Rethinking Language Modeling via Path Decomposition and Selection

Anonymous ACL submission

Abstract

Recent generative language models assume a pre-defined monotonic left-to-right sequence decomposition format to learn, which has been proven very effective in current well-known decoder-only autoregressive large language models, but might be inefficient in learning many specific task such as reasoning. In this paper, we explore the potential of other feasible decomposition formats for language models to effectively compensate the autoregressive language modeling paradigm. Specifically, we aim to find the appropriate composition from multiple candidates through introducing effective path selection in both training and decoding. Experiments on total **11** zero-shot reasoning tasks and **2** language generation tasks demonstrate the effectiveness of our methods, indicating that more suitable decomposition formats beyond a left-to-right order do exist, and superior performance can be achieved by simply selecting and optimizing the decoding paths.

1 Introduction

Most of generative language models, from ngram-based models (Bahl et al., 1983) to neural language models (Bengio et al., 2000), including the current well-known decoder-only large language models (Touvron et al., 2023a,b; OpenAI, 2023), rely on a monotonic left-to-right order to decompose the neural language texts to learn their internal dependencies during training and leverage the same determined order in decoding. Although the above monotonic modeling and generation paradigm has always been the mainstream in the NLP community in recent years, we still wonder if there exist fungible or even superior sequence decomposing formats for language models to learn and generate the target sequences, especially after witnessing the success of several non-monotonic model variants (Yang et al., 2019; Welleck et al., 2019; Shih et al., 2022). Furthermore, efficiently selecting the relatively suitable decomposing formats for different training instances is a critical but challenging aspect for the success of language models.

In this paper, we frame the problem of finding the superior decomposing formats of language texts as a decoding path selection process. Specifically, with the decoding path for several different typical language models shown in Figure 1, e.g., autorgressive (Vaswani et al., 2017), non-autoregressive (Gu et al., 2018), and BERT-family (Devlin et al., 2018), the former two types of models have the unique decoding path while BERT-family can allow various paths to generate target sequences. Therefore, to best explore the impacts of different decomposing formats of texts, we pre-train a new BERT-family variant for generation tasks to conduct evaluation experiments. Specifically, we aim to find the appropriate composition formats from multiple candidates during inference via the ***path selection*** method, and then further leverage the outputs achieved from these compositions to optimize the language models to learn the path preference through the ***path selection**** method.

To evaluate our proposed new methods, we conduct detailed experiments on various zero-shot reasoning and language generation tasks, and mainly observe that **(1)** *there do exist superior decoding paths beyond monotonic left-to-right decomposition for language models to achieve better generation outputs;* **(2)** *although BERT-family models are recognized as not proficient in these evaluation tasks, simply selecting and optimizing the decoding path enables them to perform on par with current competitive AR models of comparable capacity (model scale), demonstrating great potential.* Our observations can provide new insights into the generative modeling and inference methods for language models in the future, thus motivating the researchers to seek more effective solutions in learning the dependency of language texts and conducting the generation process with more suitable

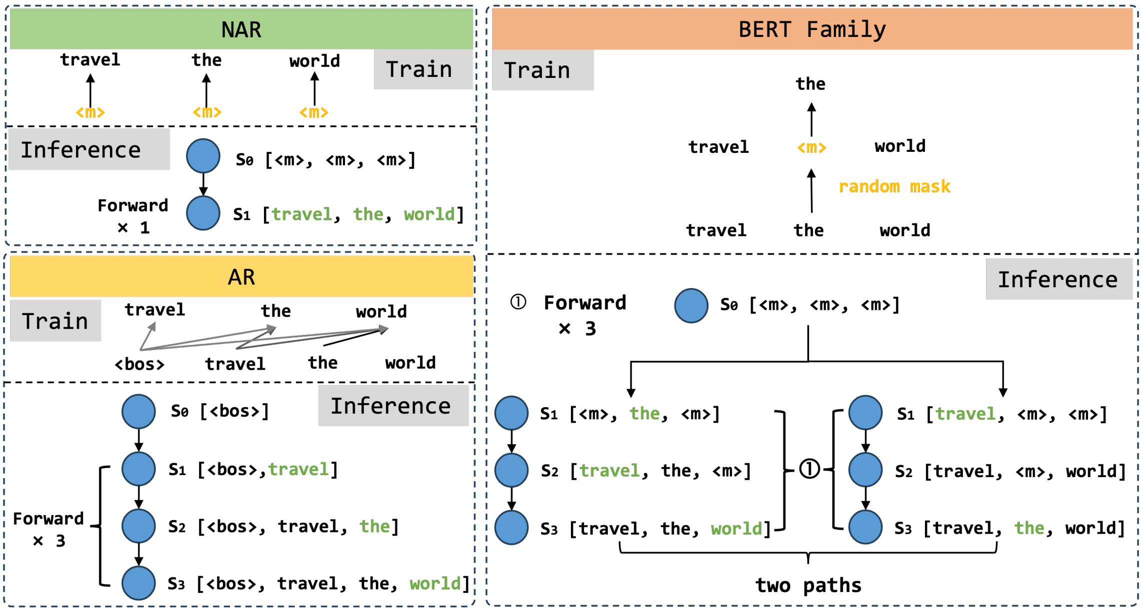

Figure 1: The sequence decomposition for training, and composition methods (i.e., decoding path to achieve the outputs sequence) for different language models.

decomposing formats.

2 Preliminary

2.1 Utilizing BERT-family for Language Generation Tasks

Since the traditional BERT-family are not designed and pre-trained for language generation tasks, several efforts should be made in adapting them to generating language texts. Previous works (Dong et al., 2019; Wang and Cho, 2019) have theoretically indicated that the BERT-family can be utilized for generating texts by predicting the masked positions in the target sequence. Despite early efforts by researchers to leverage BERT-family for language generation tasks (Chan and Fan, 2019; Jiang et al., 2021; Su et al., 2021), these attempts yielded suboptimal results compared to the mainstream generative models. Subsequently, researchers attempt to adapt BERT-family to NAR scenarios (Liang et al., 2023b,a; Xiao et al., 2024) via the the Mask-Predict decoding algorithm (Ghazvininejad et al., 2019), which first predicts the entire masked target sequence in the first decoding step, and then refines the target sequence by replacing the unreliable parts with masked tokens and re-generating them in parallel in the subsequent decoding step as details shown in the Appendix A, and receives relatively positive feedback regarding performance. During training, these models learn to predict the masked parts in the target sequence, whose loss can be computed as $\mathcal{L} = -\sum_{y_i \in Y_{mask}} \log \mathcal{P}(y_i|Y_{obs}, X; \theta)$, where X denotes the source sequence, Y_{mask} and Y_{obs} are the masked and unmasked parts in the target sequence Y, respectively. In this paper, we further delve into the essential technological advancements of BERT-family that leverage the Mask-Predict decoding algorithm to achieve better performance in generation tasks.

