ELMER: A Non-Autoregressive Pre-trained Language Model for Efficient and Effective Text Generation

Junyi Li[1,3,4], **Tianyi Tang**[1], **Wayne Xin Zhao**[1,4*], **Jian-Yun Nie**[3] and **Ji-Rong Wen**[1,2,4]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]School of Information, Renmin University of China
[3]DIRO, Université de Montréal
[4]Beijing Key Laboratory of Big Data Management and Analysis Methods
{lijunyi,steven_tang}@ruc.edu.cn batmanfly@gmail.com

Abstract

We study the text generation task under the approach of pre-trained language models (PLMs). Typically, an auto-regressive (AR) method is adopted for generating texts in a token-by-token manner. Despite many advantages of AR generation, it usually suffers from inefficient inference. Therefore, non-autoregressive (NAR) models are proposed to generate all target tokens simultaneously. However, NAR models usually generate texts of lower quality due to the absence of token dependency in the output text. In this paper, we propose **ELMER**: an **E**fficient and effective P**LM** for NAR t**E**xt gene**R**ation to explicitly model the token dependency during NAR generation. By leveraging the early exit technique, ELMER enables the token generations at different layers, according to their prediction confidence (a more confident token will exit at a lower layer). Besides, we propose a novel pre-training objective, Layer Permutation Language Modeling, to pre-train ELMER by permuting the exit layer for each token in sequences. Experiments on three text generation tasks show that ELMER significantly outperforms NAR models and further narrows the performance gap with AR PLMs (*e.g.,* ELMER (29.92) vs BART (30.61) ROUGE-L in XSUM) while achieving over 10 times inference speedup.

1 Introduction

Since the advant of GPT-2 (Radford et al., 2019), pre-trained language models (PLMs) have achieved state-of-the-art performance across text generation tasks, which aim to generate human-like texts on demand (Brown et al., 2020; Li et al., 2022c). These PLMs usually adopt an *auto-regressive* (AR) fashion to generate texts token-by-token: the next token is predicted based on all previously generated tokens. A major limitation of this approach is that it is hard to be parallelized for the inference process, thus leading to a relatively high inference latency (Gu et al., 2018). Such a limitation prevents AR models from wide deployment in online real-time applications, such as query rewriting in search engines and online chat-bot. Moreover, AR models are prone to suffering from the exposure bias problem since there is a gap between AR training and inference (Zeng and Nie, 2021). These concerns have sparked extensive interests in *non-autoregressive* (NAR) models for text generation (Gu et al., 2018).

Compared to AR models, NAR models predict target tokens in all positions simultaneously and independently (Gu et al., 2018). This full parallelism leads to an efficient and low-latency inference process. However, the independence assumption prevents NAR models from learning the dependency among target tokens, resulting in accuracy degradation (Zhan et al., 2022). One widely-used solution to improve the NAR generation quality is to iteratively refine outputs (Gu et al., 2019; Ghazvininejad et al., 2019), which however leads to the loss in the speed-up advantage. In addition, many studies aim to learn the input-output mapping for more accurate generation via embedding mapping (Guo et al., 2019), latent alignment (Libovický and Helcl, 2018), and discrete variables (Ma et al., 2019). While easing the difficulty of NAR generation to some extent, these methods still struggle for generating complex sentences. Therefore, inspired by Zhan et al. (2022), we argue that the key to NAR text generation is to enhance the learning of token dependency—each token should be generated depending on forward and backward generated tokens.

In this paper, we propose **ELMER**: an **E**fficient and Effective P**LM** for NAR t**E**xt gene**R**ation, to *explicitly* learn the bi-directional token dependency. Typically, most NAR models predict tokens simultaneously only at the last layer, thus making the token prediction unaware of tokens generated in other positions. To address this issue, we propose

*Corresponding author

to generate tokens at different layers and the upper-layer token generation can depend on lower-layer generated tokens from both left and right. In this way, our model can explicitly learn the dependency between tokens from different layers while enjoying full parallelism in NAR decoding, as shown in Figure 1. To this end, we propose to extend the early exit technique (Li et al., 2021c) to NAR text generation: if there is sufficient confidence to generate a token at a lower layer, the model is allowed to exit at this layer and make the prediction without passing through the upper layers.

Furthermore, instead of exiting at a fixed layer for a token, we aim to predict each token at different layers for learning diverse token dependencies in NAR text generation. Thus, inspired by XLNet (Yang et al., 2019), we further propose a novel pre-training objective based on early exit, *i.e.,* Layer Permutation Language Modeling (LPLM), to help ELMER learn complex token dependencies. Given a sequence, LPLM will permute the exit layer (from 1 to the maximum layer) for each token and maximize the NAR text generation probability *w.r.t.* all possible exit layer permutations of the sequence. Through LPLM, each token is able to exit at different layers and attend to all other tokens from both forward and backward positions. In this way, LPLM could effectively capture diverse token dependencies from large-scale corpora. Pre-trained with the general LPLM, ELMER can adapt to downstream text generation tasks and datasets by using specific early exit strategies.

To the best of our knowledge, we are the first to introduce the idea of early exit to NAR text generation. We fine-tune ELMER on three popular text generation tasks. Experiments show that ELMER significantly improves the best NAR models by +5.71 ROUGE-1 on XSUM, +1.09 METEOR on SQuAD v1.1, and +2.26 Distinct-2 on PersonaChat, and narrows the performance gap with auto-regressive PLMs (*e.g.,* ELMER (29.92) vs BART (30.61) ROUGE-L on XSUM) while achieving over 10x faster inference.

2 Related Work

Pre-trained Language Models. Recent years have witnessed remarkable achievement of PLMs in text generation tasks (Li et al., 2021b). Most PLMs adopt an AR paradigm to generate texts during pre-training and fine-tuning. The work based on GPT (Radford et al., 2019; Brown et al., 2020) converts different tasks into language modeling by sequentially predicting tokens. BART (Lewis et al., 2020) employs an auto-regressive decoder to recover the corrupted text in pre-training. T5 (Raffel et al., 2020) masks word spans from input texts and then sequentially predicts masked tokens. Tang et al. (2022) pre-trains a text generation model using labeled datasets with multi-task learning. Li et al. (2022b) leverages prompts to effectively fine-tune text generation models. Differently, our PLM, ELMER, adopts a NAR schema to generate texts, which leads to a very low latency in inference.

