Learning to Rewrite for Non-Autoregressive Neural Machine Translation

Xinwei Geng[†] **Xiaocheng Feng**[†‡] **Bing Qin**[†‡]
[†] Harbin Institute of Technology [‡] Peng Cheng Laboratory
{xwgeng,xcfeng,qinb}@ir.hit.edu.cn

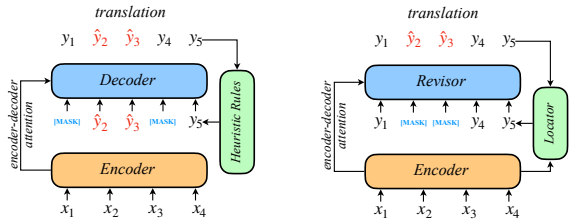
Abstract

Non-autoregressive neural machine translation, which decomposes the dependence on previous target tokens from the inputs of the decoder, has achieved impressive inference speedup but at the cost of inferior accuracy. Previous works employ iterative decoding to improve the translation by applying multiple refinement iterations. However, a serious drawback is that these approaches expose the serious weakness in recognizing the erroneous translation pieces. In this paper, we propose an architecture named REWRITENAT to explicitly learn to rewrite the erroneous translation pieces. Specifically, REWRITENAT utilizes a *locator* module to locate the erroneous ones, which are then revised into the correct ones by a *revisor* module. Towards keeping the consistency of data distribution with iterative decoding, an iterative training strategy is employed to further improve the capacity of rewriting. Extensive experiments conducted on several widely-used benchmarks show that REWRITENAT can achieve better performance while significantly reducing decoding time, compared with previous iterative decoding strategies. In particular, REWRITENAT can obtain competitive results with autoregressive translation on WMT14 En↔De, En→Fr and WMT16 Ro→En translation benchmarks[1].

1 Introduction

State-of-the-art neural machine translation (NMT) systems use autoregressive decoding where the decoder generates a target sentence word by word, and the generation of the latter words depends on previously generated ones (Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017). Instead of sequential decoding as in the autoregressive translation (AT), non-autoregressive neural machine translation (NAT) (Gu et al., 2018; Guo et al., 2019; Ma et al., 2019; Wei et al., 2019; Sun et al., 2019; Ghazvininejad et al., 2020a; Zhou et al., 2020; Ding et al., 2021a,b) generates the whole target sentence simultaneously. To enable parallel decoding, NAT imposes a conditional independence assumption among words in target sentences, yielding significantly faster inference speed than AT. However, since intrinsic dependencies within target sentence are omitted, NAT suffers from severe inconsistency problem (Wang et al., 2019), leading to inferior translation quality, especially when capturing highly multimodal distribution of target translations (Gu et al., 2018).

Towards tackling above fundamental problem, iterative decoding (Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019; Guo et al., 2020b; Ghazvininejad et al., 2020b) is proposed to improve NAT by repeatedly refining previously generated translation. Instead of enforcing NAT to generate accurate translation by one-pass decoding, these approaches are expected to revise incorrect translation pieces through several refinements (Xia et al., 2017; Zhang et al., 2018; Geng et al., 2018). With the introduction of iterative decoding, NAT further



(a) Masked LM-based NAT (b) RewriteNAT

Figure 1: Illustration of the difference in masking words between (a) conventional masked LM-based NAT (Ghazvininejad et al., 2019) and (b) our proposed REWRITENAT. Instead of using inefficient *heuristic rules* which perhaps mask correct words in some case (*e.g.,* $y_1 y_4$), REWRITENAT utilizes an additional *locator* module to learn to explicitly distinguish erroneous translation pieces (*e.g.,* $\hat{y}_2 \hat{y}_3$), annotated as special symbol (*i.e.,* [MASK]).

[1]Our code is publicly available at https://github.com/xwgeng/RewriteNAT.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3297–3308
November 7–11, 2021.

boosts translation quality, bridging performance gap between NAT and AT models.

However, existing iterative NAT models expose the weakness in distinguishing the erroneous words. The dominant approach to identify the mistakes is mask-predict algorithm (Ghazvininejad et al., 2019; Guo et al., 2020b), which employs inefficient heuristic rules to roughly choose the least confident words as the erroneous. In some case, mask-predict may mistake to rewrite correct words while maintain erroneous ones, acting as noises to make a negative impact on subsequent iterations. Without explicitly classifying translated words into wrong or right, the translations decode in constant number of iterations, hindering the further improvement of inference speed. Besides, decoder inputs of prevailing iterative NAT models (Kasai et al., 2020; Guo et al., 2020b) almost come from the ground-truth during training, while target sentences generated at different refinement steps are taken as decoder inputs in inference, creating a discrepancy that can hurt performance.

In this paper, we propose an architecture named REWRITENAT, which explicitly learns to rewrite erroneous translation pieces. Specifically, we introduce a locator module to locate incorrect words within previously generated translation. The located words will be masked out and revised by the revisor module in subsequent refinement. We frame learning to rewrite, comprised of two steps: *locate* and *revise*, as an iterative training procedure, where *locate* and *revise* operations are supervised by comparing the generated translation with the ground-truth. Towards keeping the consistency with iterative decoding, iterative training is utilized to further improve the training procedure. Experimental results on several typical machine translation datasets demonstrate that REWRITENAT achieves consistent improvement over iterative decoding baselines, but with substantially less decoding time. Further analysis show that REWRITENAT prefers to generate the "*easy*" words at the early decoding iteration, and leaves the more complicated choice later.

