

000 001 002 003 004 005 PATH SELECTION MAKES BERT-FAMILY GOOD GEN- 006 ERATORS 007 008 009

010 **Anonymous authors**
 011 Paper under double-blind review
 012
 013
 014
 015
 016
 017
 018
 019
 020
 021
 022
 023
 024
 025
 026
 027
 028
 029
 030
 031
 032
 033

ABSTRACT

034 Previous research endeavors have sought to tailor BERT-family for non-
 035 autoregressive generation tasks through task-specific fine-tuning, and more re-
 036 cent work has also attempted to evolve them into instruction followers after in-
 037 struction tuning, but their generation quality, even with more powerful iterative
 038 decoding methods, continues to lag behind the competitive autoregressive (AR)
 039 models. Furthermore, these progress rely on additional fine-tuning procedures,
 040 while the generative potential of the BERT-family without fine-tuning still re-
 041 mains unexplored. Hence, we delve deeper into the key issues leading to the
 042 performance gaps of BERT-family in generation tasks and put forth innovative
 043 solutions. Specifically, existing studies often overlook the significance of the train-
 044 ing sequence decomposition format. Unlike autoregressive models that naturally
 045 decompose text sequences in a left-to-right fashion during both training and in-
 046 ference, BERT-family are trained using a random decomposition approach (i.e.,
 047 randomly masking the texts) but strive to identify an optimal composition (re-
 048 ferred to decoding paths) during inference. To alleviate this mismatching, we
 049 introduce a **path selection** method to expand the search space during inference,
 050 facilitating the discovery of more suitable compositions. Additionally, we pro-
 051 pose **path selection***, which further integrates path selection into the training pro-
 052 cess, enabling the model to learn preferences for specific paths. Our experiments
 053 across a range of zero-shot common sense reasoning and reading comprehension
 054 tasks and several task-specific generation tasks showcase the substantial perfor-
 055 mance enhancements for the BERT-family using the proposed methods, reaching
 056 the performance levels that are on par with, and in some cases surpassing the AR
 057 pre-trained models. Our models and code will be publicly accessible on GitHub.
 058

059 1 INTRODUCTION

060 In recent years, the progress of large language models, such as Llama (Touvron et al., 2023a;b),
 061 Gemini (Team et al., 2023), and GPT-4 (OpenAI, 2023), has revolutionized the natural language pro-
 062 cessing (NLP) tasks, demonstrating remarkable capabilities across a diverse range of applications.
 063 The majority of these models follow a decoder-only autoregressive architecture, drawing inspiration
 064 from the successful GPT series models (Radford et al.; 2019; Brown et al., 2020). Conversely, the
 065 BERT-family series models (Devlin et al., 2018; Liu et al., 2019; Conneau et al., 2020; He et al.,
 066 2020), which have long been recognized for their excellent performance in various language under-
 067 standing tasks, appear to be progressing at a relatively slower pace recently. Researchers attribute
 068 this deceleration to challenges encountered in language generation scenarios, where BERT-family
 069 models struggle to achieve competitive performance when compared to autoregressive (AR) models.
 070

071 Previous works have delved into strategies for leveraging the BERT-family for language generation
 072 tasks, particularly following the introduction of the Mask-Predict algorithm (Ghazvininejad et al.,
 073 2019), which offers a promising approach for adapting BERT-family to non-autoregressive (NAR)
 074 generation scenarios. Researchers have attempted to customize BERT-family for simple specific
 075 generation tasks (Chan & Fan, 2019; Jiang et al., 2021; Su et al., 2021; Liang et al., 2023b;a), and
 076 recent advancements have also explored the potential for BERT-family in becoming instruction fol-
 077 lowers after instruction tuning (Xiao et al., 2024). Despite their efforts, performance gaps persist
 078 when compared to AR models. Furthermore, these endeavors need additional fine-tuning proce-
 079

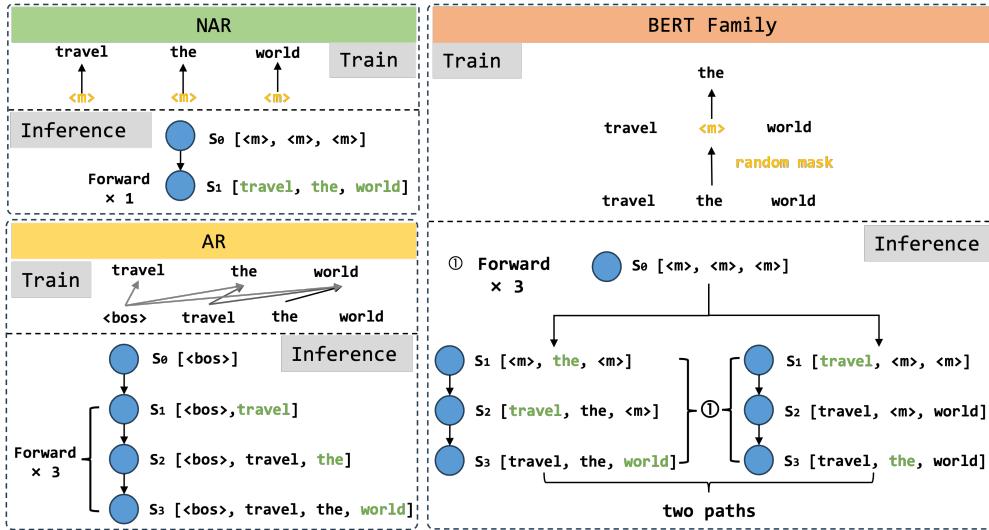


Figure 1: The sequence decomposition for training, and composition methods (i.e., decoding path to achieve the outputs sequence) for different language models.

dures to endow BERT-family models with generation capabilities, thereby leaving the BERT-family without fine-tuning under-explored.

In this paper, we delve deeper into the key issues leading to the performance gaps of BERT-family in generation tasks and put forth innovative solutions. One of the central challenges may stem from the sequence decomposition format. As shown in the Figure 1, AR models naturally decompose language modeling as the task of next-token prediction in a left-to-right order and leverage the corresponding composition strategy during inference. In contrast, BERT-family adopt a random decomposition approach during training (i.e., randomly masking several tokens) but follow certain given criteria to identify an optimal composition (denoted as decoding path in this paper) to achieve target sequences during inference, resulting in a significant training-inference gap. Regarding this mismatching, we propose **path selection** which enables BERT-family models to choose a more appropriate composition from multiple candidates during inference. Additionally, we present **path selection***, a novel approach that incorporates path selection into the training process, which aims to instruct BERT-family on learning preferences for the outputs achieved by various compositions.

