P³LM: Probabilistically Permuted Prophet Language Modeling for Generative Pre-Training

Junwei Bao[†]*, **Yifan Wang**[†], **Jiangyong Ying**[‡], **Yeyun Gong**[♯],
Jing Zhao[†], **Youzheng Wu**[†], **Xiaodong He**[†]
[†]JD AI Research [‡]Huawei Technologies [♯]Microsoft Research Asia
baojunwei@jd.com yingjiangyong@huawei.com yegong@microsoft.com

Abstract

Conventional autoregressive left-to-right (L2R) sequence generation faces two issues during decoding: limited to unidirectional target sequence modeling, and constrained on strong local dependencies. To address the aforementioned problem, we propose **P³LM**, a **p**robabilistically **p**ermuted **p**rophet **l**anguage **m**odel, which strengthens the modeling of bidirectional information and long token dependencies for sequence generation. Specifically, P³LM learns to generate tokens in permuted order upon an order-aware transformer decoder, as well as to generate the corresponding future N tokens with a multi-stream attention mechanism. Extensive experiments are conducted on the GLGE benchmark, which includes four datasets for summarization, two for question generation, one for conversational question answering, and one for dialog response generation, where P³LM achieves state-of-the-art results compared with strong publicly available generative pre-training methods.[1]

1 Introduction

Natural language generation (NLG), aiming to automatically generate a sequence of tokens, are widely explored on tasks such as summarization, question answering and generation, dialog response generation, and machine translation. Recently, generative pre-training models (Radford et al.; Song et al., 2019; Dong et al., 2019; Lewis et al., 2020; Raffel et al., 2019; Zhang et al., 2019; Bi et al., 2020; Xiao et al., 2020; Qi et al., 2020), which accumulate knowledge based on large-scale unsupervised conditional language modeling, have achieved remarkable improvements on downstream NLG tasks compared with conventional methods. A typical generative pre-training model (Song et al., 2019; Lewis et al., 2020) follows the transformer (Vaswani et al., 2017) framework which contains an encoder and a decoder, where the decoder usually learns to generate a sequence in a left-to-right (L2R) order. The L2R decoding strategy usually faces two issues during the modeling of target sequences: (1) limited to unidirectional context information, and (2) constrained on strong local dependencies.

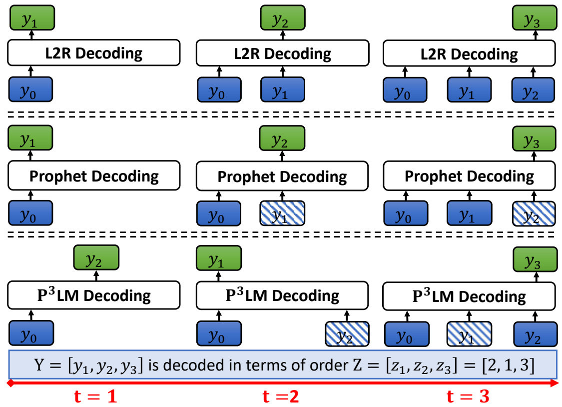

Figure 1: An illustration of L2R, Prophet, and P³LM decoding. **L2R decoding**: y_t is predicted based on $y_{\le t-1}$. **Prophet decoding**: y_t is predicted based on $y_{\le t-1}$, or $y_{\le t-2}$ with y_{t-1} being masked. **P³LM decoding**: Y is autoregressively decoded in terms of order Z, where y_{z_t} is predicted based on $y_{z_{\le t-1}}$, or $y_{z_{\le t-2}}$ with $y_{z_{t-1}}$ being masked. $y_0 = \langle s \rangle$ is the start of a sentence.

In order to enable a language model to learn bidirectional context information, auto-encoding ones, such as BERT (Devlin et al., 2019) known as a masked language model (MLM), are pre-trained based on randomly masked token prediction. In addition, autoregressive ones, such as XLNet[2] (Yang et al., 2019) known as a permutation language model (PLM), are designed to reconstruct a partial sequence in permuted order. However, directly applying these methods on language generation is not feasible, since they are designed for natural language understanding (NLU), which are usually handled by just one encoder or decoder (Song et al.,

*Corresponding author: baojunwei001@gmail.com

[1]The code is available at https://github.com/JunweiBao/P3LM.

[2]We clarify the differences between our P³LM and XLNet in Appendix A in detail.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 6663–6675
December 7-11, 2022

2019). To prevent overfitting on strong local dependencies during decoding, ProphetNet (Qi et al., 2020) is proposed to predict N future tokens. However, the future token prediction strategy predicts at most N (typically $N = 2$) continuous tokens, which has limited ability on long dependency modeling. Besides, due to the L2R decoding, the unidirectional target context modeling issue still exists.

To further enhance the ability of long dependency modeling, as well as capturing bidirectional information of target sequences, we propose P^{3}LM, a **p**robabilistically **p**ermuted **p**rophet **l**anguage **m**odel. P^{3}LM learns to generate tokens in permuted order with an order-aware transformer decoder, as well as predicting the corresponding N future tokens with a multi-stream attention mechanism. Figure 1 illustrates the idea of the proposed P^{3}LM. For instance, given a target sequence $Y = [y_1, y_2, y_3]$=*sequence*→*order*→*matters* and a permuted order $Z = [2, 1, 3]$, P^{3}LM learns to generate sequence Y in order Z, i.e., *order*→*sequence* → *matters*. Meanwhile, it also learns to predict future tokens in terms of Z, e.g., predicting *sequence* as the future token of *order* at time step $t = 1$. The above design makes P^{3}LM capable of capturing bidirectional information of target sequence, and strengths the modelling of long dependencies.

Extensive experiments are conducted on the GLGE (Liu et al., 2021) benchmark, a general language generation evaluation benchmark consisting of four datasets for summarization, two for question generation, one for conversational question answering, and one for dialog response generation, where our proposed P^{3}LM achieves 0.9 absolute and 2.5% relative improvements on the overall score compared with the public available state-of-the-art model, i.e., ProphetNet. To conclude, the contributions are as follows: (I) We propose P^{3}LM, a permutation over prophet decoding net, for generative pre-training, which utilizes bidirectional context information and enhances long token dependency modeling on target sequences; (II) We conduct extensive experiments on downstream language generation tasks and show that P^{3}LM obtains new state-of-the-art results on GLGE benchmark compared with published methods; (III) Three P^{3}LM models, which cost about 100,000 dollars, are pre-trained based on large scale datasets and will be released for further research on generative pre-training and language generation for the NLP community.