2.2 Decoding Paths for BERT-family

Formally, we consider the process of generating a sequence of discrete tokens $Y = (y_1, ..., y_N)$, where $y_i \in V$, a finite vocabulary specific to a language model. This generation process can be interpreted as deterministically sampling a series of successive state spaces S, where each state $s_i \in S$ corresponds to a sequence of tokens sampled from V, and relies on a policy π to transition to the next state. A policy π serves as a determinate mapping from states to actions, outlining how the model processes the current sequence and achieves the subsequent sequence in the next state. We denote this specific process to compose the target sequence as the decoding path P of a given language model, where each node represents the current state s_i in ith decoding step and each edge represents the policy π_i indicating the actions for transitioning from state s_i to s_{i+1}.

As shown in Figure 1, different language models have their specific decoding paths to compose the target sequence. The traditional AR and NAR language models typically have a single decoding path for composing a specific target sequence, while BERT-family can explore multiple optional decoding paths, resulting in varied output sequences of differing generation qualities. Selecting a specific decoding path from the multitude of optional paths available in BERT-family is crucial for achieving high-quality outputs. With approximately 2^{TN} possible paths for a BERT-family model, as detailed in the Appendix B, determining the optimal path is essential for the success of these models. In Ghazvininejad et al. (2019) where the Mask-Predict decoding algorithm was first proposed, the authors heuristically regulate the policy π_t in tth decoding step as predicting the masked parts in current Y and selecting the specific n_t tokens which are with lowest prediction probabilities to be re-masked, where the number of re-masked tokens can be computed as $n_t = (1 - t/T) * N$, N denotes the total number of tokens in Y, t and T denote the current and total decoding step, respectively. While the Mask-Predict algorithm provides a heuristic approach to selecting decoding paths, it may not always yield optimal results. There exist other decoding paths in the candidate space leading to better composition of target sequences (Kreutzer et al., 2020). Hence, we aim to identify an optimal decoding path from such multitudinous candidates by introducing ***path selection*** method. Moreover, we further propose ***path selection**** which empowers the model to learn the preference between different decoding paths. Our methods seek to enhance the BERT-family's ability to navigate through the complex decoding spaces and generate higher-quality output.

3 Methods

3.1 Path Selection

We first sample several optional decoding paths from the candidate spaces and select the best one with the highest total prediction probability. Specifically, we follow most of the practice in the Mask-Predict algorithm, except for the selection of the re-masked tokens in each decoding step. As shown in the right of Figure 2, rather than just selecting a specific number of tokens with the lowest prediction probabilities to transform to the unique next state (i.e., the first beam), we allow total k candidate selections for re-masked tokens with the lowest-k total prediction probabilities for each decoding path, where k is the position beam number set in advance, and total prediction probabilities are the sum of each token's probability in the sequence. Notice we always keep the number of candidate states in each decoding step as k, which is similar to the beam search algorithm for AR models (Meister et al., 2020). However, the search times to select the lowest-k candidates is quite large especially when N is large, i.e., given the total decoding step T, generated target tokens N, and the position beam number k, the total search times is $k * \sum_{t \in \{1,2,...,T\}} C_{n_t}^N$, where its detailed proof is in Appendix C. Therefore, to reduce the search overhead, we further introduce a simplified version that transforms the search times in tth decoding step from $C_{n_t}^N$ to k in which only one position in masked parts can be replaced by the one in unmasked parts to obtain the candidate decoding states, thus the upper bound of search times can be reduced to $T * k^2$. For example, as shown in Figure 2, after obtaining the first beam sequence by Mask-Predict algorithm, we can choose one token in its masked parts with the largest prediction probability (i.e., *go*) to replace the one in its unmasked parts with the least prediction probability (i.e., *often*) to obtain the second beam sequence in each decoding step.

3.2 Path Selection*

Motivated by the recent direct preference optimization (DPO) algorithm (Rafailov et al., 2024) which adopts positive-negative pair samples to train human preferences for language models, we aim to teach BERT-family the decoding path preference by training with positive-negative pair samples achieved from composition methods. Specifically, as shown in the right of Figure 2, given a specific instance in which several tokens in the target sequence are replaced with masked tokens, denoted as Y_{mask}, we randomly[1] sample two different decoding paths to generate these masked tokens in multiple steps, then achieve two different output sequences, and the specific output tokens of Y_{mask} are denoted as Y_{out}^1 and Y_{out}^2, the details of the sampling methods are presented in Appendix D. Subsequently, we use a score function $\texttt{Score}(\cdot)$, which can be the exact match accuracy with ground truth tokens or the BLEU score (Papineni et al., 2002), to identify the specific positive and negative ones. Once $\texttt{Score}(Y_{out}^1) > \texttt{Score}(Y_{out}^2)$, we adopt Y_{out}^1 as the positive output Y_w and Y_{out}^2 as the negative

[1]We randomly sample the number and specific positions of re-masked tokens to transform to the next state in each decoding path rather than that according the rule in the Mask-predict algorithm mentioned above.

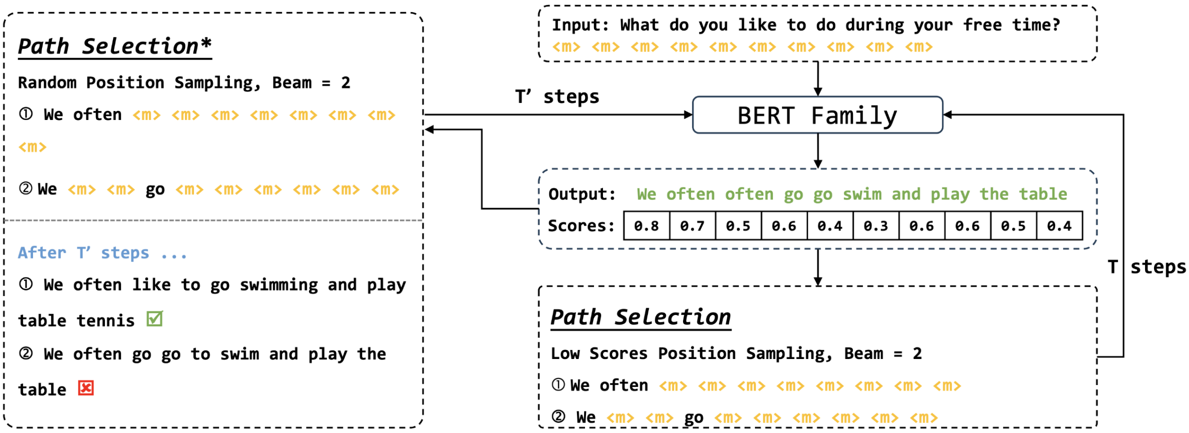

Figure 2: Overview of the path selection and path selection* methods. As for the path selection method during inference, we select the positions for masked tokens with the lowest-k prediction probabilities, while the path selection* randomly samples the positions for masked tokens.

