Fixed Input Parameterization for Efficient Prompting

Eunbi Choi[1] **Yongrae Jo**[1] **Joel Jang**[1] **Joonwon Jang**[2*] **Minjoon Seo**[1]
[1]KAIST AI [2]POSTECH
{eunbi,yongrae,joeljang,minjoon}@kaist.ac.kr
kaoara@postech.ac.kr

Abstract

Recent works have shown that attaching prompts to the input is effective at conditioning Language Models (LM) to perform specific tasks. However, prompts are always included in the input text during inference, even when they are fixed, thus incurring substantial computational and memory overhead. Also, there is currently no straightforward method of utilizing prompts that are longer than the maximum input length of the LMs without incurring additional costs during inference. We formally define Fixed Input Parameterization (FIP) problem that focuses on injecting the fixed prompt into the parameters of an LM to be an efficient alternative to attaching fixed prompts to the input. We show that in scenarios with long fixed prompts, FIP can be up to 280 times more efficient in terms of total FLOPs than previous approaches. We further explore methodologies for FIP and show promising results in persona-dependent conversation, semantic parsing, and zero-shot learning with task instructions. Through these explorations, we show that FIP can be a promising direction for conditioning language models, in scenarios with long and fixed prompts[1].

1 Introduction

Contemporary works on Language Models (LMs) (Raffel et al., 2020; Brown et al., 2020; Sanh et al., 2022; Thoppilan et al., 2022) have shown that attaching prompts to the input is effective at conditioning LMs to perform specific tasks. Note that the *prompt* in this work refers to a broader aspect of prompts which includes both the prompts used to induce specific behavior as well as prompts used to provide some contextual knowledge such as persona for dialogue agents. LMs are trained to condition on the given prompts in hopes of generalizing to unseen prompts during inference. Unseen prompts can be a persona for persona-dependent conversation (Zhang et al., 2018; Xu et al., 2022), database schema for semantic parsing (Yu et al., 2018; Hazoom et al., 2021), and task instruction for zero-shot learning with task instructions (Wei et al., 2022; Sanh et al., 2022). In these tasks, a new prompt is fixed to the input at every inference. For instance, in persona-dependent conversation, a persona description is appended to the dialogue history, so that the LM can always be conditioned on the persona. For another example, in semantic parsing, the LM is conditioned on the database schema as well as natural language questions to generalize to a new database. Lastly, zero-shot learning with task instructions involves adding natural language instructions to the inputs for adapting LMs to novel tasks.

However, concatenating prompts to input sequences for prompt-dependent inference has two major limitations. (1) During inference, prompts are always included in the input text and thus incur computational and memory overhead (Liu et al., 2022). (2) It is challenging to fit a long text such as the detailed description of a persona as a prompt into Transformer-based models whose input lengths are often fixed (Tay et al., 2022). For instance, in persona-dependent conversation, the model constantly refers to the persona description along with the dialogue history (Wolf et al., 2019; Roller et al., 2021), as shown in the left side of Figure 1. Moreover, in real-world scenarios, a persona may consist of a long detailed text description of a character or person, not just a few profile sentences. Naively concatenating long prompts to the input sequences is challenging due to the quadratic cost in time and memory of Transformer-based architectures with regard to the input sequence length. Other approaches specialized for processing long inputs (Beltagy et al., 2020; Katharopoulos et al., 2020; Izacard and Grave, 2021), or those that augment the LM with a retrieval mechanism (Han et al.,

[*] Work done during internship at KAIST AI.
[1] Code used for the experiments is available at this link

Findings of the Association for Computational Linguistics: ACL 2023, pages 8428–8441
July 9-14, 2023 ©2023 Association for Computational Linguistics

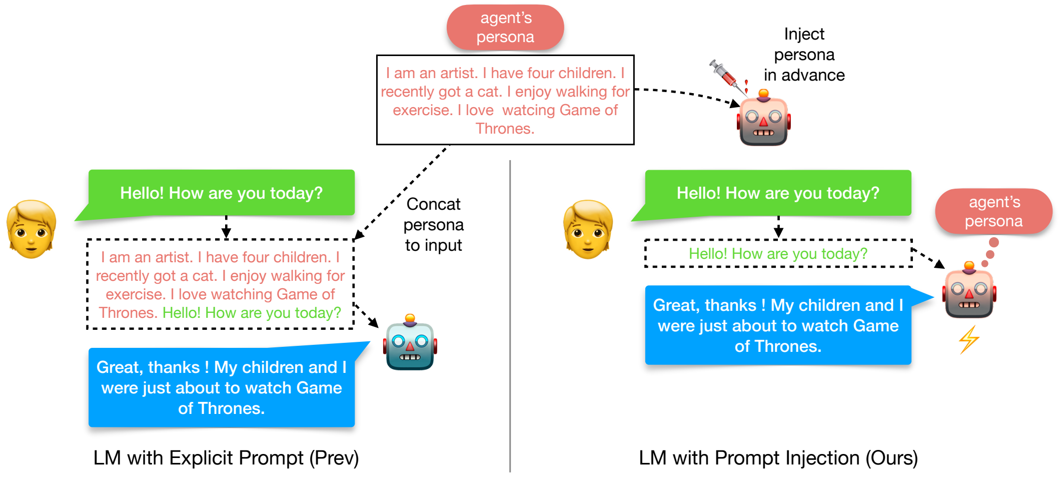

agent's persona
I am an artist. I have four children. I recently got a cat. I enjoy walking for exercise. I love watcing Game of Thrones.
Inject persona in advance
Concat persona to input
Hello! How are you today?
I am an artist. I have four children. I recently got a cat. I enjoy walking for exercise. I love watching Game of Thrones. Hello! How are you today?
Great, thanks ! My children and I were just about to watch Game of Thrones.
LM with Explicit Prompt (Prev)
Hello! How are you today?
agent's persona
Hello! How are you today?
Great, thanks ! My children and I were just about to watch Game of Thrones.
LM with Prompt Injection (Ours)

