TAPR: Enhancing LLM Performance with a Task-Aware Prompt Rewriter

Oliver Savolainen[1], Emanuele Bastianelli[2], Hosein Azarbonyad[2] and Ana Lucic[1]

[1]University of Amsterdam, Amsterdam, The Netherlands
[2]Elsevier, Amsterdam, The Netherlands

Abstract

Large Language Models (LLMs) often require carefully crafted prompts to unlock their full potential, which can be a barrier for non-expert users. This work addresses the challenge by introducing a Task-Aware Prompt Rewriter (TAPR), a model that reformulates user prompts into task-optimized prompts with the explicit goal of improving downstream LLM performance. We train TAPR using reinforcement learning with Group Relative Policy Optimization (GRPO), where rewards are derived from LLM-as-judge evaluations of both the reformulated prompt and the corresponding task output. Experimental results on diverse tasks, such as question answering, summarization, and arithmetic reasoning, show that our method yields consistent gains over base models in prompt rewriting ability. Fine-tuning Phi-4-mini-instruct (as the base model for TAPR) produces prompts that contain clearer and more instructive language, leading to higher accuracy on established benchmarks such as Natural Questions and GSM8K. Our code is available at: https://github.com/OliverSavolainen/task-specific-prompt-rewriter

Keywords

Automated Prompt Rewriting, Large Language Models (LLMs), Reinforcement Learning (RL), LLM-as-a-Judge

1. Introduction

Large Language Models (LLMs) have revolutionized the field of natural language processing, providing unprecedented capabilities in text generation, comprehension, and various other applications [1, 2]. Despite their impressive performance, these models often require carefully crafted input prompts to produce the best possible responses [1, 3]. Users, particularly those without expertise in prompt engineering, may find it challenging to formulate prompts that fully leverage the capabilities of these advanced models. This gap underscores the necessity for an automated system that can refine and optimise user prompts, making high-quality interaction with LLMs more accessible [4, 5].

Existing methods for prompt optimization rely largely on manual adjustments and iterative testing, which are not only time-consuming but also require a certain level of expertise. Automated prompt rewriting can bridge this gap by refining under-specified queries into clear, robust instructions that ensure reliable performance for all users.

In this paper, we present a **Task-Aware Prompt Rewriter (TAPR)**, a lightweight model that automatically refines and adapts user prompts for LLMs, with the aim of enhancing downstream task performance. This system involves a small, specialized model dedicated to rewriting prompts, coupled with a larger LLM that executes the task based on these optimized prompts. To train TAPR, we build on

[PromptEng] Third International Workshop on Prompt Engineering for Pre-Trained Language Models co-located with the ACM WebConf, April.13, 2026, Dubai, United Arab Emirates
oliver.savolainen@student.uva.nl (O. Savolainen); e.bastianelli@elsevier.com (E. Bastianelli); h.azarbonyad@elsevier.com (H. Azarbonyad); a.lucic@uva.nl (A. Lucic)

the recent advancements on using reinforcement learning to improve LLM performance across a range of tasks. Therefore, the main research question of this work is:
How can we create a task-aware prompt rewriter such that it enhances the performance of LLMs in downstream tasks?

We use reinforcement learning to train a smaller LLM to optimize default prompts by rewriting them, using feedback derived from the performance of a frozen Task LLM responding to the rewritten prompts. Prior work has explored using reinforcement learning for this purpose. PRewrite [5] applied PPO (Proximal Policy Optimization) [6] to fine-tune LLMs on individual datasets, yielding a single optimal prompt per dataset and demonstrating gains on several benchmarks.

Our approach extends this in two key ways. First, beyond the verifiable task-performance rewards used by prior work, we introduce using LLM-as-a-judge rewards for semantically more accurate evaluations, including a reward based on the quality of the prompts themselves, and we validate its impact through a dedicated ablation study. Second, we employ Group Relative Policy Optimization (GRPO) [7] in place of PPO and achieve great consistency with this algorithm. We also test generating multiple prompts per dataset and having TAPR select the best one, but we do not find this addition leading to further performance gains consistently.

To evaluate our method, we measure the performance of the Task LLM before and after rewriting on various tasks, including question answering, summarization, and arithmetic reasoning. Our framework leads to improvement in the prompt rewriting abilities of Phi-4-mini-instruct and LLaMA-3.2-3B-Instruct on multiple tasks. For Phi-4-mini-instruct, the highest metric score comes from a TAPR variant for each dataset.
In summary, our contributions are:

- We introduce **TAPR**, a task-aware prompt rewriter trained with reinforcement learning to optimize task-specific prompts using feedback derived from both rewritten prompts and responses from LLM performing the tasks.
- We leverage the LLM-as-a-judge paradigm to obtain semantically informed feedback and design a dedicated **prompt quality reward**, demonstrating that using LLM-based evaluations improves training stability and effectiveness.
- We perform various experiments across question answering, summarization, and arithmetic reasoning benchmarks, showing that prompts rewritten by TAPR enhance task performance compared to baseline prompts and rewrites from base models.

2. Related Work

In order to understand how to design the method for training a model to rewrite prompts, we look at previous work on LLM fine-tuning, prompt engineering, and existing prompt rewriting works. In addition, we look at the LLM-as-a-judge paradigm to explore alternative ways of evaluating and rewarding models.

2.1. LLM Fine-tuning

Supervised fine-tuning (SFT) is a general technique for adapting any pretrained LM to a labeled dataset; in the context of generative models it's often called instruction-tuning, since the labels take the form of

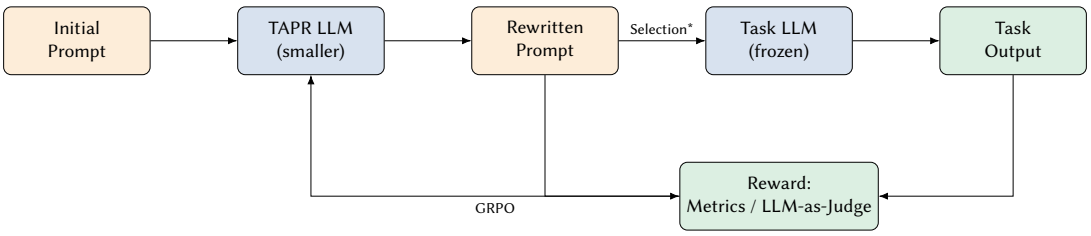

Figure 1: Overview of the training pipeline for TAPR. **Orange** nodes are *prompts* (inputs to models), **blue** nodes represent *models* (trainable or frozen), and **green** nodes are *outputs*—either the task answer or the scalar reward. Solid arrows show the forward data flow; the green reward is fed back via GRPO (curved arrow).

(prompt, response) pairs [8]. Although SFT provides strong initialization for instruction following, it does not explicitly optimize for long-term objectives, such as user satisfaction or safety. Reinforcement learning from human feedback (RLHF) addresses this by training a scalar reward model on human preference judgments between model outputs, and then using Proximal Policy Optimization (PPO) to maximize that learned reward [9, 6]. In the InstructGPT pipeline, a policy initialized by SFT is refined via RLHF, yielding substantial gains in helpfulness and coherence over purely supervised approaches [10].