Non-autoregressive Text Generation. Recently, there is a wide range of studies for NAR text generation (Gu et al., 2018; Ghazvininejad et al., 2019; Qi et al., 2021). Among them, Gu et al. (2018) is the first to propose NAR paradigm to reduce the inference latency. Ghazvininejad et al. (2019) iteratively masks and predicts a fraction of tokens that the model is least confident about. Several groups aim to learn accurate input-output mapping. For example, Saharia et al. (2020) and Libovický and Helcl (2018) use connectionist temporal classification to perform latent alignment in NAR models. Our work is closely related to BANG (Qi et al., 2021), a PLM bridging the NAR and AR generation. We differ in that we use early exit to predict tokens at different layers, which can help NAR models learn the forward and backward token dependency. Moreover, we propose a novel pre-training objective based on early exit, LPLM, for learning diverse token dependencies by permuting the exit layer for each token.

3 Preliminaries

Generally, the goal of text generation is to model the conditional probability $\Pr(\mathcal{Y}|\mathcal{X})$, where $\mathcal{X} = \langle x_1, \ldots, x_n \rangle$ and $\mathcal{Y} = \langle y_1, \ldots, y_m \rangle$ denote the input text and output text respectively and each consists of a sequence of tokens from a vocabulary $\mathcal{V}$. There are three common generation paradigms to model the conditional probability $\Pr(\mathcal{Y}|\mathcal{X})$, *i.e.,* autoregressive (AR), non-autoregressive (NAR), and semi-nonautoregressive (Semi-NAR) generation.

AR Generation. AR generation models predict the output text based on a left-to-right factorization as:

$$\Pr(\mathcal{Y}|\mathcal{X}) = \prod_{t=1}^{m} \Pr(y_t|y_{<t}, \mathcal{X}), \tag{1}$$

where each token y_t is generated based on the input text $\mathcal{X}$ and previous tokens $y_{<t}$. Note that AR generation only models the forward token dependency. The token-by-token fashion makes AR generation process hard to be parallelized. Most of existing text generation PLMs adopt AR approach (Radford et al., 2019; Lewis et al., 2020; Raffel et al., 2020).

NAR Generation. In contrast to AR models, NAR text generation models predict each token in output text simultaneously as follows, without modeling the forward or backward token dependency:

$$\Pr(\mathcal{Y}|\mathcal{X}) = \prod_{t=1}^{m} \Pr(y_t|\mathcal{X}), \tag{2}$$

where each token y_t is predicted only based on the input text $\mathcal{X}$. The independence assumption makes NAR generation process parallelizable, thus significantly accelerating the inference speed (Gu et al., 2018). While, in the absence of token dependency, the generation quality of NAR models is lower than their AR counterparts (Wang et al., 2019).

Semi-NAR Generation. Semi-NAR generation is formalized between AR and NAR generation as:

$$\Pr(\mathcal{Y}|\mathcal{X}) = \prod_{t=1}^{m} \Pr(y_t|\mathcal{Y}_{c_t}, \mathcal{X}), \tag{3}$$

where each token y_t is conditioned on the input text $\mathcal{X}$ and a visible part $\mathcal{Y}_{c_t}$ of the output text $\mathcal{Y}$. $\mathcal{Y}_{c_t}$ is designed differently to balance inference latency and accuracy (Stern et al., 2019; Lee et al., 2018). Note that the lower-layer generated tokens in our model is similar to the visible part $\mathcal{Y}_{c_t}$. While, our model keeps the advantage of full parallelism in contrast to iterative Semi-NAR methods.

In this paper, we mainly focus on the NAR approach, considering both effectiveness and efficiency for text generation models.

4 Approach

Our proposed NAR text generation PLM, ELMER, is depicted in Figure 1. ELMER aims to enhance the modeling of token dependency for NAR models. With early exit, tokens exiting at different layers can build the bi-directional token dependency with each other. Moreover, we design Layer Permutation Language Modeling (LPLM) to pre-train ELMER by permuting the exit layer for each token. Next, we will describe each part in detail.

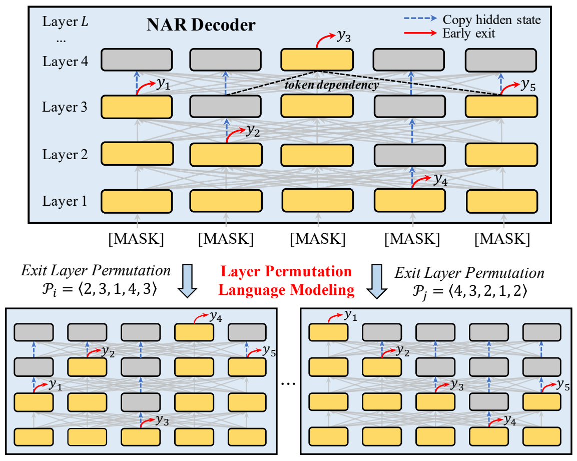


Figure 1: Overview of our proposed model ELMER.

4.1 Early Exit for Dependency Modeling

Most NAR models simultaneously predict tokens only at the last layer (Jiang et al., 2021; Zhan et al., 2022), which makes the current token generation unaware of the tokens generated in other positions. Thus, to model the bi-directional token dependency (both forward and backward), we propose to predict tokens at different layers by leveraging the early exit technique (Li et al., 2021c). In this way, the upper-layer token generation can depend on tokens generated at lower layers from both left and right.

NAR Transformer. ELMER is built on the Transformer encoder-decoder architecture (Vaswani et al., 2017). Both encoder and decoder consist of L stacked layers where each layer contains several sub-layers (*e.g.,* multi-head self-attention and feed-forward network). Unlike the original Transformer decoder that auto-regressively generates text, our model uses NAR fashion to generate tokens simultaneously. Given a pair of input-output text $\langle \mathcal{X}, \mathcal{Y} \rangle$, $\mathcal{X}$ is fed into the encoder and processed as hidden states $\boldsymbol{S} = \langle \boldsymbol{s}_1, ..., \boldsymbol{s}_n \rangle$. We then feed a sequence of "`[MASK]`" tokens into the NAR decoder to generate every token in output text $\mathcal{Y}$ in parallel.

Specifically, we first replace the original masked multi-head attention in decoder with bi-directional multi-head attention akin to the encoder. For each "`[MASK]`" token in the t-th position, the L decoder layers process it to hidden states $\{\boldsymbol{h}_t^l\}_{1 \le l \le L}$ as:

$$\boldsymbol{h}_t^l = \text{Layer}^l(\boldsymbol{h}_{1 \le t \le T}^{l-1}, \boldsymbol{S}), \tag{4}$$

$$\boldsymbol{h}_t^0 = \text{Embed}(\texttt{[MASK]}), \tag{5}$$

where $\text{Layer}^l(\cdot)$ denotes the l-th layer, $\text{Embed}(\cdot)$ is the sum of word embedding and position embedding, and T is the maximum length of the decoder.

