PILLOW: Enhancing Efficient Instruction Fine-tuning via Prompt Matching

Zhenting Qi[*♣,♡] **Xiaoyu Tan**[* ♡†] **Shaojie Shi**[♢] **Chao Qu**[♡] **Yinghui Xu**[♠] **Yuan Qi**[♠]
[♣] Zhejiang University [♡] INF Technology (Shanghai) Co., Ltd.
[♠] AI^3 Institute, Fudan University [♢] Shanghai University of Engineering Science
`zhenting.19@intl.zju.edu.cn, yulin.txy@inftech.ai`

Abstract

Instruction fine-tuning has conventionally been employed to adapt Large Language Models (LLMs) to a variety of tasks. Nonetheless, this technique often necessitates substantial computational resources, making it impractical for deployment by individuals or small-scale entities. Recently, Low-Rank Adaptation (LoRA) has become a promising alternative, offering high capabilities on par with full tuning with reduced resource overhead. However, attaining satisfactory performance through the fine-tuning of LoRA is a non-trivial challenge. In this paper, we propose PILLOW, which aims to improve LoRA's performance by a discrimination-based prompting method, leveraging LLMs' In-Context Learning ability. PILLOW incorporates a matching network that selects prompts from a user-defined prompt pool, concatenates the selected prompts with the user instruction as input, and performs inference using the LoRA-fine-tuned LLMs. Trained with Reinforcement Learning, PILLOW exhibits commensurate performance on various evaluation metrics compared with typical instruction fine-tuning methods, utilizing only consumer-grade GPU resources and exhibiting a large reduction in computational costs.

1 Introduction

In recent years, the impressive achievements of large language models (LLMs) have become increasingly evident. Online LLM products, e.g., Claude (Bai et al., 2022) and ChatGPT (OpenAI, 2023), have been widely recognized by the industry for their strong capabilities and are utilized in a myriad of industrial tasks (Liu et al., 2023b; Zhao et al., 2023). The achievement of such success highly hinges on the usage of supervised fine-tuning (SFT) (Mishra et al., 2021; Sanh et al., 2021; Wei et al., 2021).

Nevertheless, as these models become larger, so does the intricacy of SFT. These fine-tuning procedures typically demand a large scale of computational resources to accommodate training all the model parameters. Consequently, this can be economically challenging for independent developers and smaller entities, who often have their own specific needs and budget limitations. In addition, data privacy standards prevent them from using third-party APIs, adding another layer of constraint for them to utilize the LLMs. Thus, while LLMs have been evolutionary in various applications, their scalability and cost-effectiveness still pose challenges in deployment.

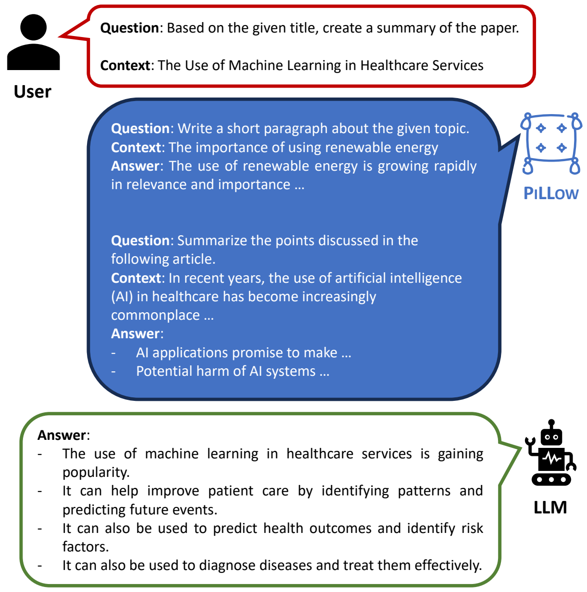
To solve the aforementioned problem, some have applied parameter-efficient finetuning which updates a relatively small portion of parameters, making fine-tuning more manageable under resource limitation (Hu et al., 2021; Dettmers et al., 2023; Chavan et al., 2023; Li and Liang, 2021; Lester et al., 2021; Liu et al., 2021b). Hu et al. (2021) introduce LoRA to train dense layers by optimizing their rank decomposition matrices, thus considerably minimizing the number of trainable parameters and not adding to inference latency. However, LLM's performance may be limited as LoRA only trains a subset of the model parameters. Furthermore, LoRA may not achieve good performance on some tasks with unique characteristics because it can hardly adapt to diverse datasets due to its static fine-tuning strategy (Chavan et al., 2023).