2 Background

2.1 Autoregressive Machine Translation

Autoregressive neural machine translation (AT) draws much attention due to its convenience and effectiveness on various machine translation tasks (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015). Given a source sentence $X = \{x_1, \cdots, x_T\}$ and target sentence $Y = \{y_1, \cdots, y_{T'}\}$, AT decomposes translation distribution $p_{\text{AT}}(X|Y)$ into a chain of conditional probabilities in a unidirectional manner:

$$p_{\text{AT}}(X|Y) = \prod_{t=1}^{T'} p(y_t|y_{<t}, X) \quad (1)$$

where $y_{<t}$ represents the set of generated tokens before time-step t. Besides, T and T' is the length of the source and the target sequence, respectively. Sine AT generates translation in an autoregressive manner, it suffers from low inference speed.

2.2 Non-Autoregressive Machine Translation

Towards alleviating this issue, NAT (Gu et al., 2018) removes sequential dependencies within target sentence, and generates target words, simultaneously. NAT models conditional probabilities $p_{\text{NAT}}(Y|X)$ of translation from X to Y as a product of conditionally independent per-step distributions:

$$p_{\text{NAT}}(Y|X) = p(T'|X) \prod_{t=1}^{T'} p(y_t|X) \quad (2)$$

Since each target word y_t only depends on the source sentence X, the target distributions $p(y_t|X)$ can be computed in parallel at inference time.

Nevertheless, this desirable property of parallel decoding comes at the cost that the translation quality is largely sacrificed. Since the intrinsic dependencies within target sentence (y_t depends $y_{<t}$) are abandoned from decoder input, NAT shows its weakness in exploiting inherent sentence structure for prediction. Hence, NAT has to figure out such target-side information by itself, merely conditioned on source-side information. In contrast, AT produces current target word, conditioned on previously generated words, which provides strong target side context information. Consequently, with less and weaker information, NAT suffers from inferior translation quality.

3 Architecture

As depicted in Figure 2, our proposed REWRITENAT literally consists of three major components: an encoder, a *revisor* and a *locator*. The encoder utilizes transformer encoder, comprisedof N^e transformer blocks (Vaswani et al., 2017), to convert source sentence into the contextual representations, similar to previous work (Gu et al., 2018). The

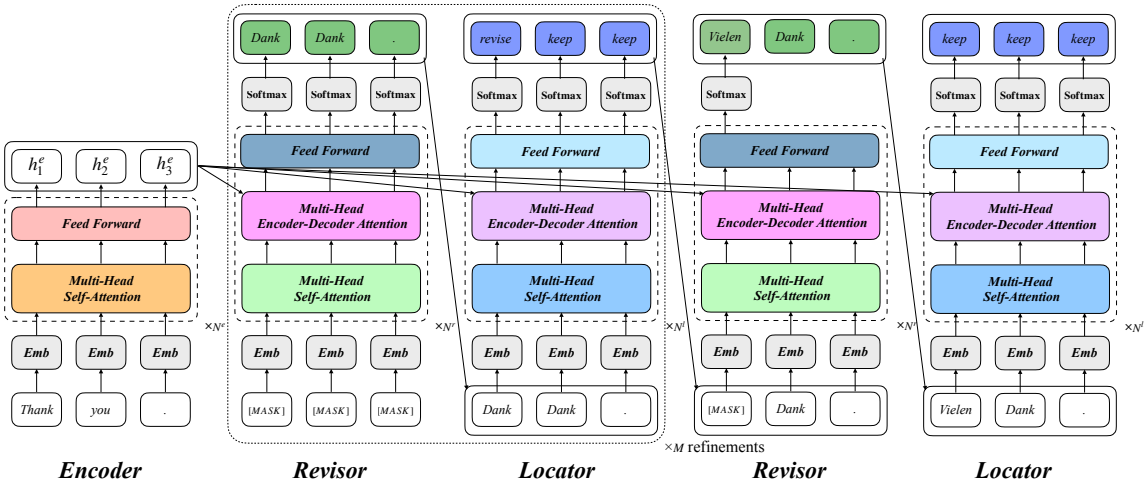


Figure 2: Architecture of our proposed REWRITENAT model, which consists of three major components: an *encoder*, a *revisor* and a *locator*. The encoder is utilized to convert the source sentence into contextual representations. During the decoding, the revisor converts the erroneous words annotated as "`[MASK]`" into the correct ones, while the incorrect words within previously generated hypothesis are distinguished, by classifying the words into two classes: *revise* and *keep*. Given previously located hypothesis, M refinements, each of which utilizes a revisor and a following locator refine the hypothesis, are applied to obtain the final translation. We take an instance from English→German translation as example, where source sentence is "*Thank you .*". REWRITENAT applies two refinements into the initial hypothesis, merely comprised of "`[MASK]`". Subsequently, the decoding terminates since the locator categorizes the entire sequence into *keep*, meaning that any word is not required to be revised.

revisor and locator, composing into an decoder, are employed to revise and locate the incorrect words within previously generated translation, respectively. We will elaborate the revisor and locator in the following.