The output distribution for predicting the t-th token y_t is computed by feeding the last decoder state $\boldsymbol{h}_t^L$ into a softmax classifier (parameterized by $\boldsymbol{W}_c$) as:

$$\Pr(y_t|\boldsymbol{h}_t^L) = \text{softmax}(\boldsymbol{W}_c \boldsymbol{h}_t^L). \tag{6}$$

Prior NAR models require to determine the output length, thus an extra module for length prediction is always needed. Instead, following Su et al. (2021), we let ELMER dynamically adjust the output length by emitting an end token (*i.e.,* `[EOS]`) at any position. After the entire generation is completed, the final sequence ranges from the beginning to the position where the first end token is emitted.

Early Exit Off-Ramps. Instead of predicting all tokens simultaneously at the last layer, we propose to generate tokens at different layers for learning the bi-directional token dependency. Hence, by leveraging early exit (Li et al., 2021c), if a token is predicted with sufficient confidence at a lower layer, the model is allowed to exit and predict without passing through the upper layers. In this way, the upper-layer tokens can be generated depending on forward and backward tokens generated at the lower layers. The bi-directional token dependency can further alleviate the hallucination issue in NAR generation (Gu et al., 2018), where the generated texts tend to be ungrammatical with repetitions.

Specifically, we inject the "*off-ramps*" (Li et al., 2021c), which make predictions with intermediate hidden states, at each decoder layer. The off-ramp can be simply implemented by a softmax classifier. For an off-ramp at the l-th layer, it makes the token prediction as:

$$\begin{aligned} \Pr(y_t|\boldsymbol{h}_t^l) &= \text{Off-Ramp}^l(\boldsymbol{h}_t^l), && (7) \\ &= \text{softmax}(\boldsymbol{W}_c^l \boldsymbol{h}_t^l), && (8) \end{aligned}$$

where the l-th off-ramp is parameterized by $\boldsymbol{W}_c^l$. These off-ramps can be specified independently or share the weights across L layers. Different from previous early-exit work that makes sentence-level prediction (Xin et al., 2020; Liao et al., 2021), our early exit is built at token level.

During training, if a token has been predicted at the l-th layer early, the hidden state $\boldsymbol{h}_t^l$ will not be updated in upper layers. Thus, in our model, we directly copy the last hidden state $\boldsymbol{h}_t^l$ (exit at layer l) to the subsequent layers following (Elbayad et al., 2020; Li et al., 2021c). Since the last hidden state predicts tokens with sufficient confidence, it has contained the predicted token information provided for the upper-layer tokens generation.

4.2 Layer Permutation Pre-training

The NAR models equipped with early exit predict each token at a fixed layer. To learn diverse token dependencies, we design a novel pre-training objective based on early exit, Layer Permutation Language Modeling (LPLM), where each token can exit at different layers. This is different from most prior work that focuses on designing small-scale NAR models for specific tasks such as translation. In contrast, we pre-train a general large-scale PLM on massive corpora following Qi et al. (2021). Our PLM can adapt to various downstream tasks.

Layer Permutation Language Modeling. Permutation language modeling was first proposed in XLNet (Yang et al., 2019) by permuting the factorization order. For a sequence with length T, there are $T!$ permutation orders to consider in autoregressive factorization. In our LPLM, rather than performing permutation on the sequence length, we permute the exit layer for each token. In particular, for each token in a sequence, we assume that it can exit at any of L layers. Therefore, there are L^T exit layer permutations for a sequence. Intuitively, if model parameters are shared across all exit layer permutations, each token can build diverse semantic dependencies with tokens in all positions.

Formally, let $\mathcal{P}_\mathcal{Y} = \{\boldsymbol{p} : \langle l_1, ..., l_t, ..., l_T \rangle\}$ be the set of all possible exit layer permutations of the sequence $\mathcal{Y}$ with length T. We use l_t ($1 \le l_t \le L$) to denote the exit layer of the t-th token. Then, for a permutation $\boldsymbol{p} \in \mathcal{P}_\mathcal{Y}$, the NAR generation probability (Eq. 2) based on LPLM can be expressed as:

$$\Pr(\mathcal{Y}|\mathcal{X}) = \prod_{t=1}^{T} \Pr(y_t|\mathcal{X}) = \prod_{t=1}^{T} \Pr(y_t|\boldsymbol{h}_t^{l_t}), \tag{9}$$

where the model exits at the l_t-th decoder layer and predicts the t-th token y_t with the hidden state $\boldsymbol{h}_t^{l_t}$. $\Pr(y_t|\boldsymbol{h}_t^{l_t})$ can be computed using Eq. 8.

During pre-training, for a sequence in our corpora, we sample k layer permutations at each time and compute the output probability with Eq. 9. With the layer permutation operation, each token can exit at different layers, thus LPLM can effectively learn diverse token dependencies from large-scale corpora. Moreover, typical early exit methods usually need to set thresholds to estimate the exit layer (Elbayad et al., 2020; Li et al., 2021c), which is not flexible for large-scale pre-training. Our proposed layer permutation naturally avoids to make exit estimations in large-scale pre-training.

Pre-training Tasks. Following BART (Lewis et al., 2020), our model is trained by first feeding the corrupted text to encoder and then reconstructing the original text by decoder in a NAR manner using the above LPLM. We mainly adopt two useful document corruption methods:

• *Sentence Shuffling:* The original text is first divided into sentences according to full stops, and then these sentences are randomly shuffled.

• *Text Infilling:* Based on the shuffled text, we sample 15% spans with lengths drawn from a Poisson distribution ($\lambda = 3$). Following BART (Lewis et al., 2020), each span is replaced with a single "`[MASK]`" token and the model can learn how many tokens in a span should be predicted.

4.3 Downstream Fine-Tuning

Our pre-trained model can be fine-tuned for various downstream text generation tasks. In fine-tuning phase, it becomes possible to precisely estimate the exit layer for each token with small-scale and task-specific datasets. Here, we mainly consider two early exit methods, *i.e.,* hard and soft early exit.

Hard Early Exit. The most straightforward way is to calculate the exit confidence and set a threshold. Following prior work (Xin et al., 2020), we quantify the exit confidence for token prediction using the *entropy* of the output probability distribution:

$$H(y_t)_{entropy} = -\sum \Pr(y_t|\boldsymbol{h}_t^l) \cdot \log \Pr(y_t|\boldsymbol{h}_t^l), \tag{10}$$

where $\Pr(y_t|\boldsymbol{h}_t^l)$ is computed as Eq. 8. In this way, a low entropy means a high confidence. Specifically, at the l-th off-ramp, our ELMER model will compute the entropy of its output distribution $H(y_t)$ and then compare with a pre-defined threshold δ to determine whether the model should exit here or continue to the next layer.