In our study, we introduce a newly developed variant of the BERT-family, which we refer to as **Generative BERT (GeBERT)**, to launch a fair comparison with AR models. GeBERT is built on an encoder-only architecture with a bidirectional attention mechanism akin to the conventional BERT-family. Additionally, in line with several current mainstream AR pre-training models (Zhang et al., 2022; Black et al., 2022; Biderman et al., 2023; Peng et al., 2023), we integrate several innovative techniques and adopt a more widely-used training corpus to pre-train GeBERT. Initially, we assess our proposed methods across a range of zero-shot common sense reasoning and reading comprehension tasks using GeBERT without fine-tuning. Our results illustrate that our methods lead to significant performance improvements compared to the baseline GeBERT model, surpassing all AR baselines with comparable model parameters. Besides, we evaluate our methods on two traditional language generation tasks after fine-tuning the models. The corresponding results demonstrate the effectiveness of our proposed methods, with GeBERT achieving state-of-the-art performance in various testing scenarios compared to previous AR and NAR pre-training models.

2 PRELIMINARY

2.1 UTILIZING BERT-FAMILY FOR LANGUAGE GENERATION TASKS

Previous works (Dong et al., 2019; Wang & Cho, 2019) have theoretically indicated that the BERT-family can be utilized for generating texts by predicting the masked positions in the target sequence.

108 Despite early efforts by researchers to leverage BERT-family for language generation tasks (Chan &
 109 Fan, 2019; Jiang et al., 2021; Su et al., 2021), these attempts yielded suboptimal results compared
 110 to the mainstream generative models. Subsequently, researchers attempt to adapt BERT-family to
 111 NAR scenarios (Liang et al., 2023b;a; Xiao et al., 2024) via the the Mask-Predict decoding algo-
 112 rithm (Ghazvininejad et al., 2019), which first predicts the entire masked target sequence in the first
 113 decoding step, and then refines the target sequence by replacing the unreliable parts with masked
 114 tokens and re-generating them in parallel in the subsequent decoding step as details shown in the
 115 Appendix A.1, and receives relatively positive feedback regarding performance. During training,
 116 these models learn to predict the masked parts in the target sequence, whose loss can be computed
 117 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}
 118 are the masked and unmasked parts in the target sequence Y , respectively. In this paper, we further
 119 delve into the essential technological advancements of BERT-family that leverage the Mask-Predict
 120 decoding algorithm to achieve better performance in generation tasks.
 121

2.2 DECODING PATHS FOR BERT-FAMILY

123 Formally, we consider the process of generating a sequence of discrete tokens $Y = (y_1, \dots, y_N)$,
 124 where $y_i \in V$, a finite vocabulary specific to a language model. This generation process can be
 125 interpreted as deterministically sampling a series of successive state spaces S , where each state
 126 $s_i \in S$ corresponds to a sequence of tokens sampled from V , and relies on a policy π to transition
 127 to the next state. A policy π serves as a determinate mapping from states to actions, outlining how
 128 the model processes the current sequence and achieves the subsequent sequence in the next state.
 129 We denote this specific process to compose the target sequence as the decoding path P of a given
 130 language model, where each node represents the current state s_i in i th decoding step and each edge
 131 represents the policy π_i indicating the actions for transitioning from state s_i to s_{i+1} .
 132

133 As shown in Figure 1, different language models have their specific decoding paths to compose the
 134 target sequence. The traditional AR and NAR language models typically have a single decoding
 135 path for composing a specific target sequence, while BERT-family can explore multiple optional
 136 decoding paths, resulting in varied output sequences of differing generation qualities. Selecting a
 137 specific decoding path from the multitude of optional paths available in BERT-family is crucial for
 138 achieving high-quality outputs. With approximately 2^{TN} possible paths for a BERT-family model,
 139 as detailed in the Appendix A.2, determining the optimal path is essential for the success of these
 140 models. In Ghazvininejad et al. (2019) where the Mask-Predict decoding algorithm was first pro-
 141 posed, the authors heuristically regulate the policy π_t in t th decoding step as predicting the masked
 142 parts in current Y and selecting the specific n_t tokens which are with lowest prediction probabilities
 143 to be re-masked, where the number of re-masked tokens can be computed as $n_t = (1 - t/T) * N$,
 144 N denotes the total number of tokens in Y , t and T denote the current and total decoding step,
 145 respectively. While the Mask-Predict algorithm provides a heuristic approach to selecting decoding
 146 paths, it may not always yield optimal results. There exist other decoding paths in the candidate
 147 space leading to better composition of target sequences (Kreutzer et al., 2020). Hence, we aim to
 148 identify an optimal decoding path from such multitudinous candidates by introducing **path selection**
 149 method. Moreover, we further propose **path selection*** which empowers the model to learn the pre-
 150 ference between different decoding paths. Our methods seek to enhance the BERT-family’s ability
 151 to navigate through the complex decoding spaces and generate higher-quality output sequences.
 152

3 APPROACH

3.1 PATHS SELECTION

154 We first sample several optional decoding paths from the candidate spaces and select the best one
 155 with the highest total prediction probability. Specifically, we follow most of the practice in the
 156 Mask-Predict algorithm, except for the selection of the re-masked tokens in each decoding step.
 157 As shown in the left of Figure 2, rather than just selecting a specific number of tokens with the
 158 lowest prediction probabilities to transform to the unique next state (i.e., the first beam), we allow
 159 total k candidate selections for re-masked tokens with the lowest- k total prediction probabilities
 160 for each decoding path, where k is the position beam number set in advance. Notice we keep
 161 the number of candidate states in each decoding step as k , which is similar to the beam search
 162 algorithm for AR models (Meister et al., 2020). However, the search times to select those with
 163

