Learning to Recover from Multi-Modality Errors for Non-Autoregressive Neural Machine Translation

Qiu Ran[*], **Yankai Lin**[*†], **Peng Li**[*†], **Jie Zhou**
Pattern Recognition Center, WeChat AI, Tencent Inc., China
{soulcaptran,yankailin,patrickpli,withtomzhou}@tencent.com

Abstract

Non-autoregressive neural machine translation (NAT) predicts the entire target sequence simultaneously and significantly accelerates inference process. However, NAT discards the dependency information in a sentence, and thus inevitably suffers from the multi-modality problem: the target tokens may be provided by different possible translations, often causing token repetitions or missing. To alleviate this problem, we propose a novel semi-autoregressive model RecoverSAT in this work, which generates a translation as a sequence of segments. The segments are generated simultaneously while each segment is predicted token-by-token. By dynamically determining segment length and deleting repetitive segments, RecoverSAT is capable of recovering from repetitive and missing token errors. Experimental results on three widely-used benchmark datasets show that our proposed model achieves more than 4× speedup while maintaining comparable performance compared with the corresponding autoregressive model.

1 Introduction

Although neural machine translation (NMT) has achieved state-of-the-art performance in recent years (Cho et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017), most NMT models still suffer from the slow decoding speed problem due to their autoregressive property: the generation of a target token depends on all the previously generated target tokens, making the decoding process intrinsically nonparallelizable.

Recently, non-autoregressive neural machine translation (NAT) models (Gu et al., 2018; Li et al., 2019; Wang et al., 2019; Guo et al., 2019a; Wei et al., 2019) have been investigated to mitigate the slow decoding speed problem by generating all target tokens independently in parallel, speeding up the decoding process significantly. Unfortunately, these models suffer from the multi-modality problem (Gu et al., 2018), resulting in inferior translation quality compared with autoregressive NMT. To be specific, a source sentence may have multiple feasible translations, and each target token may be generated with respect to different feasible translations since NAT models discard the dependency among target tokens. This generally manifests as repetitive or missing tokens in the translations. Table 1 shows an example. The German phrase "*viele Farmer*" can be translated as either "*lots of farmers*" or "*a lot of farmers*". In the first translation (Trans. 1), "*lots of*" are translated w.r.t. "*lots of farmers*" while "*of farmers*" are translated w.r.t. "*a lot of farmers*" such that two "*of*" are generated. Similarly, "*of*" is missing in the second translation (Trans. 2). Intuitively, the multi-modality problem has a significant negative effect on the translation quality of NAT.

Src.	es gibt heute viele Farmer mit diesem Ansatz
Feasible Trans.	there are lots of farmers doing this today there are a lot of farmers doing this today
Trans. 1	there are lots of of farmers doing this today
Trans. 2	there are a lot farmers doing this today

Table 1: A multi-modality problem example: NAT models generate each target token independently such that they may correspond to different feasible translations, which usually manifests as repetitive (Trans. 1) or missing (Trans. 2) tokens.

Intensive efforts have been devoted to alleviate the above problem, which can be roughly divided into two lines. The first line of work leverages the iterative decoding framework to break the independence assumption, which first generates an initial translation and then refines the translation

[*] indicates equal contribution
[†] indicates corresponding author

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3059–3069
July 5 - 10, 2020.

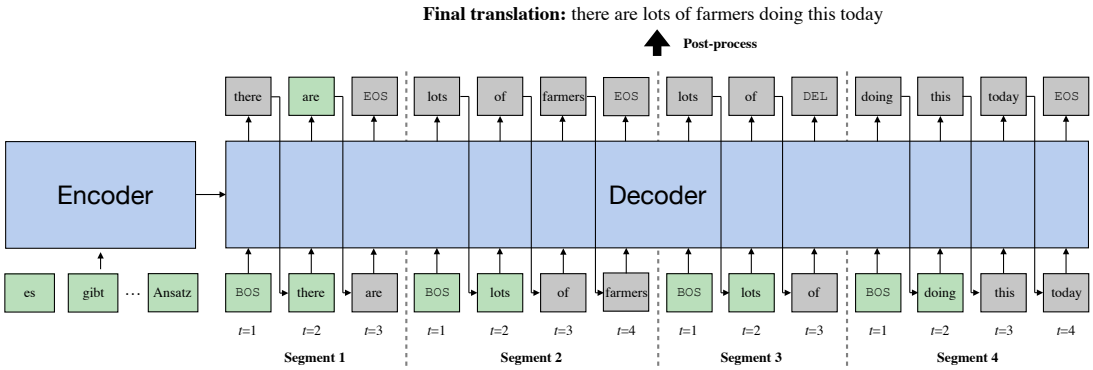

Figure 1: An overview of our RecoverSAT model. RecoverSAT generates a translation as a sequence of segments. The segments are generated simultaneously while each segment is generated token-by-token conditioned on both the source tokens and the translation history of all segments (e.g., the token "*are*" in the first segment is predicted based on all the tokens colored green). Repetitive segments (e.g., the third segment "*lots of*") are detected and deleted automatically.

iteratively by taking both the source sentence and the translation of last iteration as input (Lee et al., 2018; Ghazvininejad et al., 2019). Nevertheless, it requires to refine the translations for multiple times in order to achieve better translation quality, which hurts decoding speed significantly. The other line of work tries to improve the vanilla NAT model to better capture target-side dependency by leveraging extra autoregressive layers in the decoder (Shao et al., 2019a; Wang et al., 2018), introducing latent variables and/or more powerful probabilistic frameworks to model more complex distributions (Kaiser et al., 2018; Akoury et al., 2019; Shu et al., 2019; Ma et al., 2019), guiding the training process with an autoregressive model (Li et al., 2019; Wei et al., 2019), etc. However, these models cannot alter a target token once it has been generated, which means these models are not able to recover from an error caused by the multi-modality problem.