Soft Early Exit. The hard early exit method only makes one prediction for each token. Following prior work (Huang et al., 2021), the soft variant predicts the outputs at every decoder layer and the prediction is fed into the next layer for further improvement. In particular, at the l-th layer for the t-th position, we predict the most probable word $\hat{y}_t^l$ using the l-th off-ramp (Eq. 8) as follows:

$$\hat{y}_t^l = \arg\max \Pr(y_t^l|\boldsymbol{h}_t^l). \tag{11}$$

Then, we concatenate the word embedding of $\hat{y}_t^l$ with the current hidden state $\boldsymbol{h}_t^l$ and process it by a linear layer as follows:

$$\tilde{\boldsymbol{h}}_t^l = \boldsymbol{W}[\text{Embed}(\hat{y}_t^l); \boldsymbol{h}_t^l], \tag{12}$$

where $\boldsymbol{W}$ is a learnable matrix and $\tilde{\boldsymbol{h}}_t^l$ is the updated hidden state of the l-th layer, which will be taken as input to the next $l+1$-th layer. Compared with the hard early exit, the soft early exit is able to calibrate the token prediction at each layer.

4.4 Time Complexity Analysis

Since AR and NAR models adopt the same encoder architecture, the difference in time complexity is dominated by decoders. To generate a sequence with length T, an AR decoder with L layers has a time complexity $\mathcal{O}(LT^2)$ quadratic in sequence length. In contrast, a NAR decoder has a linear time complexity $\mathcal{O}(LT)$. This is because the attention computation in NAR decoders can be parallelized across all positions. While, our ELMER model generates tokens at different layers. Let $\overline{L}(< L)$ denote the average exit layer for a sequence, the time complexity will further be decreased to $\mathcal{O}(\overline{L}T)$.

5 Experiments

In this section, we detail the experimental setup and then highlight the main takeaways of our results.

5.1 Experimental Setup

Pre-training Setup. We pre-train ELMER based on the 16GB corpus (including English Wikipedia and BookCorpus). For our model, we use 6 layers in both encoder and decoder, with a dimension of 768 in hidden states, consistent with the base version of many AR and NAR pre-trained models (Lewis et al., 2020; Qi et al., 2020, 2021). We pre-train our model from scratch with a learning rate of 2e-4 and a minibatch size of 4096. We adopt the dictionary from BART (Lewis et al., 2020). In pre-training, we share the off-ramp weights across all layers and sample 10 exit layer permutations for a sequence.

Fine-Tuning Datasets. Following prior work (Qi et al., 2021), we fine-tune ELMER on three text generation tasks and datasets: (1) **XSUM** (Narayan et al., 2018) is a news summarization dataset containing 227K news article and single-sentence summary pairs; (2) **SQuAD v1.1** (Rajpurkar et al., 2016) is a question generation dataset, containing 98K triples of passage, question, and answer. We concatenate the passage and answer as input and predict the question; (3) **PersonaChat** (Zhang

Type	Models	XSUM			Latency↓ (ms/Sample)	SQuAD v1.1			Latency↓ (ms/Sample)
		R-1↑	R-2↑	R-L↑		R-L↑	B-4↑	MT↑	
AR	**Transformer**	30.66	10.80	24.48	262.47 (22.0x)	29.43	4.61	9.86	159.49 (13.4x)
	MASS	39.70	17.24	31.91	196.17 (16.4x)	49.48	20.16	24.41	132.46 (11.1x)
	BART	38.79	16.16	30.61	185.19 (15.5x)	42.55	17.08	23.19	114.00 (9.6x)
	ProphetNet	39.89	17.12	32.07	817.80 (68.5x)	48.00	19.58	23.94	456.51 (38.4x)
Semi-NAR	**InsT**	17.65	5.18	16.05	63.37 (5.3x)	29.98	2.34	8.15	67.61 (5.7x)
	iNAT	26.95	6.88	22.43	31.27 (2.6x)	32.34	3.16	9.18	31.59 (2.7x)
	CMLM	29.12	7.70	23.04	113.64 (9.5x)	29.60	3.89	9.70	106.84 (9.0x)
	LevT	25.33	7.40	21.48	101.01 (8.5x)	30.81	2.68	9.40	116.41 (9.8x)
	BANG	34.71	11.71	29.16	109.77 (9.2x)	47.39	17.62	21.69	111.11 (9.3x)
NAR	**NAT**	24.04	3.88	20.32	17.47 (1.5x)	31.51	2.46	8.86	17.11 (1.4x)
	iNAT	24.02	3.99	20.36	16.94 (1.4x)	32.44	2.33	8.84	16.52 (1.4x)
	CMLM	23.82	3.60	20.15	16.88 (1.4x)	31.58	2.51	8.85	16.41 (1.4x)
	LevT	24.75	4.18	20.87	27.72 (2.3x)	31.38	2.27	9.14	27.52 (2.3x)
	BANG	32.59	8.98	27.41	15.97 (1.3x)	**44.07**	12.75	18.99	15.69 (1.3x)
	ELMER-Hard	34.54	9.78	26.08	12.39 (1.0x)	37.94	11.77	18.01	12.24 (1.0x)
	ELMER-Soft	**38.30**	**14.17**	**29.92**	**11.94** (1.0x)	40.22	**13.49**	**20.08**	**11.90** (1.0x)

Table 1: A comparison between ELMER and baselines on XSUM and SQuAD v1.1 datasets. R-1/2/L, B-4, and MT are short-hands for ROUGE-1/2/L, BLEU-4, and METEOR. **ELMER-Hard** and **ELMER-Soft** denote fine-tuning ELMER with hard and soft early exit strategies, respectively. **Bold** and underline fonts denote the best and second best methods within NAR models. Our baseline results are collected from (Liu et al., 2021) and (Qi et al., 2021).

et al., 2018) is a dialog generation dataset, containing 150K triples of persona profile, dialogue history, and response. We concatenate the profile and dialogue history as input and generate the response. The statistics of datasets are shown in Appendix A.

Baselines. We compare **ELMER** to existing popular AR, NAR, and Semi-NAR generation models.

For AR generation, we experiment with a vanilla Transformer model and three PLMs:

• **Transformer** (Vaswani et al., 2017). It is an AR generation model without pre-training. To date, Transformer has become the backbone of many text generation PLMs and our ELMER model.

