POS-Constrained Parallel Decoding for Non-autoregressive Generation

Kexin Yang[♠] **Wenqiang Lei**[♢*] **Dayiheng Liu**[♠] **Weizhen Qi**[♣] **Jiancheng Lv**[♠]
[♠]College of Computer Science, Sichuan University
[♢]National University of Singapore
[♣]University of Science and Technology of China
{kexinyang0528, wenqianglei}@gmail.com

Abstract

The *multimodality problem* has become a major challenge of existing non-autoregressive generation (NAG) systems. A common solution often resorts to sequence-level knowledge distillation by rebuilding the training dataset through autoregressive generation (hereinafter known as "teacher AG"). The success of such methods may largely depend on a latent assumption, i.e., the teacher AG is superior to the NAG model. However, in this work, we experimentally reveal that this assumption does not always hold for the text generation tasks like text summarization and story ending generation. To provide a feasible solution to the multimodality problem of NAG, we propose incorporating linguistic structure (Part-of-Speech sequence in particular) into NAG inference instead of relying on teacher AG. More specifically, the proposed POS-constrained Parallel Decoding (POSPD) method aims at providing a specific POS sequence to constrain the NAG model during decoding. Our experiments demonstrate that POSPD consistently improves NAG models on four text generation tasks to a greater extent compared to knowledge distillation. This observation validates the necessity of exploring the alternatives for sequence-level knowledge distillation.

1 Introduction

Unlike autoregressive generation (AG) that generates tokens step-by-step, non-autoregressive generation (NAG) parallelly generates all tokens in one time step and thus the inference could be significantly speeded up (Ma et al., 2019; Ran et al., 2020; Susanto et al., 2020). Despite the computational advantage of NAG, it has faced the *multimodality problem* (Gu et al., 2018) caused by the conditionally independent decoding. A typical example of the problem is illustrated in Figure 1, where either

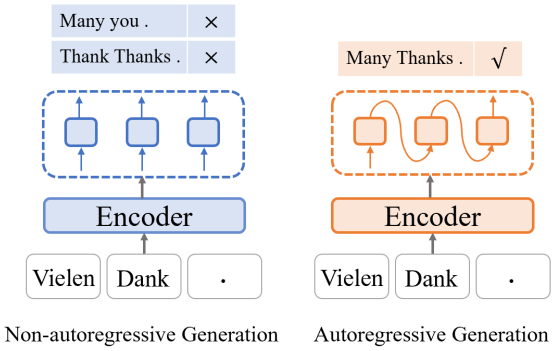

Figure 1: An example to explain "multimodality problem". The German sentence "Vielen Dank." can be translated into "Many Thanks." and "Thank you.".

of "Thank you." and "Many Thanks." is the correct translation (i.e., *generation modes*). In this example, a mixed mode "Many you." / "Thank Thanks." will be generated by NAG. It is because the conditional dependence among target words will be broken in parallel decoding. A typical manifestation is that words are usually missing (e.g., "Many you.") and repeating (e.g., "Thank Thanks.") in NAG's sentences. To solve this problem, the key is helping NAG models to deal with various generation modes.

To date, one of the most widely used solutions is sequence-level knowledge distillation (Kim and Rush, 2016) which aims to reduce the generation modes of the raw data (Zhou et al., 2019). Taking machine translation as an example, the knowledge distillation based methods rebuild the target sequence in the training set by employing an AG model to translate the training samples. The assumption is that the target sentences generated by one AG model tend to have less modality. Despite the success of the above studies, there are still two major limitations: (1) Most existing works mainly focus on machine translation where the performance of AG is generally assumed to be better

* Correspondence to Wenqiang Lei.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 5990–6000
August 1–6, 2021.

than NAG. Clearly, such a solution will degrade the performance of NAG on the task where the AG model cannot obtain a better result. As demonstrated in our experiments (See § 4.5), there are a number of such tasks beyond the assumption like text summarization and story ending generation. (2) The knowledge distillation based methods may cost a tremendous amount of time to rebuild a large-scale training set with AG, which runs counter to the initial goal of NAG to improve the speed.

