Designing Universal Prompts for LLM

Matvei Kreinin
kreinin.mv@phystech.edu
Petr Babkin
babkin.pk@phystech.edu
Maria Nikitina
nikitina.mariia@phystech.edu
Iryna Zabarianska
akshiira@yandex.ru

Abstract

The emergence of large language models (LLMs) has resulted in considerable advancements in performance across various natural language processing (NLP) tasks. Nevertheless, it quickly became apparent that the success of these models is heavily reliant on the techniques of prompt engineering. The solution to these challenges lies in an automated and reliable model capable of suggesting optimized prompts and adapting to various LLMs. Previous research has mainly concentrated on training learnable vectors or pinpointing discrete prompts, which were effective for earlier, smaller language models. However, modern LLMs necessitate coherent text prompts that are customized to their specific training guidelines. In this paper, we address this gap by proposing a methodology for training a lightweight model that not only produces optimized prompts but also adapts to different LLMs.

1 Introduction

With the increasing diversity of LLMs and the emergence of powerful pre-trained models such as GPT, BERT, and RoBERTa, optimal prompt design problems have received significant attention as an alternative to traditional fine-tuning methods that rely directly on the unique intrinsic sensitivity of a particular model. Prompts have gained prominence as a critical methodology for leveraging LLM capabilities to solve a wide range of NLP problems without requiring costly parameter updates.

The investigation of prompt tuning to adapt LLMs for specific tasks can be categorized into two main areas of research: soft-prompt tuning and discrete (hard) prompts.

Soft-prompt tuning

In the domain of soft-prompt tuning, as described in studies [6, 8], trainable vectors, also termed virtual tokens, are appended to the input layer to enable efficient task-specific tuning of LLMs. These virtual tokens are adapted to individual tasks and integrated with the standard user prompts during the inference stage. Recent research has proposed various ways to generate soft prompts, such as training of auxiliary models or differentiable representations of Prompts [3, 8]. However, these efforts often assume access to the internal state variables of the LLM [6, 10], which is usually not accessible to practitioners who engage with LLMs via APIs. Moreover, the soft prompts generated through these methods are less interpretable in natural language [9], and the natural language representations of these soft prompts can be quite misleading, as they tend to lose their effectiveness when translated back into natural language [7].

Alternatively, some research employs discrete manipulations of prompts using Reinforcement Learning techniques or LLMbased feedback mechanisms [11, 12]. Nevertheless, these algorithms might require low-level access to

the LLM, generate outputs that are difficult to comprehend or rely on undirected Monte-Carlo search strategies across the semantic space of prompts.

It is important to note that a significant limitation of these approaches is the lack of generalizability across various language models and bad interpretability of prompts. Powerful models such as GPT-3.5 exhibit a reluctance to process nonsensical tokens and prioritize legible text, owing to their fine-tuning on instruction-based tasks [2]. This trait underscores the importance of crafting coherent and legible prompts for effectively prompting these models. The opacity of these virtual tokens has catalyzed further research into discrete prompt optimization, which seeks to provide greater interpretability for model-tuning processes.

From soft prompts to hard prompts

Hard-prompt methods automate the manual prompt engineering process by integrating understandable word tokens into the original prompt framework. For example, AutoPrompt employs a template-based method in which a specific number of trigger tokens are optimized using gradient descent for tasks like sentiment classification [10]. This approach emphasizes the potential of prompt tuning as a practical alternative to extensive model fine-tuning, especially with masked language models.

Recent research has introduced techniques like RLPrompt, which employs a smaller language model with a trainable head to transform user prompts into a specific set of discrete tokens [3]. This method has demonstrated effectiveness in tasks such as few-shot classification and style transfer for models like DistilGPT and GPT-2. However, it has been noted that the discrete tokens produced can often be nonsensical and hard to interpret when applied to LLMs, which reduces their effectiveness in newer instruction-tuned and dialogue-based LLMs.

Universal prompts

Our focus is on developing a model that generates universal prompts for different LLMs. The proposed solution draws inspiration from the methodology employed in Promptist, which optimizes a small language model to generate discrete prompts specifically tailored for a Stable Diffusion model, leading to better aesthetic scores while maintaining relevance to the prompt [4]. Extending this approach to generative language models, we apply parameter-efficient fine-tuning to a smaller language model, enhancing evaluation scores consistently across a varied set of language models.This strategy not only addresses the shortcomings of generalizability seen in soft-prompt methods but also enhances the transparency and efficacy of the prompts, thereby broadening the practical deployment of LLMs in specialized tasks.

2 Proposed Methodology

We propose a two-step fine-tuning process, as depicted in Figure 1. Now we will discuss these two steps in detail.

