

Non-Autoregressive Text Generation with Pre-trained Language Models

Yixuan Su[◇] Deng Cai[♡] Yan Wang[♣] David Vandyke[♣]

Simon Baker[◇] Piji Li[♣] Nigel Collier[◇]

[◇]Language Technology Lab, University of Cambridge

[♡]The Chinese University of Hong Kong

[♣]Tencent AI Lab

[♣]Apple

{ys484, sb895, nhc30}@cam.ac.uk

thisisjcykcd@gmail.com, dvandyke@apple.com

{brandenwang, pijili}@tencent.com

Abstract

Non-autoregressive generation (NAG) has recently attracted great attention due to its fast inference speed. However, the generation quality of existing NAG models still lags behind their autoregressive counterparts. In this work, we show that BERT can be employed as the backbone of a NAG model to greatly improve performance. Additionally, we devise mechanisms to alleviate the two common problems of vanilla NAG models: the inflexibility of prefixed output length and the conditional independence of individual token predictions. Lastly, to further increase the speed advantage of the proposed model, we propose a new decoding strategy, *ratio-first*, for applications where the output lengths can be approximately estimated beforehand. For a comprehensive evaluation, we test the proposed model on three text generation tasks, including text summarization, sentence compression and machine translation. Experimental results show that our model significantly outperforms existing non-autoregressive baselines and achieves competitive performance with many strong autoregressive models. In addition, we also conduct extensive analysis experiments to reveal the effect of each proposed component.¹

1 Introduction

Autoregressive generation (AG) models achieve state-of-the-art performance on a wide range of text generation tasks, such as machine translation (Vaswani et al., 2017) and text summarization (Rush et al., 2015). Such models generate a token sequence in a left-to-right, token-by-token fashion. The prediction for the next token is conditioned on all previously generated tokens. This characteristic makes it impossible to parallelize the computational overhead for token predictions in different

positions, which leads to a relatively high latency in inference. On the other hand, non-autoregressive generation (NAG) models (Gu et al., 2018) have emerged as a promising alternative due to their fast inference speed. NAG models omit the sequential dependencies within the output-side sequence and predict tokens in all positions simultaneously once the output length has been determined beforehand. While NAG models enjoy full parallelism and faster inference, the generation quality of NAG models often lags behind their autoregressive counterparts.

In this work, we explore the potential of large-scale pre-trained language models for improving the performance of non-autoregressive generation. Specifically, we utilize BERT (Devlin et al., 2019) as the backbone for NAG modelling and extend the architecture of BERT with a CRF output layer (Lafferty et al., 2001; Sun et al., 2019) for better capturing the output-side dependencies.

In addition, we analyze two significant limitations that NAG models currently suffer from: (1) the inflexibility of prefixed output length, and (2) the conditional independence of individual token predictions. Accordingly, we devise two solutions to these two problems.

First, prior NAG models require the output length to be determined before token generation, thus an extra module for output length prediction is always required. Nevertheless, the most likely length from the prediction module is not necessarily the best-suited one for the token generation model. To this end, previous works (Gu et al., 2018; Ma et al., 2019) usually rely on length-parallel decoding (LPD) (Wei et al., 2019) for performance enhancement; that is, generating and re-ranking the results from different output length candidates. In this work, we propose a simple and elegant decoding mechanism that lets the model determine the output length on-the-fly. Specifically, our model dynamically adjusts the output sequence length via

¹All related code, data, and models can be found in <https://github.com/yxuansu/NAG-BERT>.

emitting an `[eos]` token at any output position to indicate the ending of the generated sequence. Therefore, we can avoid the additional efforts of output length prediction and results re-ranking.

Second, most existing NAG models assume the token predictions in different positions are conditionally independent. As a consequence, they often tend to generate results that are ungrammatical with repetitions (Wang et al., 2019b). To alleviate this problem, we propose a context-aware learning objective which impels the model to output different tokens at adjacent positions, thereby reducing the possibility of repetitive generation.

Furthermore, for tasks like text summarization, the output sequence (summary) is known to be shorter than the source sequence (article). In such cases, to further improve the model’s inference efficiency, we introduce a new *ratio-first* decoding strategy. Specifically, instead of performing inference on all source-side hidden states, ratio-first generates the result only based on a subset of source hidden states. The subset size is jointly determined by the source length T and a predefined ratio α that is set based on our prior knowledge from the data statistics. In the experiments, we show that ratio-first can significantly improve the inference speed while maintaining the generation quality.

We evaluate the proposed model on three typical text generation tasks, including text summarization, sentence compression and machine translation. Experimental results show that our model significantly outperforms many strong non-autoregressive baselines, and even performs competitively with several strong autoregressive models. In addition, we conduct extensive analysis experiments to study the effect of individual proposed components.