Models	**Structure**	**Tasks**	**Features During Decoding**		
			Order	**LongDep**	**BiDir**
BERT	Enc	NLU	-	-	-
RoBERTa	Enc	NLU	-	-	-
XLNet	Enc	NLU	-	-	-
ELECTRA	Enc	NLU	-	-	-
ALBERT	Enc	NLU	-	-	-
GPT	Dec	NLU&NLG	L2R	Shallow	No
UniLM	Enc/Dec	NLU&NLG	L2R&R2L	Shallow	Shallow
T5	Enc-Dec	NLU&NLG	L2R	Shallow	No
BART	Enc-Dec	NLU&NLG	L2R	Shallow	No
PEGASUS	Enc-Dec	NLG	L2R	Shallow	No
PALM	Enc-Dec	NLG	L2R	Shallow	No
MASS	Enc-Dec	NLG	L2R	Shallow	No
ProphetNet	Enc-Dec	NLG	L2R	Medium	No
P^{3}LM	Enc-Dec	NLG	Permuted	Strong	Strong

Table 1: Features about typical pre-trained models. **Enc**: encoder. **Dec**: decoder. **Order**∈{L2R, R2L, Permuted}: decoding order of target sequence. **LongDep**∈{Shallow, Medium, Strong}: long token dependencies in target sequence. **BiDir**∈{No, Shallow, Strong}: bidirectional information of target sequences.

2 Related Work

Typical pre-trained language models are shown in Table 1, which can be roughly classified into two categories: for natural language understanding (NLU) and for natural language generation (NLG). Models (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Lan et al., 2020; Clark et al., 2020) that contain a single encoder, have been proved effective for dozens of downstream NLU tasks, e.g., XLNet (Yang et al., 2019) reconstruct a sentence fragment in permuted order. Another line of research is generative pre-training for NLG. Effective methods have been designed to enhance NLG performance. These models usually pre-train the decoder as a left-to-right (L2R) autoregressive language model. GPT-3 (Brown et al., 2020) pre-train a transformer decoder with extremely large corpus and parameters, which is not finetuned on downstream tasks, while our model follows the pre-train then finetune framework. UniLM (Dong et al., 2019) pre-train a transformer encoder/decoder with both MLM task and sequence-to-sequence task, considering two unidirectional orders, i.e., L2R and R2L, while our model leverages permuted orders. Additional strong generative pre-trained models including MASS(Song et al., 2019), BART(Lewis et al., 2020), T5 (Raffel et al., 2019), and PEGASUS (Zhang et al., 2019) utilize a transformer encoder-decoder framework to pre-train generative models, all of which are limited to train a L2R decoder, while our model learns to decode tokens in permuted order. ProphetNet (Qi et al., 2020) is the most similar approach to ours, which propose a future n-gram prediction mechanism for generative pre-training, while still limited to L2R decoding.

3 Approach

3.1 Model Overview

In this paper, probabilistically permuted prophet language modeling (P^3LM) is proposed for sequence generation. The idea of P^3LM is learning to autoregressively generate a sequence in a probabilistically permuted order, meanwhile, multiple future tokens (in the perspective of that order) are jointly predicted at each decoding time step. The above design of P^3LM makes it capable of capturing bidirectional information of a target sequence, as well as strengths the modelling of long dependencies in natural language.

3.1.1 Prophet Language Modeling

To alleviate the problem of strong local dependencies during sequence generation, we introduce prophet decoding. The original prophet modeling predicts N words after current word. It is first utilized in Word2Vec (Mikolov et al., 2013), where increasing range N improves the word vector quality. ProphetNet (Qi et al., 2020) introduces it into sequence generation by predicting the future N tokens. Formally, given $X = [x_1, ..., x_S]$ as a source sequence, and $Y = [y_1, ..., y_T]$ as a target sequence. The learning of a prophet language model (PLM) is to optimize the objective defined as follows:

$$\mathcal{L}_{plm}(Y|X) = \frac{1}{N}\sum_{n=1}^{N} \log p_\theta^n(Y|X)$$

where θ represents trainable parameters. $p_\theta^n(Y|X)$ is the probability of generating Y by skipping $n \in \{1, ..., N\}$ tokens at each decoding time step t defined as follows:

$$p_\theta^n(Y|X) = \prod_{t=1}^{T} p_\theta(y_t|y_{\leq t-n}, X)$$

In details, the prophet decoding can be viewed as a kind of masking strategy on previous generated sequence, namely, only $y_{\leq t-n}$ are feasible for predicting y_t. In practice, to train models within reasonable computational complexity, N is typically set as a small number, e.g., N is 4 for Word2Vec, and 2 for ProphetNet. This limits its ability of modeling long dependencies existing in natural language, such as long distance coreferences, clause dependencies, and discourse relations. Based on prophet decoding, we introduce P^3LM to address the problem in next section.

3.1.2 P^3LM: Probabilistically Permuted Prophet Language Modeling

Although prophet decoding is capable of alleviating the problem of strong local dependencies, its ability of long dependency modeling is still limited by small N as described above, and it is constrained on unidirectional information due to L2R decoding. The L2R order is a strong inductive bias, as it is natural for most human-beings to read and write sequences from left to right. Nevertheless, L2R is not the only option for generating sequences (Gu et al., 2019). For instance, people sometimes tend to think of central phrases first before building up a whole sentence. Previous work has shown that order matters for sequence generation (Vinyals et al., 2016; Emelianenko et al., 2019). Based on the above facts, a natural idea is to involve sequence order information into decoding. To this end, we propose P^3LM to strengthen prophet language model with probabilistically permuted sequence order, which is capable of directly modeling long dependencies and bidirectional information of target sequences. Formally, as previous study (an, 2019), we condition the whole process on an input sequence X to indicate that the proposed model is applicable to both conditional and unconditional sequence generation ($X = \varnothing$). Specifically, the probability of generating Y with prophet is defined as the expectation of its posterior probability $p_\theta^n(Y|X, Z)$ over all possible orders as follows:

$$p_\theta^n(Y|X) = \mathbb{E}_{Z\sim p(Z)} p_\theta^n(Y|X, Z)$$

where order $Z = [z_1, ..., z_T] \in P^*(T)$[3], which is a permutation of positions in Y, subjects to a prior distribution $p(Z)$. The decoding is further factorized according to order Z as

$$p_\theta^n(Y|X, Z) = \prod_{t=1}^{T} p_\theta(y_{z_t}|y_{z_{\leq t-n}}, X)$$

where y_{z_t} represents the t-th generated token and z_t is its absolute position in Y. Training such a model needs to enumerate all the $T!$ permutations, which is impractical. Instead, we maximize the lower bound $\mathcal{L}(Y|X)$ of the log likelihood $\mathcal{L}_{p^3lm}(Y|X)$ by sampling an order $\tilde{Z}$ according to the prior dis-

[3] $P^*(T)$ is the set of all permutations of $\{i\}_{i=1}^{i=T}$.