Stage 1 - Supervised Finetuning of Lightweight LM

The primary objective of this step is to generate coherent text that incorporates essential tokens relevant to the specific LLM, while ensuring the text remains easy to read and contextually appropriate. In contrast to handling random or nonsensical text, contemporary LLMs trained on instructional data require input that is logical and meaningful. To achieve this, a lightweight language model, which focuses on proposing optimized prompts, is enhanced through supervised fine-tuning. This process involves using a dataset containing prompts and their optimized versions, which will be discussed in more detail in Section 3.

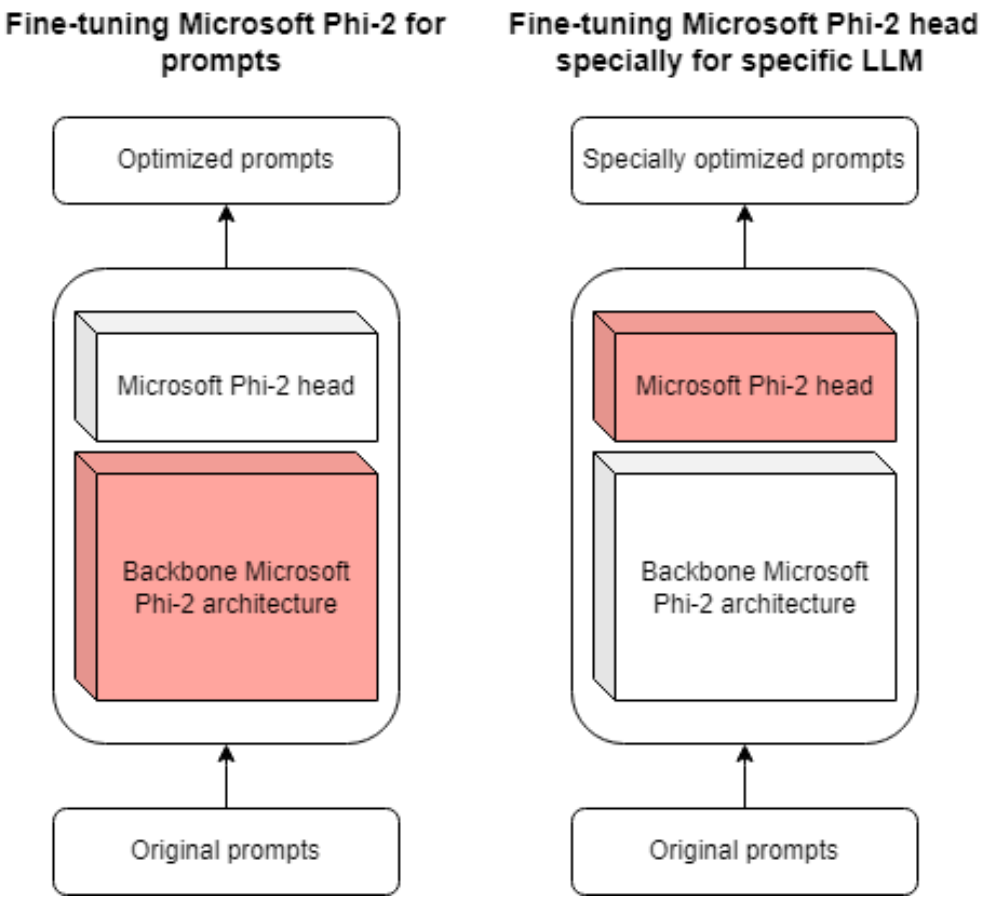

Fig. 1: Proposed approach for training prompter model

Stage 2 - Language Head Fine Tuning

Stage 1 guarantees the training of the language model for a specific task, namely generating optimized prompts based on the dataset pattern. However, it does not ensure that the optimized prompt includes sensitive tokens specific to LLMs. While this makes the prompt more precise and detailed, performance can vary significantly across different LLMs since it is trained solely on a specific dataset. In the process of language modeling, the LM head maintains the same input dimensions, but the output dimensions correspond to the vocabulary size, allowing for a probability assessment of each token's suitability

in a given position. Therefore, the language head is tasked with modeling the probabilities across the vocabulary. Considering this as essential for adapting the prompter model across different LLMs, we introduce a single MLP layer and make both this layer and the language head trainable in Stage 2.

Let $E \in \mathbb{R}^{d \times n}$ represent the encoding matrix produced by the language model, where d is the dimension of the encoded representation and n is the number of tokens. Denote by X the input tokens, then

$$E = Encoder(X),$$

where $Encoder(.)$ represents the fixed trained language model up to the encoding layer in Stage 1.