Therefore, *can we attain a similar performance level to SFT by merely employing a comparable amount of resources as used by LoRA? Can we realize it using LoRA-fine-tuned LLMs' reserved in-context learning (ICL) capacity?* Our approach, named **PILLOW**, trains a **P**rompt matching net using Re**I**nforcement **L**earning to improve fine-tuning LLMs under **LOW**-resource settings. We train an RL agent to select exemplars from a comprehensive "prompt set" which can be defined by

---

[*] Equal Contributions.
[†] Corresponding author.

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 471–482
December 6-10, 2023 ©2023 Association for Computational Linguistics



Question: Based on the given title, create a summary of the paper.

Context: The Use of Machine Learning in Healthcare Services

Question: Write a short paragraph about the given topic.
Context: The importance of using renewable energy
Answer: The use of renewable energy is growing rapidly in relevance and importance ...

Question: Summarize the points discussed in the following article.
Context: In recent years, the use of artificial intelligence (AI) in healthcare has become increasingly commonplace ...
Answer:
- AI applications promise to make ...
- Potential harm of AI systems ...

Answer:
- The use of machine learning in healthcare services is gaining popularity.
- It can help improve patient care by identifying patterns and predicting future events.
- It can also be used to predict health outcomes and identify risk factors.
- It can also be used to diagnose diseases and treat them effectively.

Figure 1: A demonstration of 2-shot PILLOW.

users or split from the training data, and subsequently merge these examples to form a prompt which is then added at the beginning of the input text that is fed into LoRA model. Figure 1 shows an example where a 2-shot prompt is given by PILLOW agent based on the user input. Our approach becomes particularly beneficial in commercial applications where user input styles vastly deviate from those found in the pre-training corpus but are confined to limited variations. In these circumstances, the RL agent can efficiently learn to choose examples that best resonate with the specific query posed to the LLMs and therefore achieve comparable performance with direct SFT training. Our contributions can be summarized as follows:

- We propose a new framework PILLOW to achieve SFT-comparable performance by utilizing LoRA and ICL with limited resources.
- We make PILLOW easy to use and widely applicable because a pre-trained LLM can be shared and used to build many LoRA adapters and matching networks for different tasks.
- Experiments show that our proposed PILLOW is effective in instruction-finetuning datasets that contain diverse tasks in various domains.

2 Method

We present PILLOW, a novel RL-based prompting matching framework, designed to enhance the performance of the fine-tuned LoRA model leveraging in-context learning. To provide a better understanding of our work, we first give a brief overview of the necessary background information. Following that, we depict our task by framing it within the Reinforcement Learning settings and subsequently detail the components of our framework.

2.1 Preliminary

2.1.1 Supervised Fine-tuning and LoRA

The technique of supervised fine-tuning (SFT) is employed for enhancing the capabilities of pre-training language models by subjecting them to additional training on labeled datasets for the purpose of task-specific or domain-specific adaptation. This process involves the recalibration of model parameters by minimizing a defined loss function, thereby aligning its predictive capacity with the anticipated outputs. SFT takes advantage of accumulated prior knowledge to augment the efficiency in subsequent tasks, such as text categorization, named entity recognition, sentiment analysis, and etc. However, comprehensive fine-tuning LLMs through SFT becomes less practical as the size of LLMs increases, especially for individual developers and studios. A promising solution to this predicament is LoRA (Hu et al., 2021), which proposes the training of rank decomposition matrices for each layer in the model architecture. This method significantly curtails the number of trainable parameters for subsequent tasks without imposing inference latency. However, for general instruction following tasks that experience large distributional shifts between different tasks, LoRA cannot achieve comparable performance with SFT due to the relatively low capacity.

2.1.2 In-Context Learning

In-context learning (ICL) is a method that enhances LLMs by supplying specific contexts using a handful of examples, or prompts, to steer the model's behavior and produce the required results (Dong et al., 2022). Efficient prompts can direct the model's responses by offering pertinent information via sentences, keywords, instructions, or examples. ICL enables users to tailor the model for specialized tasks or fields, leveraging a relatively smaller dataset composed of examples and intended outputs. Nevertheless, as Zhao et al. (2021) emphasizes, ICL can be highly sensitive to the setup of prompts, encompassing prompt templates, in-context examples, and the order of the examples.

(For more related work in prompting, please refer to Section A)

2.2 PiLLow

2.2.1 Motivation

Our objective is to construct an interpretable and resource-efficient automated prompting framework. Despite the superior performance they exhibit, continuous prompting methods do not provide human interpretable results and mandate the utilization of costly gradient information (Liu et al., 2023a). Recent advancements in the discrete prompting field have brought forward generation-based (Deng et al., 2022) and editing-based (Shin et al., 2020; Zhang et al., 2022) methods which have demonstrated their efficacy across various task domains. However, these approaches encounter significant challenges in terms of their computational intensity during the training phase, which is the main issue PiLLow aims to address.