More recently, Group Relative Policy Optimization (GRPO) was introduced which generalizes PPO by updating the policy based on a group of sampled trajectories, thereby removing explicit value function learning and enabling multi-task or multi-agent training regimes, leading to impressive performance for the DeepSeek R1 model [7, 11]. Both GRPO and PPO have also been used to enhance other abilities, for example, Search-R1 [12] showed the effectiveness of both algorithms to make LLMs better in the use of search engines. Our work is the first to explore GRPO for enhancing prompt rewriting abilities.

2.2. Prompt Engineering

Task framing has been proven to have a major impact on LLM performance. One of the first important breakthroughs for this was few-shot prompting, where providing just a handful of examples enables generalization without task-specific training [1, 13]. Chain-of-thought prompting (i.e., "Let's think step by step") elicits intermediate reasoning and boosts accuracy on arithmetic and logical tasks [14, 13]. Few-shot chain-of-thought prompting remains essential for challenging reasoning benchmarks like GSM8K and StrategyQA, teaching models to break problems into steps [15].

Clear, structured instructions also matter. Specifying output formats, especially JSON, improves consistency and makes results easier to parse [16, 17, 18]. Yet prompt design is still difficult, with no universally optimal formulation. Our approach tackles this by automatically rewriting prompts using prompt engineering principles.

2.3. Prompt Rewriting

Automated prompt engineering or prompt rewriting employs diverse optimization techniques to refine prompts that steer LLM behavior. AutoPrompt uses gradient-based search over discrete token spaces to assemble sequences that elicit desired outputs, though often producing non-interpretable prompts [4]. Evolutionary methods such as PromptBreeder treat prompt design as a population-based search,

applying crossover and mutation to candidate prompts and selecting offspring by performance [19], but rely on carefully chosen initial prompts, limiting the search space.

Reinforcement learning offers a more sample-efficient alternative by framing prompt generation as a sequential decision process. RLPrompt trains an agent to edit prompts token by token with rewards from task performance (e.g., classification accuracy, BLEU score), discovering prompts that generalize across datasets but remain hard to interpret [20]. TEMPERA improves stability by defining a custom action space over high-level edits (e.g., paraphrasing, task description insertion) and using Q-learning to maximize a composite reward of performance and linguistic diversity [21], though its constrained actions may miss novel formulations.

PRewrite adopts PPO to fine-tune LLMs (e.g., PaLM 2-S/L [22]) to rewrite user prompts in a task-aware manner [5]. Unlike token-level policies or fixed edit spaces, it processes the entire prompt and outputs a rewritten version optimized for metrics such as exact match rate on the Natural Questions dataset or solution rate for GSM8K. Empirical results show that PRewrite improves task performance and yields more interpretable and concise prompts than RLPrompt, without TEMPERA's constraints. However, it only uses automated metrics such as exact match accuracy, which can lead to prompts that do not optimize for real-world utility. Our work leverages LLM-based judging to account for that limitation.

2.4. LLM-as-a-Judge

Traditional automatic evaluation metrics such as BLEU [23], ROUGE [24], METEOR [25], and BERT-Score [26] rely on n-gram overlap or embedding similarity with static reference texts, which can poorly reflect real-world utility in open-ended tasks such as summarization, dialogue, or long-form question answering [27]. These metrics often penalize valid paraphrases or novel content, and cannot account for pragmatics, coherence, or factuality beyond surface overlap. To address these shortcomings, the LLM-as-a-judge paradigm repurposes LLMs to score or rank model outputs by simulating human evaluators [28, 29]. By prompting the judge model with evaluation criteria such as relevance, fluency, and informativeness, it can produce scalar scores or pairwise preferences that align more closely with human judgments and adapt dynamically to new tasks without costly reference annotation.

Beyond zero-shot scoring, LLM judges have been integrated directly into model training loops. For example, RLAIF demonstrates that synthetic preferences generated by an auxiliary LLM can approximate human feedback for RLHF, enabling large-scale reinforcement learning at a lower cost with minimal quality drop-off [30, 31]. This does not mean that using LLM judges leads to a perfect and faultless metric for every situation, but using LLMs to judge can often offer a better way to evaluate or calculate rewards for text generation, potentially leading to better LLM fine-tuning performance for various tasks. Our method is the first to integrate LLM-as-a-judge scores for rewards and evaluations to guide and improve automated prompt rewriting.

3. Task-Aware Prompt Rewriter

In this section, we describe the details of our method for creating the Task-Aware Prompt Rewriter (TAPR). The overall training method for TAPR can be seen in Figure 1. The training procedure works as follows: for each sample, we rewrite the initial prompt using the TAPR LLM, then we give the rewritten prompt to the Task LLM, which performs the task, and finally, we calculate the rewards using both the rewritten prompt and the task. Based on the reward values, the model is trained with a reinforcement learning algorithm.

Dataset	Example	Initial Prompt
Natural Questions (NQ) [32]	Who was the first president of the United States?	"Answer the question." [5]
HotpotQA [33]	Which novel did Mary Shelley write that was published in 1818?	"Answer the question based on the context."
CNN/Daily Mail [34]	*Article excerpt:* An Iraqi boy, badly burned, is flown to the U.S. for treatment funded by a charity.	"Summarize the text."
SciTLDR [35]	*Abstract excerpt:* We propose a 2-simplicial Transformer for deep RL and show it improves reasoning.	"Summarize the text."
GSM8K [36]	Katy mixes sugar and water in a 7:13 ratio. If she used 120 units total, how many units of sugar were used?	"SOLUTION." [5]

Table 1
All the datasets used in the experiments, example samples from them, and initial baseline prompt for rewriting we are using for them.

3.1. Problem Formulation

TAPR is a smaller LLM that reformulates queries for a bigger frozen Task LLM. We formulate the problem similarly to [5] and [37]. Let $\mathcal{P}$ denote the space of textual prompts. We consider the TAPR LLM $\pi_\theta : \mathcal{P} \to \mathcal{P}$ that given an original prompt $p \in \mathcal{P}$ outputs a revised prompt $\tilde{p} = \pi_\theta(p)$. The revised prompt is then fed into a frozen Task LLM L, yielding an output $y = L(\tilde{p})$. Let y^* be the desired (ground-truth) output corresponding to input p. We define two reward functions: a task performance reward $T(y, y^*)$ (such as LLM-as-a-judge score or accuracy), measuring how well y matches y^*, and a prompt quality reward $Q(\tilde{p})$ that encourages the rewritten prompt to be accurate and follow common prompt engineering principles.

For reinforcement learning, we treat π_θ as a policy over the discrete prompt space. We define a combined reward

$$R(y, \tilde{p}) = \alpha\, T(y, y^*) + \beta\, Q(\tilde{p}),$$

where $\alpha, \beta > 0$ are weighting coefficients. The reinforcement learning objective is to maximize the expected reward under the data distribution $\mathcal{D}$ of prompts:

$$\mathcal{J}_{\mathrm{RL}}(\theta) = \mathbb{E}_{\substack{p \sim \mathcal{D} \\ \tilde{p} \sim \pi_\theta(\,\cdot\,|p)}} \big[R\big(L(\tilde{p}),\, \tilde{p}\big) \big].$$

We optimize the expectation via policy gradient methods such as PPO [6] or GRPO [7], treating prompts as policies. This formulation follows the optimization perspective of prompt tuning (treating prompts as differentiable policies and maximizing downstream task metrics).