Figure 1: Fixed Input Prarameterization (FIP) example on a persona-dependent conversation. The left side presents an inference procedure of a previous approach where the persona (prompt) is concatenated to every input. The right side describes FIP, where the persona is *injected* into the model in advance, so that the model is able to generate responses without constantly referring to the persona description. Thus, FIP approach takes less time to generate responses than the previous method.

2022) may be used but still come with increased overall memory and computations, ultimately leading to a delay in generating responses. This problem becomes critical in situations where the LMs are deployed, and fast inference speed is required.

In this work, we formally define Fixed Input Prarameterization (FIP) problem, where we focus on *injecting* a given fixed prompt into the parameters of an LM to address the two limitations mentioned above. With FIP, LMs can produce prompt-dependent outputs without the computational overhead of appending fixed prompts at inference time (the right side of Figure 1), and it also enables the injection of longer prompts in a wholistic way.

More specifically, we first show that Fixed Input Prarameterization (FIP) is much more efficient (up to 280 times) in terms of total FLOPs compared to previous approaches that may be used for handling long prompts such as Fusion-in-Decoder (Izacard and Grave, 2021) or Linear Transformer (Katharopoulos et al., 2020). Next, we explore different methodologies as baselines for FIP, including the continued pre-training approach on the prompt as well as a novel distillation approach called Pseudo-INput Generation (PING) for successful FIP. We apply these FIP methods to three different tasks with fixed prompts: persona-dependent conversation, semantic parsing, and zero-shot learning with instructions. We compare the methods against LMs with explicit prompts as the upper bound as well as the LM without both the prompt and FIP as the lower bound. Experimental results show meaningful improvements with respect to the lower bound, but also exhibit a non-trivial gap with the upper bound. Despite the performance and efficiency trade-off, we still believe that FIP is a direction worth exploring considering its computational benefit, especially when inference costs are critical in real-world applications.

In sum, our main contributions are three folds:

- We formally define the Fixed Input Parameterization (FIP) problem and demonstrate its necessity in terms of computation and memory efficiency, in scenarios with long prompts.
- We explore baseline approaches for FIP, showing that performance can approach the upper bound (unconstrained) performance in some cases.
- We show that the *injection* of long prompts (e.g., detailed description of persona) can be achieved through FIP and show its efficiency in comparison with previous methods, being up to 280 times more efficient during inference.

2 Related Work

Prompting Prompting is an emerging paradigm for modeling LMs, especially for few-shot and zero-shot learning (Radford et al., 2019; Brown et al., 2020; Wei et al., 2022; Sanh et al., 2022). With the help of appropriate prompts, one can exploit knowledge learned by a pre-trained LM and manipulate the LM's behavior. However, for the in-context learning scenario, processing prompts that involve many training examples for each inference incurs substantial computational and memory overhead (Liu et al., 2022). Given training data, Liu et al. (2022) replace in-context learning with fine-tuning a small set of parameters for tackling the above issue. We tackle the same issue but assume a stricter scenario where there is no training data for the given prompt.

Efficient Transformers One can consider using efficient Transformer-based (Vaswani et al., 2017) architectures for handling long input sequences (Tay et al., 2022). The main challenge of using a vanilla Transformer architecture is the quadratic cost in time and memory with regard to the input sequence length due to the self-attention operation. There has been a surge of recent works addressing this problem (Dai et al., 2019; Beltagy et al., 2020; Katharopoulos et al., 2020; Zhu et al., 2021; Guo et al., 2021). They are primarily dedicated to improving either the efficiency of the self-attention mechanism or the general efficiency of the Transformer architecture through sparse models. Also, there has been an attempt to distill a unique prompt to handle long inputs (Askell et al., 2021). Our Fixed Input Prarameterization (FIP) approach tackles the efficiency problem of performing prompt-dependent tasks by keeping the input sequences short (without prompts), bounding the time and memory complexity to a constant invariant of the length of the prompt. In contrast to (Askell et al., 2021), Our work focuses on formally framing the problem into a more general and realistic setting since we aim to inject new prompts with no corresponding training data instead of only one prompt with corresponding training data.

Persona-dependent Conversation Endowing a chabot with a persona (Zhang et al., 2018; Xu et al., 2022) is challenging, but it enables the chatbot to deliver more personal, specific, consistent, and engaging conversations (Zhang et al., 2018) and gain user trust (Liu et al., 2020; Song et al., 2019; Qian et al., 2018). To achieve this, previous works have attached a persona to the dialog history at every inference time, so that the model can always be conditioned on the persona. However, when given a long persona description or long conversation history as a persona, this approach brings the critical problem of increased overall memory and computations, resulting in delayed response generation. FIP allows a dialogue agent to generate responses without a persona description as the explicit input once the persona is injected.