On the premise that discrimination is much less computationally intensive than generation or editing, we propose to build a discrimination-based prompting framework. In essence, PiLLow aims to identify the optimal prompt that aligns with the user's input, as opposed to generating or editing one. To begin with, the process of training a matching neural network exhibits greater resource efficiency by eschewing the necessity for direct operation on texts. Secondly, many downstream tasks exhibit a restricted diversity of types of questions and answers, leading to a scenario where a multitude of user inputs can be guided with several analogous examples. For organizations operating under computational resource constraints, the establishment of a compact suite of "standard" question-answer pairs is sufficient to prompt LoRA fine-tuned LLM to accomplish a designated task via the ICL capacity reserved by LoRA.

2.2.2 RL-based Prompt Matching

Prompt Matching Problem Our goal is to select a series of optimal prompts $V = \{v_1, ..., v_m\}$ from a user-defined prompt set $P = \{p_i\}_{i=0}^{n-1}$, where m is the number of shots and n is the size of the prompt set, to maximize some performance measure R. The R should be defined domain-specifically and will be discussed in Section 3.2. Each prompt p_i to be selected is a triple of $(\mathrm{question}, \mathrm{context}, \mathrm{answer})$, where $\mathrm{question}$ represents the user instruction, $\mathrm{context}$ denotes the extra information provided by the users (optional), and answer is the expected output. We formulate the task of *prompt matching* as follows:

$$\max_{V \subset P} R(y_{\mathrm{LM}} \sim M_{\mathrm{LM}}(\cdot|v_0, v_1, ..., v_m, x)), \quad (1)$$

where v_0 denotes a pre-defined initial system prompt and the response y_{LM} is sampled by the LoRA fine-tuned LLM $M_{\mathrm{LM}}(\cdot|v_0, v_1, ..., v_m, x)$ given the condition of user instruction x and the prompts $\{v_i\}_{i=0}^{m}$ added to its front.

RL Formulation The prompt matching task can be formulated as a Markov Decision Process (MDP) as follows: given an initial state $s_0 = (v_0, x)$, at each time step t, an RL agent π_θ with parameter θ selects a prompt index $k = a_t$ from the action space A according to policy $\pi_\theta(a_t|s_{<t}, x)$. We define the transition function as: $\mathcal{T} : S \times A \rightarrow S$ to be the state before and after selecting a new prompt $(v_0, ..., v_t) \times a_t \rightarrow (v_0, ..., v_t, v_{t+1})$, where $v_{t+1} = p_k$, and the process stops when $t = m$. Then, we can optimize the policy π_θ by maximize the cumulative rewards:

$$\max_\theta \mathbb{E}[\sum_{t=0}^{m} \gamma^t r(y_{\mathrm{LM},t})], \mathrm{s.t.}, y_{\mathrm{LM},t} \sim M_{\mathrm{LM}}(\cdot|\hat{s}_t, x), \quad (2)$$

where $\hat{s}_t \sim \prod_{i=0}^{t} \pi_\theta(a_i|s_{<i}, x)$, r is the reward measurement, and γ is the discount factor. We discuss the necessity of using RL in our task in Section 3.6.1.

Action Space The action space is simply the set of the indices of all candidate prompts. We preprocess the prompt set P by encoding its QA pairs into an embedding set $P' = \{f_i\}_{i=0}^{n-1}$ so that each prompt index k corresponds to one embedding vector f_k. Suppose we have n user-defined candidate prompts, then at each stage, the agent chooses an integer from 0 to $n-1$, and the discrete action space size will be n.

State Representation Before matching, the user instruction input is encoded as an embedding g, and the embeddings of the selected prompts are aggregated and averaged to a new representation h. Then, we can get the state representation as $l = \mathrm{concat}(g, h)$ by concatenating two embeddings. To track state changes in the RL environment, we use a list of indices of chosen prompts instead of prompt context to further reduce the time and space complexity. The initial state will be a list containing -1 index only, i.e. $s_0 = [-1]$. As the episode proceeds, the list will be enlarged with new prompt indices appended, i.e. $s_t = \mathrm{append}(s_{t-1}, a_t)$.