3.2. Rewards

We train TAPR using the GRPO algorithm [7], leveraging its recent success in advancing reasoning and other abilities within language models. It has also been used more commonly with verifiable rewards

such as accuracy rather than rewards coming from a model trained on preference data, such as PPO and DPO [38, 6, 7].

The TAPR model gets the initial prompts as input, and rewrites them. Then both the prompts and responses from the Task LLM are evaluated. The initial prompt can be a general instruction such as *"Answer the question"*, or it can also include the question and context if one exists. Example meta-prompt is given in Appendix A.1. We use two types of rewards with equal weights: Task LLM performance rewards and the prompt quality reward. For the former, we use scores from an LLM-as-a-judge evaluator rather than traditional metrics such as exact match accuracy and ROUGE due to their limitations.

This approach enables the evaluation of semantic correctness, accounting for cases in which LLMs generate responses that are more verbose than the ground-truth labels but nonetheless accurate[1].

The **prompt quality reward** is a score on a scale of 0 to 5 coming from an LLM judge based on whether the rewritten prompt matches the meaning of the initial instruction and improves it according to common prompt engineering principles. This decreases the chances of generating prompts with drastic inaccuracies (for example, by completely changing the meaning of the initial instruction or even just essentially repeating the rewriting instruction). As LLM-as-a-judge usage can lead to unwanted bias, we try to mitigate this by using two different models as judges.

3.3. Prompt Selection Mechanism

To further improve rewriting performance, we introduce a **Selection** mechanism, where the trained TAPR model selects the most promising prompt from multiple generated options sampled with a higher temperature during inference. We hypothesize that, while this selection is not the main task of the TAPR LLM, the trained model has gathered enough knowledge about prompt engineering to evaluate the quality of prompts in addition to generating them. We compare this approach to an approach where prompts are generated by models using a temperature set to zero.

4. Experimental Setup

We evaluate the performance of TAPR across different tasks, including question answering, summarization, and arithmetic reasoning (GSM8K) [36]. Although we also report standard metrics for these datasets, our primary focus is on LLM-as-a-judge evaluations for question answering and summarization, as these better address the limitations of conventional evaluation methods as detailed in Section 2.4.

In Table 1, we show examples from each dataset we are using alongside the initial baseline prompts we are using for rewriting.

We evaluate prompt rewriting performance across a variety of models, tasks, and training methods. Each experiment involves two distinct LLMs: the TAPR LLM, which generates improved prompts, and the Task LLM, which solves the downstream task using these rewritten prompts. Our main experiments use LLaMA-3.1-8B-Instruct [39] as the Task LLM and compare two open-source TAPR LLMs: Phi-4-mini-instruct [40], LLaMA-3.2-3B-Instruct [41]. Results with Qwen3-4B [42] are in Appendix C.

Reinforcement Learning Training Setup and Stopping Criterion: We train each TAPR LLM using reinforcement learning, precisely the GRPO [7] algorithm, until convergence, defined by a lack of

[1]We verified the effectiveness of this approach through running an internal test, and we evaluated that only 1 out of 100 of the judgments on the Natural Questions were incorrect.

improvement in the 25-step moving average of the reward for 100 consecutive steps. This stopping criterion prevents excessive training once the model's performance plateaus. The rewards used are specified in Section 3.2. To reduce bias and stabilize the reward signal, we average the LLM-as-a-judge scores from two different judge models (LLaMA-3.1-8B-Instruct and GPT-4o-mini).

Hyperparameters for training are listed in Appendix B. To maximize experimental coverage rather than repeat identical runs with different seeds, each training and evaluation is conducted only once.

We observed that the TAPR LLM might generate responses containing information other than the rewritten queries, such as explanations for the rewriting decisions made or responses to the initial task. This might lead to the Task LLM being unable to answer the query. Therefore, we instruct TAPR to use a JSON format. Apart from this change, we closely follow most templates and initial prompts used in PRewrite [5]. This gives us reliable baselines and a starting point to improve any other prompts. A full example of a meta-prompt can be seen in Appendix A.1.

Evaluation Protocol: Final evaluations are conducted on 1,000 samples from the validation or test split of each dataset. During evaluation, we use GPT-4o-mini as the sole judge model to score answers (avoiding any bias from using the same model for generation and judgment). In our internal tests (Section 3.2), verdicts from GPT-4o-mini were nearly perfectly accurate on the question-answering tasks.

Decoding Strategies: For consistency and reproducibility, we use a temperature of zero for the Task LLM's answer generation and for all judge evaluations. For TAPR's generation, we compare two strategies: (a) *TAPR*, where the rewriter uses temperature zero and returns a single fixed rewrite; and (b) *TAPR + Selection*, where the rewriter samples five candidate prompts using a moderate temperature (0.5) and then selects the best candidate based on its judgment. We chose five samples to balance diversity and computational efficiency.

Ablation Studies: To isolate the contribution of the GRPO algorithm, prompt quality reward, and using SFT before TAPR training, we also conduct ablation studies using the Phi-4-mini-instruct as TAPR LLM with LLaMA-3.1-8B-Instruct as the Task LLM. We consider Phi-4-mini-instruct to be a capable model for its size, which responds well to reinforcement learning training in our experiments. LLaMA-3.1-8B-Instruct was chosen as the Task LLM as it is a slightly bigger LLM than Phi-4-mini-instruct, while not being from the same model family.

We also perform experiments using full-prompt rewriting, and alternative Task LLMs, and report these results in the Appendices D and E. To assess how task-aware and generalizable our method is, we also report experimental results across tasks in Appendix F.

5. Experimental Results

We compare four conditions for each base model in each experiment:

- **Baseline Prompt**: The unmodified initial instruction.
- **Base Model**: Rewriting with the base version of the LLM before training.
- **TAPR**: Rewriting after training the rewriter LLM with our method.
- **TAPR + Selection**: Generating multiple rewritten prompts after training and selecting one of them.

This allows us to answer our research question and analyze whether our method is effective in enhancing the prompt rewriting abilities of LLMs. All results are reported based on the quality of the Task LLM responses.

5.1. Impact of Prompt Rewriting on Different Tasks

Here, we evaluate whether TAPR enhances the performance of LLMs in downstream tasks.

Question Answering Table 2 reports accuracy based on LLM judgments on the Natural Questions (NQ) dataset. In Appendix Table 14, we demonstrate that standard exact-match metrics often yield low scores not aligned with the actual answer quality, suggesting that LLM-as-a-judge evaluation can capture practical performance better. For Phi-4-mini-instruct and LLaMA-3.2-3B-Instruct, both deterministic rewrites and TAPR + Selection variants outperform their Base Model versions and surpass the Baseline prompt's accuracy.

TAPR LLM	Rewriting Variant	Accuracy ↑
Baseline prompt	–	56.30
Phi-4-mini-instruct	Base Model	48.80
Phi-4-mini-instruct	TAPR	**59.20**
Phi-4-mini-instruct	TAPR + Selection	57.30
LLaMA-3.2-3B-Instruct	Base Model	58.30
LLaMA-3.2-3B-Instruct	TAPR	**62.20**
LLaMA-3.2-3B-Instruct	TAPR + Selection	59.10

Table 2
NQ dataset results, each value is an accuracy percentage (%), measured by GPT-4o-mini, reflecting answer correctness for each prompt variant.