To overcome the aforementioned limitations, we explore to alleviate the multimodality problem in a different manner. In short, we aim to constrain NAG generation modes in the inference stage, rather than directly reducing generation modes in the training stage. More specifically, our basic idea is that the linguistic structure of the target sentence could be helpful to alleviate the multimodality problem. In this paper, we show that the Part-of-Speech (POS) sequence, one of most simple solutions in modeling the linguistic structure (Cutting et al., 1992), could effectively verify our idea and show promising performance in four different tasks. In more details, the proposed POS-constrained Parallel Decoding (POSPD) trains a POS predictor to obtain POS tags of target sequences. In the inference stage, POSPD constrains NAG models to choose the final outputs that satisfy the pre-specified POS sequence. As the POS predictor with a shallow decoder is separately trained, our POSPD could act as a plug-and-play method to assistant NAG models with negligible extra time. Meanwhile, it also shows the speed advantage of our method even considering the time cost in building the POS dataset, since POS tagging is much faster than sentence generating due to the small POS dictionary.

To conduct a comprehensive empirical evaluation, we examine the generalizability of POSPD by applying it to two widely-used NAG models (i.e., CMLM and DisCo) over four text generation tasks, including text summarization, story ending generation, question generation, and machine translation. Experiments demonstrate that POSPD significantly and consistently improves the two NAG models and beats the sequence-level knowledge distillation with a considerable performance gap. The main contributions of this work could be summarized as follows:

- For the first time, we experimentally reveal that the implicit assumption of knowledge distillation does not always hold for the tasks (e.g., text summarization, story ending generation, as demonstrated in our experiments). In other words, AG cannot guarantee better performance than NAG, thus resulting in the undesirable performance of NAG if using knowledge distillation to alleviate the multimodality problem. This empirical result could provide novel insight to revisiting the role of the knowledge distillation in NAG.
- To alleviate the multimodality problem in various tasks, we propose POSPD by employing POS sequences to constrain the NAG generation modes in the inference stage. It is simple but effective, being able to act as a plug-and-play assistant for NAG models. Such a linguistic structure based solution shows an effective and efficient alternative to the knowledge distillation paradigm in alleviating the multimodality problem[1].

2 Related Works

In this section, we first analyze related works on alleviating the multimodality problem. Then, we review some representative works which introduce the linguistic structure into some text generation scenarios.

2.1 The Multimodality Problem in NAG

Recently, various attempts have been made to alleviate the multimodality problem, which can be roughly divided into two types: (1) Reducing the diversity of generation modes in training; (2) Helping models select one generation mode in inference. The first type usually trains the NAG model under the guidance of an AG model (called teacher AG), e.g., sequence-level knowledge distillation (Kim and Rush, 2016), learning from AG model's hidden state (Li et al., 2019) and the curriculum learning with AG model (Liu et al., 2020d; Guo et al., 2020a). However, these methods implicitly assume that the teacher AG can achieve better performance than NAG models, otherwise it may degrade the performance of the NAG models. As two typical methods of the second type, iterative and dynamic programming methods have achieved promising performance. In short, iterative models generate the target sentence by iteratively refining the latest output (Ghazvininejad et al., 2019; Kasai et al.,

[1]The source code and dataset are available at https://github.com/yangkexin/POSPD

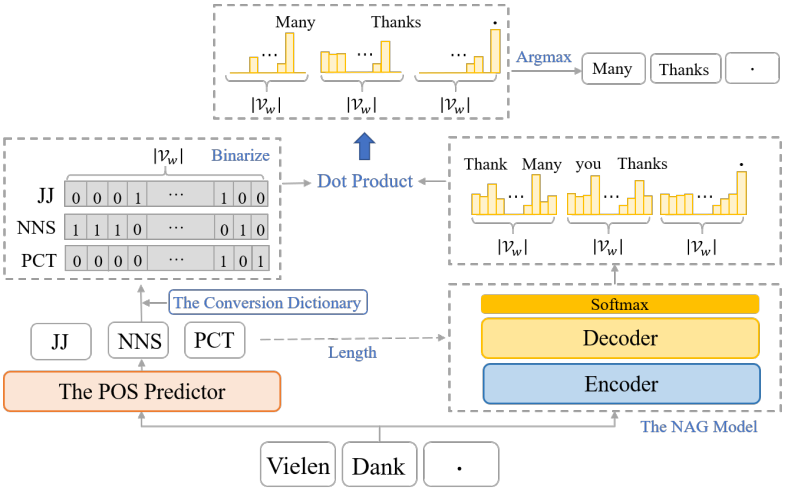

Figure 2: An overview of the POS-constrained Parallel Decoding

2020a; Guo et al., 2020b). Alternatively, dynamic programming methods use a heuristic searching strategy to select a better output from multiple decoded candidates (Sun et al., 2019; Saharia et al., 2020; Ghazvininejad et al., 2020). The biggest difference is prespecifying the linguistic structure to constrain the generation of NAG in a plug-and-play way. Extensive experiments verify the effectiveness and efficiency of our idea.