Semantic Parsing Semantic parsing is the task of mapping a natural language query into a SQL query executable on a database. Specifically, cross-domain (cross-database) semantic parsing, where models are trained and tested on different domains (databases) (Yu et al., 2018) introduces many generalization challenges (Hazoom et al., 2021). Previous works concatenate the natural language query with the serialized database schema as the input to address the problem (Suhr et al., 2020; Deng et al., 2021; Xie et al., 2022). With FIP, the model is adapted to a new database schema in advance, so that it can map natural language queries to SQL queries on the new database without explicitly referring to the schema during inference.

Zero-shot Learning with Task Instructions Recent works (Sanh et al., 2022; Wei et al., 2022) have addressed zero-shot generalization to new tasks (Brown et al., 2020; Kim et al., 2021) by multi-task prompted training. With multi-task prompted training, the models learn to use task instructions as prompts to generalize to unseen tasks. It is demonstrated that this approach improves generalization ability to novel tasks and offers an effective substitute for unsupervised language model pre-training. Through FIP, the LM can be aware of a novel task instruction before performing the task and thus does not require the instruction, which can be lengthy, to make predictions.

3 Fixed Input Prarameterization

In this section, we formally define Fixed Input Prarameterization (FIP) as a task and describe the benefits of the formulation. Prompt-dependent generation is a task of generating an output sequence $\boldsymbol{y}$ that is a proper response to the input sequence $\boldsymbol{x}$ and coherent to the prompt $\boldsymbol{z}$. Utilizing the prompt during inference, the generated sentence is obtained by $\boldsymbol{y} = f(\boldsymbol{z}, \boldsymbol{x})$ where f denotes an LM such as T5 and GPT-2. Fixed Input Prarameterization (FIP), i.e., parameterization of prompts, allows LMs to perform prompt-dependent generation without using prompts during inference. To achieve this, we need to design a FIP method H to inject a prompt $\boldsymbol{z}$ into an LM f. The process of FIP can be represented as

$$f_{\boldsymbol{z}} = H(\boldsymbol{z}, f) \tag{1}$$

where $f_{\boldsymbol{z}}$ denotes an LM injected with the prompt. Then the prompt-dependent output sequence can be obtained by $\boldsymbol{y} = f_{\boldsymbol{z}}(\boldsymbol{x})$.

FIP can also be applied for long prompts whose length exceeds the LM's input sequence length. Given a long prompt $\boldsymbol{z}$, we decompose it into multiple sub-prompts $\{\boldsymbol{z}_i\}$ each of which fits the LM's input length, i.e., $\boldsymbol{z} = \boldsymbol{z}_{1:n} = [\boldsymbol{z}_1; \boldsymbol{z}_2; ...; \boldsymbol{z}_n]$. Then the FIP process can be executed iteratively, injecting each sub-prompt sequentially while the LM is aware of the previous sub-prompts:

$$f_{\boldsymbol{z}_1} = H(\boldsymbol{z}_1, f) \tag{2}$$

$$f_{\boldsymbol{z}_{1:2}} = H(\boldsymbol{z}_2, f_{\boldsymbol{z}_1}) \tag{3}$$

$$\cdots$$

$$f_{\boldsymbol{z}_{1:n}} = H(\boldsymbol{z}_n, f_{\boldsymbol{z}_{1:n-1}}) \tag{4}$$

The above formulation can be seen as a high-level abstraction of iterative FIP that we aim to approximate. In practice, in order to fully inject $\boldsymbol{z}_{1:n}$, we repeat (2)-(4) multiple times (i.e., multiple epochs).

Why is Fixed Input Prarameterization necessary? FIP brings advantages in terms of efficiency when applied to prompt-dependent tasks. The previous approach of appending prompts to the input sequences has the drawback of the model repeatedly referring to the prompt at each inference time. This becomes critical in scenarios requiring long prompts, as Transformer architecture has quadratic computational and memory costs due to the limitation of the self-attention operation. We propose FIP as a solution to this computation bottleneck. Once a prompt is injected into the LM in advance, the LM no longer needs to refer to the prompt during inference. As a result, the model's input length remains independent of the length of prompts and is able to utilize prompts of any length efficiently. We discuss the efficiency gain of FIP in Section 6.1.

Evaluation Metric for FIP FIP can be evaluated by the evaluation metric of the fixed prompt-dependent task at hand. We also introduce a metric called the FIP score (FIP score) to measure the degree of injection. The metric is agnostic of the target task by comparing the results with that of an LM given actual prompts during inference. Let $X_{w/\ prompt}$ denote the LM's task score with the prompt as an additional input (upper bound) and $X_{w/o\ prompt}$ denote the LM's task score without the prompt (lower bound). We define **FIP score** as the min-max scaling score of X_{FIP}, where X_{FIP} represents the score of the LM on the target task after FIP, *i.e.*, **FIP score** $= \max(0, X_{FIP} - X_{w/o\ prompt})\ /\ (X_{w/\ prompt} - X_{w/o\ prompt})$. We limit using FIP only in situations where $X_{w/\ prompt} > X_{w/o\ prompt}$ because there is no reason to inject a prompt if task performance degrades when using the prompt. Even if the range of individual task scores may vary from task to task, FIP score represents the overall injection effectiveness of the FIP methods, agnostic of the individual task score range.