TAPR LLM	Rewriting Variant	Accuracy ↑
Baseline prompt	–	53.20
Phi-4-mini-instruct	Base Model	53.20
Phi-4-mini-instruct	TAPR	**56.70**
Phi-4-mini-instruct	TAPR + Selection	53.16
LLaMA-3.2-3B-Instruct	Base Model	53.15
LLaMA-3.2-3B-Instruct	TAPR	**56.00**
LLaMA-3.2-3B-Instruct	TAPR + Selection	55.00

Table 3
HotpotQA dataset results, each value is an accuracy percentage (%), measured by GPT-4o-mini, reflecting answer correctness for each prompt variant.

Example prompts illustrating the rewriting for Natural Questions are provided in Table 4. The results show that the Phi-4-mini-instruct's base model rewrite is focused on describing the process, while RL fine-tuning yields more targeted and concise instructions. The best-performing prompt, generated by Llama 3.2-3B-Instruct TAPR, combines accuracy requirements with explicit formatting and clarity constraints.

TAPR LLM	Rewriting Variant	Prompt
–	Initial Prompt	Answer the question
Phi-4-mini-instruct	Base Model	Provide a detailed explanation of the process you used to answer the question, including any relevant examples or illustrations.
Phi-4-mini-instruct	TAPR	Answer the question and provide a brief explanation for your answer.
LLaMA-3.2-3B-Instruct	TAPR	Provide a well-supported answer to the question in a concise paragraph (approximately 50–75 words) or a list of 3–5 bullet points. Ensure your response is accurate and relevant, and include evidence or examples to support your answer. Use clear, concise language and avoid ambiguity.

Table 4
Example prompts for the Natural Questions dataset. Rows compare the baseline, the pre-RL rewrites, and the TAPR outputs.

Summarization For summarization, we report LLM-as-a-judge scores out of 5 in Tables 5 and 6 on the CNN/Daily Mail and SciTLDR datasets.

The baseline summarization instruction performs noticeably worse than several rewritten prompt variants. On both datasets, Phi-4-mini-instruct trained variants yield the highest scores.

The best overall scores come from TAPR and TAPR+Selection variants of Phi-4-mini-instruct on the datasets, respectively. These prompts are shown in Table 8. Both prompts are well-matched to their respective dataset styles and provide strong length constraints, leading to more focused summaries. In contrast, over-specified prompts (such as LLaMA-3.2-3B-Instruct's multi-sentence summary format) can degrade performance by instructing the use of formats that do not match the reference summaries. LLaMA results overall are weaker here, which could be due to the hyperparameters being optimized based on experiments with Phi-4-mini.

TAPR LLM	Rewriting Variant	Judge Score (1-5) ↑
Baseline prompt	–	3.373
Phi-4-mini-instruct	Base Model	3.772
Phi-4-mini-instruct	TAPR	**3.782**
Phi-4-mini-instruct	TAPR + Selection	3.774
LLaMA-3.2-3B-Instruct	Base Model	**3.794**
LLaMA-3.2-3B-Instruct	TAPR	3.435
LLaMA-3.2-3B-Instruct	TAPR + Selection	3.411

Table 5
CNN/DM dataset summarization results, each value is an LLM-as-a-judge score out of 5, as rated by GPT-4o-mini, reflecting summary quality for each prompt variant.

Pairwise Win-Rate Results for Summarization To complement the numeric scores, we also report win-rates comparing summaries generated from the baseline prompts with the ones generated with the rewritten prompts in Table 7. These results confirm that TAPR variants are often preferred over both

TAPR LLM	Rewriting Variant	Judge Score (1-5) ↑
Baseline prompt	–	3.182
Phi-4-mini-instruct	Base Model	3.802
Phi-4-mini-instruct	TAPR	3.710
Phi-4-mini-instruct	TAPR + Selection	**3.869**
LLaMA-3.2-3B-Instruct	Base Model	**3.749**
LLaMA-3.2-3B-Instruct	TAPR	3.107
LLaMA-3.2-3B-Instruct	TAPR + Selection	3.041

Table 6
SciTLDR dataset summarization results, each value is an LLM-as-a-judge score out of 5, as rated by GPT-4o-mini, reflecting summary quality for each prompt variant.

the baseline and base model outputs. None of the TAPR models had a win-rate lower than 40%; notably, Phi-4-mini-instruct achieves almost 85% win-rates on SciTLDR, while LLaMA-3.2-3B achieves between 47%–63% win-rates despite the numeric score showing worse performance. We do observe a strong bias of the LLM-as-a-judge towards the second shown option, which is why we shuffled outputs randomly for each sample at comparison time.

Dataset	TAPR LLM	vs. Baseline		vs. Base	
		Overall ↑	Placed 1st/2nd ↑	Overall ↑	Placed 1st/2nd ↑
CNN/Daily Mail	Phi-4-mini	46.11	18.13/72.99	40.66	20.41/59.69
	LLaMA-3.2-3B	62.89	37.45/89.09	67.03	39.07/94.28
SciTLDR	Phi-4-mini	84.58	75.23/94.88	84.42	86.09/82.80
	LLaMA-3.2-3B	47.97	37.11/59.53	50.73	39.88/62.89

Table 7
Win–rate comparison on CNN/Daily Mail and SciTLDR datasets. "Overall" is the percentage of cases where TAPR was preferred by GPT-4o-mini; "Placed 1st/2nd" reports preferences when shown first vs. second, compared to both baseline and base model prompts.

Arithmetic Reasoning (GSM8K) For the arithmetic reasoning task with the GSM8K dataset, we report accuracy based on the last integer found in the model's response, matching the reward used for training. While this approach may occasionally misattribute the answer if multiple numbers are present, it also serves as a strong indicator of whether prompts guide models to output formats suited to the evaluation method. Results can be seen in Table 9.

The baseline "SOLUTION" prompt is surprisingly robust and yields high accuracy relatively to many of the more detailed rewritten prompts. Nevertheless, the Phi-4-mini-instruct TAPR variant surpasses it, while also exhibiting more than a 70% improvement over the base model.

Table 10 includes example prompt rewrites for GSM8K. The Phi-4-mini-instruct's base model rewrite asks for a Python function to sum integers, likely leading to confusion for the Task LLM and inability to produce the final correct answer with the limited maximum response tokens. This leads to poor

Dataset	TAPR LLM	Rewriting Variant	Prompt
CNN/DM, SciTLDR	–	Initial Prompt	Summarize the text
CNN/DM	Phi-4-mini-instruct	TAPR	Please provide a concise summary of the provided text, aiming for a succinct overview suitable for a general audience. The summary should be no longer than one-third of the original text length, maintaining a neutral and informative tone. Ensure that the key points and main ideas are preserved while eliminating any redundant or less critical information.
SciTLDR	LLaMA-3.2-3B-Instruct	TAPR	Condense the given text into a 3- to 5-sentence summary, highlighting key points and main ideas. Present the summary in a neutral, third-person voice, without using the author's name or any personal pronouns. For example, A recent study found that [key finding]...

Table 8
Example prompts for the CNN/DM and SciTLDR datasets. Rows compare the baseline, the pre-RL rewrites, and the TAPR outputs.

performance, which all other rewrites clearly surpass.

In contrast, the TAPR variant produces a highly relevant, task-specific chain-of-thought prompt that encourages explicit step-by-step reasoning, justifications for each step, and a clear final answer. This style directly supports the kind of solutions needed for GSM8K, resulting in substantial performance improvements.