output Y_l, and vice versa. Finally, following the common practice in online DPO algorithm, given the reference model π_{ref} and the policy model π_θ, we first use π_{ref} to sample the positive-negative pair samples, then update π_θ with the DPO loss:

$$\begin{aligned}\mathcal{L}_{\text{DPO}}(\pi_\theta;\pi_{\text{ref}}) = & -\log\sigma[\beta(\frac{\pi_\theta(Y_w|Y_{obs},X)}{\pi_{\text{ref}}(Y_w|Y_{obs},X)} \\ & - \frac{\pi_\theta(Y_l|Y_{obs},X)}{\pi_{\text{ref}}(Y_l|Y_{obs},X)})],\end{aligned} \tag{1}$$

where X denotes the source sequence, Y_{obs} denotes the unmasked parts in Y, σ denotes the sigmoid function, β is the hyperparameter controlling the DPO loss, $\pi_\theta(Y_w|Y_{obs},X) = \sum_{y_i \in Y_w} \mathcal{P}(y_i|Y_{obs},X;\theta)$, etc. Besides, we add two penalty terms to reduce the failure cases of DPO as mentioned in (Pal et al., 2024), i.e., the model reduces the probabilities of positive outputs and meanwhile more significantly reduces the probabilities of negative outputs, then the probability gap between two outputs will be larger, and the DPO loss will be smaller. However, reducing the probabilities of positive outputs is contrary to our expectations. The penalty terms can be computed:

$$\begin{aligned}\mathcal{L}_{\text{PEN}}(\pi_\theta;\pi_{\text{ref}}) = & \max\left(0, \log\frac{\pi_{\text{ref}}(Y_w|Y_{obs},X)}{\pi_\theta(Y_w|Y_{obs},X)}\right) \\ & + \max\left(0, \log\frac{\pi_{\text{ref}}(Y_l|Y_{obs},X)}{\pi_\theta(Y_l|Y_{obs},X)}\right).\end{aligned} \tag{2}$$

Then, combining the above DPO loss and the penalty terms with the traditional masked language modeling loss in BERT-family as mentioned in Section 2.1, which aims to predict the masked tokens:

$$\mathcal{L}_{\text{MLM}}(\pi_\theta) = -\sum_{y_i \in Y_{mask}} \log\mathcal{P}(y_i|Y_{obs},X;\theta). \tag{3}$$

Our final training loss is computed as $\mathcal{L} = \mathcal{L}_{\text{MLM}} + \lambda_1\mathcal{L}_{\text{DPO}} + \lambda_2\mathcal{L}_{\text{PEN}}$, where λ_1 and λ_2 are the hyper-parameters to balance the different loss items.

4 Experiments

4.1 Implementation Details

Backbone Models For better evaluation of various generation tasks, we pre-train new variants of BERT-family with a modified masking mechanism during training, which aims to better equip these masked language models for tasks involving generation (Liang et al., 2023b; Xiao et al., 2024), thus we name our model as **Ge**nerative **BERT** **(GeBERT)**. Details of our pre-training task are presented in the Appendix E. During training, we adopt the Pile (Gao et al., 2020; Biderman et al., 2022) dataset to pre-train our models based on an encoder-only language model with a bi-directional attention mechanism following the follow the most practice in previous BERT-like models, and further incorporate several effective techniques such as Rotary Positional Embedding (RoPE) (Su et al., 2024) and swiglu (Shazeer, 2020) activation function. Details of training settings are presented in the Appendix F Based on our modified pre-training task, we pre-train two versions of GeBERT containing 124M and 352M parameters which are similar to the base and large versions of other previous pre-trained language models (Devlin et al., 2018; Lewis et al., 2019; Raffel et al., 2020), denoted as

Models	LogiQA	Sciq	ARC-E	ARC-C	Wino.	BoolQ	PIQA	SIQA	Race	Hella.	Truth.	AVG.
≈ 150M parameters												
OPT-125M	27.93	75.2	43.52	22.78	50.28	61.07	62.02	37.21	30.05	31.25	23.99	42.31
GPT-neo-125M	28.88	76.5	43.73	23.12	50.43	62.02	62.46	37.21	27.56	30.40	25.83	42.56
Pythia-160M	24.27	75.4	43.64	23.63	51.30	62.14	61.97	36.90	28.71	30.30	24.97	42.11
RWKV-169M	24.73	75.2	47.52	23.46	50.67	62.17	64.04	37.00	26.89	32.25	22.25	42.41
GeBERT-124M	27.65	80.3	42.13	22.10	50.75	62.17	60.66	36.49	28.90	29.76	24.60	42.27
+ Path Selection	27.65	81.8	42.09	22.36	51.87	62.17	59.69	36.80	29.28	31.70	25.70	42.89
+ Path Selection*	28.88	80.5	42.47	22.18	52.72	62.17	60.88	36.94	29.76	32.25	25.74	**43.14**
≈ 350M parameters												
OPT-350M	28.57	74.90	44.19	23.98	52.49	61.87	64.74	39.30	29.76	32.66	23.50	43.27
Pythia-410M	29.34	81.30	52.10	24.32	53.20	61.68	67.08	38.95	30.91	40.52	23.50	45.72
RWKV-430M	24.42	79.00	52.23	25.17	52.80	62.05	68.44	38.84	28.71	40.78	22.28	44.98
GeBERT-352M	28.88	83.10	51.43	23.86	52.93	62.17	65.21	39.02	30.68	40.12	24.35	45.01
+ Path Selection	29.87	83.60	51.65	24.24	52.87	62.17	65.03	39.26	30.83	41.03	25.58	46.03
+ Path Selection*	30.33	83.30	51.97	24.18	53.19	62.17	65.78	39.51	31.00	41.30	25.80	**46.21**

Table 1: Results on zero-shot common sense reasoning and reading comprehension tasks. The first line of GeBERT denotes the baseline which adopts the left-to-right composition. **Bold** values denote the best average result (**AVG.**) through all models. underlined values denote the result of our methods outperforming the baseline GeBERT. The abbreviations **Wino.**, **Hella.**, and **Truth.** denote the WinoGrande, Hellaswag, and Truthfulqa datasets, respectively.

GeBERT-124M and **GeBERT-352M**. We utilize the Megatron-Deepspeed[2] library to train GeBERT on 8 NVIDIA A100-PCIE-80GB GPU cards.