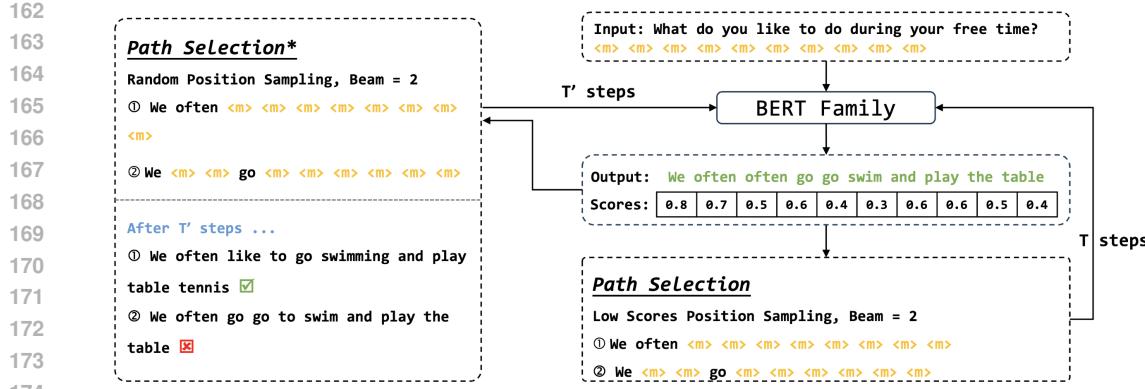


Figure 2: Overview of the path selection and path selection* methods. As for the path selection method during inference, we select the positions for masked tokens with the lowest- k prediction probabilities, while the path selection* randomly samples the positions for masked tokens.

the lowest- k total prediction probabilities 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 $k * \sum_{t \in \{1, 2, \dots, T\}} C_{n_t}^N$, where its detailed proof is in Appendix A.3. Therefore, to reduce the search overhead, we further introduce a simplified version that transforms the search times in t th 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 PATHS 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 randomly¹ 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 A.4. Subsequently, we use a score function $Score(\cdot)$, such as 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 $Score(Y_{out}^1) > Score(Y_{out}^2)$, we adopt Y_{out}^1 as the positive output Y_w and Y_{out}^2 as the negative 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:

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{ref}) = -\log \sigma \left[\beta \left(\frac{\pi_\theta(Y_w|Y_{obs}, X)}{\pi_{ref}(Y_w|Y_{obs}, X)} - \frac{\pi_\theta(Y_l|Y_{obs}, X)}{\pi_{ref}(Y_l|Y_{obs}, X)} \right) \right], \quad (1)$$

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

¹We 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.

probabilities of positive outputs is contrary to our expectations. The penalty terms can be computed as follows:

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

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:

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

Our final training loss can be 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 hyperparameters 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 **Generative BERT (GeBERT)**. Specifically, unlike earlier BERT-like models that only mask 15% of tokens in each instance for prediction (Devlin et al., 2018; Liu et al., 2019), we first decompose the training instance into two parts which simulates a scenario akin to conditional generation. Drawing inspiration from prior practice (Song et al., 2019; Li et al., 2022b; Guo et al., 2020; Xiao et al., 2023), we assign different masking methods for these two parts to enable GeBERT to learn both understanding and generation capabilities. Details of our pre-training task are presented in the Appendix A.5. 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; Huang et al., 2023), denoted as **GeBERT-124M** and **GeBERT-352M**.

Pre-training Settings As for the model architecture, we follow the most practice in previous BERT-like models to build an encoder-only language model with a bi-directional attention mechanism and further incorporate several effective techniques: 1) We use Rotary Positional Embedding (RoPE) (Su et al., 2024) to inject positional information into GeBERT rather than the traditional absolute/relative position encoding. 2) We adopt swiglu Shazeer (2020) as our activation function rather than the traditional ReLU. For pre-training corpus, we adopt the Pile (Gao et al., 2020; Biderman et al., 2022), which is a curated collection of English language datasets containing around 300B tokens and has been widely used for training language models (Zhang et al., 2022; Black et al., 2022; Biderman et al., 2023; Peng et al., 2023). During training, we set the max length as 2048 and pre-train GeBERT for 150k update steps (1 epoch on the Pile) with a batch size of 1024 samples (i.e., 2M tokens). We use Adam optimizer (Kingma & Ba, 2014) with a weight decay of 0.01, the learning rate warms up to 6e-4 in the first 1.5k steps and then decreases gradually with cosine decay strategy. We utilize the Megatron-Deepspeed² library to train GeBERT on 8 NVIDIA A100-PCIE-80GB GPU cards. We present the details for the model and pre-training in the Appendix A.6.

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-autoregressive 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,

²<https://github.com/microsoft/Megatron-DeepSpeed>

270
 271 Table 1: Results on zero-shot common sense reasoning and reading comprehension tasks. The first
 272 line of GeBERT denotes the baseline which adopts the same decoding path as AR models. **Bold**
 273 values denote the best average result (**AVG.**) through all models. underlined values denote the result
 274 of our methods outperforming the baseline GeBERT. The abbreviations **Wino.**, **Hella.**, and **Truth.**
 275 denote the WinoGrande, Hellaswag, and Truthfulqa datasets, respectively.
 276

Models	LogiQA	Sciq	ARC-E	ARC-C	Wino.	BoolQ	PIQA	SIQA	Race	Hella.	Truth.	AVG.
$\approx 150M$ 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	<u>81.8</u>	42.09	<u>22.36</u>	<u>51.87</u>	62.17	59.69	<u>36.80</u>	<u>29.28</u>	<u>31.70</u>	<u>25.70</u>	<u>42.89</u>
+ Path Selection*	28.88	<u>80.5</u>	<u>42.47</u>	<u>22.18</u>	<u>52.72</u>	62.17	<u>60.88</u>	<u>36.94</u>	<u>29.76</u>	<u>32.25</u>	<u>25.74</u>	43.14
$\approx 350M$ 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	<u>29.87</u>	<u>83.60</u>	<u>51.65</u>	<u>24.24</u>	52.87	62.17	65.03	<u>39.26</u>	<u>30.83</u>	<u>41.03</u>	<u>25.58</u>	<u>46.03</u>
+ 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

290 we train the model with 5 epochs. As for the DPO training of the vanilla GeBERT, we initialize the
 291 policy and reference model with the final saved checkpoint during pre-training. We sampled a small
 292 subset from the pile to conduct DPO training and avoid introducing extra training data.
 293