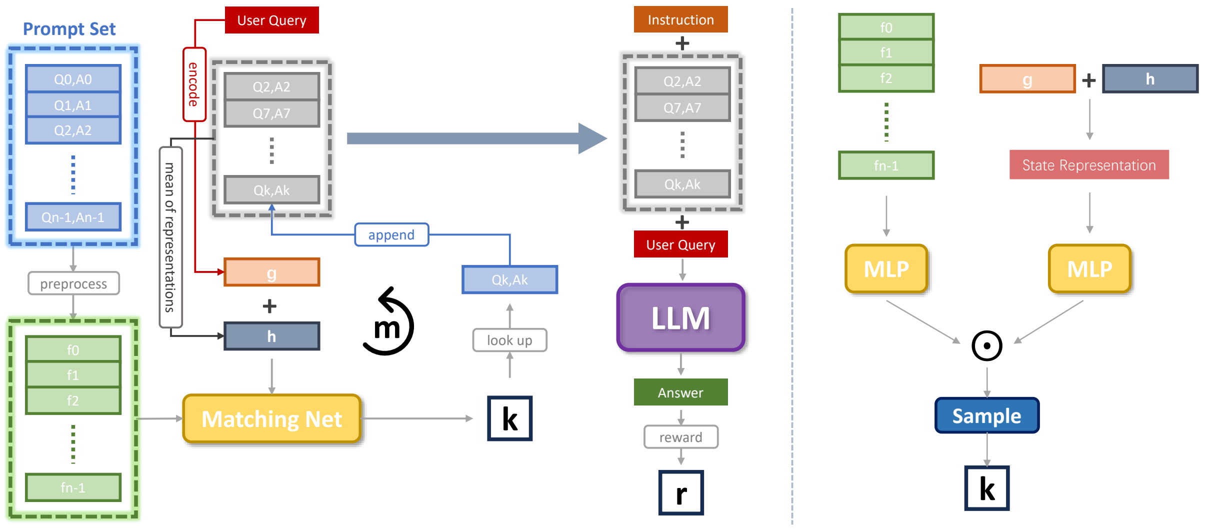


Figure 2: Illustration of PILLOW. The left figure shows how the matching net is trained: At each step (out of m steps), one prompt is selected from the prompt set by the matching net according to the user query and current matched prompts. After prompts are collected, they are passed to the LLM to get the answer, from which we calculate a reward. The right one shows the detailed pipeline of the matching network: The left MLP transforms the prompts into a set of vectors, with which we calculate dot products with the vector transformed by the right MLP from the state representation, and we obtain a probability distribution over the prompts.

Policy Network We build the policy network $\pi_\theta(a_t|s_{<t}, x)$ with a deep text-matching network. The right-hand side of Figure 2 shows the network Z_θ, which consists of two Multi-Layer Perceptrons (MLPs) to match two parts of features. The transformed prompt set $P' = \{f_i\}_{i=0}^{n-1}$ is further encoded by an MLP: $c_i = Z_1(f_i)$, where $P'' = \{c_i\}_{i=0}^{n-1}$ is named "keys", and the state representation l is encoded by another MLP: $q = Z_2(l)$, where q is named "query". We compute the similarities of the query and keys, scale by a normalization factor, and obtain a probability distribution X after the softmax layer. Finally, we sample an integer number k as the index of the matched prompt.

Framework Based on the RL settings defined above, we design the entire training procedure as shown on the left-hand side of Figure 2. Given P', g, and h, the RL agent π_θ consistently selects new prompt index k, looks up in the prompt set P, and appends the selected QA pair p_k to previously selected prompts, until the number of prompts reaches m, which is the pre-defined number of shots. Then, all the m selected prompts together with the user input are fed into the LoRA fine-tuned LLM M_{LM}. The response will be scored by the reward function and the reward signal r is used to update the parameters θ through off-the-shelf RL algorithms (Deng et al., 2022; Zhang et al., 2022). During the inference stage, the trained agent follows the same manner as aforementioned to select prompts and compose the LLM input.

3 Experiments

We conduct a comparative evaluation of our proposed framework PILLOW against two typical baseline methods: LoRA and SFT. SFT requires a high quantity of resources with high-quality response, while LoRA operates effectively under constrained resources but the performance is inferior to SFT. Nevertheless, empirical findings from our experimental studies suggest that PILLOW has the capability to yield performance in parity with SFT, even under low-resource constraints.

3.1 Datasets

We use comprehensive instruction fine-tuning datasets that are designed to align the LLMs as helpful human assistants to follow almost all kinds of instructions. The following datasets are chosen because they encompass a variety of text-to-text generation tasks and contain repetitive QA patterns. **Alpaca** (Taori et al., 2023) contains 52,000 instructions and demonstrations which are generated by OpenAI's text-davinci-003 model given new prompts that explicitly outline the requirements, aiming at conducting instruction-tuning to make LLMs follow instructions better. Using Self-Instruct (Wang et al., 2022), the authors built the