2.2 Leveraging the Linguistic Structure

Text generation involves multiple tasks, such as style transfer (Liu et al., 2020a) and text filling (Liu et al., 2019). Dating back to the period of statistical machine translation (Liu et al., 2006; Galley et al., 2006), linguistic structure prediction has long been investigated for it. Previous works often model and leverage syntactic structures on the decoder side, such as modeling long-distance word correspondence by syntactic dependency trees (Wu et al., 2017), implicitly incorporate linguistic prior in decoder (Eriguchi et al., 2017) and joint decoding with syntactic structure (Feng et al., 2020). In NAG, linguistic structures can also be helpful. As a global pattern of target sentence, it could serve as the complementary to the parallel decoding by helping models capture words dependency. However, directly incorporating aforementioned methods into NAG are less portable for current NAG models, since they are originally designed for AG. In comparison, POSPD can act as a plug-and-play component that uses a separate POS predictor to constrain NAG models during inference. Therefore, the NAG model can enjoy the benefits of the syntactical structure constraining while retaining its original model structure.

3 Methodology

In this section, we elaborate our POSPD for the NAG model. To ease of presentation, we start from a toy example to illustrate the overview of POSPD in § 3.1, and then give a detailed explanation of the implementation in § 3.2. After that, we present the training details of POSPD in § 3.3.

3.1 Overview

An overview of our POSPD method is demonstrated in Figure 2, where a toy example of machine translation is used as a showcase. To be exact, the German sentence "Vielen Dank." is fed simultaneously into both the POS predictor and the NAG model, and then the POS predictor generates a POS sequence `JJ NNS PCT` which is further converted into a binarized mask matrix through a conversion dictionary. Meanwhile, the NAG model generates the primary probability distributions through a softmax layer. Here, from Figure 1, words "Many" and "you" get the highest probability, resulting in the mix mode "Many you" if following the primary distribution. To avoid such an undesirable result, our POSPD automatically adjusts the probability according to the binarized mask matrix. For example, the probability of "you" is adjusted to 0, since the POS tag of "you" is `PRP` rather than `NNS`. As a result, "Many Thanks." gets the highest probability hence to be generated as the output.

3.2 POSPD in Details

In this part, we detail the POSPD by introducing the conversion dictionary building, the workflow of POSPD, and the core module—the POS predictor.

Building a Conversion Dictionary The key idea of POSPD is filtering out the words that dissatisfy the prespecified POS sequence in the primary results of NAG. To implement our idea, we need a conversion dictionary $\mathcal{D}_c$ that contains the mapping from POS tags to words. Given a target vocabulary $\mathcal{V}_w$ with the length of $|\mathcal{V}_w|$ and a POS tag set $\mathcal{V}_s$, each key of $\mathcal{D}_c$ is a POS tag in $\mathcal{V}_s$ and the value is a set of words that can be assigned to this POS tag. It is worth noting that a word may have multiple POS tags. Therefore, one word may appear in multiple sets in $\mathcal{D}_c$.

The POSPD Workflow The workflow of POSPD is as follows: given a source sentence $\boldsymbol{x}$, POSPD feeds it into both the NAG model's encoder and the POS predictor. After that, the POS predictor outputs a POS sequence $\boldsymbol{s} = (s_1, s_2, ..., s_L)$ for the target sentence. Meanwhile, the decoder of the NAG model generates a preliminary distribution matrix $\boldsymbol{D} = (\boldsymbol{d_1}, \boldsymbol{d_2}, ..., \boldsymbol{d_L})$, where $\boldsymbol{d_i}$ represents the distribution of all words[2] in the i-th position. Note that, the sentence length follows the length of the predicted POS tag L.

For the ease of implementation, the POS sequence s is converted into a binarized mask matrix $\mathcal{M} = (\boldsymbol{m_1}, \boldsymbol{m_2}, ..., \boldsymbol{m_L})$. In details, for each POS tag s_i, the corresponding binarized vector is $\boldsymbol{m_i} = (m_i^1, m_i^2, ..., m_i^{|\mathcal{V}_w|})$ and the j-th position m_i^j is defined as:

$$m_i^j = \begin{cases} 1, & w_j \in \mathcal{D}_c^{s_i}; \\ 0, & w_j \notin \mathcal{D}_c^{s_i}, \end{cases} \tag{1}$$

where w_j is the j-th word token in $\mathcal{V}_w$. As a result, the POS sequence s is replaced by $\mathcal{M}$. Finally, we get the new generation results by:

$$\boldsymbol{y} = \arg\max(\mathcal{M} \cdot \boldsymbol{D}). \tag{2}$$

The POS Predictor As the core module of the POSPD, our POS predictor is dedicated to output the POS tag sequence of the target sentence when accepting the source sentence as the input. To train the POS predictor, we need to create a POS dataset where each sample is a pair consisting of a source sentence and a POS sequence of the target sentence[3]. As shown in Figure 3, the architecture of our POS predictor is a variant of the standard Transformer (Vaswani et al., 2017). As shown in the gray arrow flow, the main difference between our POS predictor and the vanilla Transformer is the layer number of encoder and decoder. To be specific, unlike the vanilla Transformer which contains six layers for both encoder and decoder, we use a multi-layer encoder and a one-layer decoder to reduce the inference time, because the complexity for decoding the POS sequence is much lower than that for the original sentence.

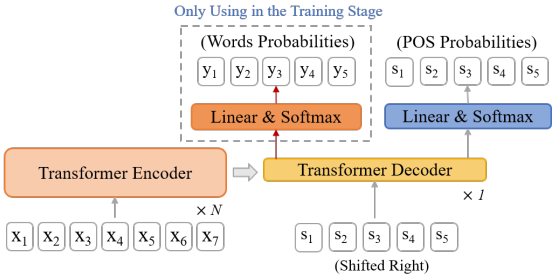

Figure 3: The overview of the POS predictor in POSPD. The linear layer (the red arrow points to) is only used in the training stage.

POS Predictor Optimization To optimize the POS predictor, we take a multi-task learning (Evgeniou and Pontil, 2004) paradigm to jointly decode the word sequence and POS sequence on the target side. The underlying hypothesis is that the target word sentence is highly related to the POS sequence. Given a source sentence $\boldsymbol{x}$, a POS sequence $\boldsymbol{s}$ and a target sentence $\boldsymbol{y} = (y_1, y_2, ..., y_L)$, the learning objective is then defined as the sum of the POS tagging loss (the first term) and the sentence prediction loss (the second term):

$$\mathcal{L} = \mathcal{L}_{pos} + \mathcal{L}_{word}, \tag{3}$$

where the POS sentence prediction loss can be written as:

$$\mathcal{L}_{pos} = \sum_{t=1}^{L} \log P(s_t | s_{<t}, \boldsymbol{x}), \tag{4}$$

and the target sentence prediction loss is:

$$\mathcal{L}_{word} = \sum_{t=1}^{L} \log P(y_t | s_{<t}, \boldsymbol{x}). \tag{5}$$

[2]The length of $\boldsymbol{d_i}$ is $|\mathcal{V}_w|$.

[3]We use NLTK POS tagger to create the POS sequence, which can be found at https://www.nltk.org/book/ch05.html.

In our method, the POS predictor uses an extra linear layer after the decoder to generate the target sentence, as shown in Figure 3. After training, we only need the POS predicting linear layer for inference, thus enjoying the better performance for the POS sequence prediction.

3.3 Training under the BPE Condition

Almost all NAG models use the Byte Pair Encoding (BPE) (Sennrich et al., 2016) technique to build the word vocabulary with subword-level tokens. However, these tokens cannot be tagged by the mainstream POS Taggers (Yarowsky and Ngai, 2001), which makes difficulties in building the POS dataset. To address this issue, we propose a simple but effective subword-level POS tagging method for our POS predictor. A simple example is demonstrated in Table 1, the NLTK toolkit tags the word "gutacht" as `NN` in the original sentence but cannot handle the BPE form "gut ##ach ##t". Intuitively, we can assign the BPE form to have the POS tag as "gutacht" (i.e. `NN NN NN`). However, this method increases the number of repeated tokens in generation sentences of NAG models and even worsens the performance. The possible reason is that the aforementioned method cannot explicitly distinguish whether a POS tag is associated with a BPE token or a complete word. In contrast, our method tags the BPE form as `NN1 NN2 NN3`. As a result, the conversion dictionary is more sparse while improving the mapping between the POS tag and the corresponding words. In addition, the word "question" is tagged as `NN`, since it doesn't have any sub-word tokens after the BPE.

Ori.	yesterday , gutacht' s mayor gave a clear answer to this question.
BPE	yesterday , gut ##ach ##t ' s mayor gave a clear answer to this question .
WP	`NN , NN SYM$ JJ NN VBD DT JJ NN TO DT NN .`
SWP	`NN PCT NN1 NN2 NN3 SYM$ JJ NN VBD DT JJ NN TO DT NN PCT`

Table 1: An example of the subword-level POS tagging method, where "WP" denotes the POS sequence generated by NLTK, and "SWP" is the "WP" of sub-word level. "##" denotes the subword token marker.