• **MASS** (Song et al., 2019), **BART** (Lewis et al., 2020), and **ProphetNet** (Qi et al., 2020). These are three representative PLMs for AR text generation, whose pre-training objectives vary from denoising text to future n-gram prediction. For a fair comparison, we adopt the base version of these PLMs.

For NAR and Semi-NAR generation, we evaluate six models with varying decoding strategies:

• **NAT** (Gu et al., 2018). It is the first proposed NAR text generation model. This baseline adds a module in the encoder to predict fertilities, acting as a global plan for the parallel generation.

• **InsT** (Stern et al., 2019). It is a Semi-NAR text generation model leveraging insertion operations. It repeatedly inserts tokens at multiple locations based on the partially inserted sequence.

• **iNAT** (Lee et al., 2018), **CMLM** (Ghazvininejad et al., 2019), **LevT** (Gu et al., 2019), and **BANG** (Qi et al., 2021). These four baselines are both NAR and Semi-NAR text generation models. Among them, BANG is the recent state-of-the-art PLM for NAR text generation.

Evaluation Metrics. To evaluate the *effectiveness* of different models, we adopt four evaluation metrics: ROUGE-n assesses the text quality by computing the overlapping n-grams between the generated and real texts (Lin, 2004); BLEU-n computes the co-occurrence ratio of n-grams between the generated and real texts (Papineni et al., 2002); METEOR assesses word-to-word matches between the generated and real texts based on the harmonic mean of the unigram precision and recall (Banerjee and Lavie, 2005); and Distinct-n measures the diversity degree by calculating the number of distinct n-grams in generated texts (Li et al., 2016). To examine the *efficiency*, we set the batch size as 1 at inference to calculate the per-sample inference latency under the same parameter setting following (Qi et al., 2021). More details about our experiments can be found in Appendix B.

5.2 Main Results

Table 1 and Table 2 display the results of ELMER and baselines on three text generation datasets.

First, ELMER-Soft outperforms NAR and Semi-NAR baselines on almost all datasets and metrics.

Type	Models	PersonaChat					Latency↓ (ms/Sample)
		BLEU-1↑	BLEU-2↑	Distinct-1↑	Distinct-2↑	Overall↑	
AR	**Transformer**	41.56	32.95	0.30	0.80	18.90	138.31 (11.9x)
	MASS	41.06	35.75	1.40	6.90	21.28	112.13 (9.7x)
	BART	47.60	39.36	1.10	6.10	23.54	106.15 (9.2x)
	ProphetNet	46.00	38.40	1.30	7.30	23.25	392.79 (33.9x)
Semi-NAR	**InsT**	12.63	9.43	0.10	0.30	5.62	65.27 (5.6x)
	iNAT	41.17	32.13	0.10	1.10	18.63	43.25 (3.7x)
	CMLM	44.38	35.18	0.10	0.80	20.12	105.82 (9.1x)
	LevT	24.89	18.94	0.10	0.60	11.13	80.26 (6.9x)
	BANG	39.82	30.72	1.90	14.20	21.66	109.17 (9.4x)
NAR	**NAT**	**31.53**	**24.17**	0.10	0.80	14.15	17.86 (1.5x)
	iNAT	30.56	23.38	0.10	0.70	13.69	16.40 (1.4x)
	CMLM	31.44	24.06	0.10	0.60	14.05	16.26 (1.4x)
	LevT	26.92	20.47	0.00	0.40	11.95	27.56 (2.4x)
	BANG	31.11	23.90	2.50	22.70	20.05	14.89 (1.3x)
	ELMER-Hard	29.43	21.89	1.56	21.45	18.58	12.01 (1.0x)
	ELMER-Soft	31.45	23.99	**3.66**	**24.96**	**21.02**	**11.59** (1.0x)

Table 2: A comparison between ELMER and baselines on PersonaChat with respect to automatic evaluation metrics.

Compared to the best NAR BANG, ELMER-Soft achieves prominent gains in effectiveness metrics such as +5.71 ROUGE-1 in XSUM, +1.09 METEOR in SQuAD v1.1, and +2.26 Distinct-2 in PersonaChat. These considerable gains clearly demonstrate the *effectiveness* of our ELMER model. In contrast to these NAR and Semi-NAR models, our model leverages the early exit technique to explicitly model the forward and backward token dependency during parallel NAR decoding.

Second, ELMER consistently achieves comparable or better results than the AR baseline without pre-training, *i.e.,* Transformer, and further narrows the performance gap between NAR and AR PLMs. For example, the performance gap between BANG and BART in XSUM and SQuAD is 3.20 ROUGE-L and 4.20 METEOR, which has now been decreased by ELMER-Soft to 0.69 and 3.11. ELMER-Soft also obtains the best Distinct scores in PersonaChat. Diversity is critical for dialogue generation to avoid boring or useless responses. Besides, the lower BLEU scores than NAT probably because the diverse generated texts by ELMER-Soft makes a reasonable difference from real texts. For those models without pre-training such as CMLM and NAT, they tend to generate highly frequent words and common phrases such as "*I, an, the*", which share much overlapped and repetitive parts with the target output. By contrast, our pre-trained model generates diverse responses, which is more critical to avoid boring or useless responses in dialogue generation.

Finally, in terms of *efficiency*, ELMER achieves much faster inference speed than all NAR and AR models, *w.r.t.* the per sample inference latency. For example, the inference latency of AR Transformer model in XSUM, SQuAD v1.1, and PersonaChat are 262.47ms, 159.49ms, and 138.31ms per sample, while those of ELMER-Soft are 11.94ms, 11.90ms, and 11.59ms, which amount to 22.0, 13.4, and 11.9 times speedup. Compared to AR PLMs, we achieve over 10x inference speedup, especially the highest 68.5x speedup in XSUM. Moreover, ELMER-Soft is slightly faster than other NAR baselines. We speculate that this speedup is mainly due to the fact that prior NAR models require to predict the output length such as NAT or rely on multiple insertion and deletion operations such as LevT.

Models	XSUM		
	ROUGE-1	**ROUGE-2**	**ROUGE-L**
NAR LevT	24.75	4.18	20.87
ELMER-Hard	34.54	9.78	26.08
-w/o pre-training	28.56	5.33	21.96
ELMER-Soft	38.30	14.17	29.92
-w/o pre-training	30.45	7.37	24.00

Table 3: Ablation study on XSUM dataset.

5.3 Detailed Analysis

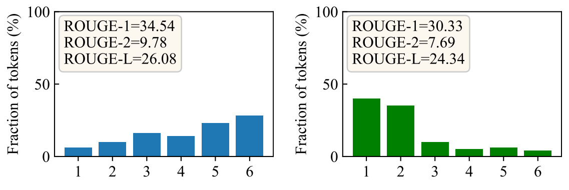
We report detailed analysis of our model on XSUM dataset – we have similar findings on other datasets.