294 4.2 EVALUATION DETAILS

295 **Datasets and Metrics** We evaluate our proposed methods on common downstream task-specific
 296 generation tasks, which have been widely used in previous pre-trained AR and NAR works, and
 297 various zero-shot common sense reasoning and reading comprehension tasks, which are popular to
 298 evaluate the vanilla version of current large language models without fine-tuning (Zeng et al., 2022;
 299 Touvron et al., 2023a;b). To the best of our knowledge, we are the first to evaluate the pre-trained
 300 NAR models for these zero-shot tasks. Specifically, For downstream task-specific generation tasks,
 301 we adopt XSUM (Narayan et al., 2018) for the summarization task and MSQG (MicroSoft Ques-
 302 tion Generation) dataset for the question generation task from the GLGE benchmark (Liu et al.,
 303 2021). For the evaluation metrics, we adopt ROUGE F1 (ROUGE-1/2/L) (Lin & Hovy, 2002)
 304 for XSUM, and BLEU (BLEU-4) (Papineni et al., 2002), Rouge-L and METEOR (Lavie & Agar-
 305 wal, 2007) for MSQG. For zero-shot common sense reasoning and reading comprehension tasks,
 306 we adopt ARC-easy, ARC-challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), PIQA (Bisk
 307 et al., 2020), SIQA (Sap et al., 2019), WinoGrande (Sakaguchi et al., 2021), Race (Lai et al., 2017),
 308 Sciq (Johannes Welbl, 2017), LogiQA (Liu et al., 2020), Hellaswag (Zellers et al., 2019), and Truth-
 309 fulqa (Lin et al., 2021), which are all widely used for evaluating recent language models. We adopt
 310 Language Model Evaluation (Gao et al., 2021) framework to evaluate these datasets under a zero-
 311 shot setting (Biderman et al., 2023). We adopt normalized accuracy for PIQA, ARC-challenge,
 312 LogiQA, Hellaswag, and accuracy for other tasks following previous works (Biderman et al., 2023).
 313

314 **Baseline Models** For the downstream task-specific generation tasks, we adopt the vanilla Trans-
 315 former baseline (Vaswani et al., 2017) and previous pre-trained AR models including MASS (Song
 316 et al., 2019), BART Lewis et al. (2019), and ProphetNet (Qi et al., 2020) which are included in the
 317 official GLGE evaluation leaderboard as autoregressive baselines. For NAR baselines, we adopt
 318 the previous pre-trained NAR models including BANG (Bang et al., 2023), ELMER (Li et al.,
 319 2022a) and PreDAT (Huang et al., 2023). Besides, we also include MIST (Jiang et al., 2021) and
 320 DEER (Liang et al., 2023a) which also fine-tune the traditional BERT-family to complete the gener-
 321 ation tasks. For common sense reasoning and reading comprehension tasks, which are only widely
 322 used after the popularity of large language models and never been included in the evaluation of pre-
 323 vious NAR models, we adopt the recent large language models which are also trained on the Pile for
 324 around 300B tokens and contains the comparable model parameters with GeBERT, including OPT-
 325 125M/350M (Zhang et al., 2022), GPT-neo-125M (Black et al., 2022), Pythia-160M/410M (Bider-
 326 man et al., 2023).

324
 325 Table 2: Results on task-specific generation tasks. **Bold** denotes the best result. underlined values
 326 denote the result of our methods outperforming the baseline GeBERT.

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 ($\approx 150M$ 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	<u>40.52</u>	<u>17.11</u>	<u>32.71</u>	39.06	9.52	<u>23.51</u>	1.2x
+ Path Optimization	40.92	17.39	33.08	39.46	<u>9.72</u>	23.68	1.2x
Large Version ($\approx 350M$ 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	<u>44.33</u>	<u>21.23</u>	<u>36.40</u>	<u>39.38</u>	10.21	<u>23.90</u>	-
+ Path Optimization	<u>44.84</u>	<u>21.89</u>	<u>36.89</u>	<u>39.78</u>	10.29	24.32	-

347 man et al., 2023), and RWKV-169M/430M (Peng et al., 2023). We re-run all the baseline models
 348 under the same Language Model Evaluation framework (Gao et al., 2024) using their open-source
 349 Hugging Face models to ensure consistent evaluation procedures.
 350

351 4.3 MAIN RESULTS

353 **Zero-shot common sense reasoning and reading comprehension** We present the results on various
 354 zero-shot common sense reasoning and reading comprehension tasks in Table 1. Compared
 355 with GeBERT and the previous AR models, we can find that: (1) GeBERT can also complete these
 356 zero-shot tasks and achieve comparable performance while adopting the same decoding path during
 357 inference³. (2) Our final models (i.e., GeBERT with path selection and path selection*) achieve
 358 the best performance through all the previous AR models on average, outperforming the previous
 359 best models (i.e., GPT-neo-125M and Pythia-410M) by around 0.8 and 0.5 score. (4) GeBERT is
 360 better at reading comprehension tasks which enable the model to answer questions given supports
 361 or evidences such as Sciq and LogiQA, we attribute this to the bi-directional attention mechanism of
 362 GeBERT. Besides, Compared with baseline GeBERT which adopts the same decoding path as AR
 363 models and those with our path selection and path selection* methods, we can find that: (1) With the
 364 path selection method, GeBERT outperforms the baseline GeBERT in most of the evaluation tasks,
 365 leading to 0.6/1.0 performance improvements on GeBERT-124M/352M. (2) Further, with the path
 366 selection* method, GeBERT can outperform the baseline GeBERT in 10 of 11 evaluation tasks and
 367 be on par in BoolQ, leading to around 1.0 performance improvements on average. (3) By comparing
 368 GeBERT only with the path selection method and with both proposed methods, the former can
 369 achieve performance improvements on most tasks, indicating the effectiveness of the path selection*
 370 method. However, the path selection* may also result in performance declines in several tasks, such
 371 as Sciq for GeBERT-124M and ARC-C for GeBERT-352M.

372 **Task-specific generation** Table 2 presents the results on task-specific generation task. We can find
 373 that: (1) For the summarization task, though GeBERT-352M underperforms BART-large GeBERT-
 374 124M, it outperforms all the other baseline models in all evaluation metrics, indicating that GeBERT
 375 can generate more informative and reasonable summaries. (2) For the question generation task,
 376 GeBERT-124M outperforms all the baseline models on Rouge-L and METEOR and only presents

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

378
 379 Table 3: Results of different methods to select the decoding paths for zero-shot common sense
 380 reasoning and reading comprehension tasks. **Hyper.** denotes the corresponding hyperparameter.

