08**

Table 2: Experiment results on WMT'14 En→De and WMT'18 Zh→En.

5.2 Results on NIST Chinese-to-English

We adopt the following related baselines: 1) Deliberation Networks (Xia et al., 2017) adopts a second decoder to polish the raw sequence produced by the first-pass decoder; 2) ABD-NMT (Zhang et al., 2018) uses a backward decoder to generate a translation and a forward decoder to refine it with attention mechanism; 3) Adaptive Multi-pass Decoder (Geng et al., 2018) utilizes RL to model the iterative rewriting process.

Table 1 shows the results of the proposed models and the baselines on NIST. Baseline BLEU scores are from Geng et al. (2018). There are three observations. Firstly, *Rewriter-Evaluator* significantly improves the translation quality of NMT models. The averaged BLEU score of RNNSearch is raised by 3.1% and that of Transformer is increased by 1.05%. Secondly, the proposed architecture notably outperforms prior multi-pass decoding methods. The performance of RNNSearch w/ *Rewriter-Evaluator* surpasses those of Deliberation Network by 2.46%, ABD-NMT by 2.06%, and Adaptive Multi-pass Decoder by 1.72%. Because all of these systems use the same backbone of RNN-based NMT models, these results validate that *Rewriter-Evaluator* is superior to other alternative methods. Lastly, the proposed architecture can improve Transformer backbone by 1.05% on average, and the improvements are consistently observed on tasks from MT03 to MT06.

5.3 Results on WMT Tasks

To further confirm the effectiveness of the proposed architecture, we make additional comparisons on WMT'14 En→De and WMT'18 Zh→En. The results are demonstrated in Table 2. Because the above methods don't have results on the two datasets, we re-implement Adaptive Multi-pass Decoding for comparisons.

These results are consistent with the observations in Sec. 5.2. We can see that the new architecture can improve BLEU scores on both RNNSearch and Transformer backbones. For example, the improvements on RNNSearch backbone are 2.13% on WMT'14 and 2.24% on WMT'18. On Transformer backbone, scores are raised by 1.38% on WMT'14 and 1.43% on WMT'18 . Furthermore, RNNSearch w/ *Rewriter-Evaluator* outperforms Adaptive Multi-pass Decoder by 1.31% and 1.32%, respectively, on the two tasks. Interestingly, the proposed architecture on RNNSearch backbone even surpasses Transformer on these two datasets. For example, the BLEU score on WMT'14 increases from 27.53% to 27.86%.

5.4 Oracle Experiment

We conduct oracle experiments on the test set of WMT'14 En→De to understand potential improvements of our architecture. An oracle selects the iteration that the corresponding rewrite has the highest BLEU score. Its BLEU scores are shown on the

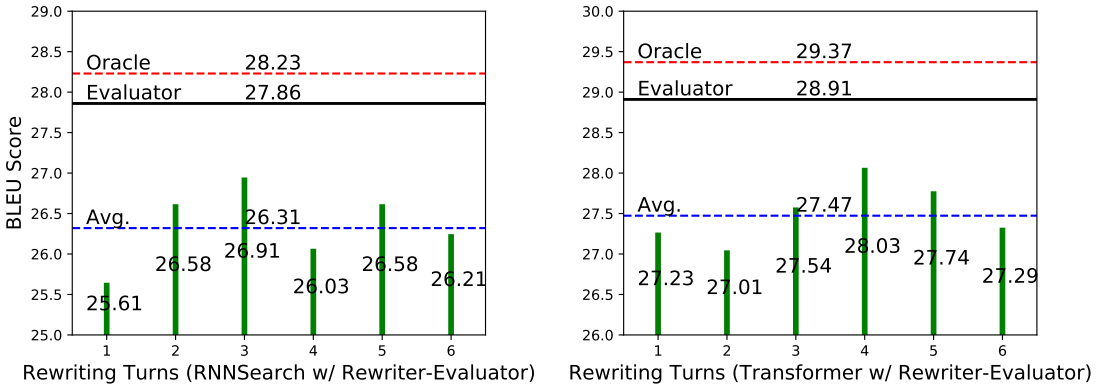

Figure 4: The oracle experiment conducted on WMT'14 En→De.