Dataset	Model Size	Method	C-Score	PPL	R/w	M-Score
Alpaca	560m	-	2.71	192.24	4.87	0.00/0.16/0.84
		SFT	2.87	106.07	5.30	0.04/0.40/0.56
		LoRA	2.56	149.93	4.89	0.00/0.32/0.68
		PiLLow	2.63 (+0.07)	140.57 (-9.36)	4.68 (-0.21)	0.02/0.21/0.77
	1b1	-	3.01	108.2	5.71	0.00/0.17/0.83
		SFT	3.29	52.02	6.48	0.12/0.43/0.45
		LoRA	3.09	78.81	5.83	0.09/0.21/0.70
		PiLLow	3.21 (+0.12)	67.36 (-11.45)	5.89 (+0.06)	0.14/0.39/0.47
	7b1	-	3.14	161.19	5.88	0.00/0.23/0.77
		SFT	3.84	64.34	6.49	0.31/0.54/0.15
		LoRA	3.27	120.70	5.94	0.17/0.46/0.37
		PiLLow	3.76 (+0.49)	103.65 (-17.05)	6.07 (+0.13)	0.29/0.44/0.27
Dolly	560m	-	2.83	247.45	4.26	0.00/0.18/0.82
		SFT	3.01	218.61	5.01	0.07/0.42/0.51
		LoRA	2.64	221.16	4.34	0.00/0.33/0.67
		PiLLow	2.74 (+0.1)	191.9 (-29.26)	4.70 (+0.36)	0.05/0.39/0.56
	1b1	-	3.13	227.43	4.74	0.00/0.19/0.81
		SFT	3.37	67.93	5.87	0.14/0.51/0.35
		LoRA	3.08	140.40	4.79	0.07/0.32/0.61
		PiLLow	3.31 (+0.23)	112.78 (-27.62)	5.31 (+0.52)	0.11/0.47/0.42
	7b1	-	3.24	244.09	4.86	0.00/0.26/0.73
		SFT	3.89	56.64	5.61	0.39/0.51/0.10
		LoRA	3.33	146.92	4.93	0.21/0.48/0.31
		PiLLow	3.81 (+0.48)	113.09 (-33.83)	5.08 (+0.15)	0.36/0.47/0.17

Table 1: Results on 1-shot PiLLow on Alpaca and Dolly. The score differences that indicate better performance than LoRA are marked with red color, while those showing worse performance are marked with blue color.

data generation pipeline to align pre-trained LMs with instructions generated by themselves.

Dolly (Conover et al., 2023) is a human-annotated dataset of 15,000 instruction-following records, including various categories like brainstorming, classification, closed QA, generation, and summarization. The annotators are given instructions to refrain from using data from any online source except Wikipedia (for specific subsets of instruction categories), and most importantly, they avoid using generative AI in writing instructions or responses.

3.2 Reward Function

Since we test PiLLow on general text-to-text generation tasks, we simply use the weighted sum of textual similarity and semantic similarity as the score function ζ instead of conducting task-specific reward engineering (Deng et al., 2022; Zhang et al., 2022):

$$\zeta(y,\hat{y}) = \lambda \cdot S_{\text{textual}}(y,\hat{y}) + (1-\lambda) \cdot S_{\text{semantic}}(y,\hat{y}), \tag{3}$$

where S_{textual}, S_{semantic} are textual similarity (based on fuzzy matching) and semantic similarity (based on cosine similarity between sentence representations), and y, $\hat{y}$ are LLM's output and expected output, respectively, and λ is a balancing factor. Note that in deployment, people can choose desired reward formulations based on their specific tasks. Finally, the reward r is obtained by scaling the score with a constant α: $r = \alpha \cdot \zeta(y, \hat{y})$.

3.3 Experiment Setup

We use Bloomz-560m, Bloomz-1b1, and Bloomz-7b1 (Muennighoff et al., 2022) as backbone models to show PiLLow's effectiveness on LLMs of different sizes. For both Alpaca and Dolly, we use the entire dataset for LoRA/SFT training. Then, we randomly select 1,200 data items: 100 for the user-defined prompt set, 900 for RL training, and 200 for testing. For model training, we implement the LoRA-/PiLLow-related experiments on one V100 GPU and SFT with one A100 GPU for efficiency.

We conduct the evaluation experiments of PILLOW on one NVIDIA GeForce RTX 3090 GPU.

3.4 Evaluation

Automatic Scores We automatically score the LLM output by reward (r/w), ChatGPT score (C-Score), and perplexity (PPL). We eliminate abnormal values and then take the average to obtain the metric measurement. For C-Score evaluation, we utilize the prompt introduced by Zhou et al. (2023) and reduce the score bias by randomly organizing the instruction and response orders (Wang et al., 2023).

Manual Scores We also evaluate the LLM output with a manual score (M-Score). This process is an absolute analysis which is similar to the method used by Zhou et al. (2023). We invite five human annotators to label each response with three labels: **Excellent**, **Pass**, **Fail**, which have the same criteria as Zhou et al. (2023). For each experiment, we randomly select 50 samples for labeling.