Fine-tuning Settings We follow the training procedure in previous works (Liang et al., 2023b; Xiao et al., 2024) to fine-tune GeBERT on downstream datasets for non-autoreressive sequence generation tasks. For the fine-tuning settings, we tune the learning rate from {1e-5, 2e-5, 5e-5, 1e-4} for different downstream tasks. We train for a total of 50 epochs and validate the model after each epoch, then obtain the final model with the best validation performance. During the training of the path selection* method, we initialize the policy and reference model with that after fine-tuning for downstream sequence generation tasks. Then, we freeze the parameters of the reference model and only update the parameters of the policy model with the same dataset adopted in fine-tuning. We set the learning rate as 2e-5 and other training hyperparameters the same in the fine-tuning stage. Then, we train the model with 5 epochs. As for the DPO training of the vanilla GeBERT, we initialize the policy and reference model with the final saved checkpoint during pre-training. We sampled a small subset from the pile to conduct DPO training and avoid introducing extra training data.

Datasets and Metrics We evaluate our proposed methods on common downstream task-specific generation tasks, which have been widely used in previous pre-trained AR and NAR works, and various zero-shot common sense reasoning and reading comprehension tasks, which are popular to evaluate the vanilla version of current large language models without fine-tuning (Zeng et al., 2022; Touvron et al., 2023a,b). To the best of our knowledge, we are the first to evaluate the pre-trained NAR models for these zero-shot tasks. Specifically, For downstream task-specific generation tasks, we adopt XSUM (Narayan et al., 2018) for the summarization task and MSQG (MicroSoft Question Generation) dataset for the question generation task from the GLGE benchmark (Liu et al., 2021). For the evaluation metrics, we adopt ROUGE F1 (ROUGE-1/2/L) (Lin and Hovy, 2002) for XSUM, and BLEU (BLEU-4) (Papineni et al., 2002), Rouge-L and METEOR (Lavie and Agarwal, 2007) for MSQG. For zero-shot common sense reasoning and reading comprehension tasks, we adopt ARC-easy, ARC-challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), WinoGrande (Sakaguchi et al., 2021), Race (Lai et al., 2017), Sciq (Johannes Welbl, 2017), LogiQA (Liu et al., 2020), Hellaswag (Zellers et al., 2019), and Truthfulqa (Lin et al., 2021), which are all widely used for evaluating recent language models. We adopt Language Model Evaluation (Gao et al., 2021) framework to evaluate these datasets under a zero-shot setting (Biderman et al., 2023). We adopt

[2]https://github.com/microsoft/Megatron-DeepSpeed

Model	XSUM			MSQG			Speedup
	Rouge-1	Rouge-2	Rouge-L	Rouge-L	BLEU-4	METEOR	
Transformer	30.66	10.80	24.24	29.43	4.61	9.86	-
Base Version (≈ 150M parameters)							
BANG	32.59	8.98	27.41	-	-	-	-
ELMER	37.30	13.17	29.92	-	-	-	-
PreDAT	39.79	17.38	32.71	-	-	-	-
MIST	34.63	11.29	28.70	-	-	-	-
DEER	39.10	16.80	32.40	38.70	9.70	23.30	-
MASS-base	39.70	17.24	31.91	38.90	**10.20**	23.30	-
BART-base	38.79	16.16	30.61	38.20	**10.20**	22.90	1.0x
ProphetNet-base	39.89	17.12	32.07	37.10	9.10	22.30	-
GeBERT-124M	40.32	16.90	32.54	39.13	9.66	23.50	3.1x
+ Path Selection	40.52	17.11	32.71	39.06	9.52	23.51	1.2x
+ Path Optimization	**40.92**	**17.39**	**33.08**	**39.46**	9.72	**23.68**	1.2x
Large Version (≈ 350M parameters)							
MASS-middle	39.10	16.50	31.40	38.90	9.50	23.50	-
BART-large	**45.10**	**22.20**	**37.20**	38.80	9.20	24.30	-
ProphetNet-large	44.40	21.30	36.40	38.30	9.60	23.30	-
GeBERT-352M	44.12	21.03	36.27	39.32	10.23	23.87	-
+ Path Selection	44.33	21.23	36.40	39.38	10.21	23.90	-
+ Path Optimization	44.84	21.89	36.89	**39.78**	**10.29**	**24.32**	-

Table 2: Results on task-specific generation tasks. **Bold** denotes the best result. underlined values denote the result of our methods outperforming the baseline GeBERT.

normalized accuracy for PIQA, ARC-challenge, LogiQA, Hellaswag, and accuracy for other tasks following previous works (Biderman et al., 2023).

Baseline Models For the downstream task-specific generation tasks, we adopt the vanilla Transformer baseline (Vaswani et al., 2017) and previous pre-trained AR models including MASS (Song et al., 2019), BART (Lewis et al., 2019), and ProphetNet (Qi et al., 2020) which are included in the official GLGE evaluation leaderboard as autoregressive baselines. For NAR baselines, we adopt the previous pre-trained NAR models including BANG (Bang et al., 2023), ELMER (Li et al., 2022) and PreDAT (Huang et al., 2023). Besides, we also include MIST (Jiang et al., 2021) and DEER (Liang et al., 2023a) which also fine-tune the traditional BERT-family to complete the generation tasks. For common sense reasoning and reading comprehension tasks, which are only widely used after the popularity of large language models and never been included in the evaluation of previous NAR models, we adopt the recent large language models which are also trained on the Pile for around 300B tokens and contains the comparable model parameters with GeBERT, including OPT-125M/350M (Zhang et al., 2022), GPT-neo-125M (Black et al., 2022), Pythia-160M/410M (Biderman et al., 2023), and RWKV-169M/430M (Peng et al., 2023). We re-run all the baseline models under the same Language Model Evaluation framework (Gao et al., 2024) using their open-source Hugging Face models to ensure consistent evaluation procedures.

4.2 Main Results

Zero-shot common sense reasoning and reading comprehension We present the results on various zero-shot common sense reasoning and reading comprehension tasks in Table 1. Compared with GeBERT and the previous AR models, we can find that: (1) GeBERT can also complete these zero-shot tasks and achieve comparable performance while adopting the same decoding path during inference[3]. (2) Our final models (i.e., GeBERT with path selection and path selection*) achieve the best performance through all the previous AR models on average, outperforming

[3]For GeBERT, we append a masked token after the current sequence and enable the model to predict it, thus realizing the same decoding path as AR models that adopt the policy as predicting the next token.