In summary, our contributions are: (1) We propose a novel framework that utilizes BERT for text generation under the non-autoregressive generation paradigm; (2) We propose a decoding mechanism that allows the model to dynamically determine the output length, and a new context-aware learning objective that reduces errors stemming from the output-side conditional independence assumption; (3) We introduce a ratio-first decoding strategy that further improve the model’s inference efficiency.

2 Background

Autoregressive generation (AG) models generate sequences based on a left-to-right factorization. As shown in Figure 1, given the source sequence \mathbf{X} ,

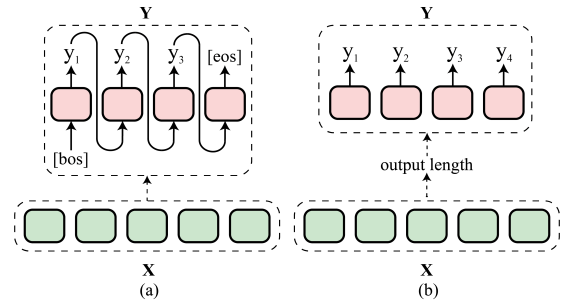


Figure 1: (a) Autoregressive; (b) Non-Autoregressive

the target sequence \mathbf{Y} with length T' is generated via a chain of conditional probabilities based on the left-to-right sequential dependencies as:

$$p(\mathbf{Y}|\mathbf{X}) = \prod_{i=1}^{T'} p(y_i|y_{<i}, \mathbf{X}), \quad (1)$$

where $y_{<i}$ denotes the tokens before the i -th step. This property of autoregressive factorization makes the generation process hard to be parallelized as the result is generated token by token.

Unlike AG models, non-autoregressive (NAG) models generate sequences without modelling the output-side dependencies. As shown in Figure 1, given the prespecified output length T' , the probability of the target sequence \mathbf{Y} is then modelled as:

$$p(\mathbf{Y}|\mathbf{X}) = \prod_{i=1}^{T'} p(y_i|\mathbf{X}, i, T'). \quad (2)$$

With this conditional independence assumption, NAG models can fully parallelize their generation process, which significantly improves the inference speed. However, it has been shown that, the choice of the prespecified output length has a notable impact on the model’s generation quality (Gu et al., 2018). In addition, the removal of output-side sequential dependency also causes the generation quality of NAG models to be inferior to their autoregressive counterparts (Wang et al., 2019b).

3 Proposed Model

In this section, we give a detailed explanation of the proposed model. First, we describe how to utilize BERT as a non-autoregressive generation model. Then we discuss the decoding mechanism which allows the model to determine the output length dynamically. Finally, we introduce the new ratio-first decoding strategy which further improves the model’s decoding efficiency.

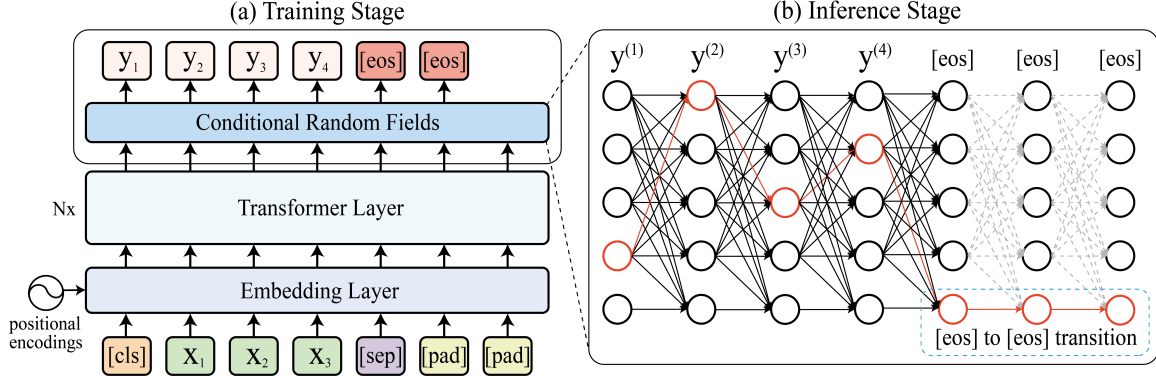


Figure 2: The overall illustration of the proposed model: During training, the model parameters are only updated on the positions of the target sequence. During inference, once the decoded trajectory (colored in red) gets into the $[\text{eos}]$ state, it will only transit to the $[\text{eos}]$ state in the remaining steps. The final result is obtained by removing the generated $[\text{eos}]$ tokens from the entire decoded trajectory.

3.1 Model Architecture

The architecture of the proposed model is presented in Figure 2, in which the embedding layer and the stack of transformer layers are initialized with BERT (Devlin et al., 2019).

