A Universal Prompt Generator for Large Language Models

Gurusha Juneja
Microsoft Research
t-gujuneja@microsoft.com

Amit Sharma
Microsoft Research
amshar@microsoft.com

Abstract

LLMs are primarily reliant on high-quality and task-specific prompts. However, the prompt engineering process relies on clever heuristics and requires multiple iterations. Some recent works attempt to automate this process by improving upon human written prompts. However, creating high-quality prompts from scratch is still an unresolved challenge owing to its inherent complexity. In this work, we propose UniPrompt, a novel technique for generating high-quality human-like prompts from scratch. To do so, we identify characteristic features of human-generated prompts such as being detailed and consisting of multiple sections. Our proposed method, UniPrompt, takes as input a single sentence description of the task and generates human-like sectioned prompts using an auxiliary language model. We train the model in two stages. First, the model is finetuned on multiple tasks using a novel dataset curated using GPT-4 across over 500 tasks. Second, we align the auxiliary model to generate task-relevant (high accuracy) prompts by collecting a prompt preference dataset and optimizing the model using the Direct Preference Optimization method. Importantly, UniPrompt is task-agnostic: once trained, it can be used to generate prompts for any task. We find that UniPrompt outperforms human-generated prompts, GPT-generated prompts, and other prompt optimization techniques across diverse tasks on medicine, causality, and hate speech by up to 5.1%, 7.2%, and 11.1% respectively.

1 Introduction

While large language models (LLMs) such as GPT-3.5 and GPT-4 exhibit remarkable zero-shot abilities on various tasks by following instructions in a user-specified prompt [15, 3], their performance can be highly sensitive to the choice of words and phrasing in prompts [24]. As a result, prompt engineering becomes a complex process involving considerable manual effort.

Recent works [17, 25, 18, 6] seeks to automate this process of manual prompt optimization by iteratively editing a human-written base prompt. These methods perform specific edit actions on the base prompt to find the best modification according to performance on a validation set. Despite showing gains in accuracy, these methods are limited by the quality of the base prompt and the set of edit actions defined. Generating high-quality optimized prompts from scratch still remains a challenging task owing to the combinatorial nature of the search space.

To develop human-quality prompts from scratch, we first identify properties of human-generated prompts. Human prompt engineers tend to write long descriptive prompts containing necessary information to solve the task. In addition, recent work [22, 1, 2] emphasizes the importance of structure in a prompt containing multiple sections such as Introduction, Motivation, Output Format, etc. Structure provides fundamental contextual information as a series of key ideas to the LLM so that finer task-specific details can be communicated by varying the wording among sections. Each section plays a different role: the *Introduction* section provides the overview of the task, the *Tricks* section contains paradigms that increase LLM's accuracy (e.g., chain-of-thought, "let's think step by step" [9]), *Corner cases* section contains special rules to be followed for corner cases, and the *Output Format* section ensures that the LLM output can be reliably processed.

Drawing inspiration from these observations, we propose a method to write structured human-like elaborate prompts from scratch. We first identify the common sections in a human written prompt and curate a diverse section-wise task-prompt dataset. We then train an auxiliary language model to generate the section contents for the given task and section. To further align the auxiliary model to produce high-accuracy prompts, we perform a second stage of finetuning. In this stage, the prompts can interact with the task-solver language model. We also collect language-based feedback on the prompt, using GPT-4, according to its interaction with the task-solver language model, and further refine the prompt. We curate a preference dataset that comprises of the initial prompt, the refined prompt and a preference among both. We finetune the auxiliary model on this dataset using Direct Preference Optimization (DPO) [19].

Our contributions. We develop `UniPrompt` a universal task-agnostic prompt generation language model finetuned on a novel section-wise task-prompt dataset, curated using GPT-4, for a large set of diverse tasks. Moreover, we further improve the performance of `UniPrompt` by finetuning on a dataset curated by incorporating *language feedback* from GPT-4. Experimentation with three classification tasks across varied domains - Medicine, Cauality and Hate Speech Detection show that `UniPrompt` outperforms human-generated and GPT-4 Generated prompts along with existing prompt optimization techniques.