Hyber.	**LogiQA**	**Sciq**	**ARC-E**	**ARC-C**	**Wino.**	**BoolQ**	**PIQA**	**SIQA**	**Race**	**Hella.**	**Truth.**	**AVG.**
left-to-right	27.65	80.3	42.13	22.10	50.75	62.17	60.66	36.49	28.90	29.26	24.60	42.26
right-to-left	28.57	72.2	30.73	22.35	49.17	58.87	55.22	33.78	25.55	30.12	25.70	39.30
random	25.81	79.6	41.71	21.67	50.20	62.17	56.09	33.93	28.13	29.50	24.48	41.20
easy-to-hard	29.19	80.4	41.96	22.44	52.57	62.17	59.79	36.80	29.79	31.73	25.58	42.90
hard-to-easy	26.88	80.4	41.50	21.16	50.04	62.17	58.92	36.34	26.22	31.03	24.97	41.78
$beam = 2$	28.11	81.4	42.59	22.01	52.01	62.17	59.69	36.54	30.05	31.98	25.45	42.94
$beam = 3$	27.65	81.8	42.09	22.36	51.86	62.17	60.01	36.80	29.67	32.17	25.58	42.90
$beam = 4$	28.26	81.8	42.51	22.61	51.07	62.17	60.28	36.28	29.28	32.09	25.70	42.91

Table 3: Results of different methods to select the decoding paths for zero-shot common sense reasoning and reading comprehension tasks. **Hyper.** denotes the corresponding hyperparameter.

the previous best models (i.e., GPT-neo-125M and Pythia-410M) by around 0.8 and 0.5 score. (4) GeBERT is better at reading comprehension tasks which enable the model to answer questions given supports or evidences such as Sciq and LogiQA, we attribute this to the bi-directional attention mechanism of GeBERT. Besides, Compared with baseline GeBERT which adopts the same decoding path as AR models and those with our path selection and path selection* methods, we can find that: (1) With the path selection method, GeBERT outperforms the baseline GeBERT in most of the evaluation tasks, leading to 0.6/1.0 performance improvements on GeBERT-124M/352M. (2) Further, with the path selection* method, GeBERT can outperform the baseline GeBERT in 10 of 11 evaluation tasks and be on par in BoolQ, leading to around 1.0 performance improvements on average. (3) By comparing GeBERT only with the path selection method and with both proposed methods, the former can achieve performance improvements on most tasks, indicating the effectiveness of the path selection* method. However, the path selection* may also result in performance declines in several tasks, such as Sciq for GeBERT-124M and ARC-C for GeBERT-352M.

Task-specific generation Table 2 presents the results on task-specific generation task. We can find that: (1) For the summarization task, though GeBERT-352M underperforms BART-large GeBERT-124M, it outperforms all the other baseline models in all evaluation metrics, indicating that GeBERT can generate more informative and reasonable summaries. (2) For the question generation task, GeBERT-124M outperforms all the baseline models on Rouge-L and METEOR and only presents performance gaps compared with the best baseline models on BLEU-4. GeBERT-352M achieves the best performance across various models on all evaluation metrics. (3) Compared to the GeBERT baseline, which adopts the original vanilla Mask-Predict algorithm to generate the output sequence, the path selection and path selection* methods can bring performance improvements on the XSUM dataset for both GeBERT-124M/352M, indicating that these two methods can enable the model to achieve better performance in generating relatively long targets. However, the path selection method does not lead to consistent performance improvements on the MSQG dataset, which contains relatively short targets. We attribute this to that short sequences will lead to relatively small candidate space and redundant outputs for different decoding paths, thus we can not achieve better outputs from multiple candidates. (4) We also compare the decoding efficiency of GeBERT-124M and BART-base, which contains around 140M parameters, and the results demonstrate that GeBERT can achieve 3.1x speedup with the vanilla Mask-predict algorithm due to the NAR attribute. Further, although path selection and path selection* will bring the extra search overhead for various decoding paths, GeBERT still achieves a faster generation process, leading to a 1.2x speedup compared to BART.

5 Analysis

5.1 Discussion of Different Methods to Determine the Composition formats

The results in Table 1 have demonstrated that our methods outperform the traditional left-to-right composition format, here we further compare with several other optional formats, e.g., (1) right-to-left order; (2) random order; (3) Instead of selecting the token with the highest prediction probability, which is denoted as an easy-to-hard order (Kasai et al., 2020), we include a hard-to-easy order which

!

Method	Rouge-1	Rouge-2	Rouge-L
GeMLM	40.32	16.90	32.54
w/ Token Beam	40.17	16.88	32.50
w/ Position Beam	40.52	17.11	32.71
w/ Path Selection*	40.78	17.30	33.01
w/ Token Beam	40.58	17.19	32.90
w/ Position Beam	40.92	17.39	33.08

Table 4: Results of different beam search algorithms.

generates the token with the lowest prediction probability first; (4) path selection with different beam number as 2/3/4. We present the corresponding results in Table 3, we can find that: (1) Based on different orders, the easy-to-hard order that we adopt in the path selection method performs best (i.e., $beam = 1$), while several other orders will lead to significant performance declines such as right-to-left order. (2) Adopting different beams in our path selection method performs differently for various tasks but achieves a comparable score on average, and all outperforms the left-to-right baseline, indicating the effectiveness of the path selection method. Besides, the comparisons of more composition formats are presented in the Appendix G.

5.2 Comparison with Tokens-aware Beams

The path selection method sampling several position beams to achieve multiple candidate outputs is similar to the token-aware beam search algorithm, which has been widely used in AR models (Meister et al., 2020). The token-aware beam search algorithm selects more candidate tokens during inference rather than always the one with the highest prediction probability, which can significantly improve the performance. We also extend this into BERT-family to permit more optional tokens in each decoding step. Specifically, we randomly select one token in the unmasked parts in target sequence and replace it with one whose prediction probability is below the first one in the overall probability distribution. Compared with our proposed path selection method, the position beams select candidates with different positions based on specific tokens while the token-aware beam search algorithm selects candidates with different prediction tokens based on specific positions. We adopt GeBERT-124M to conduct analytic experiments on XSUM, with the results are presented in Table 4. We find that the path selection method can achieve consistent performance improvements, but the token-aware beam search algorithm does not work in this scenario. We attribute the failure of token-aware beam search to the different modeling paradigm of BERT-family compared to AR models.

Hyperparameter	Rouge-1	Rouge-2	Rouge-L
$\lambda_1 = 0.0, \lambda_2 = 0$	40.32	16.90	32.54
$\lambda_1 = 0.5, \lambda_2 = 0$	39.85	16.88	32.52
$\lambda_1 = 0.5, \lambda_2 = 1$	40.76	17.25	32.96
$\lambda_1 = 0.5, \lambda_2 = 5$	40.78	17.30	33.01
$\lambda_1 = 0.5, \lambda_2 = 10$	40.70	17.24	32.97
$\lambda_1 = 0.0, \lambda_2 = 5$	40.22	16.90	32.52
$\lambda_1 = 0.1, \lambda_2 = 5$	40.74	17.20	32.92
$\lambda_1 = 0.5, \lambda_2 = 5$	40.78	17.30	33.01
$\lambda_1 = 1.0, \lambda_2 = 5$	40.78	17.28	33.05

Table 5: Result of different λ_1 and λ_2.