3.5 Results

We present our experiment results on 1-shot PILLOW in Table 1. We report the M-Score by reporting the average rate from human annotators in the order of **Excellent/Pass/Fail**. It can be seen that our method outperforms the LoRA model on both Alpaca and Dolly across most evaluation metrics and achieves performance very close to SFT.

We observe that as the model size increases, the performance gain compared with LoRA tends to become larger. On Alpaca, for example, the 1b1 model trained with PILLOW surpasses LoRA by 0.12 in C-Score and 11.45 in perplexity, and for the 7b1 model such gaps increase to 0.49 and 17.05, respectively. Also, we can see that PILLOW helps large models reach very close performance to SFT. On Dolly, for example, the 1b1 model trained with PILLOW reaches 98.22% of SFT's performance in ChatGPT score and 90.46% in reward, and the 7b1 model reaches 97.94% and 90.55%, respectively. The human evaluation results also demonstrate similar pattern with the C-Score evaluation.

Our observations in M-Score indicate that PILLOW significantly enhances the **Excellent** rate while reducing the **Fail** rate when compared to the LoRA model. This signifies a considerable improvement in quality in comparison to the LoRA model. Note that the 560m model trained with both PILLOW and LoRA does not improve that much and even degrades a little bit compared with the original pre-trained model, and on Alpaca PILLOW even performs worse than LoRA on the reward metric. However, there is no such problem for the 1b1 and 7b1 models. Therefore, we can conclude that our proposed PILLOW is particularly suitable for large-scale LLMs which inherently possess enhanced ICL and emergent capabilities. Importantly, the application of LoRA does not diminish these intrinsic abilities of the LLMs. We refer readers to Appendix D for example LLM inputs and outputs.

3.6 Ablation Study

3.6.1 Why RL?

While it may appear that the task introduced in this paper could be solved by simply matching the prompt most similar to the question during the inference phase, our first ablation study reveals that PILLOW outperform simple matching and LoRA model. This is due to the potential misalignment between the sentence encoder and LLM training data, which means the most semantically matched prompt might not yield the best answer. Additionally, as users may need to switch LLMs for different tasks which may require different objectives for LLMs outputs, PILLOW can optimize the outputs based on reward signals on multiple perspectives. We refer the readers to Appendix B for the experiment results and more details.

3.6.2 Number of Shots

We also investigate the impact of increasing the number of "shots". Results indicate a slight improvement when the number of shots increases. However, using more shots may introduce irrelevant prompts that can disrupt the output of the LLMs. Additionally, a higher number of shots can reduce the PILLOW's training efficiency. Hence, we recommend one prompt in practice. We refer the readers to Appendix C for the experiment results and more details.

4 Conclusion

We train a prompt matching framework PILLOW via Reinforcement Learning to enhance efficient instruction finetuning. PILLOW is evaluated on the most recent instruction finetuning datasets, Alpaca and Dolly, and achieves superior results across all evaluation metrics and model sizes compared with supervised fine-tuning under LoRA. This new area of research combining prompting, matching, and RL can inspire future work on better prompting methods for LLMs under low-resource regimes.

Limitations

PiLLow is implemented based on matching rather than generation. Despite being highly controllable, interpretable, and efficient, such a prompting method may not show superior performance on large but sparse datasets in which most question-answer pairs do not follow similar patterns because the RL agent may not be able to find appropriate prompts. In our future work, we intend to investigate the impact on PiLLow's performance by utilizing a variety of reinforcement learning (RL) algorithms. Furthermore, we are interested in exploring hybrid RL agents with the aim to optimize the number of shots and prompts for each slot.

In terms of the reward design, we only include a semantic similarity and a textual similarity, which may not be enough for giving authentic feedback to the RL agent. We believe that other popular automatic text generation evaluation metrics such as BLEURT (Sellam et al., 2020), BARTScore (Yuan et al., 2021), and GPTScore (Fu et al., 2023) can also be utilized for such purpose.

Ethics Statement

We declare that the current study strictly comply with the ACL Ethics Policy. The datasets (Taori et al., 2023; Conover et al., 2023) used to compare PiLLow with previous methods are publicly available and we did not modify any data in them. For the manual evaluation, we anonymously hire 5 experts. We make scoring each LLM output as a unit task and pay $0.2 for each unit task. On average, one human evaluator can finish 30 unit tasks per hour after short training and practice. We recommend that human evaluators devote a maximum of 2 hours per day to the evaluation work in order to maintain a comfortable pace.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. 2023. One-for-all: Generalized lora for parameter-efficient fine-tuning. *arXiv preprint arXiv:2306.07967*.