Hyper.	LogiQA	SciQ	ARC-E	ARC-C	Wino.	BoolQ	PIQA	SIQA	Race	Hella.	Truth.	AVG.
<i>multi-step-based</i>												
$T = 1$	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
<i>multi-token-based</i>												
$n_{new} = 2$	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
<i>multi-order-based</i>												
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
<i>multi-beam-based</i>												
$k = 2$	28.11	81.4	42.59	22.01	52.01	62.17	59.69	36.54	30.05	31.98	25.45	42.94
$k = 3$	27.65	81.8	42.09	22.36	51.86	62.17	60.01	36.80	29.67	32.17	25.58	42.90
$k = 4$	28.26	81.8	42.51	22.61	51.07	62.17	60.28	36.28	29.28	32.09	25.70	42.91

398
 399 performance gaps compared with the best baseline models on BLEU-4. GeBERT-352M achieves
 400 the best performance across various models on all evaluation metrics. (3) Compared to the GeBERT
 401 baseline, which adopts the original vanilla Mask-Predict algorithm to generate the output sequence,
 402 the path selection and path selection* methods can bring performance improvements on the XSUM
 403 dataset for both GeBERT-124M/352M, indicating that these two methods can enable the model to
 404 achieve better performance in generating relatively long targets. However, the path selection method
 405 does not lead to consistent performance improvements on the MSQG dataset, which contains rela-
 406 tively short targets. We attribute this to that short sequences will lead to relatively small candidate
 407 space and redundant outputs for different decoding paths, thus we can not achieve better outputs
 408 from multiple candidates. (4) We also compare the decoding efficiency of GeBERT-124M and
 409 BART-base, which contains around 140M parameters, and the results demonstrate that GeBERT
 410 can achieve 3.1x speedup with the vanilla Mask-predict algorithm due to the NAR attribute. Further,
 411 with path selection and path selection* methods, which will bring the extra search overhead for dif-
 412 ferent decoding paths, GeBERT still achieves a faster generation process, leading to a 1.2x speedup
 413 compared to BART.

5 ANALYSIS

5.1 DISCUSSION OF DIFFERENT METHODS TO DETERMINE THE DECODING PATHS

414
 415 In Table 1, we report the performance of GeBERT which adopt the same decoding path with AR
 416 models and that with our proposed path selection method. However, there still exist multitudinous
 417 optional decoding paths to generate the final target sequences as mentioned in section 3.1, we adopt
 418 GeBERT-124M to compare the different methods to determine the decoding paths here with the
 419 following rules. Notice in Table 1, we regulate the number of newly generated tokens (denoted as
 420 n_{new}) in each decoding step as $n_{new} = 1$ to keep consistent with AR models, i.e., we generate only
 421 one token in a left-to-right order or with the highest- k prediction probabilities in each decoding step,
 422 and adopt the total decoding steps adaptive to the target length. Then, we can (1) set $n_{new} = 2/3/4$,
 423 and the corresponding decoding steps as $\lceil N/2 \rceil, \lceil N/3 \rceil, \lceil N/3 \rceil$, where N denotes the total target
 424 tokens, we denote this method as multi-token-based, (2) set the total decoding steps as $T = 1/4/10$,
 425 and the corresponding $n_{new} = \lceil N/1 \rceil, \lceil N/4 \rceil, \lceil N/10 \rceil$. We denote this method as multi-step-
 426 based. Besides, with $n_{new} = 1$, there still exist different rules to achieve the specific generated
 427 token. (3) Rather than a left-to-right order in Table 1, we can also include a right-to-left or a random
 428 order to achieve the final target sequence. Instead of selecting the token with the highest prediction
 429 probability, which denoted as a easy-to-hard order (Kasai et al., 2020), we include a hard-to-easy
 430 order which generate the token with the lowest prediction probability first. These method can be

denoted as multi-order-based. (4) We can set different beam number k in our proposed path selection method, we compare $k = 2/3/4$ here, denoted as multi-beam-based.

We present the corresponding results with the above methods in Table 3, 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. (3) Based on different orders, the easy-to-hard order which we adopt in the path selection method perform best, while several other orders will lead to significant performance declines such as right-to-left order. (4) Adopting different beams in our path selection method performs differently for various tasks but achieve a comparable score on average, and all outperforms the left-to-right baseline, indicating the effectiveness of the path selection method.

5.2 COMPARISON WITH TOKENS-AWARE BEAM SEARCH

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, where the results are presented in Table 4. We can 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.

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}_{DPO} and \mathcal{L}_{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

Table 4: Results of different beam search algorithms.

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 5: Result of different λ_1 and λ_2 .

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

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 DISCUSSIONS

Discussion with other non-monotonic generation models. While BERT-family adopting the Mask-Predict algorithm can be viewed as a special case for non-monotonic generation, we also present the discussion with other works that also explore the non-monotonic generation for language models (Welleck et al., 2019; Shih et al., 2022; Shen et al., 2023). These models can also generate output sequences beyond a strict left-to-right order but still differ from the BERT-family. Specifically, these models only generate one token via one decoding step, rather than adopting the parallel decoding paradigm allowing generating multiple tokens for BERT-family. Furthermore, BERT-family adopt the masking language modeling during training, differing from other models which adopt the variants of autoregressive modeling.

Discussion with other path selection methods. In the realm of NAR translation, prior research has also delved into enhancing output quality by selecting the optimal result from multiple candidate generations, such as CTC-decoding (Graves et al., 2006; Shu et al., 2020; Shao & Feng, 2022) and DA-Transformer (Huang et al., 2022; Shao et al., 2022a; Huang et al., 2023). Nevertheless, these methods typically adhere to a left-to-right order when sampling various candidates. Conversely, our proposed path selection method eliminates order constraints, facilitating the candidates with more diversity. Besides, while some studies have advocated for training models with multiple references (Shao et al., 2022b; Huang et al., 2022; Liu et al., 2023), these methods treat all various references as equally plausible outputs and train with the same weights across them. In contrast, our path selection* method incorporates positive-negative sample pairs into the training process, enabling the model to learn preferences among different candidates.