4 Experiments

In this section, we use multiple text generation datasets to comprehensively evaluate the effectiveness and efficiency of the proposed POSPD. For an extensive comparison, we compare our POSPD with the sequence-level knowledge distillation, and provide detailed analyzes in alleviating the multimodality problem and the time cost in dataset building.

4.1 Datasets

We conduct experiments on four widely-used benchmark datasets to evaluate POSPD: XSUM for text summarization, ROCStories corpus for story ending generation, SQuAD 1.1 for question generation, and WMT14 (DE-EN) for machine translation. Meanwhile, we use BERT-based BPE tokenizer[4] for all datasets. The details are as follows:

XSUM (Narayan et al., 2018) includes the 227k British Broadcasting Corporation (BBC) online articles and the corresponding single-sentence summaries. The average sentence lengths are 358.5 words for input and 21.1 words for output.

ROCStories Corpus[5] (Mostafazadeh et al., 2016) contains 98k five-sentence stories. For each story, we use the last sentence as the target output while the other four sentences as the source input. We randomly sample 90k/4k stories for training/validation, and the remaining 4160 for testing. The average sentence lengths are 39.64 words for input and 10.72 words for output.

SQuAD 1.1[6] (Rajpurkar et al., 2016) is a machine reading comprehension data set containing 98K passage-question-answer triples (Liu et al., 2020b). After processing, we obtain a question generation dataset. Following GLGE (Liu et al., 2020c), the input sentence is formatted as ⟨answer [SEP] passage⟩. The average sentence lengths are 149.4 words for input and 11.5 words for output.

WMT14 (DE-EN)[7] contains 4.5M translation pairs and 3k/3k pairs for validation/testing. The average sentence lengths are 25.07 words for input and 26.53 words for output.

4.2 Evaluation Metrics

Follow GLGE (Liu et al., 2020c), we use ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L) (Lin, 2004) as evaluation metrics for text summarization,

[4] https://pypi.org/project/transformers/.

[5] https://cs.rochester.edu/nlp/rocstories/

[6] https://rajpurkar.github.io/SQuAD-explorer/

[7] https://www.statmt.org/wmt14/translation-task.html

while BLEU-4 (B-4) (Papineni et al., 2002), Meteor (Denkowski and Lavie, 2014), and R-L are used in question generation and story ending generation. Meanwhile, BLEU-4 is also the evaluation metric for machine translation to keep in line with previous works (Gu et al., 2018).

4.3 Baselines and Comparison

In this work, we focus on using iteration-based NAG models as backbones, because they are one of the mainstream NAG structures in current works and perform competitively to AG models without any external system (Kasai et al., 2020b). Specifically, we use two representative iteration-based NAG models from recent work, i.e., CMLM (Ghazvininejad et al., 2019) and DisCo (Kasai et al., 2020a). The details are as follows:

CMLM The conditional masked language model randomly masks some target tokens and predicts them with the remaining ones. In inference, it masks several tokens with the lower "confidence" and retains other tokens with higher "confidence" during iterations, which is called mask-predict inference. Following Ghazvininejad et al. (2019), we use same settings for all generation tasks[8].

DisCo The disentangled context transformer aims to use different context information when predicting each token, being regarded as an effective improvement of CMLM. For better comparison, we also use mask-predict inference as same as CMLM. Meanwhile, we use the model settings described in Kasai et al. (2020a) for all generation tasks[9].

Knowledge Distillation Following Gu et al. (2018) which uses a standard transformer (Vaswani et al., 2017) as the teacher model to regenerate training set in the greedy method for NAG models (hereinafter described as "Transformer-1 (6-6)"), we report NAG models' performances on all text generation task when using the distilled training dataset. In the following discussion, the "Transformer-1" and "Transformer-4" denote the beam size of 1 and 4 in the beam search, respectively. Meanwhile, we also report the results of different Transformer model structures, where the "(6-6)" and "(12-1)" denote the version of six encoder layers, six decoder layers and the version of 12 encoder layers, one decoder layers, respectively.