5.3 Ablation Study of Path Selection*

In this section, we conduct an ablation study to explore the effects on different λ_1 and λ_2 in our final training loss as mentioned in Section 3.2. We report the performance of λ_1 in $\{0.0, 0.1, 0.5, 1\}$, λ_2 in $\{0, 1, 5, 10\}$ without adopting position beams. Compared with the baseline model (i.e., $\lambda_1 = 0.0$, $\lambda_2 = 0$), we can find that (1) $\mathcal{L}_{\text{DPO}}$ and $\mathcal{L}_{\text{PEN}}$ are both necessary for performance improvements. With $\lambda_1 = 0.5$ and $\lambda_2 = 0$, the performance even declines, indicating the failure cases as mentioned in Section 3.2. (2) In other cases, the performances are close to each other with only around 0.1 gaps on all metrics, indicating that we need not spend lots of effort to tune the λ_1 and λ_2. Our DPO training objective is easy to achieve the corresponding performance improvements.

6 Conclusion

In this paper, we explore the potential of other better decomposition formats for language models to learn internal dependency of texts and generate the target sequences. To find better decomposition formats, we propose path selection to enable models to choose the best one from multiple candidates and path selection* to instruct the model on learning preference of different decoding paths. Results on various evaluation datasets demonstrate the effectiveness of our methods, with the performance of BERT-family reaching the level even outperforming the traditional autoregressive models with a monotonic left-to-right decomposition format.

Limitations

Our work demonstrates that BERT-family can perform better than AR language models by adopt our proposed path selection and path selection* methods. However, these models still require multi-step reasoning during zero-shot tasks to bridge the gap between inference and pre-training. This reasoning paradigm may affect the inference efficiency, making BERT-family models less effective than AR models in some contexts. Besides, the backbone models are relatively small (i.e., less than 1B parameters), since the large language models have demonstrated tremendous success in various language generation tasks, we should further evaluate our methods on these large language models.

References

Titu Andreescu, Zuming Feng, Titu Andreescu, and Zuming Feng. 2004. Inclusion-exclusion principle. *A Path to Combinatorics for Undergraduates: Counting Strategies*, pages 117–141.

Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, (2):179–190.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems*, 13.

Stella Biderman, Kieran Bicheno, and Leo Gao. 2022. Datasheet for the pile. *arXiv preprint arXiv:2201.07311*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Ying-Hong Chan and Yao-Chung Fan. 2019. A recurrent bert-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 154–162.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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. *Advances in Neural Information Processing Systems*, 32.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. 2021. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, page 8.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121.

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

Junliang Guo, Linli Xu, and Enhong Chen. 2020. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 376–385.

Fei Huang, Pei Ke, and Minlie Huang. 2023. [tacl] directed acyclic transformer pre-training for high-quality non-autoregressive text generation. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

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. *arXiv preprint arXiv:2110.11115*.

Matt Gardner Johannes Welbl, Nelson F. Liu. 2017. Crowdsourcing multiple choice science questions.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Parallel machine translation with disentangled context transformer. *arXiv preprint arXiv:2001.05136*.

Julia Kreutzer, George Foster, and Colin Cherry. 2020. Inference strategies for machine translation with conditional masking. *arXiv preprint arXiv:2010.02352*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: an automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, page 228–231, USA. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2022. Elmer: A non-autoregressive pre-trained language model for efficient and effective text generation. *arXiv:2210.13304*.

Xiaobo Liang, Juntao Li, Lijun Wu, Ziqiang Cao, and Min Zhang. 2023a. Dynamic and efficient inference for text generation via bert family. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2883–2897.

Xiaobo Liang, Zecheng Tang, Juntao Li, and Min Zhang. 2023b. Open-ended long text generation via masked language modeling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 223–241.

Chin-Yew Lin and Eduard Hovy. 2002. Manual and automatic evaluation of summaries. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*, pages 45–51.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Dayiheng Liu, Yu Yan, Yeyun Gong, Weizhen Qi, Hang Zhang, Jian Jiao, Weizhu Chen, Jie Fu, Linjun Shou, Ming Gong, et al. 2021. Glge: A new general language generation evaluation benchmark. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 408–420.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

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 arXiv:1907.11692*.

Clara Meister, Tim Vieira, and Ryan Cotterell. 2020. Best-first beam search. *Transactions of the Association for Computational Linguistics*, 8:795–809.

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 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1797–1807.

OpenAI. 2023. Gpt-4 technical report.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*.

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.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.

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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

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. *Journal of Machine Learning Research*, 21(140):1–67.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. 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 4463–4473.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Andy Shih, Dorsa Sadigh, and Stefano Ermon. 2022. Training and inference on any-order autoregressive models the right way. *Advances in Neural Information Processing Systems*, 35:2762–2775.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936. PMLR.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

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*, pages 234–243.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang and Kyunghyun Cho. 2019. Bert has a mouth, and it must speak: Bert as a markov random field language model. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 30–36.

Sean Welleck, Kianté Brantley, Hal Daumé Iii, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. In *International Conference on Machine Learning*, pages 6716–6726. PMLR.

Yisheng Xiao, Juntao Li, Zechen Sun, Zechang Li, Qingrong Xia, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2024. Are bert family good instruction followers? a study on their potential and limitations. In *The Twelfth International Conference on Learning Representations*.

Yisheng Xiao, Ruiyang Xu, Lijun Wu, Juntao Li, Tao Qin, Tie-Yan Liu, and Min Zhang. 2023. Amom: adaptive masking over masking for conditional masked language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13789–13797.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Details of the Mask-predict algorithm

We present an example adopting the Masked-Predict algorithm to generate the output sequence

Prompt: What do you like to do during your free time ?

Target: We often go swimming and play table tennis together .

Input 1: [m] [m] [m] [m] [m] [m] [m] [m] [m] [m]

Output 1: We often often go go swim and play the table

Input 2: We often [m] [m] [m] [m] [m] [m] [m] [m]

Output 2: We often swim swim and play the tennis tennis .

Input 3: We often [m] [m] and play [m] [m] [m] .

Output 3: We often go swim and play table tennis . .

Input 4: We often go [m] and play table tennis [m] .

Output 4: We often go swimming and play table tennis together .

Figure 3: Presentation of the Masked-Predict algorithm.

in Figure 3. Specifically, given the prompt, we first initial the input as fully masked tokens (i.e., Input 1) and send it into the model. After the model predict the outputs (i.e., Output 1), we will select specific unreliable tokens with relatively lower prediction probabilities to mask again (i.e., the yellow parts in outputs). In the subsequent decoding step, the model will predict these masked tokens and select several unreliable tokens again. We obtain the final target sequence until reaching the total number of decoding steps set advance. This decoding algorithm assume that the target sequence will be refined better through multiple decoding steps.