Discussion with other DPO methods for AR models The Direct Preference Optimization (DPO) algorithm has found widespread application in AR models (Yang et al., 2023; Rafailov et al., 2024; Pal et al., 2024), facilitating the learning of preferences for various output sequences generated through different sampling methods, which can significantly enhance the overall generation quality. Conversely, few explorations have been conducted for BERT-family. In this paper, we propose path selection* to enable BERT-family to learn the preference between different outputs achieved by different decoding paths according to the generation character of BERT-family. While AR models allow different lengths of sampled positive-negative pairs, we recognize that ensuring a consistent sampling length across different pairs is quite important for our methods for fair comparisons between different decoding paths. Another difference between our methods and those for AR models is that there is no significant quality difference between our sampled pairs, which will easily lead to the failure cases as mentioned in Section 3.2. Consequently, integrating penalty terms to stabilize the DPO training process emerges as a crucial aspect of our methods.

7 CONCLUSION

In this paper, we pre-train a new version of BERT-family to explore the potential of BERT-family for good generators. According to the specific generation formats of sequences, we propose path selection and path selection* methods to enhance the generation capabilities of these models. Results on a range of zero-shot common sense reasoning and reading comprehension tasks demonstrate that BERT-family can also achieve state-of-the-art performance by adopting our methods compared to current mainstream AR language models, and the experiments on several task-specific generation tasks further verify the effectiveness of our methods, outperforming the previous pre-trained language models. In overall, our works first pre-train a version of BERT-family from scratch for fair comparison with current AR pre-trained models, and further verify the very promising techniques to unlock the potential of BERT-family for various generation tasks both without and with fine-tuning. We hope our explorations can expand the applications of BERT-family in AI community. In the future, we will further explore larger versions after observing the scaling ability from 124M to 352M of GeBERT, then adapt GeBERT to more type of tasks, such as code generation and math reasoning.

540 REFERENCES
541

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

544 Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia,
545 Ziwei Ji, Tiezheng Yu, Willy Chung, et al. A multitask, multilingual, multimodal evaluation of
546 chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*, 2023.
547

548 Stella Biderman, Kieran Bicheno, and Leo Gao. Datasheet for the pile. *arXiv preprint
549 arXiv:2201.07311*, 2022.
550

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

555 Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical com-
556 monsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*,
557 volume 34, pp. 7432–7439, 2020.
558

559 Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace
560 He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autore-
561 gressive language model. *arXiv preprint arXiv:2204.06745*, 2022.
562

563 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,
564 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are
565 few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
566

567 Ying-Hong Chan and Yao-Chung Fan. A recurrent bert-based model for question generation. In
568 *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pp. 154–162,
569 2019.

570 Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina
571 Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint
572 arXiv:1905.10044*, 2019.
573

574 Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and
575 Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge.
576 *arXiv preprint arXiv:1803.05457*, 2018.
577

578 Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek,
579 Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsuper-
580 vised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting
581 of the Association for Computational Linguistics*, pp. 8440–8451, 2020.

582 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep
583 bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
584

585 Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou,
586 and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding
587 and generation. *Advances in Neural Information Processing Systems*, 32, 2019.
588

589 Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason
590 Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text
591 for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.
592

593 Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence
594 Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot
language model evaluation. *Version v0. 0.1. Sept*, pp. 8, 2021.
595

- 594 Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Fos-
 595 ter, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muen-
 596 nighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lin-
 597 tang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework
 598 for few-shot language model evaluation, 07 2024. URL <https://zenodo.org/records/12608602>.
 599
- 600 Marjan Ghazvininejad, Omer Levy, Yinhai Liu, and Luke Zettlemoyer. Mask-predict: Parallel
 601 decoding of conditional masked language models. In *Proceedings of the 2019 Conference on*
 602 *Empirical Methods in Natural Language Processing and the 9th International Joint Conference*
 603 *on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6112–6121, 2019.
- 604
- 605 Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist tempo-
 606 ral classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*,
 607 pp. 369–376, 2006.
- 608 Junliang Guo, Linli Xu, and Enhong Chen. Jointly masked sequence-to-sequence model for non-
 609 autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the*
 610 *Association for Computational Linguistics*, pp. 376–385, 2020.
- 611
- 612 Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert
 613 with disentangled attention. In *International Conference on Learning Representations*, 2020.
- 614 Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. Directed acyclic transformer for
 615 non-autoregressive machine translation. In *International Conference on Machine Learning*, pp.
 616 9410–9428. PMLR, 2022.
- 617
- 618 Fei Huang, Pei Ke, and Minlie Huang. [tacl] directed acyclic transformer pre-training for high-
 619 quality non-autoregressive text generation. In *The 61st Annual Meeting Of The Association For*
 620 *Computational Linguistics*, 2023.
- 621 Ting Jiang, Shaohan Huang, Zihan Zhang, Deqing Wang, Fuzhen Zhuang, Furu Wei, Haizhen
 622 Huang, Liangjie Zhang, and Qi Zhang. Improving non-autoregressive generation with mixup
 623 training. *arXiv preprint arXiv:2110.11115*, 2021.
- 624 Matt Gardner Johannes Welbl, Nelson F. Liu. Crowdsourcing multiple choice science questions.
 625 2017.
- 626
- 627 Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. Parallel machine translation with
 628 disentangled context transformer. *arXiv preprint arXiv:2001.05136*, 2020.
- 629
- 630 Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International*
 631 *Conference on Learning Representations*, 2014.
- 632 Julia Kreutzer, George Foster, and Colin Cherry. Inference strategies for machine translation with
 633 conditional masking. *arXiv preprint arXiv:2010.02352*, 2020.
- 634
- 635 Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading
 636 comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical*
 637 *Methods in Natural Language Processing*, pp. 785–794, 2017.
- 638
- 639 Alon Lavie and Abhaya Agarwal. Meteor: an automatic metric for mt evaluation with high levels
 640 of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical*
 641 *Machine Translation*, pp. 228–231, USA, 2007. Association for Computational Linguistics.
- 642
- 643 Mike Lewis, Yinhai Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer
 644 Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-
 645 training for natural language generation, translation, and comprehension. *arXiv preprint*
 646 *arXiv:1910.13461*, 2019.
- 647
- Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Elmer: A
 648 non-autoregressive pre-trained language model for efficient and effective text generation.
 649 *arXiv:2210.13304*, 2022a.