To alleviate the multi-modality problem while maintaining a reasonable decoding speedup, we propose a novel semi-autoregressive model named RecoverSAT in this work. RecoverSAT features in three aspects: (1) To improve decoding speed, we assume that a translation can be divided into several segments which can be generated simultaneously. (2) To better capture target-side dependency, the tokens inside a segment is autoregressively generated conditioned not only on the previously generated tokens in this segment but also on those in other segments. On one hand, we observe that repetitive tokens are more likely to occur within a short context. Therefore, autoregressively generating a segment is beneficial for reducing repetitive tokens. On the other hand, by conditioning on previously generated tokens in other segments, the model is capable of guessing what feasible translation candidates have been chosen by each segment and adapts accordingly, e.g., recovering from missing token errors. As a result, our model captures more target-side dependency such that the multi-modality problem can be alleviated naturally. (3) To make the model capable of recovering from repetitive token errors, we introduce a segment deletion mechanism into our model. Informally speaking, our model will mark a segment to be deleted once it finds the content has been translated in other segments.

We conduct experiments on three benchmark datasets for machine translation to evaluate the proposed method. The experimental results show that RecoverSAT is able to decode over 4× faster than the autoregressive counterpart while maintaining comparable performance. The source code of this work is released on https://github.com/ranqiu92/RecoverSAT.

2 Background

2.1 Autoregressive Neural Machine Translation

Autoregressive neural machine translation (AT) generates the translation token-by-token conditioned on translation history. Denoting a source sentence as $\mathbf{x} = \{x_i\}_{i=1}^{T'}$ and a target sentence as $\mathbf{y} = \{y_j\}_{j=1}^{T}$, AT models the joint probability as:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} P(y_t|\mathbf{y}_{<t}, \mathbf{x}). \quad (1)$$

where $\mathbf{y}_{<t}$ denotes the generated tokens before y_t.

During decoding, the translation history dependency makes the AT model predict each token after all previous tokens have been generated, which makes the decoding process time-consuming.

2.2 Non-Autoregressive Neural Machine Translation

Non-autoregressive neural machine translation (NAT) (Gu et al., 2018) aims to accelerate the decoding process, which discards the dependency of translation history and models $P(\mathbf{y}|\mathbf{x})$ as a product of the conditionally independent probability of each token:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} P(y_t|\mathbf{x}). \quad (2)$$

The conditional independence enables the NAT models to generate all target tokens in parallel.

However, independently predicting all target tokens is challenging as natural language often exhibits strong correlation across context. Since the model knows little information about surrounding target tokens, it may consider different possible translations when predicting different target tokens. The problem is known as the *multi-modality problem* (Gu et al., 2018) and significantly degrades the performance of NAT models.

3 Approach

3.1 Overview

RecoverSAT extends the original Transformer (Vaswani et al., 2017) to enable the decoder to perform generation autoregressively in local and non-autoregressively in global. An overview of the architecture of our RecoverSAT model is shown in Figure 1. As illustrated in the figure, RecoverSAT simultaneously predicts all segments "*there are* `EOS`", "*lots of farmers* `EOS`", "*a lot* `DEL`" and "*doing this today* `EOS`". And at each time step, it generates a token for each incomplete segment. The special token `DEL` denotes the segment should be deleted and `EOS` denotes the end of a segment. Combining all the segments, we obtain the final translation "*there are lots of farmers doing this today*".

Formally, assuming a translation $\mathbf{y}$ is generated as K segments $\mathbf{S}^1, \mathbf{S}^2, \cdots, \mathbf{S}^K$, where $\mathbf{S}^i$ is a subsequence of the translation[1]. For description simplicity, we assume that all the segments have the same length. RecoverSAT predicts a token for each segment conditioned on all previously generated tokens at each generation step, which can be formulated as:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{L}\prod_{i=1}^{K} P(\mathbf{S}_t^i|\mathbf{S}_{<t}^1 \cdots \mathbf{S}_{<t}^K; \mathbf{x}), \quad (3)$$

where $\mathbf{S}_t^i$ denotes the t-th token in the i-th segment, $\mathbf{S}_{<t}^i = \{\mathbf{S}_1^i, \cdots, \mathbf{S}_{t-1}^i\}$ denotes the translation history in the i-th segment, and L is segment length.

Here, two natural problems arise for the decoding process:

- How to determine the length of a segment?
- How to decide a segment should be deleted?

We address the two problems in a uniform way in this work. Suppose the original token vocabulary is V, we extend it with two extra tokens `EOS` and `DEL`. Then for the segment $\mathbf{S}^i$, the most probable token $\hat{\mathbf{S}}_t^i$ at time step t:

$$\hat{\mathbf{S}}_t^i = \underset{\mathbf{S}_t^i \in V \cup \{\texttt{EOS},\texttt{DEL}\}}{\arg\max} P(\mathbf{S}_t^i|\mathbf{S}_{<t}^1 \cdots \mathbf{S}_{<t}^K; \mathbf{x}) \quad (4)$$

has three possibilities:

(1) $\hat{\mathbf{S}}_t^i \in V$: the segment $\mathbf{S}^i$ is incomplete and the decoding process for it should continue;

(2) $\hat{\mathbf{S}}_t^i = \texttt{EOS}$: the segment $\mathbf{S}^i$ is complete and the decoding process for it should terminate;

(3) $\hat{\mathbf{S}}_t^i = \texttt{DEL}$: the segment $\mathbf{S}^i$ is repetitive and should be deleted. Accordingly, the decoding process for it should terminate.