4 Methods for Fixed Input Prarameterization

In this section, we explore methods of Fixed Input Prarameterization (FIP) that can address prompt-dependent tasks without accessing the prompt during inference. To achieve this, the model should be trained to store the prompt in its parameters. This can be seen as *parameterizing* the prompt into the model instead of feeding the prompt explicitly to the model. This is challenging as the prompt is unseen to the model and has no corresponding training data. In Section 4.1, a baseline method by continued pre-training is introduced, followed by a method for improving the baseline with curriculum learning. Section 4.2 presents a novel distillation-based method called Pseudo-INput Generation (PING) that learns to generate pseudo-inputs to inject novel prompts.

4.1 Continued Pre-training

We establish the Continued Pre-training method as a straightforward baseline for FIP. This method injects prompts into the parameters of an LM by continuing with the pre-training objective of the LM on the target prompt. The pre-training objective is a straightforward option as it works in an unsupervised manner. In our experiments, we leverage the pre-trained T5 model (Raffel et al., 2020) and thus use the masked language modeling objective which is the pre-training objective of T5. Following Raffel et al. (2020), we randomly replace 15% of a given prompt with special mask tokens; then, the model is trained to predict the sequence of masked tokens. In this process, the model learns about the prompt the same way the model learns knowledge during the pre-training stage.

Curriculum learning We further investigate the baseline method by leveraging *curriculum learning* (Bengio et al., 2009; Campos, 2021) during

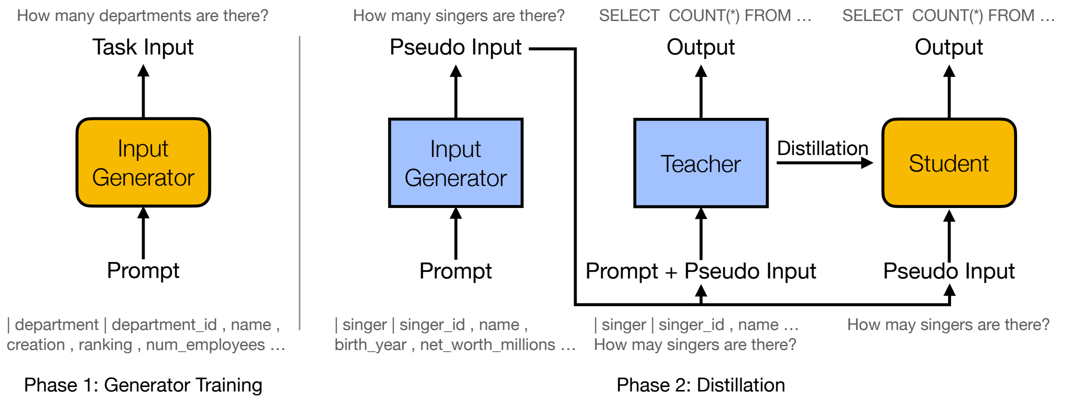

How many departments are there?
Task Input
Input Generator
Prompt
| department | department_id , name , creation , ranking , num_employees …
Phase 1: Generator Training
How many singers are there?
Pseudo Input
Input Generator
Prompt
| singer | singer_id , name , birth_year , net_worth_millions …
SELECT COUNT(*) FROM …
Output
Teacher
Prompt + Pseudo Input
| singer | singer_id , name … How may singers are there?
Distillation
SELECT COUNT(*) FROM …
Output
Student
Pseudo Input
How may singers are there?
Phase 2: Distillation

Figure 2: Illustration of the Pseudo-INput Generation (PING). During Phase 1, an input generator is trained with the task-specific training data. The inputs are prompts of a task, and the outputs are task inputs corresponding to the prompt. Input and output examples applied to semantic parsing are shown. During Phase 2, the input generator generates pseudo-inputs from the given target prompt, which are used to distill knowledge from the teacher to the student. Blue square boxes indicate frozen parameters; yellow rounded boxes indicate unfrozen parameters.

continued pre-training. We set the mask ratio as the difficulty criteria (Wettig et al., 2022) and gradually increase the ratio throughout the Continued Pre-training. As the mask ratio increases, the model should predict more masked tokens given less context. With curriculum learning, we expect the LM to gradually better adapt to the prompt, improving its prompt-dependent task performance. Throughout the experiments, we increase the mask ratio linearly from 15% to 30%, 50%, and 70% and report the best score.

4.2 Pseudo-INput Generation (PING)

The purpose of FIP is to inject a prompt into the parameters of an LM which can also be done indirectly through distillation. In this subsection, we propose a novel distillation-based method called Pseudo-INput Generation (PING) that distills a novel prompt into a student LM that does not have access to the prompt through a teacher LM that does have access to the prompt. In order for distillation, pseudo-inputs are needed since we assume a scenario where the prompt to be injected has never been seen during training and does not have separate training data. An overview of PING is illustrated in Figure 2. As shown in the figure, during Phase 1, an input generator is trained with the task-specific training data. When given a prompt of the task as the input, the generator is expected to generate the task inputs that correspond to the prompt. During Phase 2, the input generator is frozen and is used to generate pseudo-inputs from the unseen prompt, which are then given to the teacher together with the prompt, while only the pseudo-inputs are given to the student. This way, the student learns to follow the teacher and is able to learn about the prompt indirectly.