The MLP layer, responsible for transforming the encoded representations, is defined as follows:

$$Z = MLP(E),$$

where $Z \in \mathbb{R}^{d \times n}$ is the transformed representation.

The language head, which converts the transformed representations into the vocabulary space, is defined as:

$$Y = LM_{head}(Z),$$

where $Y \in \mathbb{R}^{v \times n}$ is the output logits, v is the size of the vocabulary, $LM_{head} \in \mathbb{R}^{d \times v}$ is language head.

Each LLM has a unique combination of MLP and Language Head that is dynamically loaded during inference based on the specific LLM in use. By training the MLP and Language Head separately for each LLM, our language head can adaptively assign probabilities across them. This training reveals varying high-probability tokens for different LLMs, highlighting their sensitivity to distinct tokens. At this stage, a dataset of prompts and optimized prompts is utilized to train only the MLP and Language Head.

3 Experiments

3.1 Datasets Preparation

To fine-tune prompter model, we are using the bad-improved-prompt-pairs dataset. It contains 1220 pairs of good-bad prompts on a variety of topics. To train the heads of the prompter model, bad prompts from this dataset are transferred to the LLMs, which needs to be configured to improve prompts (in our case this is GPT-4o [1] and Mistral [5]), with the task to improve it.

3.2 Prompter Model

Our prompter model is based on the Microsoft Phi-2 language model. In the initial stage we need to fine-tune the main part of this model.

1. First of all, to do this we use 32 rank LoRA layers across key transformer components. We train model for 100 epochs but in the result our model almost does not correct prompts, leaving them very similar to the input ones.

2. In the second experiment we took th dataset of 14 000 bad-good propts pairs, but this is didn't help.

3. Then we decided to train model not on pairs like good : bad, but on good: good + bad. This method also don't work well.

4. Finally, the best practice is to use during the training such messages: *System: Fix the following prompt. Prompt:* {*bad prompt*}. *Answer:* {*good prompt*}

In the second stage, we add a linear layer before the language model head, initialized as an identity matrix with zero bias. This configuration maintains the original data distribution while adding trainable parameters for the adapter head. We train our model for GPT-4o and Mistral.

Table 1: Example of the first-stage model work

Bad prompt	**Good prompt**	**Microsoft Phi-2 bad prompt**	**Microsoft Phi-2 good prompt**
How does CYP1A2 relate to coffee consumption and appetite?	Discuss the role of CYP1A2 in regulating caffeine metabolism, its effects on human health, and potential implications for weight management strategies.	What is the CYP1A2's relation to coffee consumption and appetite?	Explain how genetic variations in CYP1A2 can influence an individual's response to caffeine intake and their risk of developing obesity.

4 Conclusion

This study highlights the importance of prompt tuning for improving the performance of LLMs without the need for explicit parameter fine-tuning. Given the limitations in accessing model specifics and computational resources, our research responds to the demand for effective prompt tuning techniques that can be applied to both closed-source and open-source LLMs. The proposed training approach for the lightweight model illustrates its effectiveness in generating coherent and understandable text prompts that are compatible with modern language models. Our results consistently indicate improved performance across LLMs of different sizes.

References

[1] OpenAI at. al. Gpt-4o system card, 2024.

[2] Valeriia Cherepanova and James Zou. Talking nonsense: Probing large language models' understanding of adversarial gibberish inputs, 2024.

[3] Wang J. Hsieh C.-P. Wang Y. Guo-H. Shu T. Song M. Xing E. P. Deng, M. and Z. Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning, 2022.

[4] Chi Z. Dong L. Hao, Y. and F. Wei. Optimizing prompts for text-to-image generation, 2024.

[5] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[6] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021.

[7] Anat Kleiman-Siddharth Swaroop Finale Doshi-Velez Weiwei Pan Luke Bailey, Gustaf Ahdritz. Soft prompting might be a bug, not a feature. 2023.

[8] Guanghui Qin and Jason Eisner. Learning how to ask: Querying lms with mixtures of soft prompts, 2021.

[9] Weijia Shi, Xiaochuang Han, Hila Gonen, Ari Holtzman, Yulia Tsvetkov, and Luke Zettlemoyer. Toward human readable prompt tuning: Kubrick's the shining is a good movie, and a good prompt too?, 2022.

[10] Razeghi Y. Logan IV-R. L. Wallace E. Shin, T. and S. Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts, 2020.

[11] Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. Tempera: Test-time prompting via reinforcement learning, 2022.

[12] Muresanu A. I. Han-Z. Paster K. Pitis-S. Chan H. Zhou, Y. and J. Ba. Large language models are humanlevel prompt engineers, 2022.