The entire decoding process terminates when all the segments meet `EOS`/`DEL` or reach the maximum token number. It should be noticed that we do not explicitly delete a segment when `DEL` is encountered but do it via post-processing. In other words, the model is trained to ignore the segment to be deleted implicitly.

3.2 Learning to Recover from Errors

As there is little target-side information available in the early stage of the decoding process, the errors caused by the multi-modality problem is inevitable. In this work, instead of reducing such errors directly, we propose two training mechanisms to teach our RecoverSAT model to recover

[1]Note that, by fixing segment length (token number of each segment) instead, the segment number K can be changed dynamically according to the sentence length. In other words, we can predict the target sentence length to determine the segment number during inference. In this case, our model can also decode in constant time.

from errors: (1) Dynamic Termination Mechanism: learning to determine segment length according to target-side context; (2) Segment Deletion Mechanism: learning to delete repetitive segments.

3.2.1 Dynamic Termination Mechanism

As shown in Section 3.1, instead of pre-specifying the lengths of segments, we let the model determine the lengths by emitting the EOS token. This strategy helps our model recover from multi-modality related errors in two ways:

1. The choice of the first few tokens is more flexible. Taking Figure 1 as an example, if the decoder decides the first token of the second segment is "*of*" instead of "*lots*" (i.e., "*lots*" is not generated in the second segment), it only needs to generate "*lots*" before "EOS" in the first segment in order to recover from missing token errors. In contrast, if the decoder decides the first token is "*are*", it can avoid repetitive token error by not generating "*are*" in the first segment;

2. As shown in Eq. 3, a token is generated conditioned on all the previously generated tokens in *all the segments*. Therefore, the decoder has richer target-side information to detect and recover from such errors.

However, it is non-trivial to train the model to learn such behaviour while maintaining a reasonable speedup. On one hand, as the decoding time of our RecoverSAT model is proportional to the maximum length of the segments, we should divide the target sentences of training instances into equal-length segments to encourage the model to generate segments with identical length. On the other hand, the model should be exposed to the multi-modality related errors to enhance its ability of recovering from such errors, which suggests that the target sentences of training instances should be divided randomly to simulate these errors.

To alleviate the problem, we propose a mixed annealing dividing strategy. To be specific, we randomly decide whether to divide a target sentence equally or randomly at each training step and gradually anneal to the equally-dividing method at the end of training. Formally, given the target sentence $\mathbf{y}$ and the segment number K, we define the segment dividing indice set $\mathbf{r}$ as follows:

$$s \sim \text{Bernoulli}(p), \tag{5}$$

$$\mathbf{r} = \begin{cases} \text{EQUAL}(T, K-1) & s = 0 \\ \text{RAND}(T, K-1) & s = 1 \end{cases}, \tag{6}$$

where $\text{Bernoulli}(p)$ is the Bernoulli distribution with parameter p, $\text{EQUAL}(n, m) = \left\{\lceil \frac{n}{m+1} \rceil, \lceil \frac{2n}{m+1} \rceil, \cdots, \lceil \frac{mn}{m+1} \rceil\right\}$, $\text{RAND}(n, m)$ sampling m non-duplicate indices from $[1, n]$. A larger value of p leads to better error recovering ability while a smaller one encourages the model to generate segments with similar lengths (in other words, better speedup). To balance the two aspects, we gradually anneal p from 1 to 0 in the training process, which achieves better performance (Section 4.5).

3.2.2 Segment Deletion Mechanism

Although the dynamic termination mechanism makes the model capable of recovering from missing token errors and reducing repetitive tokens, the model still can not recover from errors where token repetition errors have already occurred. We find the major errors of our model occur when generating the first token of each segment since it cannot see any history and future. In this situation, two repetitive segments will be generated. To alleviate this problem, we propose a segment-wise deletion strategy, which uses a special token DEL to indicate a segment is repetitive and should be deleted[2].

A straightforward way to train the model to learn to delete a segment is to inject pseudo repetitive segments into the training data. The following is an example:

Target Sentence	there are lots of farmers doing this today
+ Pseudo Repetitive Segment	there are lots of farmers lots of DEL doing this today

Given the target sentence "*there are lots of farmers doing this today*", we first divide it into 3 segments "*there are*", "*lots of farmers*" and "*doing this today*". Then we copy the first two tokens of the second segment and append the special token DEL to the end to construct a pseudo repetitive segment "*lots of* DEL". Finally, we insert the repetitive segment to the right of the chosen segment, resulting in 4 segments. Formally, given the expected segment number K and the target sentence $\mathbf{y}$, we first divide $\mathbf{y}$ into $K-1$ segments $\mathbf{S}^1, \mathbf{S}^2, \cdots, \mathbf{S}^{K-1}$ and then build a pseudo repetitive segment $\mathbf{S}^i_{rep}$ by copying the first m tokens of a randomly chosen segment $\mathbf{S}^i$ and appending DEL to the end, m is uniformly

[2]It is more flexible to employ token-wise deletion strategy which could handle more complex cases. We will explore this in future.

sampled from $[1, |\mathbf{S}^i|]$. Finally, $\mathbf{S}^i_{rep}$ is inserted at the right side of $\mathbf{S}^i$. The final K segments are $\mathbf{S}^1, \mathbf{S}^2, \cdots, \mathbf{S}^i, \mathbf{S}^i_{rep}, \mathbf{S}^{i+1}, \cdots, \mathbf{S}^{K-1}$.