5 Experimental Setup

In this section, we explain the experimental setups in detail. Experiments are performed with the T5-base (Raffel et al., 2020) (220M parameters) model unless noted otherwise.

5.1 Prompt-dependent tasks

In order to evaluate the effectiveness of Fixed Input Prarameterization (FIP) methods, we select three prompt-dependent tasks—persona-dependent conversation, semantic parsing, and zero-shot learning with task instructions; all these tasks require fixed prompts during inference. Fixed prompts come in the form of a persona in persona-dependent conversation, database schema in semantic parsing, and task instruction in zero-shot learning with task instructions. As described in the introduction and Section 3, when FIP is applied for these tasks, there would be apparent benefits in real-world scenarios. With these tasks, not only the performance of the baseline FIP methods is evaluated, but also the significance of FIP is emphasized by comparison with the (unconstrained) previous approaches that concatenate prompts to the input.

5.2 Datasets

Following datasets of prompt-dependent tasks mentioned in Section 5.1 are utilized to evaluate Fixed Input Prarameterization (FIP).

PERSONA-CHAT / MSC PERSONA-CHAT (Zhang et al., 2018) is a crowd-sourced dataset intended for training agents to perform engaging and personal chit-chat by comprising the dialogues to be grounded on specific personas. For each dialogue, two speakers have a 6-8 turn conversation conditioned on a given persona. Based on PERSONA-CHAT, Multi Session Chat (MSC) (Xu et al., 2022) is a dialogue dataset collected to be comprised of long-term conversations each consisting of 5 continuing, but distinct chat sessions. In this work, we consider both the persona and dialogue history of the first two sessions as a prompt in MSC to incorporate long-term conversational context. Performance on both tasks are measured via perplexity (PPL). We randomly select 100 dialogues from the validation sets respectively as the persona-dependent conversation benchmark for testing our method. The persona descriptions are 60 tokens long on average in PERSONA-CHAT and the combined prompts average 811 tokens in MSC.

Spider Spider (Yu et al., 2018) is a large cross-domain semantic parsing and text-to-SQL dataset for developing natural language interfaces to cross-domain databases. Models must generalize to new database schemas as well as new queries to perform well on it. Evaluation metrics include Exact Matching (EM) and Execution Accuracy (EA). We utilize the development set containing 20 databases with about 50 questions per database as a semantic parsing benchmark for FIP. The database schemas range in length from 55 to 430 token lengths.

WSC / RTE / COPA For the task of zero-shot task generalization, Sanh et al. (2022) have trained the LM on a diverse set of tasks and evaluated on a held-out group of tasks to evaluate generalization performance. We choose coreference resolution, natural language inference, and sentence completion tasks, three out of their four held-out tasks, and test FIP on WSC, RTE, and COPA datasets (Wang et al., 2019). We utilize task instructions (prompts) provided from Sanh et al. (2022) and report average task scores of using them. The task instructions are comprised of about 20-30 tokens.

5.3 Implementation Details

For the Continued Pre-training method (Section 4.1), we use the Adam optimizer (Kingma and Ba, 2015) with a constant learning rate 1e-4 and batch size 8. We perform 5-20 steps of injection. For PING (Section 4.2), input generators are trained on each tasks for 1-2 epochs. We use KL-divergence for distilling the last layer's output of the decoder and perform 10-100 steps of injection. For T5-base, we use a single 16GB T4 GPU and for the larger models we use 4 32GB V100 GPUs.

In order for injection and comparison with upper-bound (W/ PROMPT) and lower-bound (W/O PROMPT) performance, we first need two different versions of the LM adapted to the given task. For the task of persona-dependent conversation and semantic parsing, W/ PROMPT model is fine-tuned together with prompts since prompts are explicitly used during inference, while W/O PROMPT model is fine-tuned on the task without being given the prompt. We perform FIP on the W/O PROMPT model since we assume having no access to prompts during inference.

For the zero-shot learning, we modify the prompts developed by Sanh et al. (2022) in the form of a fixed prompt. We replace the placeholders on their prompts with fixed words, then append the actual content to the prompt in a key-value format. For example, if the original is `If {Premise} is true, is it also true that {Hypothesis}?`, then the converted prompt is `If "Premise" is true, is it also true that "Hypothesis"? Premise:{Premise} Hypothesis:{Hypothesis}`. This ensures that the prefix is fixed, which can be injected with FIP. We use the T0-3B LM checkpoint for the zero-shot generalization.

6 Experimental Results

In this section, we first explore the inference efficiency of models performing prompt-dependent tasks and show that Fixed Input Prarameterization (FIP) leads to a meaningful gain in computational efficiency. Then the baseline and proposed methods are tested and compared on datasets discussed in Section 5.2. The results indicate that the Pseudo-INput Generation (PING) method achieves the best performance among FIP methods, sometimes even outperforming the upper bound, which uses explicit prompts during inference. In Section 6.3, we provide a concrete instance of injecting a real persona description into a conversational model, demonstrating the feasibility of long prompt injection.