- 648 Pengfei Li, Liangyou Li, Meng Zhang, Minghao Wu, and Qun Liu. Universal conditional masked
 649 language pre-training for neural machine translation. In *Proceedings of the 60th Annual Meeting*
 650 *of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6379–6391,
 651 2022b.
- 652 Xiaobo Liang, Juntao Li, Lijun Wu, Ziqiang Cao, and Min Zhang. Dynamic and efficient inference
 653 for text generation via bert family. In *Proceedings of the 61st Annual Meeting of the Association*
 654 *for Computational Linguistics (Volume 1: Long Papers)*, pp. 2883–2897, 2023a.
- 655 Xiaobo Liang, Zecheng Tang, Juntao Li, and Min Zhang. Open-ended long text generation via
 656 masked language modeling. In *Proceedings of the 61st Annual Meeting of the Association for*
 657 *Computational Linguistics (Volume 1: Long Papers)*, pp. 223–241, 2023b.
- 658 Chin-Yew Lin and Eduard Hovy. Manual and automatic evaluation of summaries. In *Proceedings*
 659 *of the ACL-02 Workshop on Automatic Summarization*, pp. 45–51, 2002.
- 660 Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human
 661 falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.
- 662 Dayiheng Liu, Yu Yan, Yeyun Gong, Weizhen Qi, Hang Zhang, Jian Jiao, Weizhu Chen, Jie Fu,
 663 Linjun Shou, Ming Gong, et al. Glge: A new general language generation evaluation benchmark.
 664 In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 408–420,
 665 2021.
- 666 Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A
 667 challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint*
 668 *arXiv:2007.08124*, 2020.
- 669 Min Liu, Yu Bao, Chengqi Zhao, and Shujian Huang. Selective knowledge distillation for non-
 670 autoregressive neural machine translation. In *Proceedings of the AAAI Conference on Artificial*
 671 *Intelligence*, volume 37, pp. 13246–13254, 2023.
- 672 Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike
 673 Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining
 674 approach. *arXiv preprint arXiv:1907.11692*, 2019.
- 675 Clara Meister, Tim Vieira, and Ryan Cotterell. Best-first beam search. *Transactions of the Association*
 676 *for Computational Linguistics*, 8:795–809, 2020.
- 677 Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don’t give me the details, just the summary!
 678 topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the*
 679 *2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1797–
 680 1807, 2018.
- 681 OpenAI. Gpt-4 technical report. 2023.
- 682 Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White.
 683 Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint*
 684 *arXiv:2402.13228*, 2024.
- 685 Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic
 686 evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association*
 687 *for Computational Linguistics*, pp. 311–318, 2002.
- 688 Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin
 689 Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. Rwkv: Reinventing rnns for
 690 the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.
- 691 Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and
 692 Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequencepre-training. In *Find-
 693 ings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2401–2410, 2020.
- 694 Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language under-
 695 standing by generative pre-training.

- 702 Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language
 703 models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- 704
- 705 Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea
 706 Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances
 707 in Neural Information Processing Systems*, 36, 2024.
- 708 Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi
 709 Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-
 710 text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>.
- 711
- 712 Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adver-
 713 sarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.
- 714
- 715 Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Common-
 716 sense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical
 717 Methods in Natural Language Processing and the 9th International Joint Conference on Natural
 718 Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.
- 719 Chenze Shao and Yang Feng. Non-monotonic latent alignments for ctc-based non-autoregressive
 720 machine translation. In *NeurIPS*, volume 35, pp. 8159–8173, 2022.
- 721 Chenze Shao, Zhengrui Ma, and Yang Feng. Viterbi decoding of directed acyclic transformer for
 722 non-autoregressive machine translation. In *Findings of EMNLP*, pp. 4390–4397, 2022a.
- 723 Chenze Shao, Xuanfu Wu, and Yang Feng. One reference is not enough: Diverse distillation with
 724 reference selection for non-autoregressive translation. In *NAACL-HLT*, pp. 3779–3791, 2022b.
- 725
- 726 Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.
- 727
- 728 Tianxiao Shen, Hao Peng, Ruoqi Shen, Yao Fu, Zaid Harchaoui, and Yejin Choi. Film: Fill-in
 729 language models for any-order generation. *arXiv preprint arXiv:2310.09930*, 2023.
- 730 Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and inference on any-order autoregressive
 731 models the right way. *Advances in Neural Information Processing Systems*, 35:2762–2775, 2022.
- 732
- 733 Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. Latent-variable non-
 734 autoregressive neural machine translation with deterministic inference using a delta posterior. In
 735 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8846–8853, 2020.
- 736 Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence
 737 pre-training for language generation. In *International Conference on Machine Learning*, pp.
 738 5926–5936. PMLR, 2019.
- 739
- 740 Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: En-
 741 hanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- 742 Yixuan Su, Deng Cai, Yan Wang, David Vandyke, Simon Baker, Piji Li, and Nigel Collier. Non-
 743 autoregressive text generation with pre-trained language models. In *Proceedings of the 16th
 744 Conference of the European Chapter of the Association for Computational Linguistics: Main
 745 Volume*, pp. 234–243, 2021.
- 746 Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu,
 747 Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly
 748 capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- 749
- 750 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
 751 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Ar-
 752 mand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation
 753 language models. *arXiv preprint arXiv:2302.13971*, 2023a.
- 754 Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko-
 755 lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open founda-
 756 tion and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

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

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

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

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

767 Yisheng Xiao, Juntao Li, Zechen Sun, Zechang Li, Qingrong Xia, Xinyu Duan, Zhefeng Wang,
 768 and Min Zhang. Are bert family good instruction followers? a study on their potential and
 769 limitations. In *The Twelfth International Conference on Learning Representations*, 2024. URL
 770 <https://openreview.net/forum?id=x8VNtpCu1I>.

771 Guangyu Yang, Jinghong Chen, Weizhe Lin, and Bill Byrne. Direct preference optimization for
 772 neural machine translation with minimum bayes risk decoding. *arXiv preprint arXiv:2311.08380*,
 773 2023.