Despite improvements, neither the summarization nor the arithmetic reasoning results show substantial improvement over the base model in the former or the short general prompt in the latter task, with some trainings and variants leading to worse task performance. In addition to the potential issue with hyperparameters, this highlights how RL training for a task that is not measured directly based on the output of the model being trained is challenging, and not always reliable. Furthermore, despite significant research, prompt engineering is still an open problem where we can not exactly know which prompt leads to the optimal performance for the Task LLM. Using Selection leads to improvement sometimes, but overall, we find no strong evidence that training for prompt rewriting consistently enhances a model's ability to judge and select among candidate prompts.

TAPR LLM	Rewriting Variant	Accuracy ↑
Baseline – "SOLUTION"	–	82.38
Phi-4-mini-instruct	Base Model	11.91
Phi-4-mini-instruct	TAPR	**83.60**
Phi-4-mini-instruct	TAPR + Selection	**83.60**
LLaMA-3.2-3B-Instruct	Base Model	74.80
LLaMA-3.2-3B-Instruct	TAPR	**82.80**
LLaMA-3.2-3B-Instruct	TAPR + Selection	81.80

Table 9
GSM8K dataset results comparing the Baseline against the base models and TAPR variants. Each value is an accuracy percentage (%), reflecting answer correctness for each prompt variant.

TAPR LLM	Rewriting Variant	Prompt
–	Initial Prompt	SOLUTION
Phi-4-mini-instruct	Base Model	Please provide a detailed solution to the following problem: Given a list of integers, write a function in Python that returns the sum of all even numbers in the list...
Phi-4-mini-instruct	TAPR	Provide a step-by-step solution to the problem, including intermediate steps, justifications for each step, and a clear explanation of the final answer.

Table 10
Example prompts for the GSM8K dataset. Rows compare the baseline, the pre-RL rewrites, and the TAPR outputs.

TAPR LLM	Rewriting Variant	NQ ↑	GSM8K ↑
Baseline	–	56.30	82.38
Phi-4-mini-instruct	Base Model	48.80	11.91
Phi-4-mini-instruct	TAPR with GRPO	**59.20**	**83.60**
Phi-4-mini-instruct	TAPR with PPO — Con.	35.90	71.10
Phi-4-mini-instruct	TAPR with PPO — Mod.	4.00	75.20
Phi-4-mini-instruct	TAPR with PPO — Agg.	52.10	78.40

Table 11
Comparison of performance using GRPO and three PPO variants for training the TAPR on NQ and GSM8K (Accuracy %). "Con." (“Conservative”), "Mod" (“Moderate”), and "Agg." (“Aggressive”) are PPO hyperparameter settings differing mainly in learning rate and KL-divergence control values.

5.2. GRPO vs PPO

Here we show results comparing using the GRPO algorithm for our method to the PPO algorithm used in PRewrite [5]. Table 11 summarizes the results on both the GSM8K and Natural Questions datasets. We observe that GRPO reliably improves the training reward and downstream task accuracy for both datasets. Depending on the hyperparameters, PPO training either results in stagnant rewards or leads to model collapse, with degenerate outputs such as repetitive symbols or nonsense token.

Although prior work in PRewrite [5] showed that PPO can be effective for prompt rewriting training, differences in model architecture, hyperparameter choices, and using a critic head instead of a full critic model may contribute to the discrepancies observed here. Due to these challenges and the lack of access to the exact implementation in PRewrite [5], we opted to use GRPO for all other experiments, and couldn’t include a PRewrite baseline to compare to.

TAPR LLM	Rewriting Variant	NQ ↑	GSM8K ↑
Baseline	–	56.30	82.38
Phi-4-mini-instruct	Base Model	48.80	11.91
Phi-4-mini-instruct	SFT with generated data	52.50	35.34
Phi-4-mini-instruct	Full SFT	54.00	71.00
Phi-4-mini-instruct	TAPR	**59.20**	**83.60**
Phi-4-mini-instruct	TAPR after SFT	55.50	70.50

Table 12
Comparison of TAPR performance on the NQ and GSM8K datasets, evaluating the impact of supervised fine-tuning (SFT) with internally generated data, externally sourced data ("Full SFT"), and reinforcement learning (RL), both with and without prior SFT. Results are reported as accuracy percentages (%).

5.3. SFT impact on TAPR

In this section, we see how effective is using Supervised Fine-Tuning (SFT) prior to reinforcement learning.

We conduct SFT for 5 epochs, with data coming from various sources showing rewrites of prompts, chain-of-thought prompts and responses, and ratings to prompts given by humans [43, 44]. In addition, we instructed GPT-4o-mini [45] to generate prompts using common prompt engineering techniques for queries coming from the datasets used in the RL phase. We compare training on our internally generated dataset with training on publicly available datasets combined with our internally generated dataset. As seen in Table 12, SFT improves the base performance of Phi-4-mini-instruct, particularly when conducted with both internally generated and external data. However, this improvement does not translate into enhanced RL training outcomes.

5.4. Impact of the Prompt Quality Reward

As an additional ablation study, we evaluate the effect of using the prompt quality reward.

As shown in Table 13 and Figures 2 and 3, integrating the prompt quality reward improves both the accuracy of the downstream task and the convergence of training. In contrast, omitting this reward severely limits training progress. Without it, the model rewrites remain essentially unchanged from the initial prompts ("Answer the question." for Natural Questions and "Provide the solution." for GSM8K), indicating negligible learning.

TAPR LLM	Rewriting Variant	NQ ↑	GSM8K ↑
Baseline	–	56.30	82.38
Phi-4-mini	Base Model	48.80	11.91
Phi-4-mini	TAPR	**59.20**	**83.60**
Phi-4-mini	TAPR without PQ Reward	57.20	82.80

Table 13
Comparison of performance with and without prompt quality reward on the NQ and GSM8K datasets using Phi-4-mini-instruct as the base model (Accuracy %).

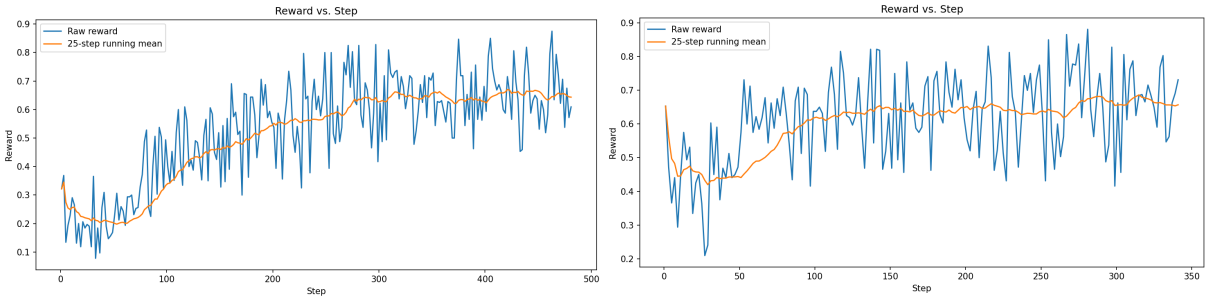

Figure 2: Training rewards comparison for the TAPR on the NQ dataset. The upper panel shows the raw per-step reward (blue) and its 25-step moving average or running mean (orange) with our method, and the bottom panel shows the training without the prompt quality reward.