However, injecting such pseudo repetitive segments to all training instances will mislead the model that generating then deleting a repetitive segment is a must-to-have behaviour, which is not desired. Therefore, we inject pseudo repetitive segment into a training instance with probability q in this work.

4 Experiments

4.1 Datasets

We conduct experiments on three widely-used machine translation datasets: IWSLT16 En-De (196k pairs), WMT14 En-De (4.5M pairs) and WMT16 En-Ro (610k pairs). For fair comparison, we use the preprocessed datasets in Lee et al. (2018), of which sentences are tokenized and segmented into subwords using byte-pair encoding (BPE) (Sennrich et al., 2016) to restrict the vocabulary size. We use a shared vocabulary of 40k subwords for both source and target languages. For the WMT14 En-De dataset, we use newstest-2013 and newstest-2014 as validation and test sets respectively. For the WMT16 En-Ro dataset, we employ newsdev-2016 and newstest-2016 as validation and test sets respectively. For the IWSLT16 En-De dataset, we use test2013 as the validation set.

4.2 Experimental Settings

For model hyperparameters, we follow most of the settings in (Gu et al., 2018; Lee et al., 2018; Wei et al., 2019). For the IWSLT16 En-De dataset, we use a small Transformer model ($d_{model} = 278$, $d_{hidden} = 507$, $n_{layer} = 5$, $n_{head} = 2$, $p_{dropout} = 0.1$). For the WMT14 En-De and WMT16 En-Ro datasets, we use a larger Transformer model ($d_{model} = 512$, $d_{hidden} = 512$, $n_{layer} = 6$, $n_{head} = 8$, $p_{dropout} = 0.1$). We linearly anneal the learning rate from 3×10^{-4} to 10^{-5} as in Lee et al. (2018) for the IWSLT16 En-De dataset, while employing the warm-up learning rate schedule (Vaswani et al., 2017) with $t_{warmup} = 4000$ for the WMT14 En-De and WMT16 En-Ro datasets. We also use label smoothing of value $\epsilon_{ls} = 0.15$ for all datasets. We utilize the sequence-level distillation (Kim and Rush, 2016), which replaces the target sentences in the training dataset with sentences generated by an autoregressive model, and set the beam size of the technique to 4. We use the encoder of the corresponding autoregressive model to initialize the encoder of RecoverSAT, and share the parameters of source and target token embedding layers and the pre-softmax linear layer. We measure the speedup of model inference in each task on a single NVIDIA P40 GPU with the batch size 1.

4.3 Baselines

We use the Transformer (Vaswani et al., 2017) as our AT baseline and fifteen latest strong NAT models as NAT baselines, including: (1) fertility-based model: NAT-FT (Gu et al., 2018); (2) iterative decoding based models: NAT-IR (Lee et al., 2018) and CMLM (Ghazvininejad et al., 2019); (3) models learning from AT teachers: imitate-NAT (Wei et al., 2019), NART (Li et al., 2019) and FCL-NAT (Guo et al., 2019b); (4) latent variable framework based models: LV NAR (Shu et al., 2019) and FlowSeq (Ma et al., 2019); (5) regularization framework based model: NAT-REG (Wang et al., 2019); (6) models introducing extra target-side dependencies: SAT (Wang et al., 2018), SynST (Akoury et al., 2019), NAT-FS (Shao et al., 2019a), PNAT (Bao et al., 2019), NART-DCRF (Sun et al., 2019) and ReorderNAT (Ran et al., 2019).

4.4 Overall Results

The performance of our RecoverSAT model and the baselines is shown in Table 2. Due to the space limitation, we only show the results corresponding to the settings of the best BLEU scores for the baselines [3]. From Table 2, we can observe that:

(1) Our RecoverSAT model achieves comparable performance with the AT baseline (Transformer) while keeping significant speedup. When $K = 2$, the BLEU score gap is moderate (from 0.06 to 0.4, even better than Transformer on the WMT16 En→Ro and Ro→En tasks) and the speedup is about 2×. When $K = 10$, the BLEU scores drop less than 5% relatively, and the speedup is considerably good (over 4×).

(2) Our RecoverSAT model outperforms all the strong NAT baselines except CMLM (on the WMT16 En→Ro and Ro→En tasks). However, the performance gap is negligible (0.16 and 0.12 respectively), and CMLM is a multi-step NAT method which is significantly slower than our model.

[3] A thorough comparison under other settings can be found in Appendix B.