3.1 Revisor

Given altered translation Y^r by the locator, the revisor is utilized to convert erroneous pieces into the correct, conditioned on source sentence. Particularly, it's expected to speculate about correct words in positions annotated as "`[MASK]`", under the context of the remaining translation. Notably, the revisor treats an input merely consisting of "`[MASK]`" as initial input, meaning that the whole input is required to be revised.

Given the hypothesis $Y^r = \{y_1^r, \cdots, y_{T'}^r\}$, we leverage a stack of transformer blocks (Vaswani et al., 2017; Gu et al., 2018) to generate the corresponding representations $H^r = \{H_1^r, \cdots, H_{T'}^r\}$, with the glimpse at source representations H^e:

$$H^r = \textit{TransformerStack}^r(Y^r, H^e) \quad (3)$$

where $\textit{TransformerStack}^r(\cdot)$ represents the stack of N^r transformer blocks with respect to the revisor. Subsequently, the generated representations H^r with respect to special symbol "`[MASK]`" are fed to a classifier π^r to generate the target words as follows:

$$\pi^r(r_t|y_t^r, X) = \textit{softmax}(W^r h_t^r + b^r) \quad (4)$$

where W^r and b^r are trainable parameters, and represent weight matrix and bias vector, respectively. The generated words by π^r are treated as the substitute of the incorrect words annotated as "`[MASK]`", yield the revised translation $Y^l = \{y_1^l, \cdots, y_{T'}^l\}$ as follows:

$$y_t^l = \begin{cases} r_t, & \text{if } y_t^r = \texttt{[MASK]} \\ y_t^r, & \text{otherwise} \end{cases} \quad (5)$$

where Y^l is fed to the locator.

3.2 Locator

Given previously generated translation as input, we employ the locator to distinguish incorrect words within entire sequence, conditioned on source sentence. Using the locator, each word within translation can be categorized into two types: *revise* (1) and *keep* (0). According to resulted classification, it is required to alter previous translation into another format, which is then fed to the revisor. In details, the words annotated as "*revise*" are substituted by special symbol, denoted as "`[MASK]`", while the remaining hold.

Given previously generated translation $Y^l = \{y_1^l, \cdots, y_{T'}^l\}$ to be located, a stack of transformer

blocks (Vaswani et al., 2017; Gu et al., 2018) are utilized to transform input translation Y^l into a sequence of hidden states $H^l = \{h_1^l, \cdots, h_{T'}^l\}$, conditioned on source contextual representations H^e:

$$H^l = \textit{TransformerStack}^l(Y^l, H^e) \quad (6)$$

where $\textit{TransformerStack}^l(\cdot)$ represents the stack of N^l transformer blocks with respect to the locator. Using induced hidden states H^l as input, an additional classifier π^l is employed to decide whether previously generated word y_t^l at step t is required to be revised, and calculated as follows:

$$\pi^l(l_t|y_t^l, X) = \textit{softmax}(W^l h_t^l + b^l) \quad (7)$$

where W^l and b^l are trainable parameters, and represent weight matrix and bias vector, respectively. Using the classifier π^l, input translation Y^l can be converted into an annotation sequence $L = \{l_1, \cdots, l_{T'}\}$. Subsequently, dependent on the annotation L, the translation Y^l is altered into $Y^r = \{y_1^r, \cdots, y_{T'}^r\}$ as follows:

$$y_t^r = \begin{cases} \texttt{[MASK]}, & \text{if } l_t = \textit{revise} \\ y_t^l, & \text{otherwise} \end{cases} \quad (8)$$

where Y^r is treated as input of the revisor.

4 Training and Inference

4.1 Training

Towards maintaining the consistency of data distribution with iterative decoding at inference time, iterative training strategy is utilized to train REWRITENAT to learn the ability to rewriting, as described in Algorithm 1. During training, at m-th refinement including *revise* and *locate* operations, we compare previously-generated translations (*i.e.,* $\hat{Y}_m^r$ and $\hat{Y}_m^l$) with ground-truth (*i.e.,* Y) to distinguish erroneous translation pieces, and construct two types of supervised signals (*i.e.,* $q(\hat{Y}_m^r)$ and $z(\hat{Y}_m^l)$) to instruct the learning of *revisor* and *locator* modules, respectively. With the introduction of iterative training with M refinements, training objective $\mathcal{L}(\theta)$ can be formalized as:

$$\mathcal{L}(\theta) = \sum_{m=1}^{M} \Big\{ \underbrace{q(\hat{Y}_m^r) \log \pi_\theta^r(Y|\hat{Y}_m^r, X)}_{\textit{revisor objective}} + \underbrace{\log \pi_\theta^l(z(\hat{Y}_m^l)|\hat{Y}_m^l, X)}_{\textit{locator objective}} \Big\} \quad (9)$$