tribution $p(Z)$ as follows:

$$\begin{aligned}\mathcal{L}_{p^3lm}(Y|X) &= \frac{1}{N}\sum_{n=1}^{N}\log p_\theta^n(Y|X)\\ &= \frac{1}{N}\sum_{n=1}^{N}\log \mathbb{E}_{Z\sim p(Z)} p_\theta^n(Y|X,Z)\\ \geq \underbrace{\mathcal{L}(Y|X)}_{\text{lower bound}} &= \frac{1}{N}\sum_{n=1}^{N}[\log p_\theta^n(Y|X,\tilde{Z})+\log p(\tilde{Z})]\\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\log p_\theta(y_{\tilde{z}_t}|y_{\tilde{z}_{\leq t-n}},X)+C\end{aligned}$$

Theoretically, different distribution $p(Z)$ will result in different P[3]LMs. Exploring the best distribution $p^*(Z)$ could be an interesting problem for future research. In this paper, we preliminarily explore an α-P[3]LM which combines L2R and URP distributions which are defined as follows:

- **L2R** order $Z^{\text{L2R}} = [1, ..., T]$ is the left-to-right position sequence of words in Y. Most previous methods train a model to generate target sequences in L2R order. The corresponding $p(Z)$ of these methods, which is a pulse distribution, is defined as follows:

$$p^{\text{L2R}}(Z) = \begin{cases} 1, & Z = Z^{\text{L2R}} \\ 0, & Z \neq Z^{\text{L2R}} \end{cases}$$

- **URP** order means an uniformly random permutation of the word positions in Y. The corresponding $p(Z)$, which is an uniform distribution over the $T!$ permutations $P^*(T)$, is defined as follows:

$$p^{\texttt{URP}}(Z) = \frac{1}{T!}, \quad Z \in P^*(T)$$

We believe that the diverse URP orders (not only L2R) can help strengthen the modeling of bidirectional information and long dependencies of target sequences. Finally, the order distribution of α-P[3]LM is straightforwardly defined as

$$p^\alpha(Z) = \alpha p^{\text{L2R}}(Z) + (1-\alpha)p^{\texttt{URP}}(Z)$$

In this paper, we empirically set $\alpha = 0.5$ according to experiments. Besides, unlike previous works that focus on automatically determining a best generation order during inference (Gu et al., 2019; Emelianenko et al., 2019; an, 2019) which requires nontrivial design, we focus on modeling orders during training and keep the L2R inference to reduce the complexity of the model.

Algorithm 1: P[3]LM Decoder $\texttt{dec}(\cdot)$

Input: Target sequence Y, order Z, place holders $\{q_{z_t}^n\}_{t=1,n=1}^{t=T,n=N}$, # of query streams N, encoder hidden states $\mathbf{h}^e$

Output: Prediction probabilities $\{p_\theta(y_{z_t}|y_{z_{\leq t-n}},X)\}_{t=1,n=1}^{t=T,n=N}$

Create 0-initialized tensors $O^0 \in \mathbb{R}^{T\times T}$ and $O^1, ..., O^N \in \mathbb{R}^{T\times 2T}$ as relative orders

Set $O^*(*, 1) = 1$

for $i \leftarrow 1$ **to** T, $j \leftarrow 1$ **to** T **do**

  let $z_{t1} = i$ and $z_{t2} = j$

  $O^0(j+1, i+1) = 1$ **if** $t1 \leq t2, i, j \neq T$

  for $n \leftarrow 1$ **to** N **do**

    $O^n(j, i+1) = 1$ **if** $t1 \leq t2-n, i \neq T$

    $O^n(j, i+T) = 1$ **if** $t1 = t2$

embed $[\langle \texttt{s} \rangle, Y]$ as $\mathbf{h}$ and q^n as $\mathbf{g}^n$, where $q^n = \{q_{z_t}^n\}_{t=1}^{t=T}$ and $\mathbf{g}^n = \{\mathbf{g}_{z_t}^n\}_{t=1}^{t=T}$

for $k \leftarrow 1$ **to** K **do**

  $\mathbf{h} \leftarrow \texttt{OSA}_{\phi^0}(\mathbf{h}, \mathbf{h}, \mathbf{h}, O^0)$

  $\mathbf{h} \leftarrow \mathbf{h}$ encoder attention on $\mathbf{h}^e$

  for $n \leftarrow 1$ **to** N **do**

    $\mathbf{g}^n \leftarrow \texttt{OSA}_{\phi^n}(\mathbf{g}^n, [\mathbf{h};\mathbf{g}^n], [\mathbf{h};\mathbf{g}^n], O^n)$

    $\mathbf{g}^n \leftarrow \mathbf{g}^n$ encoder attention on $\mathbf{h}^e$

return $p_\theta^n(y_{z_t}|y_{z_{\leq t-n}},X) = \texttt{softmax}(\mathbf{g}_{z_t}^n W)$

3.2 Neural Architecture of P[3]LM

The backbone of the proposed P[3]LM is a transformer encoder-decoder (Vaswani et al., 2017) illustrated in Figure 2. The encoder transfers X into hidden states $\mathbf{h}^e = \texttt{enc}(X)$ where $\texttt{enc}(\cdot)$ is a standard transformer encoder. According to the objective $\mathcal{L}(Y|X)$ defined above, during training, the P[3]LM decoder *simultaneously* calculates $T\times N$ probabilities as follows:

$$\{p_\theta(y_{z_t}|y_{z_{\leq t-n}}, X)\}_{t=1,n=1}^{t=T,n=N} = \texttt{dec}(Y,Z,N,\mathbf{h}^e)$$

Compared with the vanilla transformer decoder, our decoder $\texttt{dec}(\cdot)$ has two characteristics: (1) it takes an order Z as additional input to guide the autoregressive generation, and (2) it can simultaneously skip $[1, ..., N]$ (Note that $N=1$ means the next token prediction) *previous* tokens for prediction at each time step. To achieve the above two capabilities, we implement an *order-aware multi-stream* P[3]LM decoder with its workflow shown in Algorithm 1. The major effort of such a decoder is to model the absolute order Z as ***relative orders*** in multi-stream attention, aiming to control what information to use or not for decoding.