Model	Iterative Decoding	WMT14 En-De			WMT16 En-Ro			IWSLT16 En-De	
		En→	De→	Speedup	En→	Ro→	Speedup	En→	Speedup
Transformer		27.17	31.95	1.00×	32.86	32.60	1.00×	31.18	1.00×
NAT-FT+NPD ($n = 100$)		19.17	23.20	-	29.79	31.44	-	28.16	2.36×
SynST		20.74	25.50	4.86×	-	-	-	23.82	3.78×
NAT-IR ($iter = 10$)	✓	21.61	25.48	2.01×	29.32	30.19	2.15×	27.11	1.55×
NAT-FS		22.27	27.25	3.75×	30.57	30.83	3.70×	27.78	3.38×
imitate-NAT+LPD ($n = 7$)		24.15	27.28	-	31.45	31.81	-	30.68	9.70×
PNAT+LPD ($n = 9$)		24.48	29.16	-	-	-	-	-	-
NAT-REG+LPD ($n = 9$)		24.61	28.90	-	-	-	-	27.02	-
LV NAR		25.10	-	6.8×	-	-	-	-	-
NART+LPD ($n = 9$)		25.20	29.52	17.8×	-	-	-	-	-
FlowSeq+NPD ($n = 30$)		25.31	30.68	<1.5×	32.20	32.84	-	-	-
FCL-NAT+NPD ($n = 9$)		25.75	29.50	16.0×	-	-	-	-	-
ReorderNAT		26.51	31.13	-	31.70	31.99	-	30.26	5.96×
NART-DCRF+LPD ($n = 19$)		26.80	30.04	4.39×	-	-	-	-	-
SAT ($K = 2$)		26.90	-	1.51×	-	-	-	-	-
CMLM ($iter = 10$)	✓	27.03	30.53	<1.5×	**33.08**	**33.31**	-	-	-
RecoverSAT ($K = 2$)		**27.11**	**31.67**	2.16×	32.92	33.19	2.02×	**30.78**	2.06×
RecoverSAT ($K = 5$)		26.91	31.22	3.17×	32.81	32.80	3.16×	30.55	3.28×
RecoverSAT ($K = 10$)		26.32	30.46	4.31×	32.59	32.29	4.31×	29.90	4.68×

Table 2: Performance (BLEU) of Transformer, the NAT/semi-autoregressive models and RecoverSAT on three widely-used machine translation benchmark datasets. NPD denotes the noisy parallel decoding technique (Gu et al., 2018) and LPD denotes the length parallel decoding technique (Wei et al., 2019). n denotes the sample size of NPD or LPD. $iter$ denotes the refinement number of the iterative decoding method.

(3) As K grows, the BLEU scores drop moderately and the speedup grows significantly, indicating that our RecoverSAT model has a good generalizability. For example, the BLEU scores drop less than 0.45 when K grows from 2 to 5, and drop no more than 0.90 except on the WMT14 De→En task when K further grows to 10. Meanwhile, the speedup for $K = 10$ is larger than $4\times$, which is considerably good.

(4) There are only 7 baselines (SynST, imitate-NAT+LPD, LV NAR, NART+LPD, FCL-NAT+NPD, ReorderNAT and NART-DCRF+LPD) achieving better speedup than our RecoverSAT model when $K = 10$. However, only ReorderNAT and NART-DCRF+LPD achieve comparable BLEU scores with our model.The improvements of both ReorderNAT and NART-DCRF are complementary to our method. It is an interesting future work to join these works together.

4.5 Effect of Dynamic Termination Mechanism

As discussed in Section 3.2.1, the dynamic termination mechanism is used to train our RecoverSAT model to learn to determine segment length dynamically conditioned on target-side context such that it is recoverable from multi-modality related errors. In this section, we investigate the effect of this mechanism and the results are shown in Table 3.

As multi-modality related errors generally manifest as repetitive or missing tokens in the translation, we propose two quantitative metrics "Rep" and "Mis" to measure these two phenomenons respectively. "Rep" is defined as the relative increment of repetitive token ratio w.r.t. to a reference AT model. And "Mis" is defined as the relative increment of missing token ratio given the references w.r.t. to a reference AT model. Formally, given the translations $\hat{\mathbf{Y}} = \{\hat{\mathbf{y}}^1 \cdots \hat{\mathbf{y}}^k \cdots\}$ produced by the model to be evaluated and the translations $\hat{\mathbf{Y}}_{auto} = \{\hat{\mathbf{y}}^1_{auto} \cdots \hat{\mathbf{y}}^k_{auto} \cdots\}$ produced by the reference AT model, "Rep" is defined as

$$\text{Rep} = \frac{r(\hat{\mathbf{Y}}) - r(\hat{\mathbf{Y}}_{auto})}{r(\hat{\mathbf{Y}}_{auto})}, \tag{7}$$

$$r(\mathbf{Y}) = \frac{\sum_k \sum_{j=2}^{|\mathbf{y}^k|} \mathbb{1}\left(\sum_{i=1}^{9} \mathbb{1}(\mathbf{y}^k_j = \mathbf{y}^k_{j-i}) \geq 1\right)}{\sum_k |\mathbf{y}^k|}, \tag{8}$$

where $\mathbb{1}(cond) = 1$ if the condition $cond$ holds otherwise 0, and $\mathbf{y}^k_j$ is the j-th token of the translation sentence $\mathbf{y}^k$.

Given $\hat{\mathbf{Y}}$, $\hat{\mathbf{Y}}_{auto}$ and references $\bar{\mathbf{Y}} = \{\bar{\mathbf{y}}^1 \cdots \bar{\mathbf{y}}^k \cdots\}$, "Mis" is defined as

$$\text{Mis} = \frac{m(\hat{\mathbf{Y}}, \bar{\mathbf{Y}}) - m(\hat{\mathbf{Y}}_{auto}, \bar{\mathbf{Y}})}{m(\hat{\mathbf{Y}}_{auto}, \bar{\mathbf{Y}})}, \tag{9}$$

	p	BLEU	Rep	Mis	Step
NAT		24.57	50.09	9.09	1
RecoverSAT (K=10)	0.0	27.09	22.05	6.95	4.2
	0.5	29.80	12.69	3.96	5.5
	1.0	29.89	13.00	4.75	7.2
	1→0	29.90	7.09	3.56	5.1

Table 3: Effect of the dynamic termination mechanism. The results are evaluated on the IWSLT16 En-De validation set. p is the parameter of Bernoulli distribution in Eq. 5. "Rep" and "Mis" measure the relative increment (%) of repetitive and missing token ratios (see Section 4.5), the smaller the better. "Step" denotes the average number of decoding steps. And "1→0" denotes annealing p from 1 to 0 linearly.