B Details of the decoding paths for BERT-family

Lemma 1 *Given the total length N of target sequence Y, the total decoding step T, the total number of optional decoding paths is around 2^{TN}, exactly $\sum_{m=0}^{N}(-1)^m C_m^N 2^{(N-m)T}$ with the constraint that all tokens in Y should be predicted.*

Proof 1 *During each decoding step, we can select any subset of V_Y, i.e., the model can generate 1 to N different tokens at different position candidates. There exist $C_0^N + C_1^N + C_2^N + ... + C_N^N = 2^N$ candidate position sets in each decoding step, then the overall number of the decoding paths existing in the total T decoding steps is $(2^N)^T = 2^{TN}$. With the constraint that all tokens in Y should be predicted, we should omit the condition that there exist several tokens that are not be predicted during the whole decoding process from the total condition is 2^{TN}. For the specific conditions that there are a number of m tokens that are not be predicted, we should select the candidate tokens in the next $N-m$ tokens, then the number of this condition is $2^{T(N-m)}$, and we have C_m^N to select these specific m un-predicted tokens. We should consider the condition for each $m \in \{1, 2, ..., N\}$, and different $L_{\hat{Y}_i}$ have the repeat decoding paths. Actually, we can solve this problem with the Inclusion-Exclusion Principle (Andreescu et al., 2004). Thus, the total number of decoding paths is:*

$$2^{TN} - C_1^N 2^{(N-1)T} + C_2^N 2^{(N-2)T} - C_3^N 2^{(N-3)T} + ... = \sum_{m=0}^{N} (-1)^m C_m^N 2^{(N-m)T}. \tag{4}$$

C Details for the search times of vanilla path selection method.

Lemma 2 *Given the predicted length N of target sequence Y, the total decoding step T, the position beam number k, and the number of re-masked tokens in tth decoding step n_t, the total times for vanilla path selection method are $k * \sum_{t \in \{1,2,...,T\}} C_{n_t}^N$, and the search times for the simplified version are $T * k^2$.*

Proof 2 *In tth decoding step, for each beam candidate, we select n_t tokens from total N tokens to be re-masked, thus the number of total candidates for single beam is $C_{n_t}^N$, and $k * C_{n_t}^N$ for total k beams. Then, we should compute the total prediction probability for all $k * C_{n_t}^N$ candidates and select the highest k ones for next decoding step. Thus the total search times for T decoding steps are $k * \sum_{t \in \{1,2,...,T\}} C_{n_t}^N$. In the simplified version, we do not need to compute the total prediction probability for all $k * C_{n_t}^N$ candidates, we just replace one token to achieve the k candidates for each single beam, and total k^2 for k beams. Then we only need to compare the total prediction probability for these k^2 candidates and keep the highest k ones, the search times are k^2, and $T * k^2$ for T decoding steps.*

D Details of Generating the DPO Pairs

We present the details to generate the DPO pairs as mentioned in section 3.2 here. Given a specific training instance (X, Y), where Y is further decomposed into the mask parts Y_{mask} and unmasked parts Y_{obs}, the reference model π_{ref}, we achieve the training pairs as follows:

(1) We enable π_{ref} to sample the outputs of Y_{mask}, denoted as O_{mask}, where $O_{mask} = \pi_{\text{ref}}(Y_{mask}|Y_{obs})$, $\pi_{\text{ref}}(Y_{mask}|Y_{obs})$ denotes sampling the tokens in Y_{mask} based on Y_{obs}, and the

sampling method is to adopt the greedy output based on the prediction probability of π_{ref}.

(2) We randomly sample a subset of O_{mask}, denoted as Y'_{mask}, and and replace the tokens in Y'_{mask} with the masked token, where the unmasked parts of O_{mask} is denoted as Y'_{obs}.

(3) We sample the output of Y'_{mask}, denoted as O'_{mask}, where $O'_{mask} = \pi_{\text{ref}}(Y'_{mask}|Y'_{obs} \cup Y_{obs})$.

(4) We achieve one sampled output of Y_{mask} as $Y'_{obs} \cup O'_{mask}$, denoted as Y^1_{out}.

(5) We repeat the above operation to achieve the other sampled output Y^2_{out}.

After obtaining the pair samples Y^1_{out} and Y^2_{out}, we use a score function $\texttt{Score}(\cdot)$ to identify the positive and negative ones. Notice that we select the tokens with the highest prediction probabilities as the output when generating O_{mask} and O'_{mask}, which is consistent with the Mask-Predict algorithm. Besides, we only sample the decoding path with two decoding steps to reduce the overhead during training, the different ratio to sample Y'_{mask} from O_{mask} has adapted the model to various masking conditions in different decoding steps during inference. In practice, we keep the ratio to sample the Y'_{mask} the same during two sampling processes and determine it from a uniform distribution $\text{U}(0.2, 0.8)$. This is because once the ratio is large (e.g., 1.0), all tokens will be re-sampled again, and there is no difference between two sampling outputs, leading to meaningless pairs. Meanwhile, once the ratio is small (e.g., 0.01), only few tokens will be re-sampled again, there are many overlaps between two Y'_{obs}, leading the sampling outputs O'_{mask} lacking of diversity, which is not suitable for the DPO training.

E Details for Pre-training Task

We denote the pre-trained task of GeBERT as generative masked language modeling, which specially designed to fit the BERT-family to various generation tasks. This task is modified from the traditional masked language modeling (MLM) training objective, which makes the model learn to predict the specific masked tokens and has been widely used in traditional BERT-family models (Devlin et al., 2018; Liu et al., 2019). GeMLM aims to build a universal pre-trained BERT-family, which simultaneously possesses the ability of language understanding and generation. Motivated by the previous explorations in the NAR translation task (Ghazvininejad et al., 2019; Guo et al., 2020; Xiao et al., 2023) which extend the traditional MLM into the conditional generation scenery with the encoder-decoder model structure, and those that explore the potential in encoder-only models for language generation tasks (Wang and Cho, 2019; Liang et al., 2023b; Xiao et al., 2024), GeMLM first decomposes each training instance into two parts and assigns different masking strategies to help the model learn different capabilities. Besides, GeMLM further adopts the specific attention masking mechanism to enhance the consistency between the training and inference process.