Algorithm 1 Iterative Training to REWRITENAT

Input: Parallel training dataset $(\mathcal{X}, \mathcal{Y})$, revisor module π_θ^r, locator module π_θ^l, maximum refinement steps M, learning rate γ
repeat
  Sample sentence pair (X, Y) from $(\mathcal{X}, \mathcal{Y})$
  Initialize $\hat{Y}_1^r$ as a sequence of `[MASK]` with same length as Y
  for $m \leftarrow 1$ to M **do**
    Generate $\hat{Y}_m^l$ using $\pi^r(\cdot|\hat{Y}_m^r, X)$ as Eq. 5
    $\mathcal{L}_m^r \leftarrow q(\hat{Y}_m^r) \log \pi_\theta^r(Y|\hat{Y}_m^r, X)$
    Generate $\hat{Y}_m^r$ using $\pi^l(\cdot|\hat{Y}_m^l, X)$ as Eq. 8
    $\mathcal{L}_m^l \leftarrow \log \pi_\theta^l(z(\hat{Y}_m^l)|\hat{Y}_m^l, X)$
  end for
  $\mathcal{L} \leftarrow \sum_{m=1}^{M}(\mathcal{L}_m^r + \mathcal{L}_m^l)$
  Update model parameters $\theta \leftarrow \theta + \gamma \nabla_\theta \mathcal{L}$
until convergence

where the translations $\hat{Y}_m^r$ and $\hat{Y}_m^l$ are generated at m-th refinement step depending on output distributions of the revisor $\pi^l(\cdot|\hat{Y}_{m-1}^l, X)$ and locator $\pi^r(\cdot|\hat{Y}_m^r, X)$, respectively. During training, generated translation $\hat{Y}_m^r$ and $\hat{Y}_m^l$ have same length with the ground-truth Y. When calculating revisor objective, we use $q(\hat{Y}^r)$ as a weight vector to merely concentrate on optimizing at the incorrect words (annotated as `[MASK]` in $\hat{Y}^r$) but omit the losses with respect to correctly-generated ones:

$$q_t(\hat{Y}^r) = \begin{cases} 1, & \text{if } \hat{Y}_t^r = \texttt{[MASK]} \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

The locator target $z(\hat{Y}^l)$ is a vector meaning that the positions where translation $\hat{Y}^l$ is different from ground-truth Y should be categorized into *revise* (1), while the remaining are mapped into *keep* (0):

$$z_t(\hat{Y}^l) = \begin{cases} 1, & \text{if } \hat{Y}_t^l \neq Y_t \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

4.2 Inference

During training REWRITENAT generates the translations with same length as the ground-truth, while in inference we apply REWRITENAT over a sequence of "`[MASK]`" with a length predicted by length classifier (Lee et al., 2018). When locator module classifies entire sentence into *keep* or the classifications of two consecutive refinements keep the same, decoding stops (a.k.a *dynamic halting*).

Model	En→De		De→En		En→Ro		Ro→En	
	Iters.	BLEU	Iters.	BLEU	Iters.	BLEU	Iters.	BLEU
TRANSFORMER (Vaswani et al., 2017)	N	27.82	N	31.66	N	34.44	N	34.14
NAT w/ Fertility (Gu et al., 2018)	1	19.17	1	23.20	1	29.79	1	31.44
CTC (Libovický and Helcl, 2018)	1	17.68	1	19.80	1	19.93	1	24.71
NAT-REG (Wang et al., 2019)	1	24.61	1	28.90	–	–	–	–
Imitate-NAT (Wei et al., 2019)	1	24.15	1	27.28	1	31.45	1	31.81
Flowseq (Ma et al., 2019)	1	25.31	1	30.68	1	32.20	1	32.84
Hint-NAT (Li et al., 2019)	1	25.20	1	29.52	–	–	–	–
NAT-DCRF (Sun et al., 2019)	1	26.80	1	30.04	–	–	–	–
Bag-of-ngram (Shao et al., 2020)	1	20.90	1	24.61	1	28.31	1	29.29
FCL-NAT (Guo et al., 2020a)	1	25.75	1	29.50	–	–	–	–
TCL-NAT (Liu et al., 2020)	1	25.37	1	29.60	–	–	–	–
EM+ODD (Sun and Yang, 2020)	1	24.54	1	27.93	–	–	–	–
AXE (Ghazvininejad et al., 2020a)	1	23.53	1	27.90	1	30.75	1	31.54
iNAT (Lee et al., 2018)	10	21.61	10	25.48	10	29.32	10	30.19
InsT (Stern et al., 2019)	$\log_2 N$	27.41	–	–	–	–	–	–
CMLM (Ghazvininejad et al., 2019)	4	25.94	4	29.90	4	32.53	4	33.23
	10	27.03	10	30.53	10	33.08	10	33.31
LevT (Gu et al., 2019)	2.05	27.27	–	–	–	–	2.03	33.26
LaNMT (Shu et al., 2020)	4	26.30	–	–	–	–	4	29.10
SMART (Ghazvininejad et al., 2020b)	4	27.03	4	30.87	–	–	–	–
	10	27.65	10	31.27	–	–	–	–
JM-NAT (Guo et al., 2020b)	4	27.05	4	31.51	4	32.97	4	33.21
	10	27.69	10	**32.24**	10	33.52	10	33.72
DisCo (Kasai et al., 2020)	4.84	27.34	4.23	31.31	3.29	33.22	3.10	33.25
OUR PROPOSED REWRITENAT	2.70	**27.83**	**2.42**	31.52	**2.41**	**33.63**	**1.76**	**34.09**

Table 1: Evaluation of translation performance on WMT14 En↔De and WMT16 En↔Ro datasets. TRANSFORMER is a strong autoregressive baseline, which is treated as teacher model to distill the datasets. In addition to purely non-autoregressive baselines (*e.g.,* NAT-DCRF and AXE), we utilizes several typical iterative non-autoregressive models (*e.g.,* CMLM and DISCO) as comparisons. "*Iters.*" indicates the number of refinement steps taken during decoding, averaged over WMT14 En→De test set.

Besides, we further set a maximum number of iterations to guarantee constant-time complexity in worst case.