Method	NIST Zh→En	WMT'14 En→De	WMT'18 Zh→En
Adaptive Multi-pass Decoder	58.27	30.62	50.18
RNNSearch w/ *Rewriter-Evaluator*	**75.23**	71.58	**60.53**
Transformer w/ *Rewriter-Evaluator*	73.66	**72.46**	58.91

Table 3: PAT scores of different methods on NIST, WMT'14, and WMT'18.

red dashed lines in Fig. 4. The numbers on the green vertical bars are the BLEU scores of adopting a fixed number of rewriting iterations. Their averaged number is shown on the dashed blue line. BLEU score from using our evaluator is shown on the solid dark-blue line.

Results show that the evaluator, with 27.86% BLEU score and 28.91 BLEU score, is much better than the strategies of using a fixed number of rewriting turns. The gaps between oracle and the averaged performance by RNNSearch and Transformer with fixed iterations are 1.92% and 1.90%. Using the evaluator, these gaps are reduced relatively by 80.7% for RNNSearch and 75.8% for Transformer, respectively, down to 0.37% and 0.46%. These results show that the evaluator is able to learn an appropriate termination policy, approximating the performances of oracle policy.

5.5 Termination Accuracy Analysis

We define a metric, percentage of accurate terminations (PAT), to measure how precise a termination policy can be. PAT is computed as

$$\frac{1}{|U|}\sum_{(\mathbf{x},\mathbf{y})\in U}\delta(w^q(\mathbf{x},\mathbf{y}) = w^b(\mathbf{x},\mathbf{y})), \qquad (15)$$

where δ is the indicator function that outputs 1 if its argument is true and 0 otherwise. For each pair $(\mathbf{x},\mathbf{y})$ in the test set U, $w^q(\mathbf{x},\mathbf{y})$ is the turn index k with the highest quality score $\max_k q^{(k)}$ and $w^b(\mathbf{x},\mathbf{y})$ is the one with the highest BLEU score $\max_k \text{BLEU}(\mathbf{z}^{(k)},\mathbf{y})$. The translations $\mathbf{z}^{(k)}, 1 \le k \le K$ and their scores $q^{(k)}, 1 \le k \le K$ are obtained using Eq. 5 and Eq. 6.

Param. Sharing	K	NIST	WMT'14	WMT'18
✗	6	42.25	26.17	23.88
✓	2	41.83	25.64	23.26
✓	4	42.37	26.21	23.98
✓	6	42.79	**26.43**	**24.11**
✓	8	**42.83**	26.37	24.09

Table 4: Ablation studies conducted on the validation sets of NIST, WMT'14, and WMT'18.

For fair comparisons, the maximum number of rewritings is set to 6 for both Rewriter-Evaluator and Adaptive Multi-pass Decoder (Geng et al., 2018). Results in Table 3 show that PAT scores from Rewriter-Evaluator are much higher than those of Adaptive Multi-pass Decoder. For instance, RNNSearch w/ Rewriter-Evaluator surpasses Adaptive Multi-pass Decoder by 40.96% on WMT'14 and 10.35% on WMT'18.

5.6 Ablation Studies

Table 4 shows the results of ablation studies on NIST, WMT'14, and WMT'18.

Parameter Sharing. The encoders from Eq. (3) and Eq. (4) are shared between the rewriter and the evaluator. We find this improves the performances of the proposed models. For example, on NIST, sharing encoders increases our BLEU score

Method	WMT'14 En→De	
	Training	Test
RNNSearch	7h56m	11m26s
w/ *Rewriter-Evaluator*	9h17m	39m50s
Transformer	5h23m	14m11s
w/ *Rewriter-Evaluator*	6h36m	52m02s

Table 5: Running time comparisons on WMT'14.

from 42.25% to 42.79% with the same maximum iteration number of K.

Maximum Number of Iterations. Increasing the maximum number of turns K generally improves the BLEU scores. For instance, on NIST, $K = 8$ outperforms $K = 2$ by 1.0%, $K = 4$ by 0.46%, and $K = 6$ by 0.04%. However, described in Sec. 5.7, large K (e.g., 8) can increase inference time cost. Moreover, additional gains in performance from $K = 8$ is small. We therefore set $K = 6$ by default.

5.7 Running Time Comparisons

While achieving improved translation quality, the models are trained with multiple passes of translation. Therefore, a natural question is on the increase of training time and test time. We report results on 4 GPUs with the maximum rewriting turns $K = 6$ and the beam size set to 8. Results on WMT'14 are listed in Table 5.