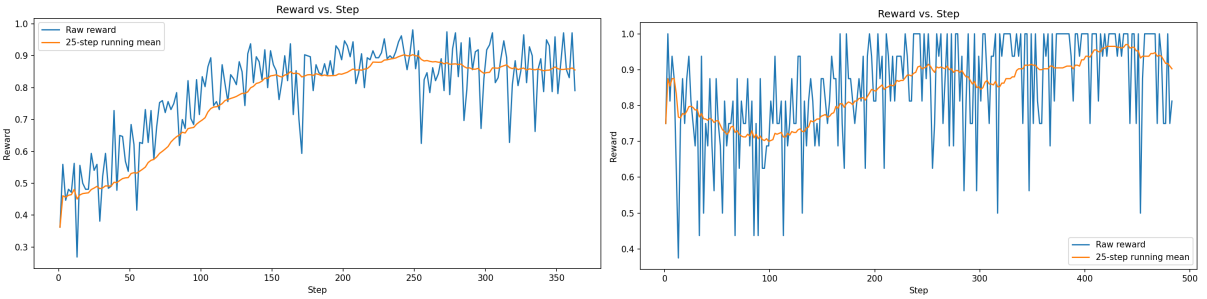

Figure 3: Training rewards comparison for the TAPR on the GSM8K dataset. The upper panel shows the raw per-step reward (blue) and its 25-step moving average or running mean (orange) with our method, and the bottom panel shows the training without the prompt quality reward.

6. Discussion

6.1. Performance Discussion

Our results indicate that, although the TAPR method can produce improved prompt rewrites and boost performance, these gains are not always consistent. In several cases, even simple or generic prompts achieve competitive or better task results than more highly optimized rewrites. This underscores the difficulty of reinforcement learning for prompt engineering: learning a rewriting policy based on rewards that do not directly reference the Task LLM's outputs is inherently challenging.

While the prompt quality reward correlates with better training rewards and often improved task scores, as seen in Section 5.4, much of the reward improvement may reflect the model's ability to satisfy prompt quality criteria rather than the end-task metrics. Thus, increases in training reward do not always translate directly into downstream accuracy gains.

Prompt engineering is, by nature, a challenging problem. The relationship between prompt structure and LLM performance is complex and sometimes unpredictable. During RL training, models can receive mixed signals: for some samples, even a suboptimal or unrelated prompt may elicit a correct answer, while a carefully crafted prompt might not.

Further complicating matters, RL for text generation is inherently unstable. Since the reward is based on the entire output rather than single decisions, training is highly sensitive to hyperparameter choices. As a result, identifying robust configurations can require significant trial and error. In our experiments,

this issue was apparent for LLaMA-3.2-3B-Instruct, likely due to the hyperparameters being optimized based on training Phi-4-mini-instruct. In summary, while the Task-Aware Prompt Rewriter approach shows promise, reliably optimizing prompts through RL remains an open challenge.

6.2. Prompt Selection Mechanism

The Selection mechanism sometimes improves over basic TAPR outputs, but these gains are inconsistent. As a result, we find no strong evidence that training for prompt rewriting consistently enhances a model's ability to judge and select among candidate prompts. This is likely due to the fact that our training only improves performance with a specific prompt without intending to improve any other abilities. Given the additional computational cost, 0-temperature generation may be preferable for simplicity, despite the risk of a single failed prompt affecting all outputs. For instance, in one Phi-4-mini-instruct training run on CNN/Daily Mail, test-time selection modestly outperformed the base model, yet 0-temperature generation produced a prompt such as "Could you tell me a light-hearted, family-friendly joke, preferably related to technology or everyday life?", a complete mismatch that caused the Task LLM to generate jokes rather than summaries for all inputs.

6.3. LLM-as-a-Judge Evaluation

Although LLM-as-a-judge evaluation is central to this work, we acknowledge its limitations and potential biases. For example, as observed with GPT-4o-mini in Section 5.1, the model exhibited a strong preference for whichever response was presented second. Still, we argue that this approach offers a more realistic assessment of open-ended generation tasks like question answering and summarization, compared to overlap-based metrics. Appendix Table 14 shows that exact-match accuracy can severely underestimate the quality of model outputs, while manual inspection confirms many responses deemed incorrect by traditional metrics are actually valid and possibly preferred due to natural phrasing or richer explanations. Despite the higher cost, we therefore recommend LLM-as-a-judge scoring as a valuable complement to traditional metrics and human evaluation for future work on LLM fine-tuning and benchmarking.

6.4. Future Work

The results and analysis in this work only partially capture the true effectiveness of the TAPR method. Both the datasets and prompts considered represent a narrow look at how people interact with LLMs in real-world scenarios. In practice, many user queries are more diverse, nuanced, and less well-formed than those found in common benchmarks. As such, small changes to questions or instructions on familiar datasets may yield limited improvements, while automatic prompt optimization could provide even greater benefits when adapting to users' unique, everyday needs. This is partially supported by our finding that rewritten prompts tended to be clearer, more detailed, and more consistent with prompt engineering best practices than the baseline instructions, suggesting that our approach may have more potential not fully reflected by our evaluation metrics. In the future, a more robust test would be to deploy this method in real agentic workflows, where crafting effective prompts is a central challenge, and directly compare the outcomes achieved by human-written prompts versus automatically optimized prompts.

Our study was limited to smaller, non-reasoning models. Although scaling up to larger models is computationally expensive, it is also plausible that larger LLMs, with their broader knowledge and

enhanced capabilities, could be more effective at both generating and benefiting from optimized prompts, particularly in the context of full-prompt rewriting where information loss is a concern. However, using advanced reasoning models that already employ chain-of-thought or step-by-step prompting may require specialized evaluation protocols and different prompt objectives to achieve maximum benefit.

Finally, one motivation for this approach was to help close the gap between small and large LLMs by maximizing the effectiveness of less capable models through better prompt engineering. If successful, this would enable cost savings by achieving similar downstream performance with less expensive models. However, our results show that the gains from prompt rewriting are not yet consistent enough to offset the additional training and inference costs. More detailed cost–benefit analyses, as well as further method refinements, will be valuable directions for future work.

7. Conclusion

In this work, we set out to address how to create a task-aware prompt rewriter model that effectively enhances LLM performance. Our findings demonstrate that a smaller model can be successfully trained through reinforcement learning to rewrite prompts that consistently improve downstream LLM outputs. Specifically, we showed that training with Group Relative Policy Optimization (GRPO) yields stable learning. Applying this trained TAPR model resulted in performance gains across various benchmarks. Notably, Phi-4-mini-instruct-based TAPR variants consistently outperformed baseline and base model prompts, delivering improved accuracy alongside qualitatively clearer, more detailed prompts adhering closely to established prompt engineering principles. Our ablation study revealed how the incorporation of LLM-as-a-judge evaluations, particularly the prompt quality reward, enhanced the training process, guiding the model towards clearer and more effective rewrites. We also observed LLM-as-a-judge evaluations to be more accurate than many traditional metrics.

Despite these promising results, our approach does have limitations, including variability in improvements across tasks and models, potential biases inherent to LLM-based reward evaluations, and increased computational demands. Future research should focus on refining the RL training and scaling to more diverse tasks and realistic user scenarios.