Specifically, as shown in figure 4, given a specific training instance with the max context length L: $C = \{c_1, c_2, ..., c_{L-1}, c_L\}$, GeMLM decomposes C into a tuple (X, Y), where $X = \{c_1, c_2, ...c_{i-1}, c_i\}$ denotes the prefix tokens, and $Y = \{c_{i+1}, c_{i+2}, ...c_{L-1}, c_L\}$ denotes the suffix tokens. The prefix tokens are used to provide context information and help the model understand the whole sentence, we randomly sample a small ratio of mask tokens, which is similar to the traditional MLM in BERT, denoted as $(X_{mask}, X_{obs}) = \texttt{RANDOM\_MASK}(X, \beta_X)$, where X_{mask} and X_{obs} denote the masked and unmasked parts in X, β_X denotes the masking ratio. The suffix tokens tend to help the model learn the generation capability, we adopt uniform masking as mentioned in CMLM (Ghazvininejad et al., 2019), denoted as $(Y_{mask}, Y_{obs}) = \texttt{UNIFORM\_MASK}(Y, \beta_Y)$, where β_Y is sampled from a uniform distribution $U(0, 1)$. Then GeMLM predicts the masked tokens based on different context.

In practice, we adopt an adaptive masking function for the masking ratio β_X as mentioned in (Xiao et al., 2023) to replace the fixed masking ratio in the traditional MLM, as $\beta_X = 0.3 - \beta_Y * 0.2$. This operation can achieve more diverse masking conditions in X for the model to learn and is based on the intuition that once more tokens in Y are masked, X should provide more context information (i.e., lower β_X). Besides, we prevent the query of each token in X attending the tokens in Y in the attention module as mentioned in figure 4 during training, which keeps consistent with the inference process since there is no target sequence in advance. Then, the final training loss of GeMLM can

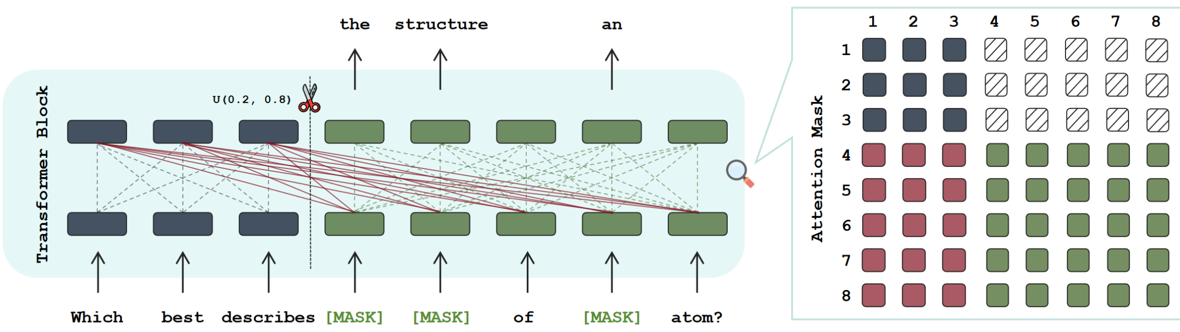

Figure 4: Presentation of generative masked language modeling.

be computed as:

$$\mathcal{L}_{\text{GeMLM}} = -\sum_{x_t \in X_{mask}} \log \mathcal{P}(x_t | X_{obs}; \theta) - \sum_{y_t \in Y_{mask}} \log \mathcal{P}(y_t | X_{obs}, Y_{obs}; \theta). \tag{5}$$

F Details for pre-training

Details of the pre-training models and settings are present in Table 6.

Parameters	GeBERT-124M	GeBERT-352M
Num_layers	12	24
Hidden_size	768	1024
Num_attn_heads	12	16
Init_std	0.02	0.02
Seq_length	2048	2048
Batch_size	1024	1024
Train_iters	153000	153000
Learning_rate	6e-4	3e-4
Lr_decay_style	cosine	cosine
Clip_grad	1.0	1.0
Adam_beta	(0.9,0.95)	(0.9, 0.95)
Weight_decay	1e-2	1e-2

Table 6: Details of the pre-training models and setting.

G More comparisons of composition formats.

Except those as mentioned in Section 5.1, we can also adopt the following composition formats: (1) Notice in Table 1, we regulate the number of newly generated tokens (denoted as n_{new}) in each decoding step as $n_{new} = 1$ to keep consistent with AR models, i.e., we generate only one token in a left-to-right order or with the highest-k prediction probabilities in each decoding step, and adopt the total decoding steps adaptive to the target length. Then, we can (1) set $n_{new} = 2/3/4$, and the corresponding decoding steps as $\lceil N/2 \rceil, \lceil N/3 \rceil, \lceil N/3 \rceil$, where N denotes the total target tokens, we denote this method as multi-token-based, (2) set the total decoding steps as $T = 1/4/10$, and the corresponding $n_{new} = \lceil N/1 \rceil, \lceil N/4 \rceil, \lceil N/10 \rceil$. We denote this method as multi-step-based. Besides, with $n_{new} = 1$, there still exist different rules to achieve the specific generated token. The corresponding results are presented in Table 7, we can find that: (1) The performance declines as the n_{new} increases, indicating that setting $n_{new} = 1$ to keep consistent with AR models, in which the model predicting one token in each decoding step, is important to achieve competitive performance. (2) With the multi-step-based method, more decoding steps lead to better performance, which also verifies the above observation, i.e., the length of targets is less than the decoding steps in several tasks, such as Sciq and SIQA, where the model will also predict one token in each decoding step. Conversely, the performance on these tasks which contain the relatively long targets such as PIQA and ARC still falls behind the left-to-right baseline.

Hyber.	**LogiQA**	**Sciq**	**ARC-E**	**ARC-C**	**Wino.**	**BoolQ**	**PIQA**	**SIQA**	**Race**	**Hella.**	**Truth.**	**AVG.**
left-to-right	27.65	80.3	42.13	22.10	50.75	62.17	60.66	36.49	28.90	29.26	24.60	42.26
multi-step-based												
$T = 1$	23.50	64.6	36.49	21.93	50.75	62.17	54.19	34.75	23.44	28.15	21.42	38.30
$T = 4$	26.73	80.4	41.20	21.33	50.99	62.17	57.24	36.64	28.71	30.74	25.95	42.01
$T = 7$	26.42	80.5	41.04	22.19	52.41	62.17	58.16	36.54	29.67	31.11	24.48	42.24
multi-token-based												
$n_{new} = 1$	29.19	80.4	41.96	22.44	52.57	62.17	59.79	36.80	29.79	31.73	25.58	42.90
$n_{new} = 2$	29.03	71.1	40.15	22.44	50.99	62.17	59.19	36.89	29.47	31.68	25.09	41.65
$n_{new} = 3$	27.96	66.8	37.79	22.36	50.12	62.17	57.07	35.31	27.94	30.83	24.97	40.30
$n_{new} = 4$	29.65	65.5	38.39	22.53	49.17	62.17	55.06	35.47	27.37	30.40	24.24	40.00

Table 7: Results of different methods to select the decoding paths for zero-shot common sense reasoning and reading comprehension tasks. **Hyper.** denotes the corresponding hyperparameter.