Input Representation Following the setup of BERT, we first append a $[\text{cls}]$ and a $[\text{sep}]$ token on both sides of the source sequence. Then we attach a number of $[\text{pad}]$ tokens at the end of source sequence to make its length equal to the predefined maximum size (e.g., 256). Thus we can make sure the source length is longer than or equal to the output length. As a special case, for tasks like text summarization where the source is known to be longer than the target, we do not attach the $[\text{pad}]$ tokens when constructing the input.

Transformer Layers Given the source sequence \mathbf{X} , it is processed by a stack of N transformer (Vaswani et al., 2017) layers. Formally, the Multi-Head Attention is defined as $\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$, where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ denotes the query, key and value respectively. The computation of the first transformer layer is then defined as:

$$\mathbf{V}^{(1)} = \text{MultiHead}(E(\mathbf{X}), E(\mathbf{X}), E(\mathbf{X})), \quad (3)$$

$$\mathbf{O}^{(1)} = \text{FFN}(\mathbf{V}^{(1)}), \quad (4)$$

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2, \quad (5)$$

where $E(\mathbf{X}) = TE(\mathbf{X}) + PE(\mathbf{X})$ in which $TE(\cdot)$ denotes the token embedding and $PE(\cdot)$ denotes the position embedding. For other layers:

$$\mathbf{V}^{(n)} = \text{MultiHead}(\mathbf{O}^{(n-1)}, \mathbf{O}^{(n-1)}, \mathbf{O}^{(n-1)}), \quad (6)$$

$$\mathbf{O}^{(n)} = \text{FFN}(\mathbf{V}^{(n)}), \quad (7)$$

where $n = 2, \dots, N$ and N is the total number of transformer layers. The final sequence representation $\mathbf{H} \in \mathbb{R}^{T \times d_{\text{model}}}$ is the output states of BERT from the last layer, where T is the source sequence length and d_{model} is the model size.

CRF Layer Then, \mathbf{H} is passed through a linear-chain CRF (Lafferty et al., 2001). Under the CRF framework, the likelihood of the target sequence \mathbf{Y} with length T' is then modelled as:

$$\begin{aligned} P_{\text{CRF}}(\mathbf{Y}|\mathbf{X}) &= \frac{e^{\mathcal{S}(\mathbf{X}, \mathbf{Y})}}{\sum_{\mathbf{Y}'} e^{\mathcal{S}(\mathbf{X}, \mathbf{Y}')}} \\ &= \frac{1}{Z(\mathbf{X})} \exp\left(\sum_{i=1}^{T'} \Phi_{y_i}(h_i) + \sum_{i=2}^{T'} t(y_{i-1}, y_i)\right), \end{aligned} \quad (8)$$

where $Z(\mathbf{X})$ is the normalizing factor and $\Phi_{y_i}(h_i)$ denotes the label score of y_i at position i . In practice, Φ is parameterized by a neural network that maps the BERT output state h_i into the label (vocabulary) space. The $t(y_{i-1}, y_i) = \mathbf{T}_{y_{i-1}, y_i}$ denotes the transition score from label y_{i-1} to y_i where $\mathbf{T} \in \mathbb{R}^{|V| \times |V|}$ is the transition matrix.

Approximation In the context of text generation, the size of the label space (vocabulary size) $|V|$ is typically large, e.g., 32k. Therefore, it is intractable to directly model the transition matrix \mathbf{T} and the normalizing factor $Z(\mathbf{X})$. To this end, we adopt the techniques proposed by Sun et al. (2019) to approximate these two terms. Specifically, the full transition matrix is approximated by the product of two low-rank matrices $\mathbf{T} = \mathbf{E}_1 \mathbf{E}_2^T$, where $\mathbf{E}_1, \mathbf{E}_2 \in \mathbb{R}^{|V| \times d}$ and d is much smaller than $|V|$. To compute the normalizing factor $Z(\mathbf{X})$, at each

time step, instead of searching through all possible paths, the number of candidates is heuristically truncated to a predefined beam size k . We refer readers to the original paper for further details.

3.2 Output Length Determination

In this section, we describe how to let the model determine the output sequence length by itself. Our basic idea is that we want the model to dynamically stop generation via emitting a special `[eos]` token. To achieve this, during training, we manually append **two** consecutive `[eos]` tokens to the end of the target sequence, as shown in the top left part of Figure 2. In this way, the model can learn a deterministic transition behaviour between two `[eos]` states, meaning that $t([\text{eos}], [\text{eos}]) = \max_{v \in \mathcal{Y}} t([\text{eos}], v)$. This is because, during training, the model never sees a transition $([\text{eos}], v)$, where $v \neq [\text{eos}]$.