Overall, this work demonstrates the viability of automated prompt rewriting as a practical and effective technique for unlocking the full potential of LLMs, reducing the reliance on manual prompt engineering.

Acknowledgments

Thanks to the developers of ACM consolidated LaTeX styles https://github.com/borisveytsman/acmart and to the developers of Elsevier updated LaTeX templates https://www.ctan.org/tex-archive/macros/latex/contrib/els-cas-templates.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT, Grammarly in order to: Grammar and spelling check, Paraphrase and reword, Improve writing style, Formatting assistance. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in Neural Information Processing Systems 33 (2020).

[2] J. Wei, Y. Tay, R. Bommasani, *et al.*, Emergent abilities of large language models, arXiv preprint arXiv:2206.07682 (2022). URL: https://arxiv.org/abs/2206.07682. arXiv:2206.07682.

[3] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, A. Kluska, A. Lewkowycz, *et al.*, Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2022. URL: https://arxiv.org/abs/2206.04615. arXiv:2206.04615.

[4] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, S. Singh, Autoprompt: Eliciting knowledge from language models with automatically generated prompts, 2020. URL: https://arxiv.org/abs/2010.15980. arXiv:2010.15980.

[5] W. Kong, S. A. Hombaiah, M. Zhang, Q. Mei, M. Bendersky, Prewrite: Prompt rewriting with reinforcement learning, 2024. URL: https://arxiv.org/abs/2401.08189. arXiv:2401.08189.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017. URL: https://arxiv.org/abs/1707.06347. arXiv:1707.06347.

[7] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, D. Guo, Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL: https://arxiv.org/abs/2402.03300. arXiv:2402.03300.

[8] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le, Finetuned language models are zero-shot learners, in: Proceedings of the 10th International Conference on Learning Representations (ICLR), 2022.

[9] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, D. Amodei, Deep reinforcement learning from human preferences, in: Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, pp. 4299–4307.

[10] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe, Training language models to follow instructions with human

feedback, in: Advances in Neural Information Processing Systems 35 (NeurIPS 2022), 2022, pp. 27730–27744.
[11] DeepSeek-AI, Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL: https://arxiv.org/abs/2501.12948. arXiv:2501.12948.
[12] B. Jin, H. Zeng, Z. Yue, J. Yoon, S. Arik, D. Wang, H. Zamani, J. Han, Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL: https://arxiv.org/abs/2503.09516. arXiv:2503.09516.
[13] L. Boonstra, Prompt engineering, Google 5-Day GenAI Intensive Whitepapers-to-Text, Kaggle, 2025. https://www.kaggle.com/code/toddgardiner/google-5-day-genai-intensive-whitepapers-to-text/output.
[14] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large language models are zero-shot reasoners, 2023. URL: https://arxiv.org/abs/2205.11916. arXiv:2205.11916.
[15] J. Wei, et al., Chain-of-thought prompting elicits reasoning in large language models, arXiv preprint arXiv:2201.11903 (2022).
[16] T. Shen, T. Lei, R. Barzilay, T. Jaakkola, Style transfer from non-parallel text by cross-alignment, in: Advances in Neural Information Processing Systems, 2017.
[17] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, E. P. Xing, Toward controlled generation of text, in: Proceedings of the 34th International Conference on Machine Learning (ICML), 2017.
[18] D. X. Long, H. N. Ngoc, T. Sim, H. Dao, S. Joty, K. Kawaguchi, N. F. Chen, M.-Y. Kan, Llms are biased towards output formats! systematically evaluating and mitigating output format bias of llms, 2024. URL: https://arxiv.org/abs/2408.08656. arXiv:2408.08656.
[19] C. Fernando, D. S. Banarse, H. Michalewski, S. Osindero, T. Rocktäschel, Promptbreeder: Self-referential self-improvement via prompt evolution, in: Proceedings of the 41st International Conference on Machine Learning (ICML), 2024, pp. 13481–13544.
[20] M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. P. Xing, Z. Hu, Rlprompt: Optimizing discrete text prompts with reinforcement learning, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2022, pp. 3369–3391.
[21] T. Zhang, X. Wang, D. Zhou, D. Schuurmans, J. E. Gonzalez, Tempera: Test-time prompting via reinforcement learning, 2022. URL: https://arxiv.org/abs/2211.11890. arXiv:2211.11890.
[22] Google, Palm 2 technical report, PaLM 2 Technical Report, 2023. URL: https://doi.org/10.48550/arXiv.2305.10403, arXiv preprint arXiv:2305.10403.
[23] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2002.
[24] C.-Y. Lin, Rouge: A package for automatic evaluation of summaries, in: Text Summarization Branches Out, ACL Workshop, 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013/.
[25] S. Banerjee, A. Lavie, Meteor: An automatic metric for mt evaluation with improved correlation with human judgments, in: Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Ann Arbor, Michigan, 2005, pp. 65–72.
[26] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, Y. Artzi, Bertscore: Evaluating text generation with bert, 2020. URL: https://arxiv.org/abs/1904.09675. arXiv:1904.09675.
[27] H. Saadany, C. Orăsan, Bleu, meteor, bertscore: Evaluation of metrics performance in assessing critical translation errors in sentiment-oriented text, in: Proceedings of the Translation and Interpreting Technology Online Conference TRITON 2021, TRITON 2021, INCOMA Ltd. Shoumen, BULGARIA, 2021, p. 48–56. URL: http://dx.doi.org/10.26615/978-954-452-071-7_006.

doi:10.26615/978-954-452-071-7_006.
[28] L. Zheng, W.-L. Chiang, Y. Sheng, et al., Judging llm-as-a-judge with mt-bench and chatbot arena, in: NeurIPS, 2023.
[29] J. Gu, X. Jiang, Z. Shi, H. Tan, X. Zhai, C. Xu, W. Li, Y. Shen, S. Ma, H. Liu, S. Wang, K. Zhang, Y. Wang, W. Gao, L. Ni, J. Guo, A survey on llm-as-a-judge, arXiv preprint arXiv:2411.15594 (2024).
[30] H. Lee, S. Phatale, H. Mansoor, T. Mesnard, J. Ferret, K. Lu, C. Bishop, E. Hall, V. Carbune, A. Rastogi, S. Prakash, Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback, 2024. URL: https://arxiv.org/abs/2309.00267. arXiv:2309.00267.
[31] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, et al., Constitutional ai: Harmlessness from ai feedback, arXiv preprint arXiv:2212.08073 (2022).
[32] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, S. Petrov, Natural questions: A benchmark for question answering research, Transactions of the Association for Computational Linguistics 7 (2019) 452–466. URL: https://aclanthology.org/Q19-1026/. doi:10.1162/tacl_a_00276.
[33] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, C. D. Manning, HotpotQA: A dataset for diverse, explainable multi-hop question answering, in: E. Riloff, D. Chiang, J. Hockenmaier, J. Tsujii (Eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 2369–2380. URL: https://aclanthology.org/D18-1259/. doi:10.18653/v1/D18-1259.
[34] K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, P. Blunsom, Teaching machines to read and comprehend, 2015. URL: https://arxiv.org/abs/1506.03340. arXiv:1506.03340.
[35] I. Cachola, K. Lo, A. Cohan, D. Weld, TLDR: Extreme summarization of scientific documents, in: T. Cohn, Y. He, Y. Liu (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, Online, 2020, pp. 4766–4777. URL: https://aclanthology.org/2020.findings-emnlp.428/. doi:10.18653/v1/2020.findings-emnlp.428.
[36] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, J. Schulman, Training verifiers to solve math word problems, 2021. URL: https://arxiv.org/abs/2110.14168. doi:10.48550/arXiv.2110.14168. arXiv:2110.14168.
[37] W. Li, X. Wang, W. Li, B. Jin, A survey of automatic prompt engineering: An optimization perspective, arXiv preprint arXiv:2502.11560 (2025). URL: https://arxiv.org/abs/2502.11560.
[38] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, C. Finn, Direct preference optimization: Your language model is secretly a reward model, 2024. URL: https://arxiv.org/abs/2305.18290. arXiv:2305.18290.
[39] M. AI, The llama 3 herd of models, 2024. URL: https://arxiv.org/abs/2407.21783. arXiv:2407.21783.
[40] Microsoft Azure AI Services Blog, Introducing phi-4: Microsoft's newest small language model specializing in complex reasoning, 2025. URL: https://techcommunity.microsoft.com/blog/aiplatformblog/introducing-phi-4-microsoft%E2%80%99s-newest-small-language-model-specializing-in-comple/4357090, microsoft Tech Community Blog.
[41] Meta AI, Llama 3.2: Revolutionizing edge ai and vision with open, customizable models, 2024. URL: https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/, meta AI Blog.
[42] Q. Team, Qwen3 technical report, 2025. URL: https://arxiv.org/abs/2505.09388.