5 Experiments

5.1 Settings

Datasets We evaluate REWRITENAT on two standard machine translation datasets: WMT14 En↔De (4.5M pairs) and En→Fr (36M pairs)[2], WMT16 En↔Ro[3] (610K pairs) and WMT17 En→Zh[4] (20M pairs). For WMT14 En↔De translation, we use script from `fairseq` (Ott et al., 2019) to preprocess dataset, where the preprocessing steps follow Vaswani et al. (2017). The *newstest2013* and *newstest2014* are treated as development and test sets, respectively. For WMT14 En→Fr, we borrow the setup of Gehring et al. (2017), validate on *newstest2012+2013* and test on *newstest2014*. For WMT16 En↔Ro translation, we use the dataset released by Lee et al. (2018), where *newsdev2016* and *newstest2016* are taken as development and test sets. For WMT17 En→Zh translation, we pre-process the dataset following Hassan et al. (2018). We treat *newsdev2017* as the development set and *newstest2017* as the test set. The datasets are tokenized into subword units using BPE (Sennrich et al., 2016). We evaluate performance with BLEU (Papineni et al., 2002) for all language pairs, except for En→Zh, where we use SacreBLEU (Post, 2018) [5].

Distillation Knowledge distillation (Kim and Rush, 2016; Hinton et al., 2015) is utilized to train the NAT models due to its effectiveness of alleviating multimodality (Gu et al., 2018) using the generated translation by TRANSFORMER (TRANSFORMER-BIG for WMT14 En↔De and En→Fr as well as WMT17 En→Zh, TRANSFORMER-BASE for WMT16 En↔Ro) as a substitute for target-side ground-truth (Ghazvininejad et al., 2019).

Hyperparameters We follow most of the standard hyperparameters for TRANSFORMER-

[2] https://www.statmt.org/wmt14
[3] https://www.statmt.org/wmt16
[4] https://www.statmt.org/wmt17

[5] SacreBLEU hash: BLEU+case.mixed+lang.en-zh +numrefs.1+smooth.exp+test.wmt17+tok.zh+version.1.4.14

BASE (Vaswani et al., 2017): 6 layers per stack, 8 attention heads per layer, 512 model dimensions, 2048 hidden dimensions. We follow the weight initialization schema from BERT (Devlin et al., 2019), and sample weights from $\mathcal{N}(0, 0.02)$, set biases to zero, and set layer normalization parameters to $\beta = 0$ and $\gamma = 1$. For regularization, we use dropout (En↔De and En↔Ro: 0.3, En→Fr: 0.1, En→Zh: 0.25), 0.01 L_2 weight decay, and smoothed cross validation loss with $\lambda = 0.1$. we adopt the Adam optimizer (Kingma and Ba, 2015) using $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1e^{-8}$. The learning rate is scheduled using `inverse_sqrt` with a maximum learning rate 0.0005, and 10,000 warmup steps except for TRANSFORMER which sets warmup steps as 4000. All the models are run on 8 Tesla V100 GPUs for 300,000 updates with an effective batch size of 128,000 tokens apart from En→Fr where we make 500,000 updates to account for the data size. During decoding, we use a beam size of $b = 5$ for autoregressive decoding, while length beam (Ghazvininejad et al., 2019) is applied to obtain the translation with respect to non-autoregressive counterpart.

5.2 Main Results

Table 1 shows the results of REWRITENAT, together with a series of purely NAT baselines and representative iterative NAT approaches. Our proposed REWRITENAT can achieve consistent and remarkable improvements over both pure and iterative NAT baseines across different translation tasks despite significantly fewer iterations on average (*e.g.,* 2.7 iterations in En→De and 1.76 iterations in Ro→En), except for De→En (*i.e.,* 31.52 vs. 32.24). The possible reason is that JM-NAT utilizes a more powerful TRANSFORMER-BASE architecture (32.69 vs. 31.66) than the one we use. Despite inferior performance, when achieving same performance (*e.g.,* 31.52 vs. 31.51), REWRITENAT takes 2.42 iterations in average, less than JM-NAT. Particularly noteworthy is that REWRITENAT obtain competitive performance with autoregressive baseline on En→De (*i.e.,* 27.83 vs. 27.82), De→En (*i.e.,* 31.52 vs. 31.66) and Ro→En (*i.e.,* 34.09 vs. 34.14) translation, demonstrating its effectiveness.

Instead of using constant number of iterations (*e.g.,* 4 or 10) for iterative NAT baselines, important advantage of REWRITENAT is dynamic halting (Section 4.2) which can choose suitable iterations with respect to different sentences, leading to high inference speed. As shown in Table 2, in most cases (*i.e.,* 89.4%), REWRITENAT can produce final translations within 4 iterations. Furthermore, it's surprising that one-shot decoding (using single iteration) makes up 20.6% of test set.

Iters.	**1**	**2**	**3**	**4**	**≥ 5**
Per. (%)	20.6	31.7	24.5	12.6	10.6

Table 2: Percentage of different iterations taken during decoding with respect to REWRITENAT on En→De test set.

Towards verifying the robustness on large-scale datasets, we evaluate the translation performance of REWRITENAT on En→Fr and En→Zh. As shown in Table 3, our proposed REWRITENAT obtains consistent improvement in both translation quality and speed compared with dominant CMLM baseline. It's particularly important that our approach can achieve competitive results (*i.e.,* 41.36 vs. 41.59) with autoregressive baseline on En→Fr despite the average of 2.11 iterations.