6.1 Inference Efficiency

The comparison of inference efficiency of a model with FIP, a baseline model that naively concate-

Model	Prompt Length	FLOPs (T)	Latency (s)
T5 W/ FIP	*	0.7	0.58
T5	512	7.2 (×10.3)	1.09 (×1.9)
	512 × 2	14.6 (×21.0)	2.38 (×4.1)
	512 × 4	OOM	-
T5 + FID	512	7.2 (×10.3)	1.09 (×1.9)
	512 × 2	14.0 (×20.2)	1.54 (×2.6)
	512 × 4	27.6 (×39.8)	2.87 (×4.9)
	512 × 8	54.9 (×79.2)	5.87 (×10.0)
	512 × 28	OOM *(×280)*	-
LINEAR-TRANSFORMER	512	9.5 (×13.8)	1.58 (×2.7)
	512 × 2	16.1 (×23.2)	2.62 (×4.5)
	512 × 4	29.2 (×42.2)	4.74 (×8.1)
	512 × 8	55.6 (×80.1)	9.11 (×15.6)
	512 × 28	OOM *(×280)*	-

Table 1: Inference efficiency of different models that can be used for performing prompt-dependent inference. We depict how many times FIP is efficient in comparison with the other approaches inside the parenthesis. When there is out-of-memory (OOM) using the 16GB T4 GPU, we estimate the FLOPs in *italics* assuming a linear correlation between prompt length and FLOPs.

nates the prompt to the input, Fusion-in-Decoder (FiD) (Izacard and Grave, 2021), and Linear Transformer (Katharopoulos et al., 2020) are shown in Table 1. We consider FiD as one of the options for processing long inputs because it processes long input sequences by encoding chunks of input sequences separately, reducing the quadratic complexity to linear. Linear Transformer also reduces the complexity to linear by linearizing the attention mechanism. We measure FLOPs and forward propagation latency via DeepSpeed Flops profiler [2] using a single 16GB T4 GPU.

As shown in Table 1, T5 W/ FIP is much more efficient than other models, especially as we assume a longer prompt length. This is because the efficiency of FIP remains the same independent of the prompt length while the costs of others increase linearly. Specifically, when the prompt length is 8 times the model's max input sequence length, one can achieve 80× computational efficiency in terms of FLOPs by applying FIP. Furthermore, in a scenario where the prompt length is 28× the model's max input sequence length (shown in Section 6.3 when trying to utilize a long persona that is over 13,000 token length long), previous approaches show an out-of-memory (OOM) issue using the 16GB T4 GPU, which means it is impossible to utilize such long prompts. FIP is estimated to be *280*× more efficient in terms of total FLOPs if the GPU RAM were hypothetically big enough.

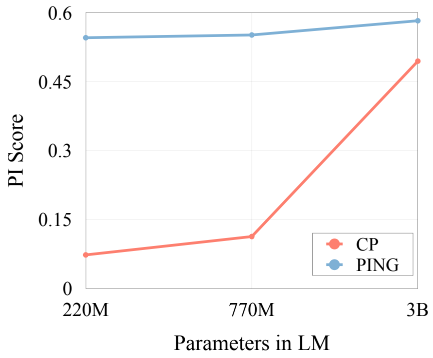


Figure 3: FIP scores in PERSONA-CHAT as we scale the sizes of the LM. There is a consistent trend of improved injection performance across FIP methods as the LM scales.

6.2 Task Performance

We report the task performance obtained by applying different FIP methods on three prompt-dependent tasks in Table 2. FIP scores are also obtained as introduced in Section 3. For all of W/ FIP methods that applied Fixed Input Prarameterization, we observe an overall increase in performance compared to W/O PROMPT, indicating successful injection of prompts into the parameters of the model through FIP methods. The standard deviations of perplexity with 5 random seeds are lower than 0.01 and 0.1 for PERSONA-CHAT and MSC, respectively, which demonstrates the statistical significance of the results. Furthermore, we find that FIP performance improves steadily with model size in PERSONA-CHAT, demonstrating that larger models benefit more from FIP as shown in Figure 3 in terms of FIP score. The task scores are reported in Appendix A.

As shown in Table 2, while CP (Continued Pre-training in Section 4.1) gives modest performance improvement over W/O PROMPT, the results of CP W/ CURR show that leveraging curriculum learning during continued pre-training is effective in some cases. CP W/ CURR performs better compared to CP in PERSONA-CHAT, MSC, Spider, and RTE; it even outperforms W/ PROMPT in RTE. On the other hand, PING significantly improves performance from CP in PERSONA-CHAT, MSC, Spider, and WSC, performing almost on par with W/ PROMPT in WSC. This sheds light on the possibility that FIP may be able to reach the upper bound performance. However, the results show at the same time that there is still a gap between the performance of FIP methods and the upper bound W/ PROMPT that

[2] https://www.deepspeed.ai/tutorials/flops-profiler/

	Dialogue				Semantic Parsing			Task Generalization					
	PERSONA-CHAT		**MSC**		**Spider**			**WSC**		**RTE**		**COPA**	
	PPL (↓)	FIP Score	PPL (↓)	FIP Score	EM	EA	FIP Score	ACC	FIP Score	ACC	FIP Score	ACC	FIP Score
W/ PROMPT	**8.40**	-	**16.42**	-	**57.9**	**61.3**	-	**63.6**	-	67.9	-	**67.3**	-
W/O PROMPT													
W/O FIP	10.72	-	23.96	-	14.5	15.1	-	44.0	-	64.2	-	60.0	-
W/ FIP													
CP	10.53	0.081	18.95	0.664	16.9	17.5	0.054	54.5	0.536	67.7	0.946	64.8	**0.658**
CP W/ CURR	10.28	0.191	18.82	0.681	17.7	18.4	0.072	50.8	0.347	**68.2**	**1.08**	64.1	0.562
PING	9.45	**0.549**	18.44	**0.731**	36.6	41.7	**0.507**	63.3	**0.985**	64.5	0.081	62.0	0.274