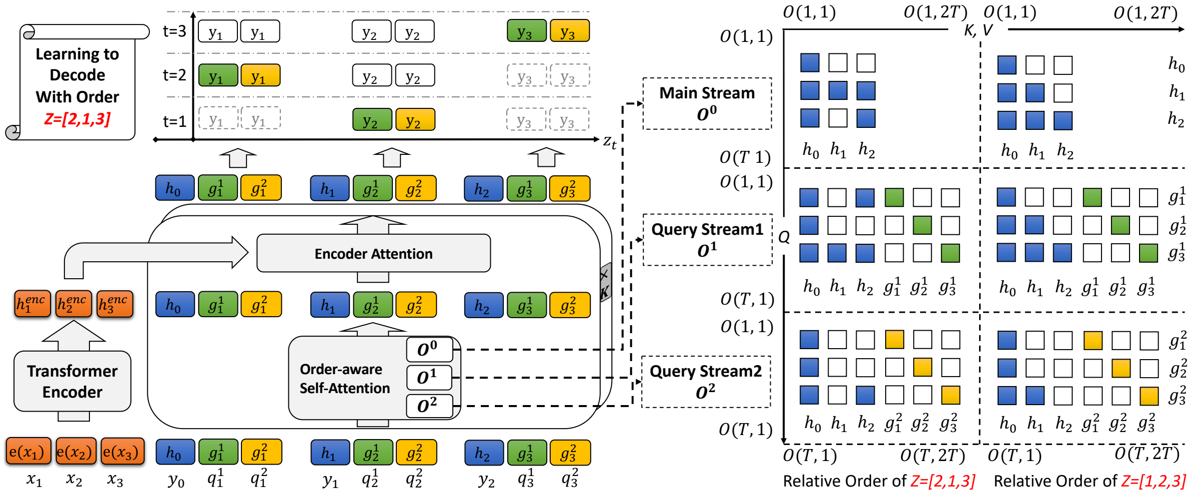

Figure 2: Neural architecture of P3LM (Left), and sampled relative orders (Right) as examples. A case is shown that sequence $y_2 \to y_1 \to y_3$ is decoded in order $Z = [2, 1, 3]$ where $y_0 = \langle s \rangle$ is fixed as the start of sentence. Encoder is in red, main stream is in blue, query stream 1 is in green, and query stream 2 is in yellow. Colored items in the masking tensors mean that the corresponding input is available for computing the corresponding output.

Multi-Stream. The original multi-stream attention has been successfully utilized in XLNet (Yang et al., 2019). Different from XLNet which leverages a 2-stream attention in *encoder for NLU* tasks, we adopt an $(N+1)$-stream attention where $N \geq 1$ in *decoder for NLG* tasks like ProphetNet (Qi et al., 2020). Unlike ProphetNet, attention in P3LM decoder is order sensitive. Specifically, as shown in Figure 2 and lines 9-16 in Algorithm 1, at each time step t, P3LM decoder leverages a main stream (in blue) as the vanilla transformer decoder to represent $y_{z_{<t}}$ as hidden states $\mathbf{h}_{z_{<t}}$. In addition, it constructs N query streams (in green and yellow) to represent N place holders $q_{z_t} = [q^1_{z_t}, ..., q^N_{z_t}]$ as hidden states $\mathbf{g}_{z_t} = [\mathbf{g}^1_{z_t}, ..., \mathbf{g}^N_{z_t}]$. Each query stream is used to predict y_{z_t} by skipping $n \in [1, ..., N]$ tokens, respectively. The above multi-stream transformation is implemented with K layers (line 10), each of which contains two sub-layers, i.e., order-aware self-attention $\texttt{OSA}(\phi)$ (line 11 and 14) introduced in next section and encoder-attention (line 12 and 15). Finally, the distribution of predicting y_{z_t} at the n-th stream is defined as $p^n_\theta(y_{z_t}|y_{z_{<t-n}}, x) = \texttt{softmax}(\mathbf{g}^n_{z_t} W)$ (line 16) where $W \in \mathbb{R}^{D\times V}$ are trainable parameters, D indicates the hidden size, and V represents the vocabulary size.

Order-aware Self-Attention (OSA). To involve the order information, an intuitive solution is to directly reorder Y into a new sequence Y' according to Z, and then learns to decode Y' with an L2R decoder. However, it will mismatch word and position embeddings, which leads to the loss of the words' original positional information. Instead, we introduce an order-aware self-attention $\texttt{OSA}_\theta(\cdot)$ which leverages relative orders and keeps the positions of the words inputted into the decoder unchanged. Specifically, the absolute order Z is converted into relative order $O^0 \in \mathbb{R}^{T\times T}$ for main stream, and a set of relative orders $O^1, ..., O^N \in \mathbb{R}^{T\times 2T}$ for query streams (lines 1-8). $O(j, i)$ indicates the item in the j-th row and i-th column of a matrix O. In short, these relative orders act as attention masks, controlling that words with their order in front are available for those behind. Finally, $\texttt{OSA}_\phi(\cdot)$ taking in packed hidden states Q, K, V and some relative order O is defined as follows:

$$\texttt{OSA}_\phi(Q, K, V, O)$$
$$= \texttt{softmax}\left(\frac{(QW^Q)\cdot(KW^K)^\top \odot O}{\sqrt{D}}\right)\cdot(VW^V)$$

where $W^Q, W^K, W^V \in \phi$ are trainable parameters.

4 Experiment

Extensive experiments are conducted. In Section 4.1, pre-training details of P3LM are introduced. In Section 4.2, we show that P3LM achieves state-of-the-art (SOTA) results on GLGE benchmark compared with published methods. In Section 4.3, we conduct experiments on text summarization dataset CNN/DM, where ablation study verifies the effectiveness of P3LM which involves sequence order information compared with conventional left-to-right (L2R) generation paradigm.

4.1 P^{3}LM Pre-training

4.1.1 Model Architecture

P^{3}LM follows the transformer encoder-decoder framework. Two model architectures, i.e., P^{3}LM$_{\texttt{base}}$ and P^{3}LM$_{\texttt{large}}$ are used for pre-training. The base architecture contains about 125M parameters including a 6-layer encoder and a 6-layer decoder with 768 embedding/hidden size and 3,072 feed-forward filter size. The architecture of the large model contains about 391M parameters including a 12-layer encoder and 12-layer decoder with 1,024 embedding/hidden size and 4,096 feed-forward filter size.

4.1.2 Corpus and Infrastructure

Following BERT and ProphetNet (Qi et al., 2020), the English Wikipedia and BookCorpus are used to pre-train P^{3}LM. In this paper, to keep up with previous work, we first collect and process the above datasets, and finally obtain about 16GB data for pre-training. We pre-train a P^{3}LM$_{\texttt{base(16G)}}$ and a P^{3}LM$_{\texttt{large(16G)}}$ model on the 16GB dataset with 64 $\times$ 32GB NVIDIA V100 GPUs from scratch without loading any other pre-trained models. Following ProphetNet on large scale pre-training, we also collect a 160GB large scale dataset which is the combination of five sources including wikipedia, books, stories, news, and web text. Based on the 160G data, we also pre-train a large scale model P^{3}LM$_{\texttt{large(160G)}}$ initialized by P^{3}LM$_{\texttt{large(16G)}}$ with 16 $\times$ 40GB NVIDIA A100 GPUS. The batch size of all the three pre-trained models are set as 1,024. The P^{3}LM$_{\texttt{base(16G)}}$, P^{3}LM$_{\texttt{large(16G)}}$, and P^{3}LM$_{\texttt{large(160G)}}$ are trained with 750k (95 epochs), 1,500k (192 epochs), and 2000k (22 epochs) iterations and cost about 1.7 days, 28.0 days, 48.6 days, respectively. We use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1e-4 for pre-training. Our implementation is based on FAIRSEQ[4]. To make a fair comparison, we set the maximum future N-token to be 2 as ProphetNet in experiments.