Model	**En→Zh**		**En→Fr**	
	Iters.	**BLEU**	**Iters.**	**BLEU**
TRANSFORMER	N	35.44	N	41.59
CMLM	4	33.18	4	39.94
	10	33.80	10	40.53
REWRITENAT	**3.06**	**34.32**	**2.11**	**41.36**

Table 3: Average number of iterations ("*Iters.*") and performance ("*BLEU*") with repsect to REWRITENAT on large-scale WMT17 En→Zh and WMT14 En→Fr datasets.

5.3 Decoding Speed

As shown above, REWRITENAT can obtain substantial improvements than strong iterative NAT baselines while reducing the number of iterations. Here we compare them in terms of speedup with respect to TRANSFORMER, as depicted in Figure 3. It can be clearly observed that REWRITENAT can obtain same performance but with substantially higher speedup than iterative NAT baselines. When maximum iteration is set as 2 (*i.e.,* $T = 2$), REWRITENAT obtains competitive result to CMLM and TRANSFORMER with $b = 1$ (*i.e.,* 27.03 vs. 27.05) but with higher speedup (*i.e.,* 7.02×). The performance of REWRITENAT benefits much from the growth of T until $T = 4$. Particularly, REWRITENAT with $T = 4$ achieves comparable result

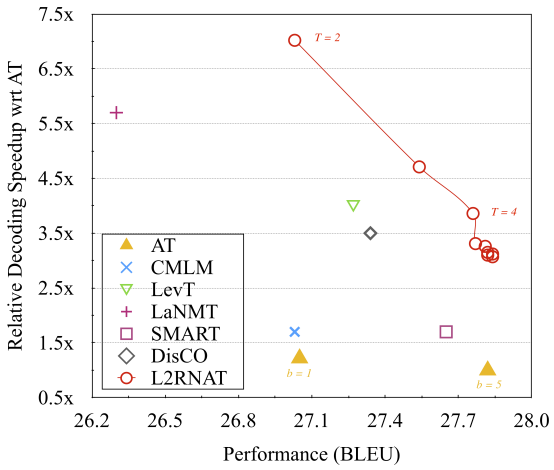


Figure 3: Relative decoding speedup on WMT14 En→De test data with respect to autoregressive model (indicated as ▲) with beam sizes $b = 1$ and $b = 5$. T denotes a (max) number of iterations taken during decoding ($T = 2, \ldots, 10$). "*Relative Decoding Speedup*" is calculated over TRANSFORMER with beam size as 5.

(27.77 vs. 27.82, 3.86×) with TRANSFORMER with $b = 5$. Furthermore, REWRITENAT with $T = 4$ outperforms the strongest SMART (*i.e.,* 27.56 vs. 27.77) but using about half of decoding time. Afterwards, performance gain is relatively subtle but with a slight decrease of speedup due to dynamic halting.

6 Analysis

6.1 Word Repetitions

With decoupling the sequential dependencies among target sentence, NAT shows the serious weakness in modeling highly multimodal distributions (Gu et al., 2018), often manifest as *word repetitions* (Wang et al., 2019) in generated translations. Towards evaluating the multi-modality, we follow Ghazvininejad et al. (2019) to measure the percentage of consecutive repetitive words as a proxy metric. As shown in Table 4, the proportion of repetitive words with respect to REWRITENAT is significantly lower than most relevant CMLM baseline, especially when decoding using single iteration (-6.05%). Simultaneously, REWRITENAT can achieve substantial performance over CMLM. These results demonstrate the superiority of REWRITENAT over CMLM in alleviating word repetitions.

Iters.	CMLM		REWRITENAT	
	BLEU	**Reps**	**BLEU**	**Reps**
$T = 1$	18.05	16.72%	21.17	10.67%
$T = 2$	22.91	5.40%	27.03	1.95%
$T = 3$	24.99	2.03%	27.54	0.97%
$T = 4$	25.94	1.07%	27.76	0.59%

Table 4: The performance ("*BLEU*") and percentage of repetitive words ("*Reps*") when decoding with a different number of iterations on WMT14 En→De test set. Notably, with respect to REWRITENAT, T denotes the max number of iterations taken during decoding.

6.2 Effect of Weight Sharing

Towards evaluating the effectiveness of weight sharing between revisor and locator modules, we conduct some experiments to make the further analysis. As shown in Table 5, the performance of REWRITENAT using sharing parameters (*i.e.,* + w/ *sharing*) shows a slight decrease (*i.e.,* 27.54 vs. 27.83) on WMT14 En→De translation task, but still surpasses the most relevant baseline (*i.e.,* CMLM). Besides, it's observed that the proposed REWRITENAT with weight sharing can consumes less iterations taken during decoding, leading to a slightly high inference speed.

Model	Iters.	BLEU	△
CMLM	10	27.03	–
REWRITENAT	2.7	27.83	+ 0.80
+ w/ *sharing*	2.5	27.54	+ 0.51

Table 5: Average number of iterations ("*Iters.*") and performance ("*BLEU*") when sharing weights of *revisor* with *locator* on WMT14 En→De test set.