[8] https://github.com/facebookresearch/Mask-Predict

[9] https://github.com/facebookresearch/DisCo

4.4 Experimental Settings

We follow the hyperparameters for standard Transformer in (Vaswani et al., 2017) for our POS predictor. One minor difference is the layers of encoder and decoder are set to 12 and 1 to make a fair comparison with AG models, respectively. All of the models are implemented based on Fairseq (Ott et al., 2019), and we follow the other specific parameter settings for both AG and NAG models in (Kasai et al., 2020b). In inference, the length beam, length penalty, and batch size are all set to 1 to calculate the main results (without any post-processing) and latency. The latency is calculated through using the built-in time statistics function in Fairseq, which is tested on a single NVIDIA Tesla P100 GPU to keep in line with previous works (Gu et al., 2018). Meanwhile, the beam size of our POS predictor is set to 5. For the number of iterations, we report the iterations when the NAG model results are converged. In practice, the iterations of two NAG models are 4, 3, 3 and 10 on XSUM, SQuAD1.1, ROCStories and WMT14 (DE-EN).

4.5 Main Results

We evaluate the performance of two NAG models (CMLM and DisCo) on four text generation datasets, and further provide the results when using sequence-level data distillation (i.e., "+Distill") and the POSPD (i.e., "+POSPD"), respectively. We report the main results in Table 2 and the inference time comparison in Table 3, from which we can make the following conclusions:

1. **POSPD consistently improve NAG models on four text generation dataset to a greater extent compared to knowledge distillation.** POSPD consistently improve NAG models on four text generation tasks while knowledge distillation may even degrade performances of the NAG models such as XSUM (row 5 vs. row 6) and SQuAD 1.1 (row 8 vs. row 9). More importantly, although the knowledge distillation improves NAG models by 1.04/1.56 (row 5 vs. row 6, row 8 vs. row 9) on BLEU-4 in WMT14 (DE-EN), POSPD still beats the knowledge distillation version by 0.24/0.19 (row 6 vs. row 7, row 9 vs. row 10) on BLEU-4.

2. **Knowledge distillation does not always improve the NAG model as the AG models may get worse performance than NAG.** In both text summarization (XSUM) and story ending generation (ROCStories) tasks, the two original NAG models CMLM and DisCo outperform the AG model.

Patterns	Models	XSUM	SQuAD 1.1	ROCStories	WMT14
	Metrics	R-1/R-2/R-L	B-4/Meteor/R-L		B-4
AG (row 1-4)	Transformer-1 (6-6)	19.53/3.38/15.36	3.87/9.73/29.34	1.89/8.70/23.98	31.61
	Transformer-1 (12-1)	17.51/2.63/14.18	2.84/7.78/26.58	1.03/6.99/20.46	27.25
	Transformer-4 (6-6)	22.98/5.88/18.56	4.69/9.95/29.76	2.45/8.67/23.85	33.07
	Transformer-4 (12-1)	17.69/2.72/14.30	3.55/7.73/28.15	1.52/7.26/20.66	28.28
NAG (row 5-10)	CMLM	24.95/5.07/19.73	3.49/10.68/30.48	1.61/9.24/25.01	26.48
	+Distill	20.22/3.49/16.29	3.03/9.13/28.91	0.30/5.14/16.58	27.28
	+POSPD	25.22/5.49/19.93	**4.29/11.00/30.66**	**1.79/9.37**/24.96	27.52
	DisCo	26.85/6.86/21.72	3.38/10.33/31.21	1.68/9.06/**25.10**	27.21
	+Distill	18.42/3.27/14.92	3.25/8.78/29.57	0.00/4.59/15.72	28.04
	+POSPD	**27.39/7.26/22.15**	4.20/10.80/30.59	1.72/9.25/25.07	**28.23**

Table 2: Results on four text generation datasets. Bold values represent the maximum values of each column in the NAG pattern.

Models	XSUM	SQuAD 1.1	ROCStories	WMT14 (DE-EN)
POSPD	105 (1.00×)	69 (1.00×)	66 (1.00×)	105 (1.00×)
CMLM	132 (0.79×)	110 (0.62×)	74 (0.89×)	172 (0.61×)
DisCo	107 (0.98×)	105 (0.66×)	76 (0.87×)	168 (0.63×)

Table 3: Inference speed (ms/sample) comparisons on four text generation datasets.

It is obvious that the adoption of sequence-level knowledge distillation limits the performance of NAG models in these case. More interestingly, in question generation, the AG model outperforms the NAG models with BLUE-4 by 0.4/0.5 (row 3 vs. row 5/row 8), but knowledge distillation degrades NAG models' performance with BLEU-4 by 0.46/0.13 (row 5 vs. row 6, row 8 vs. row 9).