During inference, the result $\tilde{\mathbf{Y}}$ is acquired as $\tilde{\mathbf{Y}} = \arg \max_{\mathbf{Y}'} S(\mathbf{X}, \mathbf{Y}')$, where the CRF scoring function $S(\mathbf{X}, \mathbf{Y}')$ in Equation (8) can be decomposed as:

$$\begin{aligned} S(\mathbf{X}, \mathbf{Y}') &= \sum_{i=1}^T \Phi_{y'_i}(h_i) + \sum_{i=2}^T t(y'_{i-1}, y'_i) \\ &= \underbrace{\Phi_{y'_1}(h_1)}_{\text{initial state}} + \sum_{i=2}^T \underbrace{\{\Phi_{y'_i}(h_i) + t(y'_{i-1}, y'_i)\}}_{\text{state transition}}. \end{aligned} \quad (9)$$

Once the decoded trajectory enters the `[eos]` state, the state transition term in $S(\mathbf{X}, \mathbf{Y}')$ will be dominated by the transition score term $t([\text{eos}], [\text{eos}])$. As a result, the model will keep transitioning to `[eos]` in the remaining steps. An example is provided in the right part of Figure 2, from which we can see that, at step 5, the decoded trajectory enters the `[eos]` state and remains at it in the rest of the generation process. In this way, our model can dynamically control the length of output sequence by entering the `[eos]` state during the generation process. After the entire generation process is completed, the final output sequence can be obtained by removing all generated `[eos]` tokens.

3.3 Ratio-First Decoding

We note that the outputs of BERT can be divided into two subsets. The first subset ranges from the beginning to the position where the first `[eos]` is emitted, and the second subset is the rest. For example, in Figure 2, the first subset are those corresponding to the output sequence “ $y^{(1)} y^{(2)} y^{(3)}$

$y^{(4)} [\text{eos}]$ ”. As for the second part, we can see that it has little effect on the final output and removing it should not change the result. This indicates that it suffices to only consider the beginning part of BERT outputs for improving the inference speed. Especially, for tasks like summarization where the target is known to be shorter than the source sequence, we are safe to only use the first $\lceil \alpha \cdot T \rceil$ outputs of BERT to perform inference. Here T denotes the source length, $\alpha \in (0.0, 1.0)$ is set based on the data statistics and $\lceil \cdot \rceil$ is the integer rounding operation. Formally, given the source sequence \mathbf{X} , the ratio-first decoding is defined as

$$\begin{aligned} \tilde{\mathbf{Y}} &= \arg \max_{\mathbf{Y}'} \mathcal{F}(\mathbf{X}, \mathbf{Y}', \alpha), \\ &= \arg \max_{\mathbf{Y}'} \left\{ \sum_{i=1}^{\lceil \alpha \cdot T \rceil} \Phi_{y'_i}(h_i) + \sum_{i=2}^{\lceil \alpha \cdot T \rceil} t(y'_{i-1}, y'_i) \right\}. \end{aligned} \quad (10)$$

When $\alpha = 1.0$, ratio-first degenerates to the standard decoding strategy in CRF-based models.

It should be noted that, $\lceil \alpha \cdot T \rceil$ only constrains the maximum length of the generated result, and the actual output length (after removing the generated `[eos]` tokens) is still decided by the model itself. In the experiment section, we demonstrate that ratio-first can notably improve the inference speed whilst maintaining the generation quality.

4 Learning

Due to the conditional independence approximation on output tokens, NAG models often tend to generate repeated tokens (Wang et al., 2019b). One way to alleviate this problem is to introduce implicit dependencies on the output side. In this work, we propose to use the unlikelihood formulation of Welleck et al. (2020) in the context of NAG, where we define the set of negative candidate as the surrounding tokens within a predefined context window c . Formally, given the source sequence \mathbf{X} and the target sequence \mathbf{Y} with length T' , the proposed context-aware objective is defined as:

$$\begin{aligned} \mathcal{L}_{\text{CA}}(\mathbf{Y}|\mathbf{X}) &= - \sum_{i=1}^{T'} \{ \log p_{\theta}(y_i|h_i; \mathbf{X}) + l_{\text{CA}}(i) \}, \\ l_{\text{CA}}(i) &= \sum_{j=i-c, y_j \neq y_i}^{j=i+c} \log(1.0 - p_{\theta}(y_j|h_i; \mathbf{X})), \end{aligned} \quad (11)$$

where h_i is the model output state at position i . At position i , the proposed objective maximizes the probability of token y_i while minimizing the probabilities of the surrounding tokens. In this way, it discourages the model from generating repetitive tokens at different time steps.

The overall learning objective is then defined as

$$\begin{aligned}\mathcal{L}_{\text{CRF}} &= -\log P_{\text{CRF}}(\mathbf{Y}|\mathbf{X}), \\ \mathcal{L} &= \mathcal{L}_{\text{CRF}} + \lambda \cdot \mathcal{L}_{\text{CA}},\end{aligned}\quad (12)$$

where λ controls the importance of different loss terms and $P_{\text{CRF}}(\mathbf{Y}|\mathbf{X})$ is described in Equation (8).