Ablation Study. Our ELMER model is the first one to adopt the early exit technique to conduct NAR text generation. To learn more diverse token dependencies, we proposed the LPLM objective (Sec. 4.2) based on early exit to pre-train ELMER. Here, we evaluate the effects of early exit and large-scale LPLM training by directly applying the hard and soft early exit methods to NAR text generation without pre-training. We compare the early-exit variants with the best non-pretrained NAR model in XSUM, *i.e.,* LevT. The results are shown in Table 3. We can observe that these two early-exit variants without pre-training perform better than LevT. This indicates that introducing the early exit mechanism can improve NAR text generation quality due to the explicit modeling of the target-side dependency. Moreover, our proposed large-scale pre-training based on the LPLM objective improves NAR results significantly and consistently in both early exit strategies. By utilizing the general LPLM, our model can capture diverse token dependencies from large-scale corpora and further adapt to downstream tasks with specific early exit methods.

Parameter Sensitivity Analysis. In ELMER-Hard, the pre-defined entropy threshold δ is critical to determine at which layer our model will exit. Here, we further examine the model performance and the fraction of tokens exiting at every layers ($1 \sim 6$) by varying the exit threshold in the set $\{0.0, 0.25, 0.5, 0.75, 1.0\}$. As the threshold δ increases gradually, more tokens exit earlier. From the experiment, we find that low and high thresholds lead to reverse early exit situations shown by the fraction of tokens exiting at each layer. Therefore, we only present the results for two representative thresholds (0.5 and 1.0) in Figure 2. We can observe that: (1) when $\delta = 0.5$, a larger fraction of tokens are generated in higher layers (≥ 3), which means a lower threshold needs more computation to achieve enough confidence. These upper-layer "difficult" tokens can depend on simple tokens such as "*I*", "*the*", "*have*" generated at lower layers. (2) when $\delta = 1.0$, it shows an opposite trend and the performance drops slightly. The reason might be that 1.0 is so high that the model makes a precipitate exiting decision. Finally, according to the model performance on the validation set, we set the threshold δ as 0.5 in our experiments.

5.4 Human Evaluation

Despite the effectiveness of automatic evaluation, human evaluation remains critical for text generation (Celikyilmaz et al., 2020). Since human evaluation is expensive, we only focus on comparing



(a) $\delta = 0.5$ (b) $\delta = 1.0$

Figure 2: Varying δ for ELMER-Hard on XSUM.

our best performing ELMER-Soft with two AR and NAR PLMs, *i.e.,* BART and BANG.

In order to reduce the variance caused by human, three workers were asked to score texts from four aspects (Li et al., 2022a), *i.e., fluency*, *informativeness*, *accuracy*, and *relevance*. *Fluency* evaluates whether the text is well-formed and logical to read; *Informativeness* measures whether the text contains useful information; *Accuracy* tests whether the text describes the given content accurately; *Relevance* measures whether the text is relevant to the given context. These four scores are rated from 1 to 5. We further design a Turing test (Turing, 1950) where a human judge is asked to detect whether the given text is generated by a human. For each method, we average the scores from three human judges and then report the average results over 500 texts. From the results in Table 4, we can see that our model is better than NAR model BANG in terms of *fluency* (3.99 vs 3.42) and *relevance* (3.63 vs 3.38). The major reason is that our model leverages the early exit to explicitly model the token dependency, which could reduce token repetitions and improves the fluency of texts. While, our model achieves a slightly worse fluency score than the AR model BART. We speculate that BART can produce some common phrases by using the AR generation mode.

We also present some generated examples by our model in Appendix C. It can be observe that our model can generate some common phrases occurred in real texts such as "*surface-to-air*", "*South China Sea*", and "*main seminary*". These phrases can improve the fluency of NAR generated texts. By incorporating early exit and generating tokens at different layers, our model can explicitly learn the forward and backward token dependency, which can deal with the multi-modality issue to some extent. While, as a NAR model, our model still inevitably generates some repetitive stop tokens such as "*on on on*", which can be further improved in future work.

Models	XSUM				
	TT (%)	Flu.	Info.	Acc.	Rel.
BART	51.01	4.11	3.99	3.64	3.79
BANG	45.34	3.42	3.45	3.46	3.38
ELMER-Soft	48.90	3.99	3.52	3.49	3.63
Gold	55.67	4.12	4.29	3.95	4.05

Table 4: Turing test (TT) and human scores on XSUM. Flu., Info., Acc. and Rel. denote fluency, informativeness, accuracy and relevance respectively.

6 Conclusion

This paper presented an efficient and effective PLM for NAR text generation, called ELMER. ELMER introduced a token-level early exit mechanism to explicitly model the semantic dependency between target tokens during parallel decoding. Moreover, we proposed a pre-training objective, Layer Permutation Language Modeling, to pre-train ELMER on large-scale corpora by permuting the exit layer for each token in a sequence. Experiments on text summarization, question generation, and dialogue generation demonstrate that our model can effectively improve the NAR generation quality compared to highly competitive NAR models and further narrow the performance gap with AR models while achieving higher inference efficiency. In future work, we will investigate the efficacy of utilizing the proposed model to iteratively generate texts and consider more early exit strategies in fine-tuning.

7 Limitations

An important limitation of ELMER compared with other NAR text generation models is the need for defining an appropriate early exit strategy. In pre-training, ELMER utilizes a general strategy, *i.e.,* layer permutation language modeling, by permuting the exit layer for each token in a sequence. However, to apply ELMER to downstream specific tasks and datasets, we need to design an effective and suitable estimation method to decide at which layer the model will exit and predict tokens. Also, as a PLM, ELMER may present biases learned from the pre-training corpus in the output text.

In this study, we evaluate ELMER on three text generation tasks and datasets, but the output length of these tasks is relatively short. We should deal with long-form text generation which raises more challenges to the NAR paradigm.

8 Ethical Concerns

Current NAR text generation techniques achieve faster inference speed but suffer from several issues like multi-modality, lack of accuracy to the input, commonsense issues etc., which makes their online real-time deployment difficult. ELMER is an effort at rectifying some of these issues, with a focus of modeling the semantic dependency between target tokens to improve NAR generation quality. However, compared to AR models, ELMER outputs continue to mix multiple candidates and repeat words at times. This should be strongly considered before any direct deployment of real-world systems.