It shows that *Rewriter-Evaluator* increases the test time by approximately 4 times, because of multiple passes of decoding. However, training time is only relatively increased by 15% and 18%, respectively on RNNSearch and Transformer, due to the large priority queue used in PGD to store previous translation cases.

6 Related Work

Multi-pass decoding has been well studied in statistical machine translation (Brown et al., 1993; Koehn et al., 2003, 2007; Och and Ney, 2004; Chiang, 2005; Dyer et al., 2013). Och (2003); Och and Ney (2002) propose training models with minimum error rate criterion on lattices from first-pass decoder. Marie and Max (2015) introduce an iterative method to refine search space generated from simple feature with additional information from more complex feature. Shen et al. (2004) investigate reranking of hypothesis using neural models trained with discriminative criterion. Neubig et al. (2015) propose to reconfirm effectiveness of reranking. Chen et al. (2008) present a regeneration of search space from techniques such as n-gram expansion. These approaches are however applied to shallow models such as log-linear models (Och and Ney, 2002).

Our work is closely related to recent efforts in multi-pass decoding on NMT. In these recent works (Xia et al., 2017; Zhang et al., 2018; Geng et al., 2018), the models generate multiple target sentences for a source sentence and, except for the first one, each of them is based on the sentence generated in the previous turn. For example, Xia et al. (2017) propose Deliberation Networks that uses a second decoder to polish the raw sequence produced by the first-pass decoder. While these methods have achieved promising results, they lack a proper termination policy for the multi-pass translation process. Zhang et al. (2018) adopt a predefined number of decoding passes, which is not flexible. Geng et al. (2018) incorporate post-editing mechanism into NMT model via RL. However, RL can be unstable for training because of the high variance in gradient estimation. The lack of a proper termination policy results in premature terminations or over-translated sentences, which can largely limit the performance gains of these methods.

7 Conclusion

This paper has introduced a novel architecture, *Rewriter-Evaluator*, that achieves a proper termination policy for multi-pass decoding in NMT. At every translation pass, given the source sentence and its past translation, a rewriter generates a new translation, aiming at making further performance improvements over the past translations. An evaluator estimates the translation quality to determine whether to complete this iterative rewriting process. We also propose PGD that facilitates training the rewriter and the evaluator both jointly and efficiently. We have applied *Rewriter-Evaluator* to improve mainstream NMT models. Extensive experiments have been conducted on three translation tasks, NIST Zh→En, WMT'18 Zh→En, and WMT'14 En→De, showing that our architecture notably improves the results of NMT models and significantly outperforms other related methods. An oracle experiment and a termination accuracy analysis show that the performance gains can be attributed to the improvements in completing the rewriting process at proper iterations.

Acknowledgments

This work was done when the first author did internship at Ant Group. We thank anonymous reviewers for their valuable suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representation*.

Léon Bottou and Olivier Bousquet. 2008. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. Curran Associates, Inc.

Justin A Boyan and Andrew W Moore. 1995. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in neural information processing systems*, pages 369–376.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Boxing Chen, Min Zhang, Aiti Aw, and Haizhou Li. 2008. Regenerating hypotheses for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 105–112, Manchester, UK. Coling 2008 Organizing Committee.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Jonas Gehring, 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 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Xinwei Geng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2018. Adaptive multi-pass decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 523–532.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

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

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Benjamin Marie and Aurélien Max. 2015. Multi-pass decoding with complex feature guidance for statistical machine translation. In *Proceedings of the*

53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 554–559, Beijing, China. Association for Computational Linguistics.

Graham Neubig, Makoto Morishita, and Satoshi Nakamura. 2015. Neural reranking improves subjective quality of machine translation: NAIST at WAT2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 35–41, Kyoto, Japan. Workshop on Asian Translation.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2016. Pre-translation for neural machine translation. *arXiv preprint arXiv:1610.05243*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

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.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 177–184, Boston, Massachusetts, USA. Association for Computational Linguistics.

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.

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 5998–6008.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4144–4150.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Advances in Neural Information Processing Systems*, pages 1784–1794.

Hao Zhang and Daniel Gildea. 2008. Efficient multi-pass decoding for synchronous context free grammars. In *Proceedings of ACL-08: HLT*, pages 209–217, Columbus, Ohio. Association for Computational Linguistics.

Wen Zhang, 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, Italy. Association for Computational Linguistics.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.