5 Related Work

Non-Autoregressive generation was first introduced by Gu et al. (2018) to reduce the inference latency in machine translation. Recent works in this area have investigated ways to mitigate the trade-off between the decoding speed and generation quality. Gu et al. (2018) utilized fertility as latent variables for better translation performance. Wang et al. (2019b) proposed two auxiliary objectives for better modelling the output states and solving the under-translation problem. To better model the intermediate alignments between source and target sides, Ma et al. (2019) proposed a model based on the generative flow framework. Ghazvininejad et al. (2019) proposed to use a masked language objective to train the NAG model. During inference, starting from a fully masked sequence, the output is generated in an iterative refinement manner. Recently, Sun et al. (2019) proposed to incorporate a conditional random field into the decoder of a NAG model for better modelling the output-side dependencies. Our work is different from prior works in two aspects: (1) we directly utilize a pre-trained language model (BERT) to perform non-autoregressive generation; (2) our model can dynamically generate the output sequence without the need of prespecified output length.

6 Experiments

We evaluate the proposed model on three typical text generation tasks: (1) text summarization; (2) sentence compression and (3) machine translation.

6.1 Experimental Setup

We implement the proposed model with PyTorch (Paszke et al., 2017). The BERT model we use is the Huggingface implementation (Wolf et al., 2019) (bert-base-uncased). To approximate the transition

matrix in the CRF layer, we set the dimension d of matrices \mathbf{E}_1 and \mathbf{E}_2 as 32. For the normalizing factor $\mathbf{Z}(\mathbf{X})$, we set the predefined beam size k as 256. As for the overall learning objective, we set the window size c as 3 and λ as 1.0. In training, we use Adam optimizer (Kingma and Ba, 2015). To measure the relative speedup, we follow the standard setup which runs inference for each individual example separately. The model’s inference speed is computed by averaging the results of test cases. For a fair comparison, we measure the inference speed of all models on the same platform.

6.2 Text Summarization

Text summarization aims to automatically generate a compact summary that retains the most important content of the original text document (Nenkova and McKeown, 2012). In this experiment, we use the Gigawords dataset (Rush et al., 2015) as our benchmark. For evaluation, standard metrics including ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-L (R-L) (Lin, 2004) are reported.

We compare our model with several representative and the latest NAG models, including NAG-NMT (Gu et al., 2018), NAR-REG (Wang et al., 2019b) and NAG-CRF (Sun et al., 2019). Following previous works, during training, we train a length predictor to predict the output length. During inference, for each NAG baseline, we adopt the length-parallel decoding strategy (LPD- k) (Wei et al., 2019), that is, generating k results using the top- k possible output length predictions from the length predictor. The results are then re-ranked by a transformer model to get the final output. In the experiment, we report the results of different NAG baselines using LPD-9 decoding. In addition, to better examine the effect of using BERT in NAG models, we add a **BNAG-CRF** baseline which adopts the same structure of the NAG-CRF model but using BERT as the encoder. We also compare our model with several strong autoregressive models, which are Luong-NMT (Luong et al., 2015), Pointer-Generator (See et al., 2017), DRGD (Li et al., 2017) and Concept Pointer (Wang et al., 2019a). To measure the relative inference speedup, we include transformer as a baseline model.

The results are shown in Table 1, from which we can see that, by using length-parallel decoding, the performance of all NAG baselines can be notably improved. However, such procedure significantly increases the inference latency. In contrast,

Models	R-1	R-2	R-L	Speedup
Autoregressive				
Luong-NMT	33.10	14.45	30.71	-
Pointer-Generator	35.98	15.99	33.33	-
DRGD	36.25	17.61	33.55	-
Concept Pointer	36.62	16.40	33.98	-
Transformer (b = 4)	35.74	16.97	33.43	1.00×
Non-Autoregressive				
NAG-NMT	27.20	8.96	25.58	9.31 ×
+LPD-9	29.76	10.03	28.04	5.28×
NAR-REG	28.56	9.79	26.83	8.64×
+LPD-9	31.23	11.14	29.55	4.74×
NAG-CRF	30.29	12.61	28.71	8.07×
+LPD-9	32.91	14.31	31.03	4.32×
BNAG-CRF	32.63	14.32	30.82	6.13×
+LPD-9	34.56	16.10	32.76	3.21×
Ours ($\alpha = 0.3$)	34.67	16.13	32.81	9.31 ×
Ours ($\alpha = 1.0$)	35.05	16.48	33.28	6.72×

Table 1: Results on Gigawords dataset, where b in the transformer baseline stands for beam search size.

our model can self-determine the output length without any re-ranking process. As shown in the results, our model outperforms the best NAG baseline (with LPD) and achieves performances that are comparable with several strong AG models.