Shaotao Chen, Xihe Qiu, Xiaoyu Tan, Zhijun Fang, and Yaochu Jin. 2022. A model-based hybrid soft actor-critic deep reinforcement learning algorithm for optimal ventilator settings. *Information sciences*, 611:47–64.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Antonio Coronato, Muddasar Naeem, Giuseppe De Pietro, and Giovanni Paragliola. 2020. Reinforcement learning for intelligent healthcare applications: A survey. *Artificial Intelligence in Medicine*, 109:101964.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

Ben Hambly, Renyuan Xu, and Huining Yang. 2023. Recent advances in reinforcement learning in finance. *Mathematical Finance*, 33(3):437–503.

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

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. 2023b. Summary of chatgpt/gpt-4 research and perspective towards the future of large language models. *arXiv preprint arXiv:2304.01852*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Xihe Qiu, Xiaoyu Tan, Qiong Li, Shaotao Chen, Yajun Ru, and Yaochu Jin. 2022. A latent batch-constrained deep reinforcement learning approach for precision dosing clinical decision support. *Knowledge-based systems*, 237:107689.

Chao Qu, Xiaoyu Tan, Siqiao Xue, Xiaoming Shi, James Zhang, and Hongyuan Mei. 2023. Bellman meets hawkes: Model-based reinforcement learning via temporal point processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9543–9551.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E Gonzalez. 2022. Tempera: Test-time prompting via reinforcement learning. *arXiv preprint arXiv:2211.11890*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

A Automatic Prompting

Since writing prompts manually is time-consuming and cost-intensive, a number of methods have been proposed to automate the prompting process. In continuous prompting (a.k.a soft prompting) (Liu et al., 2023a), prompting is performed directly in the embedding space of the language models. However, by their continuous nature, such prompts are not human-understandable. *Prefix Tuning* (Li and Liang, 2021) adds a sequence of continuous task-specific prompt embeddings to the front of input texts in each layer of LM while keeping the LM's parameters frozen. Similarly, *Prompt Tuning* (Lester et al., 2021) prepends the input texts with special tokens to form a template and directly tune the token embeddings without updating LM's parameters. Unlike the two methods, *P-Tuning* (Liu et al., 2021b) removes the restriction on adding the prompt embedding to the beginning of the input. They define that the prompt tokens can be inserted anywhere in the input sequence and can only be inserted in the input rather than any other model layer.

Approaches on discrete prompting (a.k.a hard prompting) (Liu et al., 2023a) automatically generate or edit prompts described in a discrete space, i.e. in the form of texts. *AutoPrompt* (Shin et al., 2020) edits textual prompt template in a gradient-guided manner, and find that the best final prompts are usually gibberish and not human-interpretable. *TEMPERA* (Zhang et al., 2022) is also an editing-based method, but it trains the test-time editor with RL framework and edits the initial prompts using commonly-used instructions, few-shot exemplars, and verbalizers. Similarly, *RLPrompt* (Deng et al., 2022) is also built on an RL framework, which generates better prompts word by word with black-box optimization. The authors also find that final optimal prompts are often ungrammatical texts and they are transferrable between different LMs. However, both generation and editing are hard tasks and can be computationally intensive given their large action space and long decision process. Also, the RL-based methods rely on specific reward designs, which only apply to limited tasks like few-shot text classification.

Recent work has also leveraged pre-defined exemplar pools to boost the final performance of prompting LLMs. Rubin et al. (2021) trained a dense retriever that fetches useful training examples as LLM prompts from an exemplar pool during test time. Liu et al. (2021a) suggest retrieving training pool exemplars that are semantically comparable to a test example, and they demonstrate how this can greatly improve performance. Similarly, *TEMPERA* (Zhang et al., 2022) design an attention-based exemplar selector over the embedding space and show that such an exemplar selection process can effectively choose training examples that lead to high performance.

B Why RL?

RL techniques have been widely used in multiple industrial application and achieved significant improvement in numerous domain (Qu et al., 2023; Coronato et al., 2020; Qiu et al., 2022; Chen et al., 2022; Hambly et al., 2023). Some may wonder why RL is even necessary in our settings since it seems that the tasks can be solved by simply matching the prompt with the largest similarity with the question. Here we conduct an ablation study on whether we use RL to solve the tasks. As can be seen in Table 2, with all else being equal, an RL-trained prompt matching network performs better than simple matching (SimMatch) which performs almost the same with pure LoRA. We attribute such a performance gap to the LLM in that the sentence encoder and the LLM may not be trained on the same data, so the prompt chosen by the encoder that best matches the question, i.e. seemed “matched” to humans, may not seem that “matched” to the LLM. Even if the chosen prompt is truly the most semantically similar one to the question, it may not best prompt to assist the LLM in generating an appropriate answer. In addition, users might want to switch to different LLMs to adapt to different downstream tasks, where such "appropriateness" might be defined differently, therefore an RL-based training framework is necessary.