4.1.3 Pre-Training Task

P^{3}LM is essentially a sequence-to-sequence model which takes a sequence as input and outputs a target sequence. During pre-training, the input length is set to 512 tokens. We randomly pick a starting position u in every 64 tokens, and then mask a continuous span from u. The masked length is set to 15% of the total number of input tokens, i.e., 9 continuous tokens in every 64 tokens. Following ProphetNet and MASS (Song et al., 2019), among the masked tokens, 80% of them are replaced by $[\mathbb{M}]$, 10% replaced by random tokens, and 10% unchanged. Considering the computational cost, we follow MASS to only predict the masked fragment. Different from ProphetNet and MASS, P^{3}LM predicts the target sequence in both an L2R order and a URP order. Specifically, a URP sequence generation task is to generate a target sequence word by word in a given URP sequence order. Traditional L2R sequence generation task trains a generative model which only needs to learn a fixed *one-word-right* relative positional information. In contrast, URP sequence generation requires a model to learn more complex *arbitrarily* relative positional information between words in a target sequence.

4.2 Finetune on General Generation Tasks

In this section, we show the finetune results of P^{3}LM compared with strong baselines and state-of-the-art pre-trained models on GLGE[5] (Liu et al., 2021), which is a general language generation evaluation benchmark containing 8 datasets on 4 tasks.

4.2.1 GLGE Benchmark

Table 3 shows the statistics of GLGE benchmark. GLGE is a general language generation evaluation benchmark consisting of four datasets for summarization including CNN/DM (Hermann et al., 2015; See et al., 2017), Gigaword (Rush et al., 2015; Graff et al., 2003), XSum (Narayan et al., 2018), and MSNews, two for question generation including SQuAD 1.1 (Rajpurkar et al., 2016), and MSQG, one for conversational question answering including CoQA (Reddy et al., 2019), and one for dialog response generation including PersonaChat (Zhang et al., 2018). Statistics of GLGE are show in Table 3. Evaluation metrics, including Rouge-1, ROUGE-2, and ROUGE-L (Lin, 2004) for summarization, ROUGE-L, BLEU-4 (Papineni et al., 2002), and METEOR (Banerjee and Lavie, 2005) for question generation, F1 for conversational question answering, and BLEU-1, BLEU-2, Distinct-1, and Distince-2 (Li et al., 2016) for dialog response generation, are used. GLGE calculates an overall score $s = \frac{1}{8}\sum_{d=1}^{8}\frac{1}{|S_d|}\sum_{m\in S_d} m$ where S_d indicates the evaluation metrics for the d-th dataset.

[4] https://fairseq.readthedocs.io/en/latest/

[5] https://microsoft.github.io/glge/

Models	Score	Text Summarization				Question Generation		QA	Dialog
		CNN/DM	Gigaword	XSUM	MSNews	SQuAD1.1	MSQG	CoQA	PersonaChat
Metrics		R-1/R-2/R-L				R-L/B-4/MTR		F1	B-1/B-2/D-1/D-2
Test									
LSTM	20.0	37.3/15.7/34.4	34.2/16.0/31.8	25.1/6.9/19.9	30.0/14.6/27.7	27.2/3.8/8.9	25.3/3.5/14.1	15.1	42.2/35.9/0.2/0.7
Transformer	21.9	39.5/16.7/36.7	37.1/18.4/34.5	30.5/10.4/24.2	33.0/15.4/30.0	30.7/4.8/10.9	29.3/5.1/16.6	15.7	38.3/33.6/0.2/0.7
$\text{MASS}_{\texttt{base}}$	33.6	42.1/19.5/39.0	38.7/19.7/35.9	39.7/17.2/31.9	39.4/21.0/36.1	49.4/20.1/24.4	38.9/10.2/23.3	65.4	41.0/35.7/1.4/6.9
$\text{ProphetNet}_{\texttt{base}}$	33.8	42.5/19.7/39.5	38.9/19.9/36.0	39.8/17.1/32.0	40.6/21.6/37.0	48.0/19.5/23.9	37.1/9.3/22.7	65.3	46.0/38.4/1.3/7.3
$\text{MASS}_{\texttt{middle}}$	34.3	42.9/19.8/39.8	38.9/20.2/36.2	39.1/16.5/31.4	40.4/21.5/36.8	49.9/21.3/25.2	38.9/9.5/23.5	67.6	46.0/38.2/1.2/6.2
$\text{BART}_{\texttt{large}}$	35.8	44.1/**21.2**/40.9	38.1/18.4/34.9	45.1/22.2/37.2	43.8/24.0/39.2	50.3/22.0/26.4	38.8/9.2/**24.3**	68.6	**49.9**/**40.0**/1.3/8.0
$\text{ProphetNet}_{\texttt{large}}$	36.5	44.2/21.1/41.3	39.5/**20.4**/36.6	44.4/21.3/36.4	44.1/24.4/40.2	51.5/22.5/26.0	38.3/9.6/23.3	73.0	46.7/39.0/1.3/7.5
$\text{P}^3\text{LM}_{\texttt{large(160G)}}$	**37.4**	**44.3**/21.0/**41.4**	**39.6**/20.2/**36.8**	**45.3**/**22.3**/**37.3**	**44.6**/**25.0**/**40.8**	**51.6**/**23.0**/**26.6**	**39.5**/**11.0**/23.6	**75.3**	48.8/39.4/**1.7**/**13.7**
Valid									
$\text{P}^3\text{LM}_{\texttt{large(160G)}}$	38.7	44.8/21.5/42.0	48.8/26.8/45.6	45.4/22.5/37.6	44.2/24.5/40.4	52.5/24.3/27.1	39.9/12.9/24.4	75.9	49.0/39.4/1.7/13.5

Table 2: Experiment results on the GLGE. Overall scores $s = \frac{1}{8}\sum_{d=1}^{8}\frac{1}{|S_d|}\sum_{m \in S_d} m$ where S_d is the metrics for the d-th dataset are highlighted in color. R-1: Rouge-1. R-2: Rouge-2. R-L: Rouge-L. B-4: BLUE-4. MTR: METEOR. D-1: Distinct-1. D-2:Distinct-2. It is worth noting that D-1 and D-2 are multiplied by 100. Highest scores are in bold. The gains of $\text{P}^3\text{LM}_{\texttt{large(160G)}}$ over $\text{ProphetNet}_{\texttt{large(160G)}}$ are statistically significant at $p = 0.05$.