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

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

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

782 APPENDIX

783 A.1 DETAILS OF THE MASK-PREDICT ALGORITHM

784 We present an example adopting the Masked-Predict
 785 algorithm to generate the output sequence in Figure
 786 3. Specifically, given the prompt, we first initial the
 787 input as fully masked tokens (i.e., Input 1) and send
 788 it into the model. After the model predict the out-
 789 puts (i.e., Output 1), we will select specific unreli-
 790 able tokens with relatively lower prediction proba-
 791 bilities to mask again (i.e., the yellow parts in outputs).
 792 In the subsequent decoding step, the model will pre-
 793 dict these masked tokens and select several unreli-
 794 able tokens again. We obtain the final target sequence
 795 until reaching the total number of decoding steps set ad-
 796 vance. This decoding algorithm assume that the target
 797 sequence will be refined better through multiple de-
 798 coding steps.

800 A.2 DETAILS OF THE DECODING PATHS FOR 801 BERT-FAMILY

802 **Lemma 1** Given the total length N of target sequence Y , the total decoding step T , the total number
 803 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
 804 that all tokens in Y should be predicted.

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 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.

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 $C_0^N + C_1^N + C_2^N + \dots + C_N^N = 2^N$ 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 $m \in \{1, 2, \dots, N\}$, and different L_{Y_i} have the repeat decoding paths. Actually, we can solve this problem with the Inclusion-Exclusion Principle (Andreeescu et al., 2004). Thus, the total number of decoding paths is:

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

A.3 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 t th decoding step n_t , the total times for vanilla path selection method are $k * \sum_{t \in \{1, 2, \dots, T\}} C_{n_t}^N$, and the search times for the simplified version are $T * k^2$.

Proof 2 In t th 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 $k * \sum_{t \in \{1, 2, \dots, T\}} C_{n_t}^N$. 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.

A.4 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 $O_{mask} = \pi_{ref}(Y_{mask}|Y_{obs})$, $\pi_{ref}(Y_{mask}|Y_{obs})$ 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 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 $O'_{mask} = \pi_{ref}(Y'_{mask}|Y'_{obs} \cup Y_{obs})$.
- (4) We achieve one sampled output of Y_{mask} as $Y'_{obs} \cup O'_{mask}$, denoted as Y_{out}^1 .
- (5) We repeat the above operation to achieve the other sampled output Y_{out}^2 .

After obtaining the pair samples Y_{out}^1 and Y_{out}^2 , we use a score function $Score(\cdot)$ 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 $U(0.2, 0.8)$. 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

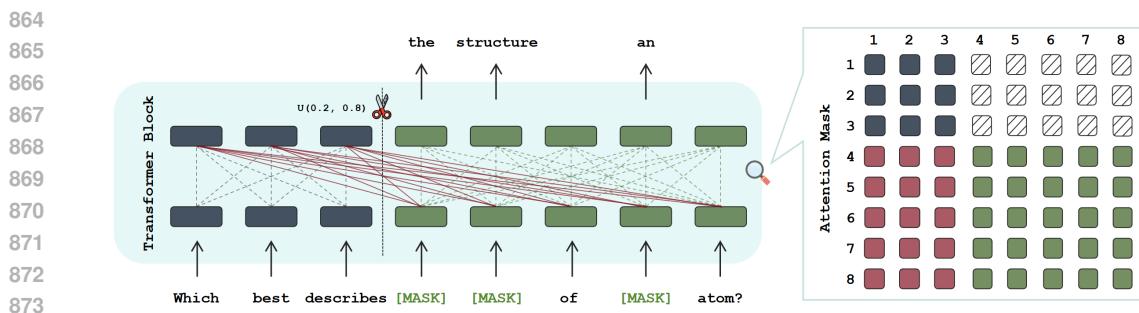


Figure 4: Presentation of generative masked language modeling.

overlaps between two Y'_{obs} , leading the sampling outputs O'_{mask} lacking of diversity, which is not suitable for the DPO training.

A.5 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 & 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 : $C = \{c_1, c_2, \dots, c_{L-1}, c_L\}$, GeMLM decomposes C into a tuple (X, Y) , where $X = \{c_1, c_2, \dots, c_{i-1}, c_i\}$ denotes the prefix tokens, and $Y = \{c_{i+1}, c_{i+2}, \dots, c_{L-1}, c_L\}$ 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 $(X_{mask}, X_{obs}) = \text{RANDOM_MASK}(X, \beta_X)$, 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 $(Y_{mask}, Y_{obs}) = \text{UNIFORM_MASK}(Y, \beta_Y)$, 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 be computed as:

$$\mathcal{L}_{\text{GeMLM}} = - \sum_{x_t \in X_{mask}} \log \mathcal{P}(x_t | X_{obs}; \theta) - \sum_{y_t \in Y_{mask}} \log \mathcal{P}(y_t | X_{obs}, Y_{obs}; \theta). \quad (4)$$

918 A.6 DETAILS FOR PRE-TRAINING
919920 Details of the pre-training models and settings are present in Table A.6.
921

922 Parameters	923 GeBERT-124M	924 GeBERT-352M
925 Num_layers	926 12	927 24
928 Hidden_size	929 768	930 1024
931 Num_attn_heads	932 12	933 16
934 Init_std	935 0.02	936 0.02
937 Seq_length	938 2048	939 2048
940 Batch_size	941 1024	942 1024
943 Train_iters	944 153000	945 153000
946 Learning_rate	947 6e-4	948 3e-4
949 Lr_decay_style	950 cosine	951 cosine
952 Clip_grad	953 1.0	954 1.0
955 Adam_beta	956 (0.9,0.95)	957 (0.9, 0.95)
958 Weight_decay	959 1e-2	960 1e-2

935
936 Table 6: Details of the pre-training models and setting.
937
938939 A.7 LIMITATIONS AND BROADER IMPACTS
940941 Our work demonstrates that BERT-family can perform better than AR language models by adopt
942 our proposed path selection and path selection* methods. However, these models still require multi-
943 step reasoning during zero-shot tasks to bridge the gap between inference and pre-training. This
944 reasoning paradigm may affect the inference efficiency, making BERT-family models less effective
945 than AR models in some contexts. On the other hand, this characteristic becomes an advantage in
946 generative tasks. BERT-family with non-autoregressive generation paradigm do not need to perform
947 sentence-length reasoning steps during decoding, leading to faster generation and reduced latency.
948 This makes BERT-family particularly well-suited for tasks where quick text generation is critical,
949 such as large-scale text generation.
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971