Method	**C-Score**	**PPL**	**R/w**	**M-Score**
LoRA	3.09	78.81	5.83	0.09/0.21/0.70
SimMatch	3.12	79.26	5.81	0.12/0.26/0.62
PILLOW	**3.21**	**67.36**	**5.89**	**0.14/0.39/0.47**

Table 2: Ablation on prompting framework. Experiments are conducted with Bloomz-1b1 on Alpaca test set.

C Number of Prompts

We ablate on the number of shots to study how the number of exemplars affects the performance. As can be seen in Table 3, as we increase the number of shots from 1 to 3, the ChatGPT score and reward increase a little, but in general, the measurements do not change too much. Intuitively, the more exemplars are given, the better the LLM output would be. However, it imposes difficulty to the matching net to ensure the selected exemplars are all helpful for prompting the LLM. It is possible that an irrelevant prompt is newly chosen and corrupts the LLM output. Also, a large number of shots makes the RL training slow. Therefore, we recommend using a small number of number of shots to balance the negative effects. In practice, we recommend just implement one prompt for PILLOW with best efficiency.

Type	**C-Score**	**PPL**	**R/w**	**M-Score**
1-shot	3.21	67.36	5.89	0.14/0.39/0.47
2-shot	3.19	**66.95**	6.05	0.14/**0.41**/**0.45**
3-shot	**3.23**	69.92	**6.17**	**0.16**/0.38/0.46

Table 3: Ablation on the number of shots. Experiments are conducted with Bloomz-1b1 on Alpaca test set.

D Example LLM Input & Output

We provide example LLM input and output here. The results are generated by Bloomz-7b1. To fit the page, we only showcase 1-shot prompts.

User Input	Matched Prompts	LLM Input	LLM Output
What sport is Ruud Gullit known for?	What is a polygon? ### A polygon is a form in Geometry. It is a single dimensional plane made of connecting lines and any number of vertices. It is a closed chain of connected line segments or edges. The vertices of the polygon are formed where two edges meet. Examples of polygons are hexagons, pentagons, and octagons. Any plane that does not contain edges or vertices is not a polygon. An example of a non-polygon is a circle.	What is a polygon? ### A polygon is a form in Geometry. It is a single dimensional plane made of connecting lines and any number of vertices. It is a closed chain of connected line segments or edges. The vertices of the polygon are formed where two edges meet. Examples of polygons are hexagons, pentagons, and octagons. Any plane that does not contain edges or vertices is not a polygon. An example of a non-polygon is a circle. ### What sport is Ruud Gullit known for ###	Ruud Gullit is a football player who played for Barcelona and England.
Rewrite the sentence using an idiom: The light suddenly disappeared	Evaluate this sentence for spelling and grammar mistakes He finnished his meal and left the resturant ### He finished his meal and left the restaurant.	Evaluate this sentence for spelling and grammar mistakes He finnished his meal and left the resturant ### He finished his meal and left the restaurant. ### Rewrite the sentence using an idiom The light suddenly disappeared ###	The light suddenly went out.
Explain how design thinking helps organizations to innovate.	Describe the structure of an atom. ### An atom is made up of a nucleus, which contains protons and neutrons, surrounded by electrons that travel in orbits around the nucleus. The protons and neutrons have a positive charge, while the electrons have a negative charge, resulting in an overall neutral atom. The number of each particle determines the atomic number and the type of atom.	Describe the structure of an atom. ### An atom is made up of a nucleus, which contains protons and neutrons, surrounded by electrons that travel in orbits around the nucleus. The protons and neutrons have a positive charge, while the electrons have a negative charge, resulting in an overall neutral atom. The number of each particle determines the atomic number and the type of atom. ### Explain how design thinking helps organizations to innovate. ###	Design thinking is a process used to solve problems through creative problem solving. It involves gathering information about the problem, brainstorming solutions, testing different ideas, evaluating them, and making decisions based on the results. Design thinking can help organizations to innovate by helping them to identify new opportunities and develop innovative solutions for their challenges.

E Hyperparameters

We set the following hyperparameters for PiLLow evaluation:

Field	**Value**
LoRA rank	8
number of RL traning epochs	150
MLP input sizes	384, 768
MLP hidden size	1024
MLP output size	512
learning rate	1e-6
trainig batch size	32
lambda (balancing factor)	0.2
LLM number of beams	1
LLM top p	0.8
LLM top k	0
LLM do sample	False
LLM number of return sequences	1
LLM temperature	1
LLM repetition penalty	1
LLM max new tokens	512
LLM length penalty	1
LLM early stopping	True