Corpus	|Train|	|Dev|	|Test|	|Src.|	|Tgt.|
CNN/DM	287,113	13,368	11,490	822.3	57.9
Gigaword	3,803,957	189,651	1,951	33.7	8.7
XSUM	204,017	11,327	11,333	358.5	21.1
MSNews	136,082	7,496	7,562	310.7	9.7
SQuAD 1.1	75,722	10,570	11,877	149.4	11.5
MSQG	198,058	11,008	11,022	45.9	5.9
CoQA	108,647	3,935	4,048	354.4	2.6
PersonaChat	122,499	14,602	14,056	120.8	11.8

Table 3: Statistics of GLGE tasks. |Train|: the number of examples in training set. |Src.|: the average number of words in source inputs.

4.2.2 Baselines

We choose the following well performed pre-trained generative models as our baselines. **LSTM** (Bahdanau et al., 2015) is implemented with the word embedding dimension, the hidden size, the number of the encoder layer, and the number of the decoder layer as 512, 512, 1, and 1, respectively. LSTM is trained for a maximum of 100 epochs with learning rate of between 1e-4 and 3e-4. **Transformer** (Vaswani et al., 2017) contains a 6-layer encoder and a 6-layer decoder with 1024 embedding and hidden size, and 4096 feed-forward filter size. Transformer is trained for a maximum of 20 epochs with learning rate of between 1e-4 and 3e-4. **MASS** (Song et al., 2019) includes $\text{MASS}_{\texttt{base}}$ and $\text{MASS}_{\texttt{middle}}$ containing a 6-layer encoder and a 6-layer decoder with 768/1024 embedding and hidden size and 3072/4096 feed-forward filter size. MASS are pre-trained on the 16GB English Wikipedia and BookCorpus dataset and finetuned for a maximum of 25 epochs. $\textbf{BART}_{\texttt{large}}$ (Lewis et al., 2020) contains a 12-layer encoder and 12-layer decoder with 1024 embedding and hidden size, and 4096 feed-forward filter size. BART is pre-trained based on the 160GB data of news, books, stories, and web text and finetuned for a maximum of 20,000 iterations. **ProphetNet** (Qi et al., 2020) includes $\text{ProphetNet}_{\texttt{base}}$ and $\text{ProphetNet}_{\texttt{large}}$ containing the same architecture as the corresponding P^3LM models, where the base model is pre-trained on the 16GB English Wikipedia and BookCorpus, and the large one on the 160GB corpora. ProphetNet is finetuned for a maximum of 10 epochs.

4.2.3 Implementation Details

$\text{P}^3\text{LM}_{\texttt{large(160G)}}$ is pre-trained on the same 160GB data as $\text{ProphetNet}_{\texttt{large}}$ as described in Section 4.1, and then finetuned on the eight datasets in GLGE, respectively. The best performing model on each development set is chosen to inference on the corresponding test set. Due to space limitation, implementation details about the eight models are show in Table 6 in Appendix B.

4.2.4 Main Results

Table 2 shows the results of $\text{P}^3\text{LM}_{\texttt{large(160G)}}$ and above strong baselines. $\text{P}^3\text{LM}_{\texttt{large(160G)}}$ outperforms all these published methods on GLGE according to the overall score. Specifically, compared with the score 36.5 of $\text{ProphetNet}_{\texttt{large}}$ which is the state-of-the-art published method, the score of our proposed $\text{P}^3\text{LM}_{\texttt{large(160G)}}$ is 37.4, which achieves 0.9 absolute and 2.5% relative improvements. From the perspective of different tasks, the average scores of our model are 34.9/29.2/75.3/25.9, which are 34.5/28.5/73.0/23.6 for $\text{ProphetNet}_{\texttt{large}}$, on text summarization, question generation, question answering, and persona dialog response generation, respectively. Our model achieves +0.6/+0.7/+2.3/+2.3 absolute im-

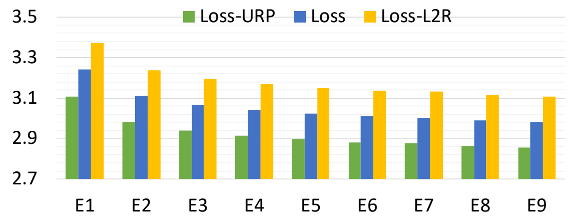

Figure 3: Losses of each epoch during the pre-training of P^{3}LM$_{\texttt{large(160G)}}$. **Same tend for perplexity**.

provements. Based on the above results, the effectiveness of P^{3}LM is verified again. Besides, L2R inference is explained in Appendix C and the effect of pre-training iterations is shown in Appendix D.

4.2.5 Order Matters for Language Modeling

To explore the effect of orders, we split the loss of α-P^{3}LM into two parts, i.e., loss-URP and loss-L2R. The first part corresponds to $\alpha p^{\texttt{URP}}$ in p^α, and the second corresponds to $(1-\alpha)p^{\texttt{L2R}}$ in p^α. Figure 3 shows that the loss-URP fits faster than loss-L2R. Since the perplexity $ppl = 2^{loss}$, we conclude that **URP order achieves lower perplexity than L2R order**, i.e., the difficulty of modeling natural language sentence in an L2R order is larger than the average level reflected by the URP order. This observation indicates that sequence order matters for language modelling. In future, we will consider to train P^{3}LM in orders considering syntactical information, e.g., a level-order traversal of the syntactic tree of a natural language sequence.

4.3 Finetuning on Text Summarization.

Abstractive text summarization as a typical NLG task, aims to generate a short and fluent summary of a long text document. In this section, we finetune and evaluate the proposed P^{3}LM on a text summarization dataset CNN/DM introduced before.

4.3.1 Experiment Settings

A base and a large models on CNN/DM with batch size as 512 are finetuned, and max epochs are set as 25 and 15, respectively. Adam optimizer is used to update the parameters of the model with a learning rate of 1e-4 and warm-up updates of 1,000. Model with the best rouge score on the validation set is used for testing. Although a URP order is used for training the P^{3}LM, we use beam search with an L2R order to generate summaries during inference. Beam size is set as 5 for both the base and large models. The length of the target sequence is limited between 45 and 110 with a length penalty as 1.2.