2 Related Work

In the past years, the impressive performance of large language models in zero-shot settings has led to a growing interest in automatic prompting. Recent methods propose an edit-based optimization technique. [25] optimizes the prompt by training an agent to edit a base prompt given the query. Whereas, [17] performs specific edits on phrases of the prompt and greedily selects from the best modifications. [18] suggests editing prompts by generating language gradients based on the behavior of the prompt on the validation set. Followed by updating the prompt using the proposed edits. [6] finetunes a language model, to enrich a prompt for text-to-image generation, using reinforcement learning to optimize aesthetic and relevance scores. Rather than editing human-generated prompts, our work aims to generate prompts from scratch.

Although these methods produce optimized prompts, the search space is limited by the set of edit actions that can be made to the initial prompt. Another direction of work explores prompting another language model to produce prompts given a few examples. [26] use a language model to generate a set of prompts given only in-context demonstrations and select the most appropriate based on the evaluation scores. Other methods like [11, 23] exploit the ability of language models to recognize performant prompts and use language models as black-box optimizers. Another class of prompt optimization techniques [5] uses evolutionary algorithms to generate new prompts by a set of operations on the given initial population of prompts.

Most of the previous work either edits a given prompt or generates short prompts (a few sentences). Moreover, these methods optimize the prompt for each task separately, hence, requiring many calls to the solver LM during test time. We aim to generate high-performance prompts that are considerably larger (multiple paragraphs) and closer to human-like prompts written for complex tasks.

3 `UniPrompt`: Training a Universal Prompt Generator

Given a large language model (LLM) and a task, the goal is to generate a prompt for solving that task. We call this LLM as the *solver LLM*. To generate prompts, we use an *auxiliary* language model, typically smaller than the *solver* LLM whose weights can be finetuned. For each task, we assume an accuracy metric on the test set that needs to be maximized.

Similar to [16], we train our model in two stages (see Fig 1), **1)** supervised finetuning over a dataset of exemplar prompts; and **2)** optimization based on pairwise feedback of generated prompts. We first generate a supervised finetuning dataset using GPT-4. The dataset contains task-specific section-wise prompts for a diverse set of tasks. This dataset is used to finetune an auxiliary language

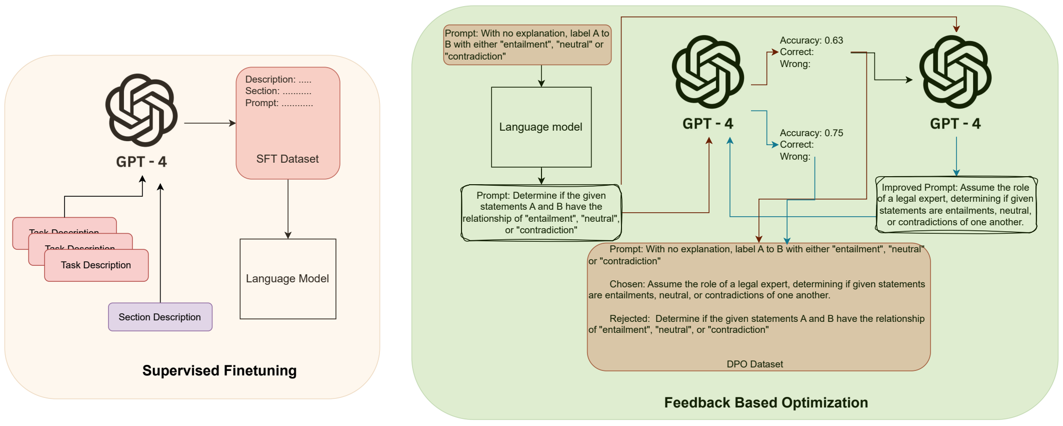
GPT - 4
Description: .....
Section: ....
Prompt: ....
SFT Dataset
Task Description
Task Description
Task Description
Section Description
Language Model
Supervised Finetuning
Prompt: With no explanation, label A to B with either "entailment", "neutral" or "contradiction"
Language model
Prompt: Determine if the given statements A and B have the relationship of "entailment", "neutral", or "contradiction"
GPT - 4
Accuracy: 0.63
Correct:
Wrong:
Accuracy: 0.75
Correct:
Wrong:
GPT - 4
Improved Prompt: Assume the role of a legal expert, determining if given statements are entailments, neutral, or contradictions of one another.
Prompt: With no explanation, label A to B with either "entailment", "neutral" or "contradiction"
Chosen: Assume the role of a legal expert, determining if given statements are entailments, neutral, or contradictions of one another.
Rejected: Determine if the given statements A and B have the relationship of "entailment", "neutral", or "contradiction"
DPO Dataset
Feedback Based Optimization