6.3 Iterations vs. Length

As described above, compared with previous iterative NATs, the number of iterations taken during decoding significantly decreases with respect to our proposed REWRITENAT. Towards exploring the impact of length, we compare the number of required iterations and the length of target sentences, as illustrated in Figure 4. It's clearly observed that REWRITENAT can properly choose the number of iterations accordingly. In general, as the length of target sentences grows, REWRITENAT also requires more iterations to produce the translation.

6.4 Analysis on Part-of-Speech

Despite proving the effectiveness, we doubt whether the number of iterations has any prefer-

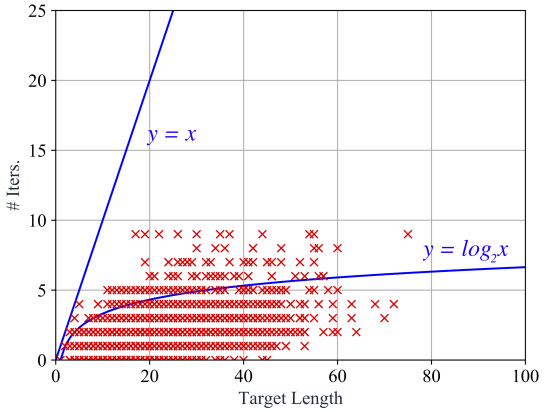


Figure 4: # Iters. vs. length of target sentences with respect to the different instances (denoted as ×) on WMT14 En→De test set.

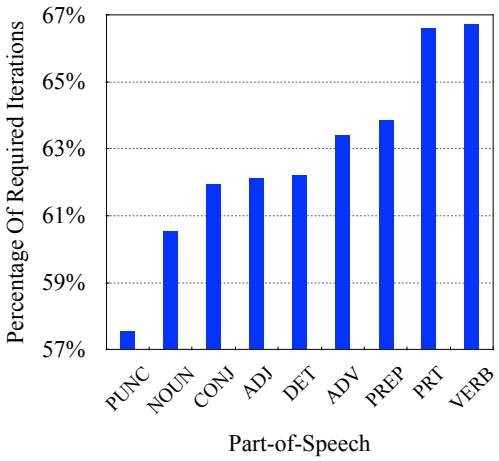


Figure 5: Average percentage of required iterations to generate different Part-of-Speechs of total iterations on De→En test set.

ence towards different Part-of-Speechs [6]. For each Part-of-Speech [7], we calculate average percentage of required iterations to produce the words with respect to different Part-of-Speechs. As shown in Figure 5, REWRITENAT tends to generate punctuation words (*i.e.,* PUNC) early in decoding. Subsequently, nouns are next easiest to predict. Conditioned on generated nouns, other Part-of-Speechs (*e.g.,* CONJ, ADJ, DET, ADV, PREP), which often act as modifiers, prefers to come out in the generated translation. Finally, the most difficult for REWRITENAT is to generate verbs (*i.e.,* VERB) and particles (*i.e.,* PRT). These observations are consistent with easy-first generation hypothesis: early decoding iterations mostly generate words which are the easiest to predict based on input data (Emelianenko et al., 2019).

6.5 Case Study

As illustrated in Figure 6, we present a translation example to compare REWRITENAT with CMLM. The number of maximum decoding iterations is set as 10. We can observe that REWRITENAT can generate the reasonable translation with 3 decoding iterations and terminate the decoding due to the locator module, automatically. In addition, the erroneous translation pieces (*e.g.,* "*are children children*") can be accurately distinguished. In contrast, strong CMLM baseline shows their weakness at tackling the incorrect ones. Consequently, CMLM generally spend more decoding iterations than REWRITENAT, but achieving inferior performance. These results confirm the effectiveness and efficiency of the proposed REWRITENAT.

7 Related Work

Gu et al. (2018) first proposed NAT to generate the translation in parallel, boosting the inference speed. Towards mitigating the performance degradation, a series of works were proposed to strengthen the capacity of capturing the dependencies among output words, including adding a lite autoregressive module (Kaiser et al., 2018; Wang et al., 2018; Sun et al., 2019), training with well-designed objectives (Guo et al., 2019; Libovický and Helcl, 2018; Shao et al., 2020; Ghazvininejad et al., 2020a; Du et al., 2021), modeling with latent varibles (Ma et al., 2019) and mimicking hidden states of autoregressive teacher (Wei et al., 2019; Li et al., 2019).

Despite above improvements, decoding inconsistency can still be observed in the translation. Towards eliminating the errors, iterative decoding (Xia et al., 2017; Zhang et al., 2018; Geng et al., 2018) was proposed to employ multiple iterations to polish previously generated translation. As an early alternative, Lee et al. (2018) corrected the original non-autoregressive output by passing it multiple times through a denoising autoencoder. Instead of generating in discrete space of sentences, continuous latent variables were utilized to improve iterative refinements (Shu et al., 2020; Lee et al., 2020). Subsequently, Ghazvininejad et al. (2019) introduced mask-predict, which first generate target words non-autoregressively, and then repeat-

[6] STANFORD CORENLP TOOLKIT (Manning et al., 2014) is utilized to annotate translation output with Part-of-Speechs.