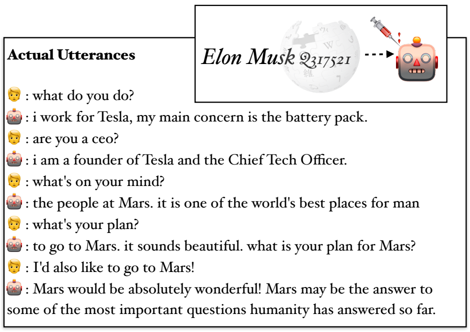
Table 2: Fixed Input Prarameterization performance on three prompt-dependent tasks. W/ PROMPT stands for the upper bound (unconstrained) method, which uses the prompt during inference by appending it to the input. W/O PROMPT depicts the lower bound method of not utilizing the prompts at all. Lastly, we show three W/ FIP methods: CP and CP W/ CURR stand for the Continued Pre-training (baseline) and the Continued Pre-training with curriculum learning, respectively, as explained in Section 4.1; PING depicts our novel proposed method utilizing distillation.

needs to be bridged in future work.

We find that the performance of different methods depends on the complexity of the input sequence structure. We believe that PING achieves a good performance in PERSONA-CHAT, MSC, Spider, and WSC because those datasets have relatively simple input sequences, such as a short utterance and simple query. In datasets with many components or multiple complex sentences (e.g., COPA and RTE), the low quality of generated pseudo-inputs degrades the performance of PING. On the other hand, CP and CP W/ CURR perform better in datasets with complex structure. These findings encourage the community to explore a more integral FIP method that can cover different datasets.

6.3 Long Prompts Injection

To demonstrate the effectiveness of FIP on injection of long prompts into LMs, we show how the method works with a real-world example. We pick a Wikipedia page (Elon Musk), considering it as a long persona description, and inject the entire article (over 13,000 tokens) into an LM trained with PERSONA-CHAT. Here, we use T5-large as a base model and apply PING. Figure 4 shows an actual instance of interactions with the LM that underwent FIP through PING. The responses show the LM successfully reflecting the description of the person on the Wikipedia page without having the description appended to the input. Moreover, the inference of FIP is 280× more computationally efficient in terms of FLOPs than the baseline, as shown in Section 6.1.



Actual Utterances

🧑 : what do you do?
🤖 : i work for Tesla, my main concern is the battery pack.
🧑 : are you a ceo?
🤖 : i am a founder of Tesla and the Chief Tech Officer.
🧑 : what's on your mind?
🤖 : the people at Mars. it is one of the world's best places for man
🧑 : what's your plan?
🤖 : to go to Mars. it sounds beautiful. what is your plan for Mars?
🧑 : I'd also like to go to Mars!
🤖 : Mars would be absolutely wonderful! Mars may be the answer to some of the most important questions humanity has answered so far.

Figure 4: A real-world example of Fixed Input Prarameterization with a long prompt. (Top) The process of injecting a Wikipedia article describing a person (Elon Musk) into a model with FIP. The article is more than 13,000 tokens long. Actual conversation between the persona injected model and a human (cherry-picked).

7 Conclusion

In this paper, we formally define Fixed Input Prarameterization (FIP) problem that focuses on injecting the prompt into the parameters of an LM, as an efficient alternative to attaching fixed prompts to the inputs for prompt-dependent tasks. Through experiments, we show that FIP is much more computationally efficient (up to 280 times) in terms of total FLOPs for handling long prompts compared to the previous alternatives. We further explore baseline methodologies for FIP and find that Pseudo-INput Generation (PING), a distillation-based approach, shows promising results in persona-dependent conversation, semantic parsing, and zero-shot learning with task instructions. Through the explorations, we show that FIP can be a promising direction for conditioning language models efficiently, in scenarios with long and fixed prompts.

Limitations While Fixed Input Prarameterization (FIP) enables performing prompt-dependent tasks efficiently, there are limitations that need to be addressed in future work. In particular, the current FIP methods cause task performance degradation. Moreover, the computational cost needed for the injection of prompts and the storage required to store the parameters of every injected model have not been extensively considered. For example, when considering *previous conversation history* as the prompt to be injected in a long-term conversation setting, fast injection may also be a requirement for real-world application. Updating or adding a relatively small number of parameters (Hu et al., 2021; Wang et al., 2021) may be a potential avenue for addressing the problems.

Acknowledgements

We would like to thank Hyunji Lee, Sohee Yang, Seonghyeon Ye, and Soyoung Yoon for helpful discussions. This work was partly supported by KT grant (2021, A study on a conversational language model that uses long external text as a prompt, 80%) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00113, Developing a Sustainable Collaborative Multi-modal Lifelong Learning Framework, 20%).