Figure 1: **Training of** `UniPrompt` occurs in two stages, first *Supervised Finetuning* where a Language model is finetuned on a dataset of diverse tasks containing section-wise prompts per task generated using GPT-4. Second stage, *Feedback Based Optimization*, where the language model is further finetuned using direct preference optimization [19] on a preference dataset. Green box depicts the process of collection of preference dataset. The language model first generates section contents for a given task. This generation is evaluated using the solver LM (GPT-4) on a validation set. The generated prompt and output of the solver LM (accuracy, correct outputs, wrong outputs) are fed into GPT-4 to get feedback and an improved prompt. Based on the relative performance of the initial and improved prompt, preference dataset is curated as shown in the figure.

model denoted as ϕ_{sft}. To further enrich the prompts produced by ϕ_{sft} we optimize it using direct preference optimization [19] over a preference dataset. To collect the pairwise preference dataset, we first evaluate the prompts produced by ϕ_{sft} against the target LM Θ, for which the prompts are to be optimized. We then use GPT-4 to provide feedback on the prompts based on their performance and suggest new prompts. Based on the performance of the prompt pair, we decide which of the prompts is preferable. Since the language model, ϕ_{dpo}, is finetuned on a wide set of tasks, it can act as a universal prompter generalizing to novel tasks.

3.1 Supervised Finetuning

As noted earlier, human-generated prompts tend to be structured [22] having multiple sections like Introduction, Examples, Motivation, Tricks, Output Format, etc. To induce the ability of structured prompt generation in a smaller language model, we curate a section-wise dataset of around 12,000 task-prompt pairs. The tasks for training dataset creation were taken from tasksource [20] library that contains around five hundred classification tasks. We extract the task description from tasksource-instruct, which contains tasksource dataset recasted with instructions. For instance, the task description for BIG-bench Entailed Polarity task [4] is, *"Given a fact, answer the following question with a yes or a no"*.

The dataset provides diverse tasks and their short description, but not the human-generated prompts for each task. To approximate human-generated prompts, we use GPT-4 as a teacher model. Initial experiments showed that generating the entire prompt from a smaller language model is difficult. Hence, we decide to finetune the language model to produce section-wise content for a given task. By prompting GPT-4 with the task description and section description, we ask it to generate the contents of the section (see Appendix B for the meta-prompt used). To ensure that the generated section-wise prompts are concise and relevant, we prompt GPT-4 to not generate more than five lines of content for each section. Based on preliminary experimentation, we decided to include ten sections for each task, which included an Introduction, Task description, Background Knowledge, Real-life application, Rules, Constraints while solving the task, Some simplification to help solve the task, challenges, or corner cases, and Tricks such as "let us think step by step".

Based on this dataset, we train our auxiliary language model, ϕ_{sft}. Each sample in the resulting dataset can be conceptualized as a tuple $<\mathcal{T},\mathcal{S},P_{gold}>$ where $\mathcal{T}$ is the task description, $\mathcal{S}$ denotes section name, and P_{gold} is the section contents generated by GPT-4 (see Appendix A for dataset example). Starting from a smaller model like Llama [21], we finetune it to generate the “ground-truth” GPT-4 prompts given a task description and section title. Denoting the text generated by ϕ_{sft} as s, We seek to optimize the following objective:

$$\min_{\phi}[-\log(p_{\phi}(P_{gold}|\mathcal{T}, \mathcal{S}, s))] \tag{1}$$

3.2 Learning from feedback using Direct Preference Optimization

While the first stage of finetuning produces legible prompts, it still lacks knowledge about the solver LLM’s behavior. To align it to the solver LM Θ, some interaction with the model is necessary. Existing methods use RLHF for such alignment [16]. RLHF is often unstable and it is difficult to train a reward model in a complex scenario. Hence, we decided to use Direct Preference Optimization [19] that eliminates the need to sample from the policy during fine-tuning and offers stable convergence.