Method	R-1	R-2	R-L
w/o pre-training			
LEAD-3	40.42	17.62	36.67
PGNet	36.44	15.66	33.42
PGNet+Coverage	39.53	17.28	36.38
Bottom-Up	41.22	18.68	38.34
w/ pre-training			
S2S-ELMo	41.56	18.94	38.47
BERT-SUMABS	41.72	19.39	38.76
BERT-SUMEXTABS	42.13	19.60	39.18
MASS	42.12	19.50	39.01
UniLM	43.33	20.21	40.51
PALM	42.71	19.97	39.71
ProphetNet$_{\texttt{base(16G)}}$	42.52	19.78	39.59
ProphetNet$_{\texttt{large(16G)}}$	<u>43.68</u>	<u>20.64</u>	<u>40.72</u>
P^{3}LM$_{\texttt{base(16G)}}$	42.90	19.98	39.93
P^{3}LM$_{\texttt{large(16G)}}$	**44.07**	**20.82**	**41.15**

Table 4: Experiment results on CNN/DM. Pre-training corpus of all methods is less than 18GB. Highest scores are in bold, and seconds are underlined. The gains of P^{3}LM$_{\texttt{base(16G)}}$ over ProphetNet$_{\texttt{base(16G)}}$, and P^{3}LM$_{\texttt{large(16G)}}$ over ProphetNet$_{\texttt{large(16G)}}$ are statistically significant at $p = 0.05$.

4.3.2 Baselines

Popular baselines are compared for evaluation. LEAD-3 (Nallapati et al., 2017) takes the first three sentences as the summary; PGNet (See et al., 2017) is Seq2Seq model incorporated with a copy mechanism; PGNet+Coverage (See et al., 2017) introduces a coverage mechanism to PGNet; BottomUp (Gehrmann et al., 2018) employs a bottom-up content selector based on Seq2Seq model; S2S-ELMo (Edunov et al., 2019) uses the pre-trained ELMo (Radford et al.) representations for generation. Several pre-training based strong baselines including BERTSUMABS (Liu and Lapata, 2019), MASS, UniLM (Dong et al., 2019), PALM (Bi et al., 2020), and ProphetNet are also compared.

4.3.3 Experiment Results

Table 4 shows experiment results of models without pre-training or pre-trained on the less than 18GB wikipedia and bookcorpus dataset, where ELMo is an exception that it is trained on a 5GB dataset. Results show that P^{3}LM outperforms the baselines and achieves the best performance. Specifically, our base and large models achieve +0.38/+0.20/+0.34 and +0.39/+0.18/+0.43 improvements compared with corresponding ProphetNet models in terms of R-1, R-2, and R-L. We think the improvements come from the P^{3}LM decoding that strengthens bi-direction information and long dependencies modeling of target sequences.

Init	Settings	R-1	R-2	R-L
w/o pre-training				
base	$N=1, p^{\mathtt{L2R}}$	40.33	17.64	37.35
	$N=2, p^{\mathtt{L2R}}$	40.65	17.94	37.67
	$N=2, p^{\mathtt{URP}}$	36.59	15.57	34.42
	$N=2, p^{\mathtt{URP}} \to p^{\mathtt{L2R}}$	41.32	18.57	38.46
	$N=2, p^{\alpha}$	**41.38**	**18.60**	**38.47**
large	$N=1, p^{\mathtt{L2R}}$	40.53	17.54	37.61
	$N=2, p^{\mathtt{L2R}}$	41.04	18.12	38.08
	$N=2, p^{\mathtt{URP}}$	36.19	15.43	33.84
	$N=2, p^{\mathtt{URP}} \to p^{\mathtt{L2R}}$	**42.00**	**19.02**	**39.06**
	$N=2, p^{\alpha}$	41.45	18.66	38.42
w/ pre-training				
base(`16G`)	$N=1, p^{\mathtt{L2R}}$	42.32	19.45	39.27
	$N=2, p^{\mathtt{L2R}}$	42.56	19.57	39.56
	$N=2, p^{\mathtt{URP}}$	38.35	17.00	36.19
	$N=2, p^{\mathtt{URP}} \to p^{\mathtt{L2R}}$	**42.92**	**19.98**	**39.93**
	$N=2, p^{\alpha}$	42.90	**19.98**	**39.93**
large(`16G`)	$N=1, p^{\mathtt{L2R}}$	43.27	20.10	40.23
	$N=2, p^{\mathtt{L2R}}$	43.43	20.18	40.46
	$N=2, p^{\mathtt{URP}}$	38.94	17.61	36.76
	$N=2, p^{\mathtt{URP}} \to p^{\mathtt{L2R}}$	43.60	20.59	40.66
	$N=2, p^{\alpha}$	**44.07**	**20.82**	**41.15**

Table 5: Ablation study of different finetuning settings on CNN/DM, with pre-training or not.

4.3.4 Ablation Study

To further verify the effectiveness of the proposed P^{3}LM, we conduct ablation study on different finetuning settings. We investigate different combinations of finetuning settings and show the results in Table 5. Specifically, $p^{\mathtt{URP}} \to p^{\mathtt{L2R}}$ means the model is firstly trained several epochs on sampled instances with orders subjecting to distribution $p^{\mathtt{URP}}$ and then several epochs to distribution $p^{\mathtt{L2R}}$. We first observe that prophet mechanism ($N=2$) brings improvements. More importantly, compared with $p^{\mathtt{L2R}}$, P^{3}LM introduces $p^{\mathtt{URP}}$, where we can see that the $p^{\mathtt{URP}} \to p^{\mathtt{L2R}}$ and p^{α} achieve the best performance when loading the pre-trained P^{3}LM$_{\mathtt{base(16G)}}$ and P^{3}LM$_{\mathtt{large(16G)}}$ models. Furthermore, when loading no pre-trained models, P^{3}LM trained based on $p^{\mathtt{URP}} \to p^{\mathtt{L2R}}$ and p^{α} still improve traditional `L2R` training a lot. P^{3}LM with only $p^{\mathtt{URP}}$ performs the worst, which is reasonable since the model only uniformly selects one permutation of a target as training data, which is completely inconsistent with the L2R inference. It further indicates that, although the L2R order is only one special case of all $T!$ permutations, it is still important and should be paid more attention as our α-P^{3}LM do.

5 Conclusion

A probabilistically permuted prophet language modeling, P^{3}LM, is proposed for generative pre-training. P^{3}LM models sequences by considering both left-to-right and random permutation orders, equipped with a prophet mechanism for future token prediction. Extensive experiments are conducted on GLGE, a general natural language generation evaluation benchmark, where P^{3}LM achieves state-of-the-art results compared with public available generative pre-training methods.