On the other hand, the text generation technology may be potentially misused for harmful applications. When deploying ELMER to online real-time platforms, the high-quality text generated by our work also makes it difficult to distinguish synthetic text from human-written text, such as fake news and stories. It is somewhat difficult to anticipate the harmful usages of our method since they often involve repurposing our model in a totally different setting or for an unexpected purpose than we planned. To alleviate this problem, we can ask for help from some classic security risk assessment frameworks such as detecting threats and potential impacts, measuring likelihood, and determining risk as a combination of likelihood and impact (Blank, 2011).

Acknowledgement

This work was partially supported by Beijing Natural Science Foundation under Grant No. 4222027, and Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098, and the Outstanding Innovative Talents Cultivation Funded Programs 2021 of Renmin University of China. Xin Zhao is the corresponding author.

Reproducibility

For reproducing and reusing our work, we release the pre-trained ELMER model and source codes at the link: `https://github.com/RUCAIBox/ELMER`. In the future, we will integrate our model into Hugging Face (Wolf et al., 2020) and TextBox (Li et al., 2021a) libraries for easy-to-use. We hope that these open-source resources will facilitate and contribute to the advancement of related research.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72. Association for Computational Linguistics.

Rebecca M Blank. 2011. Guide for conducting risk assessments.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *CoRR*, abs/2006.14799.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. 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 11179–11189.

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. AAAI Press.

Chenyang Huang, Hao Zhou, Osmar R. Zaïane, Lili Mou, and Lei Li. 2021. Non-autoregressive translation with layer-wise prediction and deep supervision. *CoRR*, abs/2110.07515.

Ting Jiang, Shaohan Huang, Zihan Zhang, Deqing Wang, Fuzhen Zhuang, Furu Wei, Haizhen Huang, Liangjie Zhang, and Qi Zhang. 2021. Improving non-autoregressive generation with mixup training. *CoRR*, abs/2110.11115.

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, Brussels, Belgium, October 31 - November 4, 2018*, pages 1173–1182. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. 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 110–119. The Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Zheng Gong, Lixin Yang, Zhuohao Yu, Zhipeng Chen, Jingyuan Wang, Xin Zhao, and Ji-Rong Wen. 2022a. Eliteplm: An empirical study on general language ability evaluation of pretrained language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3519–3539. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Gaole He, Jinhao Jiang, Xiaoxuan Hu, Puzhao Xie, Zhipeng Chen, Zhuohao Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2021a. Textbox: A unified, modularized, and extensible framework for text generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 30–39.

Junyi Li, Tianyi Tang, Jian-Yun Nie, Ji-Rong Wen, and Xin Zhao. 2022b. Learning to transfer prompts for text generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022, pages 3506–3518. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2022c. A survey of pretrained language models based text generation. *CoRR*, abs/2201.05273.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021b. Pretrained language model for text generation: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4492–4499. ijcai.org.

Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2021c. Accelerating BERT inference for sequence labeling via early-exit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 189–199. Association for Computational Linguistics.

Kaiyuan Liao, Yi Zhang, Xuancheng Ren, Qi Su, Xu Sun, and Bin He. 2021. A global past-future early exit method for accelerating inference of pre-trained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2013–2023. Association for Computational Linguistics.

Jindrich Libovický and Jindrich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3016–3021. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

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. 2021. GLGE: A new general language generation evaluation benchmark. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 408–420. 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.

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.

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

Weizhen Qi, Yeyun Gong, Jian Jiao, Yu Yan, Weizhu Chen, Dayiheng Liu, Kewen Tang, Houqiang Li, Jiusheng Chen, Ruofei Zhang, Ming Zhou, and Nan Duan. 2021. BANG: bridging autoregressive and non-autoregressive generation with large scale pre-training. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8630–8639. PMLR.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 2401–2410. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

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.

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.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5926–5936. PMLR.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5976–5985. PMLR.

Yixuan Su, Deng Cai, Yan Wang, David Vandyke, Simon Baker, Piji Li, and Nigel Collier. 2021. Non-autoregressive text generation with pre-trained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 234–243. Association for Computational Linguistics.

Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2022. MVP: multi-task supervised pre-training for natural language generation. *CoRR*, abs/2206.12131.

Alan M. Turing. 1950. Computing machinery and intelligence. *Mind*, LIX(236):433–460.

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.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. 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. AAAI Press.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2246–2251. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. 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 5754–5764.

Yan Zeng and Jian-Yun Nie. 2021. An investigation of suitability of pre-trained language models for dialogue generation - avoiding discrepancies. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 4481–4494. Association for Computational Linguistics.

Jiaao Zhan, Qian Chen, Boxing Chen, Wen Wang, Yu Bai, and Yang Gao. 2022. Non-autoregressive translation with dependency-aware decoder. *CoRR*, abs/2203.16266.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2204–2213. Association for Computational Linguistics.

Appendix

We provide some experiment-related information as supplementary materials. The appendix is organized into three sections:

- Statistics of each dataset are presented in Appendix A;
- Training settings of baselines and our model ELMER are presented in Appendix B;
- Generated examples by our model are presented in Appendix C.

A Statistics of Datasets

The detailed information of these three datasets is listed in Table 5.

Dataset	#Train	#Valid	#Test	#Output
XSUM	204,045	11,332	11,334	21.1
SQuAD v1.1	75,722	10,570	11,877	11.6
PersonaChat	122,499	14,602	14,056	11.9

Table 5: Statistics of three datasets. #Output denotes the average number of tokens in the output texts.

B Experimental Details

For AR baselines, we adopt the hyper-parameters: learning rate 2e-5, batch size 20, Adam optimizer, and the maximum input and output length of 512. We fine-tune these models on each dataset for 50 epochs and select the best model. For NAR and Semi-NAR baselines, the hyper-parameters are the same as AR baselines except the number of fine-tuning epochs, since NAR models need more epochs to converge (Qi et al., 2021). We fine-tune these baselines and ELMER for 50 epochs and save checkpoints for every epoch. We select the best checkpoint based on the performance on validation set. Following BANG (Qi et al., 2021), the difference between semi-NAR and NAR models lies in that: (1) we select the outputs from the first iteration as the NAR outputs, since the first iteration only depends on the input text; (2) we select the outputs from the maximum (we set as 10) iteration as the semi-NAR outputs, since each iteration depends on both the input text and the last generated texts. Most settings and results are collected from GLGE (Liu et al., 2021) and BANG (Qi et al., 2021). To make a fair comparison with non-pretrained baselines, we also compare ELMER with the best NAR non-pretrained baseline LevT in ablation analysis (Table 3).

C Case Study

We show some qualitative examples of these three datasets in Table 6, Table 7, and Table 8.