Given two candidate outputs by the auxiliary LM, DPO increases the log probability of the preferred output to the dispreferred one while preventing model degeneration. Formally, given a dataset of the form $< x, p_1, p_2 >$ where x is the input to the model, p_1 is the more preferred output and p_2 is the less preferred output, the gradient of loss $\mathcal{L}$ with respect to the parameters ϕ of the model is given by the following equation:

$$\nabla_{\phi}\mathcal{L}_{DPO}(\pi_{\phi}, \pi_{ref}) = -\beta\mathbb{E}\left[\sigma(\hat{r}(x, p_2) - \hat{r}(x, p_2))\left[\nabla_{\phi}\log\pi(p_1|x) - \nabla_{\phi}\log\pi(p_2|x)\right]\right] \tag{2}$$

where $\hat{r}(x, p_1) = \beta \log \dfrac{\pi_{\phi}(p_1|x)}{\pi_{ref}(p_1|x)}$ is the implicit reward function. The difference in log probabilities is scaled by the difference in the implicit reward of two outputs. This factor limits the fine-tuned policy’s divergence from the base policy.

To generate preference data for DPO, we leverage the human-like language reasoning abilities of GPT-4 to provide feedback on the prompt produced by ϕ_{sft}. For a given task and section description, we sample a prompt p_{sft} from ϕ_{sft} and evaluate its performance using a validation set on the solver LM Θ. We then prompt GPT-4 to reason why p_{sft} performs incorrectly on some of the validation samples; based on the reasoning provided, we ask GPT-4 to generate a modified prompt p_{GPT} that mitigates the present issues while retaining its performance on the correct samples (see Appendix B for meta prompt). We then evaluate p_{GPT} on the same validation set against the solver LM Θ. Based on the performance of p_{sft} and p_{GPT}, we create a preference pair: if validation accuracy of p_{sft} is higher than that of p_{GPT}, then p_{sft} is preferred, and vice-versa. Using this feedback technique, we obtain 5000 prompt pairs from the same dataset of tasks and sections as used in the first finetuning stage.

4 Experimental Results

4.1 Implementational Details

We use the following configuration.

- **Solver LM:** We present results with GPT-4 and GPT-3.5 as the solver LM.
- **Auxiliary LM that generates prompts:** LLAMA 7B model, which we finetune using LoRA adapters [7]. For the training hyperparameters see Appendix C.
- **Feedback LM:** We use the state-of-the-art Language model, GPT-4, due to its ability to provide human-like detailed feedback.

4.2 Datasets

For evaluation purposes, we use three publicly available datasets - MedQA [8], CauseEffectPairs [13] and Ethos[12]. Since these three datasets were not used during training time and the task they are trying to solve is relatively difficult as compared to the tasks in training dataset, they are a good benchmark to test the "universal" character of `UniPrompt`. MedQA [8] is a dataset of multiple-choice medical problems asked in professional board exams. It contains questions in three languages:

English, simplified Chinese, and traditional Chinese, out of which we used only English language questions. The input task description for the MedQA dataset used is, *"Multiple choice question answering based on the United States Medical License Exams (USMLE). The dataset is collected from the professional medical board exams."*

CauseEffectPairs [13] is a dataset of 108 cause-effect pairs selected from 37 datasets from various domains, like biology, engineering, economics, and medicine. The task is to discover a causal relationship between two variables and predict whether A changes B or B changes A. The input task description for the CauseEffectPairs dataset used is, *"Given a pair of variables A, B, the goal is to classify whether A causes B, B causes A"*.

Ethos [12] is an online multi-label hate speech detection dataset with 998 online comments along with their labels. The task is to identify whether the given comment conveys hate or not. The input task description for the Ethos dataset used is, *"Given the query, classify it as hate or non hate."*.