3. **POSPD does not bring significant extra time in constraining NAG models' generation while decoding**. POSPD maintains its advantage in high-speed inference across all data sets. For example, on the dataset SQuAD 1.1, the inference latency of POSPD is much lower than NAG models (1.00× vs. 0.62×/0.66×). Meanwhile, on the WMT14 (DE-EN) that has the longest average length of the target sentence, POSPD still maintains its advantage in the inference speed. Therefore, our POSPD could constrain the NAG model with the negligible extra time, since POSPD and the NAG model predict sequences (i.e., POS sequence and target sentence) in parallel.

4.6 Further Discussions

There is a loose ending towards the discussion of our POSPD solution. In this section, we conduct discussions to shed light on other interesting properties of POSPD. The discussions are guided by the following three research questions:

Q1: How does POSPD alleviate the multimodality problem?

Q2: Is it time-consuming to build the POS dataset on the new task?

Q3: Does multi-tasking learning object help the POS tag prediction?

4.6.1 Discussion on Generated Results (Q1)

To further analyze the role of POSPD and the sequence-level knowledge distillation in alleviating the multimodality problem, we conduct further statistical analyses on the generated results of four datasets. Considering the multimodality problem usually manifests as repeating or missing tokens in the generation sentences, we use two indicators, i.e., the repetition rate and the total number of tokens, to quantify them separately. Concretely, we refer to a "single-token repeat" metric (Welleck et al., 2020) and define the repetition rate here as the percentage of the repeated times between two adjacent tokens in the total number of tokens in a sentence, and then average it over the dataset.

The results are shown in Table 4, from which we can see both knowledge distillation and POSPD can reduce the repetition rate in NAG models on four datasets, and they are more effective on XSUM datasets with longer sentences. While in token numbers, using knowledge distillation significantly reduces the number of tokens generated by NAG models on XSUM. In contrast, using POSPD remarkably make the length of generated sentences by NAG models close to the reference without increasing the repetition rate.

Models	SQuAD 1.1	XSUM	WMT14	ROCStories
Metrics	Repetition / Tokens			
Reference	≈0.0/140786	≈0.0/275003	0.01/67617	≈0.0/44731
CMLM	0.09/-11036	0.15/-2616	0.01/-1957	0.06/+69
+Distill	0.05/-25418	0.06/-52643	0.03/+1214	0.03/+20058
+POSPD	0.09/+247	0.07/+4570	0.01/+1247	0.05/+1297
DisCo	0.05/-24364	0.14/-4641	0.01/-1957	0.06/+2234
+Distill	0.06/-30723	0.06/-52643	0.01/-2026	0.02/+9020
+POSPD	0.02/+1945	0.09/-10871	0.01/+1257	0.05/+1408

Table 4: Statistical analysis of NAG models' generations. "Reference" denotes the target sentence's reference. "Repetition" and "Tokens" represent the repetition rate and tokens number gap between reference and model outputs, respectively.

4.6.2 Time Cost in Building Datasets (Q2)

Considering that both POSPD and knowledge distillation require the processing of the training dataset when it comes to a new task/dataset (i.e., building the POS data set for POSPD / regenerating the training set for knowledge distillation), we further analyze the time consumption of the two processing steps. As shown in Table 5, POSPD has a significant advantage over knowledge distillation in the time consuming of dataset building. Especially on the larger dataset WMT14 (DE-EN), it can save even more time in building datasets, which is beneficial for rapid deployment on new tasks.

Method	Distill	POSPD	Samples
SQuAD 1.1	1086	45 (24.1×)	75k
ROCStories	402	49 (8.2×)	90k
XSUM	1850	240 (7.71×)	200k
WMT14	44258	5220 (8.48×)	4500k

Table 5: Time cost (seconds) comparisons in rebuilding datasets on four datasets. The batch size are 100 and 1 for the knowledge distillation and POSPD.

Models	SQuAD 1.1	XSUM
Metrics	B-4/Meteor/R-L	R-1/R-2/R-L
CMLM	3.49/10.68/30.48	24.95/5.07/19.73
MT w/o	4.18/10.80/31.04	25.12/5.24/19.91
MT w/	4.29/11.00/30.66	25.22/5.49/19.93
DisCo	3.38/10.33/31.21	26.85/6.86/21.72
MT w/o	4.17/10.56/31.05	27.00/6.89/21.81
MT w/	4.20/10.80/30.59	27.39/7.26/22.15

Table 6: The ablation study on using multi-task learning strategy in POSPD's training stage. "MT w/o" and "MT w/" denote training the POS predictor in POSPD with/without the multi-tasking learning, respectively.