Limitations

Exploring Better Distribution $p^*(Z)$

Figure 3 shows that URP loss fits faster than L2R loss. Since L2R is a special case of URP order, we think that the difficulty of modeling natural language sentence in an L2R order is larger than the average level reflected by the URP order. It indicates that sequence order really matters for language modelling and exploring other distribution $p^*(Z)$ besides p^{α} could be an interesting problem. In future, we will consider to train P^{3}LM in orders considering syntax, e.g., a level-order traversal of the syntactic tree of a natural language.

Training-Inference Consistency

P^{3}LM decodes a sequence in an order sampled from p^{α} during training. Different from training, P^{3}LM performs L2R decoding during inference. Nevertheless, P^{3}LM achieves significant improvements across multiple tasks and datasets. We think this benefits from the involving of P^{3}LM decoding which introduces more constraints to help the model to learn bidirectional context and long dependency modeling. In future, we will explore to decode a sequence in terms of an optimized order, not limited in L2R order.

Training Efficiency.

The model construction and network structure is as complex as ProphetNet. The key point of P^{3}LM is utilizing sampled orders according to a given distribution as the attention mask in transformer decoder. This makes the computation cost of P^{3}LM similar to ProphetNet when sampling one order for a target sequence. In this paper, according to experiments, we sample two orders from p^{α} for training, this makes training one instance in one epoch twice the time of ProphetNet.

Acknowledge

Supported by the National Key R&D Program of China under Grant No. 2020AAA0108600.

References

Elman Mansimov an. 2019. A generalized framework of sequence generation with application t. *ArXiv preprint*, abs/1905.12790.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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

Bin Bi, Chenliang Li, Chen Wu, Ming Yan, Wei Wang, Songfang Huang, Fei Huang, and Luo Si. 2020. PALM: Pre-training an autoencoding&autoregressive language model for context-conditioned generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8681–8691, Online. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. 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 13042–13054.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4052–4059, Minneapolis, Minnesota. Association for Computational Linguistics.

Dmitrii Emelianenko, Elena Voita, and Pavel Serdyukov. 2019. Sequence modeling with unconstrained generation order. 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 7698–7709.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

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*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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*, pages 7871–7880, Online. 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 *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

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

Dayiheng Liu, 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*, pages 408–420, Online. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ArXiv preprint*, abs/1301.3781.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3075–3081. AAAI Press.

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*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

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

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-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv preprint*, abs/1910.10683.

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*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

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*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. 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.

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.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Dongling Xiao, Han Zhang, Yu-Kun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE-GEN: an enhanced multi-flow pre-training and fine-tuning framework for natural language generation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3997–4003. ijcai.org.

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.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2019. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *ArXiv preprint*, abs/1912.08777.

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 (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

A P^{3}LM v.s. XLNet

The idea of permuted decoding is inspired by XLNet. However, P^{3}LM is different from XLNet in multiple aspects as follows. First, P^{3}LM is designed for addressing bi-directional context and long dependency problems for natural language generation (NLG), while XLNet is for natural language understanding (NLU); Second, P^{3}LM is in a transformer encoder-decoder architecture, while XLNet is only a transformer encoder; Third, P^{3}LMis trained on the full permutation of the target sequence to enhance long dependency modeling, while XLNet is trained on partial permutation of the source sequence; Fourth, P^{3}LM is implemented with multi-streams ($\# >= 3$) for predicting multiple future tokens at one time step, while XLNET is implemented with two streams ($\# = 2$) for predicting one token at a step; Fifth, P^{3}LM implements permuted decoding that requiring a shift-right operation while XLNet does not, which is due to transformer's different designs for encoder and decoder.

B Model Parameters on GLGE

Table 6 shows the parameters of our model on GLGE. Parameters are primarily searched from LR$\in${1e-4, 1e-5}, WarmUp $\in$ {0.5k, 1k}, BatchSize $\in$ {128, 256, 512}, BeamSize $\in$ [4, 10], and LenPenalty $\in$ [0.6, 1.5], except WarmUp=10k for Gigaword and LenPenalty=10.0 for PersonaChat.

Parameters	Text Summarization				QG		QA	Dialog
	CD	GG	XS	MN	SQ	MQ	CQ	PC
LR	1e-4	1e-4	1e-4	1e-5	1e-5	1e-5	1e-5	1e-4
WarmUp	1k	10k	0.5k	1k	1k	1k	1k	0.5k
BatchSize	512	128	256	128	128	128	128	128
MaxEpoch	15	6	15	15	10	10	10	15
MaxSrcLen	512	128	512	512	256	256	512	256
MaxTgtLen	128	32	128	64	64	32	32	32
BeamSize	5	5	8	8	6	4	7	10
LenPenalty	1.4	0.9	0.8	0.9	1.0	0.8	0.8	10.0
DecLen	45-110	3-32	10-64	3-64	5-32	3-32	1-32	3-32
BestEpoch	14	6	9	13	7	5	10	13

Table 6: Hyperparameters used in fine-tuning P^{3}LM on GLGE. LR: learning rate. WarmUp: warm up steps. BatchSize: batch size. MaxEpoch: max epochs in fine-tuning. MaxSrcLen: source max length. MaxTgtLen:target max lengt. BeamSize: decoding beam size. LenPenalty: decoding length penalty. DecLen: length range of generated sequence. BestEpoch: best performing epoch. CD: CNN/DM. GG: Gigaword. XS: XSUM. MN: MSNews. SQ: SQuAD-QG. MQ: MSQG. CQ: CoQA. PC: PersonaChat.

C L2R Inference

P^{3}LM decodes a sequence in both L2R and URP order with prophet mechanism during training. Different from training, our model leverages L2R decoding during inference. Nevertheless, P^{3}LM achieves significant improvements across multiple tasks and datasets. We think this benefits from the involving of P^{3}LM decoding which introduces more constraints to help the model to learn bidirectional context and long dependency modeling.

D Effect of Pre-training Iterations

We verify that the performance of a pre-trained model improves with the increasing of training iterations within current maximum iteration number. Figure 4 shows the results of finetuned models on CNN/DM with different pre-trained models. For both the base and large models, rouge scores increase when the models are pre-trained with more iterations.

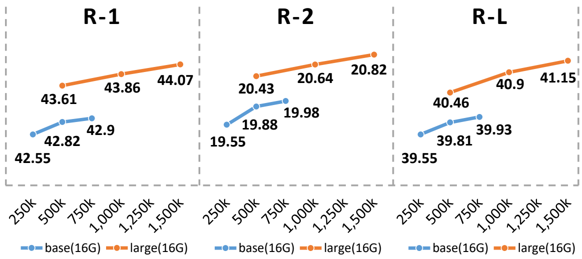

Figure 4: P^{3}LM finetuning results on CNN/DM of different pre-trained models at different iterations.