4.3 Baselines

We compare `UniPrompt` with four other methods: a human written prompt to solve the task, GPT-4 as a zero-shot prompter, Automatic prompt optimization (APO)[18] and the method using language models as black box optimizers (LLM Optimizer) proposed in [11]. The human-generated prompts for the three test datasets are taken from existing literature where the authors have manually optimized the prompts [14, 10, 18]. Oracle refers to GPT-4 which was prompted to produce structured prompts given a task and section title. This is the same model that was used as the teacher during the supervised fine-tuning stage of `UniPrompt`. GPT-4 Feedback refers to the improved prompt after taking feedback on `UniPrompt` (SFT) generated prompt directly from GPT-4 rather that the prompt generated from the model finetuned on the feedback dataset. The initial prompt for Automatic prompt optimization and "language models as black box optimizers" were taken to be the human-generated prompts.

Table 1: Performance of GPT-3.5 and GPT-4 as solver LM on MedQA, Ethos, and CauseEffectPairs datasets. `UniPrompt` consistently outperforms all the other baselines for both language models.

	GPT 3.5			GPT 4		
Prompt Model	MedQA	Ethos	Causal	MedQA	Ethos	Causal
Human	53.1	74.1	80.7	76.1	78.4	95.2
APO	52.9	74.0	75.7	76.0	78.8	94.3
Oracle (GPT-4)	49.3	65.2	75.7	77.4	80.0	94.3
GPT-4 Feedback	53.4	73.0	86.5	77.3	83.4	97.0
LLM Optimizer	53.3	65.4	81.4	58.9	81.3	95.3
`UniPrompt` (SFT)	52.6	74.0	**91.3**	77.5	81.5	94.2
`UniPrompt` (SFT+DPO)	51.2	84.6	88.1	76.1	**89.5**	97.1
Human + examples	54.2	64.4	79.8	76.5	80.0	98.0
APO + examples	53.3	64.8	82.4	76.8	79.8	95.3
Oracle (GPT-4) +examples	44.0	58.7	74.0	79.2	77.0	97.2
LLM Optimizer + examples	54.7	61.1	82.2	70.5	77.6	97.1
`UniPrompt` (SFT) + examples	53.3	64.9	81.3	78.5	80.9	**99.0**
`UniPrompt` (SFT+DPO) + examples	**55.3**	**86.4**	87.9	**81.2**	87.8	**99.0**

5 Analysis

Comparision with human prompts. Comparing the performance of the `UniPrompt` generated prompts with the Human-optimized prompts (See Table 1) we observe that `UniPrompt` (SFT) & `UniPrompt` (SFT+DPO) prompts show significant improvements on the Ethos and Causal datasets. When GPT-3.5 is used as the solver LM `UniPrompt` (SFT) outperforms the human-optimized prompts on the Causal dataset by $+10.6\%$. The model shows similar performance to the human prompts on the MedQA -0.5% and Ethos -0.1% datasets. The `UniPrompt` (SFT+DPO) model outperforms the human-optimized prompts on the Ethos $+10.5\%$ and Causal $+7.4\%$ datasets.

When GPT 4 is used as the solver LM `UniPrompt` (SFT) prompt model outperforms the human-optimized prompts on the Ethos dataset by +3.1 %. The model shows similar performance to the human prompts on the MedQA $\sim 0\%$ and Causal -1% datasets. The UniPrompt (SFT+DPO) model outperforms the human-optimized prompts on the Ethos $+11.1\%$ and Causal $+1.9\%$ datasets, whilst showing similar performance on the MedQA dataset.

Comparision with zero-shot GPT-4 The UniPrompt (SFT) & UniPrompt (SFT+DPO) models show significant improvements compared to using zero-shot GPT-4 across all three datasets. When GPT-3.5 is used as the solver LM UniPrompt (SFT) outperforms the zero-shot GPT-4 prompts on the MedQA by $+3.3\%$, Ethos by $+8.8\%$ and Causal by $+15.6\%$. The UniPrompt (SFT+DPO) model outperforms the zero-shot GPT-4 prompts on the MedQA by $+1.9\%$, Ethos by$+19.4\%$ and Causal by$+12.4\%$. This shows, that multi-task training helps the model to generalise over novel tasks.

When GPT-4 is used as the solver LM UniPrompt (SFT) prompt model outperforms the zero-shot GPT-4 prompts on the Ethos by $+1.5\%$. The model shows similar performance to the GPT-4 prompts on the MedQA and Causal datasets. The UniPrompt (SFT+DPO) model outperforms the zero-shot GPT-4 prompts on the Ethos by $+9.5\%$ and Causal by$+2.8\%$.