References

Amanda Askell, Yushi Bai, Anna Chen, Dawn Drain, Deep Ganguli, T. J. Henighan, Andy Jones, Nicholas Joseph, Benjamin Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, John Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Christopher Olah, and Jared Kaplan. 2021. A general language assistant as a laboratory for alignment. *ArXiv*, abs/2112.00861.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *ArXiv*, abs/2004.05150.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Daniel Fernando Campos. 2021. Curriculum learning for language modeling. *ArXiv*, abs/2108.02170.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-sql. In *NAACL*.

Mandy Guo, Joshua Ainslie, David C. Uthus, Santiago Ontañón, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2021. Longt5: Efficient text-to-text transformer for long sequences. *ArXiv*, abs/2112.07916.

Seungju Han, Beomsu Kim, Jin Yong Yoo, Seokjun Seo, Sangbum Kim, Enkhbayar Erdenee, and Buru Chang. 2022. Meet your favorite character: Open-domain chatbot mimicking fictional characters with only a few utterances. In *NAACL*.

Moshe Hazoom, Vibhor Malik, and Ben Bogin. 2021. Text-to-sql in the wild: A naturally-occurring dataset based on stack exchange data. *ArXiv*, abs/2106.05006.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Franccois Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*.

Boseop Kim, Hyoungseok Kim, Sang-Woo Lee, Gichang Lee, Donghyun Kwak, Dong Hyeon Jeon, Sunghyun Park, Sung ju Kim, Seonhoon Kim, Dong Hyung Seo, Heungsub Lee, Minyoung Jeong, Sungjae Lee, Minsub Kim, SukHyun Ko, Seokhun Kim, Taeyong Park, Jinuk Kim, Soyoung Kang, Na-Hyeon Ryu, Kang Min Yoo, Minsuk Chang, Soobin Suh, Sookyo In, Jinseong Park, Kyungduk Kim, Hiun Kim, Jisu Jeong, Yong Goo Yeo, Dong hyun Ham, Do-Hyoung Park, Min Young Lee, Jaewoo Kang, Inho Kang, Jung-Woo Ha, Woo Chul Park, and Nako Sung. 2021. What changes can large-scale language models bring? intensive study on hyperclova: Billions-scale korean generative pretrained transformers. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *ArXiv*, abs/2205.05638.

Qian Liu, Yihong Chen, B. Chen, Jian-Guang Lou, Zixuan Chen, Bin Zhou, and Dongmei Zhang. 2020. You impress me: Dialogue generation via mutual persona perception. In *ACL*.

Qiao Qian, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Assigning personality/profile to a chatting machine for coherent conversation generation. In *IJCAI*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric Michael Smith, Y.-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *EACL*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang A. Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M SAIFUL BARI, Canwen Xu, Urmish Thakker, Shanya Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Stella Rose Biderman, Leo Gao, T. G. Owe Bers, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *ICLR*.

Haoyu Song, Weinan Zhang, Yiming Cui, Dong Wang, and Ting Liu. 2019. Exploiting persona information for diverse generation of conversational responses. In *IJCAI*.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In *ACL*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys (CSUR)*.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam M. Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, Yaguang Li, Hongrae Lee, Huaixiu Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, I. A. Krivokon, Willard James Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Hartz Søraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravindran Rajakumar, Alena Butryna, Matthew Lamm, V. O. Kuzmina, Joseph Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. Lamda: Language models for dialog applications. *ArXiv*, abs/2201.08239.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of ACL*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *ICLR*.

Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. 2022. Should you mask 15% in masked language modeling? *ArXiv*, abs/2202.08005.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *ArXiv*, abs/1901.08149.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *ArXiv*, abs/2201.05966.

Jing Xu, Arthur D. Szlam, and Jason Weston. 2022. Beyond goldfish memory: Long-term open-domain conversation. In *ACL*.

Tao Yu, Rui Zhang, Kai-Chou Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Z Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur D. Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*.

Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. 2021. Long-short transformer: Efficient transformers for language and vision. In *NeurIPS*.

A Appendix

Table A1: Prompt Injection performance in PERSONA-CHAT as model size increases. There is a consistent trend of improved injection performance across PI methods as the model scales, and CP tends to increase more rapidly.

	PERSONA-CHAT					
	220M		**770M**		**3B**	
	PPL (↓)	PI Score	PPL (↓)	PI Score	PPL (↓)	PI Score
W/ PROMPT	8.40	-	7.42	-	6.66	-
W/O PROMPT						
W/O PI	10.72	-	9.54	-	8.82	-
W/ PI						
CP	10.53	0.081	9.3	0.113	7.75	**0.495**
PING	9.45	0.549	8.37	0.552	7.56	**0.583**

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
After Section 7 Conclusion

☒ A2. Did you discuss any potential risks of your work?
Our paper aims to improve the model's efficiency, without changing the model's output much.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1 Introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 5.2 Datasets and Section 5.3 Implementation Details

☑ B1. Did you cite the creators of artifacts you used?
Section 5.2 Datasets and Section 5.3 Implementation Details

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
The license of the code used in the paper will be discussed on the GitHub repository (to be released).

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
All data and models used in the paper are available for research purposes

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
All data used in the paper do not have any offensive content or identifiers.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 5.2 Datasets

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 5.2 Datasets

C ☑ Did you run computational experiments?

Section 5.3 Implementation Details

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 5.3 Implementation Details

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 5.3 Implementation Details

☒ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Our experiment results are from the average of multiple examples, with single runs.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 5.3 Implementation Details and Section 6.1 Inference Efficiency

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.