where $m(\cdot,\cdot)$ computes the missing token ratio and is defined as follows:

$$c_w(\mathbf{y}^k, \bar{\mathbf{y}}^k) = \max\left(c(\bar{\mathbf{y}}^k, w) - c(\mathbf{y}^k, w), 0\right),$$
$$m(\mathbf{Y}, \bar{\mathbf{Y}}) = \frac{\sum_k \sum_{w \in \bar{\mathbf{y}}^k} c_w(\mathbf{y}^k, \bar{\mathbf{y}}^k)}{\sum_k |\bar{\mathbf{y}}^k|}, \quad (10)$$

where $c(\mathbf{y}, w)$ is the occurrence number of a token w in the sentence $\mathbf{y}$.

From Table 3, we can observe that: (1) By using the dynamic termination mechanism ($p = 0.5$, 1.0, $1 \rightarrow 0$, where p is the parameter of Bernoulli distribution (Eq. 5)), both repetitive and missing token errors are reduced ("Rep" & "Mis"), and the BLEU scores are increased, indicating the effectiveness of the mechanism; (2) As p grows larger, the average number of decoding steps ("Step") increases significantly. The reason is that more target sentences are divided into segments equally with smaller p during training and the model is biased to generate segments with similar lengths. However, if the model is not exposed to randomly divided segments ($p = 0.0$), it fails to learn to recover from multi-modality related errors and the BLEU score drops significantly. (3) By using the *annealing dividing strategy* ($p = 1 \rightarrow 0$, see Section 3.2.1), we achieve a good balance between decoding speed and translation quality. Therefore, we use it as the default setting in this paper.

4.6 Effect of Segment Deletion Mechanism

In this section, we investigate the effect of the segment deletion mechanism and the results are shown in Table 4, where q is the probability of injecting pseudo repetitive segments to each training instance. From the results we can observe that: (1) Without using the segment deletion mechanism ($q = 0$), the BLEU score drops significantly and the repetitive token errors ("Rep") increase drastically, indicating that the mechanism is effective for recovering from repetitive token errors. (2) As q grows larger, the average number of decoding steps ("Step") increases steadily because the model is misled that to generate then delete a repetitive segment is expected. Thus, q should not be too large. (3) The repetitive token errors ("Rep") increase drastically when $q > 0.7$. We believe that the reason is that the pseudo repetitive segments are constructed randomly, making it hard to learn the underlying mapping. (4) The model achieves the best performance with $q = 0.5$. Therefore, we set $q = 0.5$ in our experiments.

	q	BLEU	Rep	Step
NAT		24.57	50.09	1
RecoverSAT ($K = 10$)	0.0	28.56	26.24	4.4
	0.1	29.73	5.11	4.7
	0.3	29.61	7.71	5.1
	0.5	29.90	7.09	5.1
	0.7	29.76	11.47	5.2
	0.9	29.25	21.38	5.3
	1.0	29.13	20.55	5.2

Table 4: Effect of segment deletion mechanism. The results are evaluated on the IWSLT16 En-De validation set. q is the probability of injecting pseudo repetitive segments to each training instance (see Section 3.2.2).

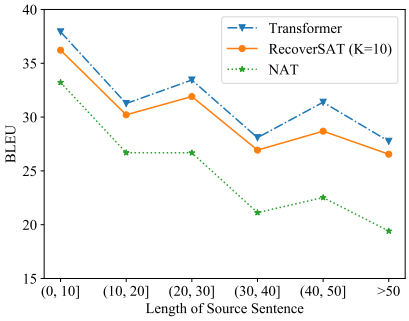

Figure 2: Translation quality on the IWSLT16 En-De validation set over sentences in different length.

4.7 Performance over Sentence Lengths

Figure 2 shows the translation quality of the Transformer, our RecoverSAT model with $K = 10$ and NAT on the IWSLT16 En-De validation set bucketed by different source sentence lengths. From the figure, we can observe that RecoverSAT surpasses NAT significantly and achieves comparable performance to the Transformer on all length buckets, which indicates the effectiveness of our model.

Source		die er_greif_endste Abteilung ist das Denk_mal für die Kinder , das zum Ged_enken an die 1,5 Millionen Kinder , die in den Konzent_rations_lagern und Gas_k_ammern vernichtet wurden , erbaut wurde .
Reference		the most tragic section is the children's mem_orial , built in memory of 1.5 million children killed in concentration camps and gas cham_bers .
NAT	Translation	the most tangible department department the monument monument the children , which was built commem_commem_orate 1.5 1.5 million children were destroyed in the concentration camps and gas cham_bers .
RecoverSAT ($K = 10$)	Translation	**A**: [1] the EOS [2] most tangible department is the EOS [3] monument for children EOS [4] built to EOS [5] commem_orate the 1.5 EOS [6] million children destroyed EOS [7] in the concentration camps and EOS [8] in DEL [9] gas EOS [10] cham_bers . EOS
	Forced Translation	**B**: [1] the EOS [2] most tangible department is the EOS [3] monument for children EOS [4] built to EOS [5] commem_orate EOS [6] the 1.5 million children destroyed EOS [7] in the concentration camps and EOS [8] in DEL [9] gas EOS [10] cham_bers . EOS
		C: [1] the EOS [2] most tangible department is the EOS [3] monument for children EOS [4] built to EOS [5] commem_orate the 1.5 million children EOS [6] destroyed EOS [7] in concentration camps and EOS [8] in DEL [9] gas EOS [10] cham_bers . EOS
		D: [1] the EOS [2] most tangible department is the EOS [3] monument for children EOS [4] built to EOS [5] commem_orate the 1.5 million children destroyed EOS [6] in the concentration camps and EOS [7] in the DEL [8] in DEL [9] gas EOS [10] cham_bers . EOS

Table 5: Translation examples of NAT and RecoverSAT. "Forced Translation" denotes the generated sentence when we manually force the model to generate a certain token (colored green) at a certain position. We use yellow color to label repetitive tokens, red color to label missing tokens, and gray color to label the segments to be deleted. We use "_" to concatenate sub-words and subscript numbers (e.g., [1]) to mark the beginning of each segment.