Comparision with existing methods When compared to the existing prompt optimization techniques, UniPrompt shows significant improvement (See Table 1). When GPT-3.5 is used as the solver LM, UniPrompt outperforms the best baseline by 19.2% on Ethos, and 15.6% on casual datasets. Using GPT-4 as the solver LM UniPrompt outperforms baselines by 1.5% on MedQA, 10.7% on Ethos and 2.8% on causal datasets.

Performance on novel tasks Due to the diverse task training of UniPrompt, it consistently outperforms the baselines on out-of-distribution novel tasks. This eliminates the need for task-specific test-time optimization which was a key component of the previous works.

Sometimes examples hurt: Although it may seem counter-intuitive, but providing in-context examples in the prompt may reduce its performance (See Table 1). We see that for Ethos dataset, when using GPT-3.5 as the solver LM, providing examples decreases performance for all prompt models except UniPrompt (SFT+DPO). When using GPT-4 as the solver LM, the performance decrease is relatively lower. The MedQA and Causal datasets benefit from examples especially when using a bigger model like GPT-4 as the solver LM. This could be attributed to the random selection of examples, or the fact that information conveyed by the examples is already present in the prompt. A better analysis of example selection could provide more insight on this observation.

6 Discussion

In this work, we addressed the challenging task of generating human-like structured prompts from scratch, which distinguishes our approach from previous works that primarily focused on prompt editing. We approach the problem of generating structured prompts by training a smaller language model on a novel task-prompt dataset. Through extensive experiments conducted on three diverse datasets, our method consistently outperformed existing techniques, demonstrating its effectiveness in improving the quality and relevance of generated prompts. Our contributions not only highlight the significance of prompt engineering but also emphasize the potential of generating prompts as a crucial step in enhancing the performance of language models.

7 Limitations and Future Work

A potential limitation of this paper is that current method prompts GPT-4 to give feedback on entire prompt. It is difficult for GPT-4 to provide specific edits in a highly elaborate sectioned prompt. Further, exploring the prompt space using language feedback might not be the optimal way. Future work mignt incorporate better principled approaches to explore the prompt space.
Moving forward, several promising avenues for future research emerge from our findings. Firstly, investigating techniques to exercise more control over the prompter language model's output could lead to even more fine-grained prompt generation. Additionally, exploring the impact of incorporating examples into the prompt generation process represents another exciting direction. By analyzing how examples influence the performance of the language model, we can gain valuable insights into the capabilities of the prompt generator LM.

References

[1] A developer's guide to prompt engineering and llms. `https://github.blog/2023-07-17-prompt-engineering-guide-generative-ai-llms/`.

[2] Full sydney pre-prompt (including rules and limitations and sample chat). `https://www.reddit.com/r/bing/comments/11398o3/full_sydney_preprompt_including_rules_and/?rdt=62233`.

[3] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[4] Aarohi Srivastava et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2023.

[5] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers, 2023.

[6] Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation, 2022.

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

[8] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams, 2020.

[9] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[10] Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality, 2023.

[11] Shihong Liu, Samuel Yu, Zhiqiu Lin, Deepak Pathak, and Deva Ramanan. Language models as black-box optimizers for vision-language models, 2023.

[12] Ioannis Mollas, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumakas. ETHOS: a multi-label hate speech detection dataset. *Complex &amp Intelligent Systems*, 8(6):4663–4678, jan 2022.

[13] Joris M. Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. Distinguishing cause from effect using observational data: methods and benchmarks, 2015.

[14] Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. Capabilities of gpt-4 on medical challenge problems, 2023.

[15] OpenAI. Gpt-4 technical report, 2023.

[16] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[17] Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models, 2023.

[18] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search, 2023.

[19] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.

[20] Damien Sileo. tasksource: Structured dataset preprocessing annotations for frictionless extreme multi-task learning and evaluation. *arXiv preprint arXiv:2301.05948*, 2023.

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

[22] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023.

[23] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2023.

[24] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

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

[26] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers, 2023.