Comparing the results of **BNAG-CRF** and NAG-CRF, we can see that incorporating BERT as encoder helps to improve the model performance. Nonetheless, our model still outperforms **BNAG-CRF** with LPD-9 decoding. This is because the dynamic length decoding mechanism allows our model to generate results with optimal length, leading to stronger model performances.

Finally, we analyze the proposed ratio-first decoding. From the results, we observe a moderate performance drop when using ratio-first ($\alpha = 0.3$). It comes from the fact that, for some input documents with length T , the reference summary is longer than $[\alpha \cdot T]$. In such cases, ratio-first fails to generate the complete reference summary, leading to the drop of performance. On the other hand, we can see that, ratio-first can notably improve the inference speedup. With $\alpha = 0.3$, our model achieves the highest inference speedup while still outperforms all compared NAG models.

6.3 Sentence Compression

Sentence compression aims at compressing a long sentence into a short one by deleting redundant words. In this experiment, we use the Google sentence compression dataset (Filippova and Altun, 2013) as our benchmark. For evaluation, we use

Models	F1	R-1	R-2	R-L	Speedup
Autoregressive					
Bi-LSTM-Dep	82.3	81.5	74.1	81.3	-
Tagger	82.8	81.1	72.4	80.9	-
Tagger+ILP	79.0	76.1	64.6	75.8	-
HiSAN-Dep	82.7	82.1	74.9	81.9	-
HiSAN	83.2	82.9	75.8	82.7	-
Transformer (b = 4)	82.4	82.0	74.6	81.8	1.00×
Non-Autoregressive					
NAG-NMT	72.5	72.1	59.9	71.8	10.71 ×
+LPD-9	73.8	73.6	61.0	73.1	6.09×
NAG-REG	73.7	73.1	61.5	73.0	10.00×
+LPD-9	75.6	75.1	63.4	74.9	5.49×
NAG-CRF	75.1	74.4	66.8	74.2	9.41×
+LPD-9	77.3	76.5	69.0	76.3	5.04×
BNAG-CRF	77.1	76.2	68.9	76.0	7.21×
+LPD-9	79.3	78.5	71.7	78.2	3.91×
Ours ($\alpha = 0.7$)	79.5	79.0	72.1	78.7	10.00×
Ours ($\alpha = 1.0$)	80.7	80.3	73.6	80.1	8.42×

Table 2: Results on sentence compression task

the standard token-kept-F1 (F1) score. In addition, We also report the results of other standard metrics including ROUGE-1, ROUGE-2 and ROUGE-L.

We compare the proposed model with the same NAG baselines as in the previous experiment. We also compare our model with several strong autoregressive models, including Bi-LSTM-Dep (Filippova et al., 2015), Tagger and Tagger+ILP (Wang et al., 2017), HiSAN-Dep and HiSAN (Kamigaito et al., 2018). To measure the inference speedup, we include transformer as a baseline model.

The results are presented in Table 2, from which we see that our model outperforms the best reported NAG baseline (with LPD) in terms of both the generation quality and inference speed. Comparing with the strong autoregressive models, our model can achieve competitive performance with a over 8.42× inference speed up. We also report the results of our model using the ratio-first decoding strategy. By setting α as 0.7, it achieves a 10.00× inference speedup while still outperforming other compared NAG baselines.

6.4 Machine Translation

Machine translation aims at translating text from the source language to the target language. In this task, we use the IWSLT14 German-to-English (DE-EN) dataset as our benchmark. Following previous works, we use the sequence-level knowledge distillation (Gu et al., 2018) during training. For evaluation, we report results in BLEU scores (Papineni et al., 2002). In this experiment, we use the BERT model in German language.

We compare our model with a range of strong

Models	BLEU	Speedup(\times)
Autoregressive		
LSTM-based	28.53	-
CNN-based	32.84	-
Transformer (b = 4)	33.31	1.00
Non-Autoregressive		
ENAG-E	24.13 (27.30)	15.08 (7.39)
ENAG-P	25.09 (28.60)	14.48 (7.24)
NAG-REG	23.89 (28.04)	16.45 (9.05)
NAG-NMT	23.04 (26.79)	13.92 (7.24)
NAG-CRF	26.39 (29.21)	11.74 (6.03)
BNAG-CRF	26.73 (29.67)	9.42 (5.01)
Ours ($\alpha = 0.8$)	29.71	13.92
Ours ($\alpha = 1.0$)	30.45	11.31

Table 3: Results on IWSLT14 De-En dataset. The numbers in () are results using length-parallel decoding.

BERT	CRF	R-1	R-2	R-L
✓	✓	35.05	16.48	33.28
×	✓	32.41	14.19	30.53
✓	×	32.16	11.33	30.34
×	×	27.02	8.81	25.25

Table 4: Ablation study on Gigawords dataset.