4.8 Case Study

We present translation examples of NAT and our RecoverSAT model on the WMT14 De→En validation set in Table 5. From the table, we can observe that: (1) The multi-modality problem (repetitive and missing tokens) is severe in the sentence generated by NAT, while it is effectively alleviated by RecoverSAT (see translations A to D); (2) RecoverSAT can leverage target contexts to dynamically determine the segment length to reduce repetitive token errors (see translation B) or recover from missing token errors (see translations C and D); (3) RecoverSAT is capable of detecting and deleting the repetitive segments, even if there are multiple such segments (see translation D).

5 Related Work

There has been various work investigating to accelerate the decoding process of sequence generation models (Kalchbrenner et al., 2018; Gu et al., 2018). In the field of neural machine translation, which is the focus of this work, Gu et al. (2018) first propose non-autoregressive machine translation (NAT), which generates all target tokens simultaneously. Although accelerating the decoding process significantly, NAT suffers from the multi-modality problem (Gu et al., 2018) which generally manifests as repetitive or missing tokens in translation. Therefore, intensive efforts have been devoted to alleviate the multi-modality problem in NAT. Wang et al. (2019) regularize the decoder hidden states of neighboring tokens to reduce repetitive tokens; Sun et al. (2019) utilize conditional random field to model target-side positional contexts; Shao et al. (2019a) and Shao et al. (2019b) introduce target-side information via specially designed training loss while Guo et al. (2019a) enhance the input of the decoder with target-side information; Kaiser et al. (2018), Akoury et al. (2019), Shu et al. (2019) and Ma et al. (2019) incorporate latent variables to guide generation; Li et al. (2019), Wei et al. (2019) and Guo et al. (2019b) use autoregressive models to guide the training process of NAT; Ran et al. (2019) and Bao et al. (2019) consider the reordering information in decoding. Wang et al. (2018) further propose a semi-autoregressive Transformer method, which generates segments autoregressively and predicts the tokens in a segment non-autoregressively. However, none of the above methods explicitly consider recovering from multi-modality related errors.

Recently, multi-step NAT models have also been investigated to address this issue. Lee et al. (2018) and Ghazvininejad et al. (2019) adopt an iterative decoding methods which have the potential to re-

cover from generation errors. Besides, Stern et al. and Gu et al. (2019) also propose to use dynamic insertion/deletion to alleviate the generation repetition/missing. Different from these work, our model changes one-step NAT to a semi-autoregressive form, which maintains considerable speedup and enables the model to see the local history and future to avoid repetitive/missing words in decoding. Our work can further replace the one-step NAT to improve its performance.

6 Conclusion

In this work, we propose a novel semi-autoregressive model RecoverSAT to alleviate the multi-modality problem, which performs translation by generating segments non-autoregressively and predicts the tokens in a segment autoregressively. By determining segment length dynamically, RecoverSAT is capable of recovering from missing token errors and reducing repetitive token errors. By explicitly detecting and deleting repetitive segments, RecoverSAT is able to recover from repetitive token errors. Experiments on three widely-used benchmark datasets show that our RecoverSAT model maintains comparable performance with more than $4\times$ decoding speedup compared with the AT model.

Acknowledgments

We would like to thank all anonymous reviewers for their insightful comments.

References

Nader Akoury, Kalpesh Krishna, and Mohit Iyyer. 2019. Syntactically supervised transformers for faster neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1269–1281.

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

Yu Bao, Hao Zhou, Jiangtao Feng, Mingxuan Wang, Shujian Huang, Jiajun Chen, and Lei Li. 2019. Non-autoregressive transformer by position learning. *arXiv preprint arXiv:1911.10677*.

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*, pages 1724–1734.

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 6114–6123.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *Proceedings of International Conference on Learning Representations*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Proceedings of Advances in Neural Information Processing Systems 32*, pages 11181–11191.

Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. 2019a. Non-autoregressive neural machine translation with enhanced decoder input. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, volume 33, pages 3723–3730.

Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2019b. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. *arXiv preprint arXiv:1911.08717*.

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, pages 2390–2399.

Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. 2018. Efficient neural audio synthesis. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2410–2419.

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.

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.

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 5712–5717.

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 4273–4283.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2019. Guiding non-autoregressive neural machine translation decoding with reordering information. *arXiv preprint arXiv:1911.02215*.

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.

Chenze Shao, Yang Feng, Jinchao Zhang, Fandong Meng, Xilin Chen, and Jie Zhou. 2019a. Retrieving sequential information for non-autoregressive neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3013–3024.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. 2019b. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. *arXiv preprint arXiv:1911.09320*.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2019. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. *arXiv preprint arXiv:1908.07181*.

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

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 32*, pages 3011–3020.

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 30*, pages 5998–6008.

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.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, volume 33, pages 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.