Due to the explicit modeling of the bi-direction token dependency, our model can generate some common phrases occurred in real texts such as "surface-to-air", "South China Sea", and "main seminary". These phrases can improve the fluency of NAR generated texts. But, as a NAR model, our model still inevitably generates some repetitive tokens such as "on on on". The generated examples in PersonaChat also explain that our model generates more diverse responses, which are very different from the real responses.

News	Reports on Wednesday suggested more than one iguana was actually filmed, with scenes then stitched together. But the BBC has said only one animal was chased by the snakes - with other iguanas only filmed for close-ups. The scene quickly went viral when it was aired last year and later won a Bafta for must-see moment. The iguana hatchling, filmed in the Galapagos, eventually got away - much to viewers' relief. The Daily Mail claimed the episode was embroiled in a "fakery row" after producer Elizabeth White told the Media Production Show: "It wasn't the same iguana, no, and often we have to augment it with other clips. "Unfortunately lizards, snakes and iguanas aren't good at 'takes'." But the BBC defended the Sir David Attenborough-fronted programme, with a spokeswoman saying: "The BBC strongly refutes any suggestion that the award-winning iguana v snakes sequence was 'faked'. "The final iguana chase in which one iguana escapes the snakes was - unusually for natural history filming - shot using two cameras, allowing us to follow both the individual iguana and the snakes' point of view. "What was captured in the field was extraordinary animal behaviour which had never been witnessed or filmed before." She added: "As is common in natural history film-making, pick-up shots were filmed separately - for example close-ups of iguana eyes - to make the story of the sequence as clear as possible for the audience. "This is absolutely in keeping with the norms of natural history film-making - and absolutely in line with the BBC's editorial policy guidelines, and was a true representation of animal behaviour."	Satellite images taken on 14 February appear to show two batteries of eight missile launchers and a radar system on Woody or Yongxing Island in the Paracels. The presence of missiles would significantly increase tensions in the acrimonious South China Sea dispute. China's Foreign Minister Wang Yi said reports were a Western media invention. But Mr Wang defended "the limited and necessary self-defence facilities" on islands inhabited by Chinese personnel as "consistent with the right for self-preservation and self-protection.... under the international law". Asked about the reports, US Secretary of State John Kerry attacked China's increased "militarisation" of the contested region, saying it was a "serious concern". Taiwan's defence ministry said it had "learned of an air defence missile system deployed" by the Chinese on Woody Island. It would not say how many missiles had been deployed or when, but told the BBC they would be capable of targeting civilian and military aircraft. The commander of the US Pacific Fleet confirmed the deployment to Reuters news agency. Adm Harry Harris said such a move would be "a militarisation of the South China Sea in ways" China's President Xi Jinping had pledged not to make. Japan's Chief Cabinet Secretary Yoshihide Suga said there were "serious concerns" over China's "unilateral move to change the status quo" in the region, and "we cannot accept this fact". China has been carrying out extensive land reclamation work in the region, which it says is legal and for civilian purposes. But the work has angered other countries which also claim the territory, and there is growing concern about the implications of the area becoming militarised.
Summaries	The BBC has denied claims award-winning series Planet Earth II faked a nail-biting scene showing a baby iguana being chased by racer snakes .	China has deployed surface-to-air missiles on a disputed island in the South China Sea, Taiwan says .
ELMER-Hard	The BBC has defended claims the BBC nature documentary of the iguana footage of a "faked storm " by a Davidborough.	China has confirmed that new reports of on surface-air on on on of in the South China Sea .
ELMER-Soft	The BBC has defended claims a BBC documentary about a series of from natural drama Sir David Atten was "faked".	China is the first of of air defence missiles on a disputed disputed island of the South China Sea, according to US

Table 6: Qualitative examples on XSUM dataset.

Paragraphs	Moreau Seminary [SEP] The university is the major seat of the Congregation of Holy Cross (albeit not its official headquarters , which are in Rome) . Its main seminary , Moreau Seminary , is located on the campus across St . Joseph lake from the Main Building . Old College , the oldest building on campus and located near the shore of St . Mary lake , houses undergraduate seminarians . Retired priests and brothers reside in Fatima House (a former retreat center) , Holy Cross House , as well as Columba Hall near the Grotto . The university through the Moreau Seminary has ties to theologian Frederick Buechner . While not Catholic , Buechner has praised writers from Notre Dame and Moreau Seminary created a Buechner Prize for Preaching .	eight [SEP] The College of Engineering was established in 1920 , however , early courses in civil and mechanical engineering were a part of the College of Science since the 1870s . Today the college , housed in the Fitzpatrick , Cushing , and Stinson - Remick Halls of Engineering , includes five departments of study ? @ S aerospace and mechanical engineering , chemical and biomolecular engineering , civil engineering and geological sciences , computer science and engineering , and electrical engineering ? @ S with eight B . S . degrees offered . Additionally , the college offers five - year dual degree programs with the Colleges of Arts and Letters and of Business awarding additional B . A . and Master of Business Administration (MBA) degrees , respectively .
Questions	What is the primary seminary of the Congregation of the Holy Cross ?	How many BS level degrees are offered in the College of Engineering at Notre Dame ?
ELMER-Hard	What is the name of Frederick Binaryinary at St. Joseph ?	How many B.S degrees offered at the College of the engineering ?
ELMER-Soft	What is the name of the main seminary at St. Joseph ?	How many B.S. degrees offered at the College of departments ?

Table 7: Qualitative examples on SQuAD v1.1 dataset.

Histories	i love to meet new people . i have a turtle named timothy . my favorite sport is ultimate frisbee . my parents are living in bora bora . autumn is my favorite season . [SEP] hello , how are you doing tonight ? i am well an loving this interaction how are you ? i am great . i just got back from the club .	i just bought a brand new house . i like to dance at the club . i run a dog obedience school . i have a big sweet tooth . i like taking and posting selkies . [SEP] hello , how are you doing tonight ? i am well an loving this interaction how are you ? i am great . i just got back from the club . this is my favorite time of the year season wise i would rather eat chocolate cake during this season . what club did you go to ? me an timothy watched tv i went to club chino . what show are you watching ? lol oh okay kind of random do you live in a house or apartment ? we watched a show about animals like him i love those shows . i am really craving cake . why does that matter any ? i went outdoors to play frisbee
Reponses	this is my favorite time of the year season wise	it matters because i have a sweet tooth .
ELMER-Hard	what team of do you do ? ultimate	i like to exercise with my dogs .
ELMER-Soft	i love club! do you have new friends ?	cool. i spend time with my dog .

Table 8: Qualitative examples on PersonaChat dataset.