NAG models, including NAG-NMT (Gu et al., 2018), ENAG-E and ENAG-P (Guo et al., 2019), NAG-REG (Wang et al., 2019b), NAG-CRF (Sun et al., 2019) and **BNAG-CRF**. For each NAG baseline, we also report the results using LPD-9 decoding. In addition, we compare our model with several strong autoregressive models, including LSTM-based (Wu et al., 2016), CNN-based (Gehring et al., 2017) and transformer model.

The results are shown in Table 3, from which we see that our model outperforms the best NAG baseline (with LPD) in terms of both the generation quality and inference speedup. Additionally, we also report the results using the ratio-first decoding. By setting α as 0.8, the inference speedup can be further boosted to $13.92\times$ while the generation quality is still higher than the best NAG baseline.

6.5 Further Analysis

In this section, we present further discussions and empirical analysis of the proposed model.

BERT & CRF To quantify the importance of each component (BERT & CRF) of our model, we evaluate the performance on Gigawords dataset by removing each component iteratively.

The results are shown in Table 4, from which we can see that by removing any of these compo-

Models	rep-1	rep-2	rep-3	rep-4	R-L
w/o CA	6.897	2.640	0.741	0.295	32.89
Ours	5.786	1.978	0.427	0.106	33.28
Transformer	4.329	1.348	0.267	0.089	33.43

Table 5: Evaluation results on n -gram repetitions.

nents, the overall performance decreases. By removing BERT from the model, we observe notable drop across all metrics. This shows that the knowledge of BERT is an important factor of the model’s strong performance. Comparing with results in Table 1, it still outperforms vanilla NAG-CRF and performs comparably with NAG-CRF using LPD decoding, which demonstrates the merit of the proposed dynamic length decoding mechanism. Another interesting finding is that, by only removing the CRF layer, the most notable drop is observed on the bigram-level metric (ROUGE-2). This shows that the bigram-level dependencies on the output side are mainly captured by the CRF module. In addition, by removing both BERT and CRF, all metrics further decrease. This confirms that each of these two components positively contributes to the model’s overall performance.

Context-Aware Objective In this part, we study the effect of the context-aware objective. As described in Equation (11), it aims at alleviating the problem of repetitive generation. To give a quantitative analysis, we use the measurement of sentence-level repetition (Welleck et al., 2020) to compute the ratio of duplicate n -grams (rep- n) in the generated result. This metric is defined as

$$\text{rep-}n(\mathbf{Y}) = 100 \times \left(1.0 - \frac{|\text{unique } n\text{-grams}(\mathbf{Y})|}{|n\text{-grams}(\mathbf{Y})|}\right). \quad (13)$$

For each generated result, rep- n is 0.0 when it has no repeating n -grams. The final result is computed by averaging over the entire evaluation set.

We conduct experiments on Gigawords dataset to evaluate the n -gram repetitions ranging from uni-gram to 4-gram. The results are shown in Table 5, where w/o CA means the model is trained without using context-aware objective and R-L denotes the model’s ROUGE-L score. Additionally, we also show the results from transformer model for a direct comparison. Comparing the two variants of our model, we see that training with context-aware objective leads to a 42% drop on rep-3 metric (0.427 vs 0.741) and a 64% drop on rep-4 metric (0.106 vs 0.295). The ROUGE-L results also indicate that

Models	Ours ($\alpha = 1.0$)	Length-Parallel Decoding		
		LPD-1	LPD-5	LPD-10
BLEU	30.45	27.15	29.62	30.37
Speedup(\times)	11.31	11.84	8.92	6.01

Table 6: Results comparison on IWSLT14 dataset

the reduction in token repetition can effectively improve the model generation quality.

Dynamic Length Determination Next, we examine the importance of the model’s ability to dynamically determine the length of the generated output. To this end, we train another model variant by removing the two $[\text{eos}]$ tokens from the target sequence. In this way, the model is not able to self-determine the output length throughout the generation process. To perform inference, we use length-parallel decoding (LPD) with different number of length candidates. Formally, for each length candidate l , the model generates the result \tilde{Y} as

$$\tilde{Y} = \arg \max_{Y'} \left\{ \sum_{i=1}^l \Phi_{y'_i}(h_i) + \sum_{i=2}^l t(y'_{i-1}, y'_i) \right\}. \quad (14)$$

The final result is acquired by re-ranking the generated results with a transformer model.

We conduct experiments on the IWSLT14 DE-EN dataset in which we try a different number of length candidates, including top-1, top-5 and top-10. The results are shown in Table 6, from which we can see, as the number of length candidates increases, the model performance increases as well. The reason is that a larger candidates set is more likely to contain the best-suited length for the generation model, leading to better performance. However, such decoding procedure inevitably increases the required computation overhead. We can see that, when setting k as 10, the inference speedup decreases from $11.84\times$ to $6.01\times$. In contrast, our proposed model is able to determine the optimal output length by itself. Without any re-ranking process, it outperforms the model with LPD-10 decoding and achieves the inference speedup that is comparable with the model using LPD-1 decoding.