A Supervised Finetuning Dataset

Task: super glue boolq
Task Description: With no explanation, label the following with either False or True.
Section: description
Prompt: As a prompt engineer, your task is to create a prompt that accurately labels statements as either True or False without providing any explanations. The statements may involve general knowledge, trivia, or facts about events, places, people, or objects. Ensure you have a strong understanding of the subject matter and write a concise and specific task description related to the given statements.

Task: glue qnli
Task Description: With no explanation, label A to B with either entailment or not entailment
Section: background
Prompt: 1. Entailment means that the information in statement B can be inferred directly from statement A.
2. Not entailment means that the information in statement B cannot be inferred directly from statement A or is unrelated.
3. Understanding the context and relationship between the two statements is crucial for accurate classification.

Task: bigbench undo permutation
Task Description: In the following sentences with shuffled words, find the correct order of word swaps to unscramble the sentence.
Section: tricks
Prompt: 1. Identify the key words or phrases in the task to understand the context of the sentence. Look for nouns, verbs, and adjectives that seem related or could logically fit together.
2. Start by solving the problem step by step and focus on one swap at a time. Breaking the problem into smaller sub-problems will make it easier to manage.
3. To make the task more manageable, first focus on swapping the words that are clearly out of place,

such as words that should be at the beginning or end of the sentence.

SFT data curation Prompt

```
You are a prompt engineer. You are given a task. This is the task description:
{task_description}
Here are some examples:
{examples}

You are asked to write a prompt to solve this task. The prompt should be able to solve
the task with 100% accuracy.
The prompt would include the following:
1. Introduction
2. Task Description
3. Examples with explanation
4. Some trics to solve the task
5. Background Knowledge
6. Real life application
7. Rules or Constraints
8. Some simplification to help solve the problem
9. Challanges or corner cases
10. Desired output format.


## Background Section
But, as for now, you only have to write the Background Knowledge section.

The section can include any background knowledge required to solve the task.
For example, if it is a math problem then the task related formulas can be provided.
Incase no background knowledge is required, leave it blank.
Give no more than 3 lines or points.
Be creative but make sure to only provide task specific knowledge. Be breif and specific.

## Constraints Section
But, as for now, you only have to write the Introduction section.

The section can include the constraints to problem or any rules to has to follow.
For example, if the task is to verify a fact then the constraint could be to not
follow any stereotype or bias.
Do not exceed 3 lines.
Be creative but make sure to only provide task specific constraints. Be breif and specific.

## Application Section
But, as for now, you only have to write the Real life application section.

The section can include any application of the task in real life that can ease the models
understanding of the task.
For example, if the task is to solve differential equation, then application
could be to understand the behaviour of a multi-particle system.
Give maximum of two applications with breif explanation.
Be creative but make sure to only provide task specific knowledge. Be breif and specific.
```

B GPT-4 Feedback Meta Prompts

```
You are given an input prompt {prompt}.
This prompt is evaluated on an LLM.
The LLM gives the following inputs, the LLM gives wrong answer.
```

```
## Wrong outputs
{wrong_outputs}

For the following inputs, the LLM gives right answer.

## Right outputs
{right_outputs}

You need to tell what is wrong with the input prompt and
finally suggest a modification to the prompt.
You can add new sections as well.
Make sure not to change the input section of input prompt.
Make sure not to change the output format section of the input prompt.
Make sure that the modified prompt does not change LLMs answer to the right outputs and
changes the LLMs answer to the wrong outputs.
At the end of your answer, rewrite the entire new prompt in the following format:
[new prompt: prompt Content]
Make sure to write the entire modified prompt between brackets
like [new prompt: prompt Content]. Do not miss this step.
```

C Hyperparameter Selection

For the supervised fine-tuning stage, we used LORA $r = 16$, LoRA $\alpha = 32$, LoRA dropout $= 0.05$.
We finetuned the model for just one epoch on a batch size of 32, learning rate 2e-4, max grad norm 0.3, warmup ratio 0.03.
For the second stage, we use LORA $r = 8$, LoRA $\alpha = 32$, LoRA dropout $= 0.05$. We optimize using DPO in this stage with $\beta = 0.2$ and other hyperparameters as default.