A Positional Encoding

Our RecoverSAT model utilizes the positional encoding method in Vaswani et al. (2017) to encode the information about the positions of source tokens. The positional embedding is defined as:

$$\mathbf{PE}_{pos}[2i] = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad (11)$$

$$\mathbf{PE}_{pos}[2i+1] = \cos\left(\frac{pos}{10000^{2i/d}}\right), \quad (12)$$

where $\mathbf{PE}_{pos}[i]$ is the i-th element of the positional embedding vector $\mathbf{PE}_{pos}$ for the position pos, and d is the dimension of the positional embedding vector. Then we can compute the input vector of the encoder for the m-th source token w as:

$$\mathbf{E}_w = \mathbf{E}_w^{token} + \mathbf{PE}_m, \quad (13)$$

where $\mathbf{E}_w^{token}$ is the token embedding vector of w.

However, we can not apply this method to *target* tokens directly. Since lengths of segments are dynamically determined, the positions of the tokens in the target sentence, except those in the first segment, are not available during generation. To solve the problem, we use the aforementioned method to independently encode the position in the corresponding segment of each token instead and adopt an absolute segment embedding method, which uses a distinct trainable vector to represent the position of each segment. Formally, the input vector of the decoder for the n-th target token v of the j-th segment is computed as:

$$\mathbf{E}_v = \mathbf{E}_v^{token} + \mathbf{PE}_n + \mathbf{E}_j^{seg}, \quad (14)$$

where $\mathbf{E}_j^{seg}$ is the segment embedding vector for the segment position j.

Model	Iterative Decoding	WMT14 En-De			WMT16 En-Ro			IWSLT16 En-De	
		En→	De→	Speedup	En→	Ro→	Speedup	En→	Speedup
Transformer		27.17	31.95	1.00×	32.86	32.60	1.00×	31.18	1.00×
NAT-FT		17.69	21.47	-	27.29	29.06	-	26.52	15.6×
NAT-FT+NPD ($n = 10$)		18.66	22.41	-	29.02	30.76	-	27.44	7.68×
NAT-FT+NPD ($n = 100$)		19.17	23.20	-	29.79	31.44	-	28.16	2.36×
SynST		20.74	25.50	4.86×	-	-	-	23.82	3.78×
NAT-IR ($iter = 1$)	✓	13.91	16.77	11.39×	24.45	25.73	16.03×	22.20	8.98×
NAT-IR ($iter = 10$)	✓	21.61	25.48	2.01×	29.32	30.19	2.15×	27.11	1.55×
NAT-FS		22.27	27.25	3.75×	30.57	30.83	3.70×	27.78	3.38×
imitate-NAT		22.44	25.67	-	28.61	28.90	-	28.41	18.6×
imitate-NAT+LPD ($n = 7$)		24.15	27.28	-	31.45	31.81	-	30.68	9.70×
PNAT		23.05	27.18	-	-	-	-	-	-
PNAT+LPD ($n = 9$)		24.48	29.16	-	-	-	-	-	-
NAT-REG		20.65	24.77	-	-	-	-	23.14	-
NAT-REG+LPD ($n = 9$)		24.61	28.90	-	-	-	-	27.02	-
LV NAR		25.10	-	6.8×	-	-	-	-	-
NART		21.11	25.24	30.2×	-	-	-	-	-
NART+LPD ($n = 9$)		25.20	29.52	17.8×	-	-	-	-	-
FlowSeq-base		21.45	26.16	<1.5×	29.34	30.44	-	-	-
FlowSeq-base+NPD ($n = 30$)		23.48	28.40	<1.5×	31.75	32.49	-	-	-
FlowSeq-large		23.72	28.39	<1.5×	29.73	30.72	-	-	-
FlowSeq-large+NPD ($n = 30$)		25.31	30.68	<1.5×	32.20	32.84	-	-	-
FCL-NAT		21.70	25.32	28.9×	-	-	-	-	-
FCL-NAT+NPD ($n = 9$)		25.75	29.50	16.0×	-	-	-	-	-
ReorderNAT		26.51	31.13	-	31.70	31.99	-	30.26	5.96×
NART-DCRF		23.44	27.22	10.4×	-	-	-	-	-
NART-DCRF+LPD ($n = 19$)		26.80	30.04	4.39×	-	-	-	-	-
SAT ($K = 2$)		26.90	-	1.51×	-	-	-	-	-
SAT ($K = 6$)		24.83	-	2.98×	-	-	-	-	-
CMLM-small ($iter = 1$)	✓	15.06	19.26	-	20.12	20.36	-	-	-
CMLM-small ($iter = 10$)	✓	25.51	29.47	-	31.65	32.27	-	-	-
CMLM-base ($iter = 1$)	✓	18.05	21.83	-	27.32	28.20	-	-	-
CMLM-base ($iter = 10$)	✓	27.03	30.53	<1.5×	**33.08**	**33.31**	-	-	-
RecoverSAT ($K = 2$)		**27.11**	**31.67**	2.16×	32.92	33.19	2.02×	**30.78**	2.06×
RecoverSAT ($K = 5$)		26.91	31.22	3.17×	32.81	32.80	3.16×	30.55	3.28×
RecoverSAT ($K = 10$)		26.32	30.46	4.31×	32.59	32.29	4.31×	29.90	4.68×

Table 6: Performance (BLEU) of Transformer and the NAT/semi-autoregressive models on three widely-used machine translation benchmark datasets. NPD denotes the noisy parallel decoding technique (Gu et al., 2018) and LPD denotes the length parallel decoding technique (Wei et al., 2019). n denotes the sample size of NPD or LPD. $iter$ denotes the refinement number of the iterative decoding method.