Ratio-First Decoding We are also interested in the effect of the ratio-first decoding strategy. To provide a quantitative analysis, we perform inference on the Gigawords dataset using ratio-first with different α . The experimental results with different α are presented in Figure 3. It can be observed that, when α reaches 0.3, the model approximately

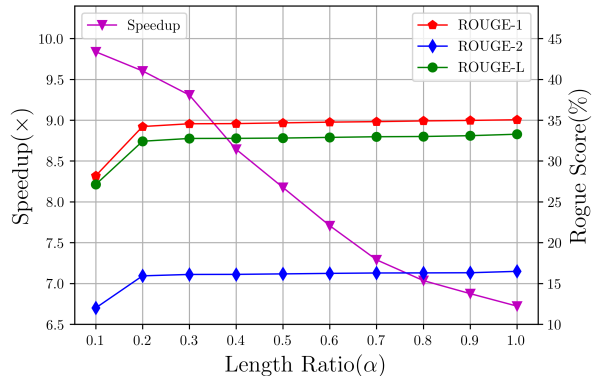


Figure 3: Experiment results on Gigawords dataset using ratio-first decoding with different α .

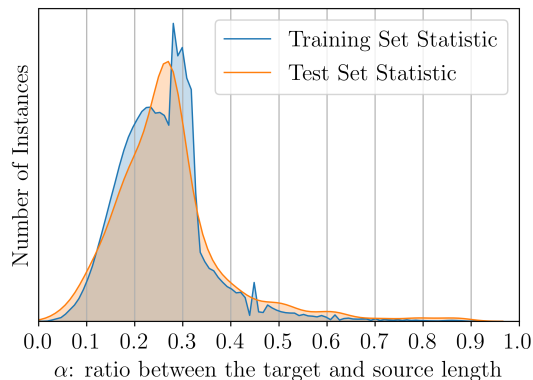


Figure 4: The distribution of target/source length ratio of the training and test set in Gigawords dataset.

achieves its optimal performance. At the same time, a notable improvement can be observed in terms of the inference speedup ($6.72\times \rightarrow 9.31\times$).

Now we illustrate why the near optimal performance can be achieved when α reaches 0.3. In Figure 4, we present the distribution of the target/source length ratio of every data instance in the Gigawords dataset. We can see that, for most cases, the ratio between the target length T' and source length T is less than 0.3. Recall the definition of ratio-first decoding in Equation (10), the $[\alpha \cdot T]$ constrains the maximum length of the generated result. Therefore, once we have a prior knowledge on the data statistic, we can easily choose a proper α that both improves the inference speed whilst maintaining the generation quality. In this case, a proper α could be 0.3 which is demonstrated by the results in Figure 3 and 4. By setting different α , ratio-first provides us an explicit way to control the balance between the inference speed and the generation quality. This property of ratio-first is especially favorable in real-life scenarios where the inference speed is the highest concern.

7 Conclusion

In this work, we explored the potential of BERT in various text generation tasks under the NAG framework. To address problems from NAG models previously having a prefixed output length, we devised a decoding mechanism which enables the model to determine the output length dynamically. To reduce errors stemming from the assumption of conditional independence of output tokens, we proposed a context-aware objective as well as using a CRF decoding. Furthermore, to maximize the inference speed advantage of our model, we introduced a ratio-first decoding strategy. We evaluated our model on three benchmark datasets and the results show that our model significantly outperforms many strong NAG baselines and performs comparably to many strong AG models.

Acknowledgments

The authors wish to thank Jialu Xu, Guanlin Li, Xing Wang for their insightful discussions and support. Many thanks to our anonymous reviewers for their suggestions and comments.

References

- 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 360–368.
- Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1481–1491.
- 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, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1243–1252.
- 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.
- 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*.
- Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. 2019. Non-autoregressive neural machine translation with enhanced decoder input. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3723–3730.
- Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2018. Higher-order syntactic attention network for longer sentence compression. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1716–1726.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289.
- Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2091–2100.
- Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

- Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.
- 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.
- Ani Nenkova and Kathleen R. McKeown. 2012. A survey of text summarization techniques. In *Mining Text Data*, pages 43–76.
- 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.
- Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.
- Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 379–389.
- Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. 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 1073–1083.
- 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, 8-14 December 2019, Vancouver, BC, Canada*, pages 3011–3020.
- 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, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.
- Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. 2017. Can syntax help? improving an lstm-based sentence compression model for new domains. 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 1385–1393.
- Wenbo Wang, Yang Gao, Heyan Huang, and Yuxiang Zhou. 2019a. Concept pointer network for abstractive summarization. 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 3074–3083.
- Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019b. Non-autoregressive machine translation with auxiliary regularization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 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 Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1304–1312.
- Sean Welleck, Iliia 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*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R’emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.