4.6.3 Multi-task Learning Strategy (Q3)

In this part, we analyze the impact of using a multi-task learning strategy in POSPD's training stage. For lack of space, we take the ablation study on two datasets of different sizes, i.e., SQuAD 1.1 and XSUM. The results are shown in Table 6. Interestingly, predicting the POS sequence directly from the original sentence (i.e., "POSPD w/o") can also improve the performance of the NAG models. More importantly, multi-task learning strategy can improve the performance of POSPD in two datasets with a tiny increase in model parameters (only one linear layer). Meanwhile, it is only used during the POSPD's training stage and does not affect the inference time of POSPD.

5 Conclusion

In this paper, we revisit the role of the knowledge distillation in alleviating the multimodality problem of NAG. In brief, we experimentally reflect that the basic assumption of these knowledge distillation methods, the AG model is superior to NAG model, does not always hold for all text generation tasks. To alleviate the multimodality problem, we show a different solution by incorporating linguistic structure into NAG. Extensive experiments demonstrate that our POSPD significantly and consistently improves the NAG models in effectiveness and computational efficacy.

As we tentatively give a successful implementation of leveraging one of the simplest linguistic structures to benefit the NAG models in inference, such paradigm deserves a closer and more detailed exploration. Thus in the future, we will investigate to make the NAG models enjoy the benefits of incorporating diverse and abundant linguistic structures in a more superior way. In addition, our

experimental results suggest that future work might need to consider wider ranges of generation tasks instead of only machine translation when assessing the performance of NAG models.

Acknowledgments

This work was supported in part by the National Key RD Program of China under Grant 2020YFB1406702, in part by NFSC under Grant 61625204, 61836006, and the Science and Technology Major Project of Sichuan province under Grant 2020YFG0478.

References

Douglass Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Third Conference on Applied Natural Language Processing*, pages 133–140.

Michael J. Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT@ACL 2014, June 26-27, 2014, Baltimore, Maryland, USA*, pages 376–380. The Association for Computer Linguistics.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 72–78. Association for Computational Linguistics.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117.

Xiaocheng Feng, Zhangyin Feng, Wanlong Zhao, Bing Qin, and Ting Liu. 2020. Enhanced neural machine translation by joint decoding with word and pos-tagging sequences. *Mob. Networks Appl.*, 25(5):1722–1728.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020. Aligned cross entropy for non-autoregressive machine translation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, 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 2019, Hong Kong, China, November 3-7, 2019*, pages 6111–6120. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 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.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020a. Non-autoregressive machine translation with disentangled context transformer. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5144–5155. PMLR.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. 2020b. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation. *CoRR*, abs/2006.10369.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1317–1327. The 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 2019, Hong Kong, China, November 3-7, 2019*, pages 5707–5712. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Dayiheng Liu, Jie Fu, Pengfei Liu, and Jiancheng Lv. 2019. Tigs: An inference algorithm for text infilling with gradient search. In *ACL*.

Dayiheng Liu, Jie Fu, Yidan Zhang, Chris Pal, and Jiancheng Lv. 2020a. Revision in continuous space: Unsupervised text style transfer without adversarial learning. In *AAAI*.

Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Jiancheng Lv, Nan Duan, and Ming Zhou. 2020b. Tell me how to ask again: Question data augmentation with controllable rewriting in continuous space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5798–5810. Association for Computational Linguistics.

Dayiheng Liu, Yu Yan, Yeyun Gong, Weizhen Qi, Hang Zhang, Jian Jiao, Weizhu Chen, Jie Fu, Linjun Shou, Ming Gong, Pengcheng Wang, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, Ruofei Zhang, Winnie Wu, Ming Zhou, and Nan Duan. 2020c. GLGE: A new general language generation evaluation benchmark. *CoRR*, abs/2011.11928.

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

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia. Association for Computational Linguistics.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard H. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 4281–4291. Association for Computational Linguistics.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 839–849. The Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1797–1807. 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 NAACL-HLT 2019: Demonstrations*.

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, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2020. Learning to recover from multi-modality errors for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3059–3069. Association for Computational Linguistics.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1098–1108. 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, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhi-Hong Deng. 2019. Fast structured decoding for sequence models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3011–3020.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. 2020. Lexically constrained neural machine translation with levenshtein transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3536–3543. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 698–707. Association for Computational Linguistics.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Chunting Zhou, Graham Neubig, and Jiatao Gu. 2019. Understanding knowledge distillation in non-autoregressive machine translation. *CoRR*, abs/1911.02727.