[7] PUNC-*punctuation*, NOUN-*noun*, PRT-*particle*, DET-*determiner*, CONJ-*conjunction*, ADJ-*adjective*, ADV-*adverb*, PREP-*preposition*, VERB-*verb*

SOURCE		Den Kindern stehen regionale Handwerker von 11 bis 17 Uhr helfend zur Seite .
CMLM	1∼8	Regional craftsmen are ~~at~~ their children from 11 a.m. to 5 p.m .
	9	Regional craftsmen are ~~assist their~~ children from 11 a.m. to 5 p.m .
	10	Regional craftsmen are helping the children from 11 a.m. to 5 p.m .
REWRITENAT	1	Regional craftsmen are ~~children children~~ children from 11 a.m. to 5 p.m .
	2	Regional craftsmen ~~are~~ assist the children from 11 a.m. to 5 p.m .
	3	Regional craftsmen will assist the children from 11 a.m. to 5 p.m .

Figure 6: An example from the WMT14 De→En translation that illustrates how REWRITENAT, together with CMLM generate text with iterative decoding. The translation pieces to be revised in next iteration are annotated as strikethrough, and the erroneous ones within the final translation are underlined. Notably, we remove the BPE tokens in the generated translation, leading to the unreasonable words (*e.g.,* cra@@ fts@@ from → *craftsfrom*).

edly mask out and re-generate the subset of words that model is least confident about (Ghazvininejad et al., 2020b; Guo et al., 2020b).

However, a serious drawback is that previous iterative NAT approaches exposes fundamental weakness in distinguishing erroneous translation pieces. Precisely, previous iterative NAT models based on mask-predict utilizes heuristic rules to consider the least confident words as the ones to be revised, but it struggles to perfectly make correct classifications simply relying on the probability distribution of the generated translation. Despite LevT (Gu et al., 2019) can alleviate the issue to some extent by adopting two basic operations (*i.e.,insert* and *delete*), a serious discrepancy in input data distribution between training and decoding exists due to the utilization of iterative strategy into decoding but not training. Towards address above issues, REWRITENAT adopts an additional locator module specialized to distinguish the erroneous translation pieces, and iterative training strategy is utilized to maintain the consistency of data distribution with iterative decoding.

8 Conclusion

In this work, we propose an architecture named REWRITENAT, which explicitly learns to rewrite the erroneous translation pieces, and iterative training is utilized to train this architecture. Extensive experimental results demonstrate REWRITENAT can achieve remarkable improvement over previous iterative NAT models, but with significantly less decoding iterations. The further analysis reveals that the generation orders of REWRITENAT measured by the percentage of decoding iterations are consistent with easy-first hypothesis.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. We also thank Heng Gong, Zhangyin Feng, Xiachong Feng, Runze Nie, Yichong Huang and Mingtao Yang for helpful discussion. This work was supported by the National Key R&D Program of China via grant 2020AAA0106502, National Natural Science Foundation of China (NSFC) via grant 61976073 and Shenzhen Foundational Research Funding (JCYJ20200109113441941).

References

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

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. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. 2021a. Rejuvenating low-frequency words: Making the most of parallel data in non-autoregressive translation.

In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3431–3441, Online. Association for Computational Linguistics.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. 2021b. Understanding and improving lexical choice in non-autoregressive translation. In *International Conference on Learning Representations (ICLR)*.

Cunxiao Du, Zhaopeng Tu, and Jing Jiang. 2021. Order-agnostic cross entropy for non-autoregressive machine translation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2849–2859. PMLR.

Dmitrii Emelianenko, Elena Voita, and Pavel Serdyukov. 2019. Sequence modeling with unconstrained generation order. In *Advances in Neural Information Processing Systems*, volume 32, pages 7698–7709. Curran Associates, Inc.

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. 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, Brussels, Belgium. Association for Computational Linguistics.

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

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China. Association for Computational Linguistics.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. 2020b. Semi-autoregressive training improves mask-predict decoding. *CoRR*, abs/2001.08785.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations (ICLR)*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, volume 32, pages 11179–11189. Curran Associates, Inc.

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

Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2020a. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7839–7846.

Junliang Guo, Linli Xu, and Enhong Chen. 2020b. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 376–385, Online. Association for Computational Linguistics.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

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

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

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, Austin, Texas. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Jason Lee, Raphael Shu, and Kyunghyun Cho. 2020. Iterative refinement in the continuous space for non-autoregressive neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1006–1015, Online. Association for Computational Linguistics.

Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Hint-based training for non-autoregressive machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5708–5713, Hong Kong, China. Association for Computational Linguistics.

Jindřich Libovický and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels, Belgium. Association for Computational Linguistics.

Jinglin Liu, Yi Ren, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. Task-level curriculum learning for non-autoregressive neural machine translation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 3861–3867. International Joint Conferences on Artificial Intelligence Organization. Main track.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4282–4292, Hong Kong, China. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. 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.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. 2020. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):198–205.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2020. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8846–8853.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5976–5985. PMLR.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. In *Advances in Neural Information Processing Systems*, volume 32, pages 3011–3020. Curran Associates, Inc.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 6000–6010. Curran Associates, Inc.

Chunqi Wang, Ji Zhang, and Haiqing Chen. 2018. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 479–488, Brussels, Belgium. Association for Computational Linguistics.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5377–5384.

Bingzhen Wei, Mingxuan Wang, Hao Zhou, Junyang Lin, and Xu Sun. 2019. Imitation learning for non-autoregressive neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1304–1312, Florence, Italy. Association for Computational Linguistics.

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*, volume 30. Curran Associates, Inc.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1):5698–5705.

Chunting Zhou, Graham Neubig, and Jiatao Gu. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations (ICLR)*.