arXiv:2505.09388.

[43] Data-is-Better-Together Collective, 10k prompts ranked, 2024. URL: https://huggingface.co/datasets/data-is-better-together/10k_prompts_ranked, accessed: 2025-06-19.

[44] R. Kundu, Prompt optimization dataset, 2025. URL: https://huggingface.co/datasets/rishavkundu/prompt_optimization_dataset, accessed: 2025-06-19.

[45] OpenAI, GPT-4o mini: advancing cost-efficient intelligence, https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, 2024. Accessed: 2025-06-19.

[46] Hugging Face, Trl: Transformer reinforcement learning, https://huggingface.co/docs/trl/en/index, 2025. Published January 28, 2025; Accessed: 2025-06-19.

[47] Z. Liu, C. Chen, W. Li, P. Qi, T. Pang, C. Du, W. S. Lee, M. Lin, Understanding r1-zero-like training: A critical perspective, 2025. URL: https://arxiv.org/abs/2503.20783. arXiv:2503.20783.

A. Prompts Used in Training and Evaluation

A.1. Meta-prompt for Instruction-Only TAPR Training

```
Rewrite the following instruction by rephrasing and/or adding specific requirements to
    ensure the best possible chance for another LLM to answer the query correctly. Use
    illustrative descriptions if needed. Don't add details that change the meaning of
    the instruction. Output your response as a JSON object with two keys: "explanation"
    and "final_rewritten_query". The "explanation" key should contain your reasoning for
     the changes you made, and the "final_rewritten_query" key should contain only the
    final rewritten instruction.

Here's a demonstrative example:
1. Original instruction: The old instruction
New Instruction: {"explanation": "Explanation of why this kind of rewriting was done", "
    final_rewritten_query": "The rewritten instruction."}

Now write the new instruction. Respond only with valid JSON. Do not write an
    introduction or summary.

Original instruction: "Answer the question"

New Instruction: {
```

A.2. LLM-as-a-Judge Prompt for Question Answering

```
You are an AI assistant specialized in judging whether a given response correctly
    answers the user's query based on the reference answer(s).
Analyze [QUERY], the candidate [RESPONSE], and the list of acceptable [REFERENCE]
    answers, then decide if the response contains a correct answer.

Guidelines
1. Match at least one reference answer exactly or via an unambiguous synonym.
2. Ignore extra, irrelevant detail; focus on whether the core answer is present.
3. Output 1 for correct, 0 for incorrect.

Return a JSON object:
{
  "explanation": "...short justification...",
  "score": 1 | 0
}

Examples

[QUERY]     : Who wrote "Pride and Prejudice"?
[RESPONSE]  : Jane Austen wrote "Pride and Prejudice" in 1813.
[REFERENCE] : ["Jane Austen"]
{
  "explanation": "Mentions Jane Austen, which matches the reference exactly.",
```

```
  "score": 1
}

(3 more examples)

Now evaluate:
[QUERY]    : "{query}"
[RESPONSE] : "{final_answer}"
[REFERENCE]: "{target_answers}"
```

B. Hyperparameters

We conduct the GRPO training using the TRL library [46], applying the hyperparameter settings described here. To stabilize learning and prevent the policy from exploiting sequence-length benefits, we adopt the *Dr. GRPO* loss formulation and mask truncated completions so that only fully generated samples contribute to the gradient [47]. Dropout is disabled to ensure that the reference model produces deterministic log-probabilities for the KL divergence term, therefore reducing variance in the penalty. We generate four completions per prompt using sampling with maximum temperature $T = 1.0$, a top-k of 40, and nucleus sampling $p = 0.95$ to increase exploration. A warm-up ratio of 5 % followed by cosine learning-rate decay prevents early optimization spikes, while a weight decay of 0.01 and global gradient clipping at 1.0 mitigate overfitting and exploding gradients. The learning rate of 1e-5 does not lead to a loss of capabilities while still allowing fairly quick convergence.

Each update comprises two GRPO iterations per batch and employs clipped-advantage bounds $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$, maintaining learning momentum without catastrophic policy drift. Training in bfloat16 precision and 4-bit quantization with LoRA adapters enables us to conduct training with limited compute. These hyperparameter choices are informed by previous experimentation and literature mentioned in Section 2.1, striking a balance between stability and exploration to preserve the model's text generation abilities while enabling effective prompt optimization. Further details can be seen in our codebase.

C. Additional Experimental Results on Natural Questions

In Table 14, we present results with another LLM in Qwen3-4B on the Natural Questions dataset. These results show how using Qwen3-4B as the base model did not prove to be as effective for our method as the other LLMs, signaling the importance of optimizing hyperparameters for each different setup. We also include F1, ROUGE, and exact-match accuracy scores to show how unreliable they can be compared to another LLM judging.

D. Experimental Results - Full-Prompt Rewriting

Here, we introduce *full-prompt rewriting*, where the TAPR LLM rewrites both the instruction and the specific input (e.g., the question). Therefore, both during training and evaluation, the model rewrites the default prompt for each sample. We observed that rewriting very long inputs can sometimes hurt performance; therefore, we limit full-prompt rewriting experiments to the two datasets with relatively

TAPR LLM	Accuracy (%)	F1	ROUGE	Accuracy LLM judge(%)
Baseline prompt	0.10	0.09	0.09	56.30
Phi-4-mini-instruct	0.00	0.02	0.02	48.80
Phi-4-mini-instruct TAPR	0.00	0.06	0.09	**59.20**
Phi-4-mini-instruct TAPR + Selection	0.00	0.07	0.08	57.30
Qwen3-4B	0.00	0.06	0.07	**53.90**
Qwen3-4B TAPR	0.00	0.07	0.08	44.40
Qwen3-4B TAPR + Selection	0.00	0.08	0.10	47.35
LLaMA 3.2 3B-instruct	0.00	0.03	0.04	58.30
LLaMA 3.2 3B-instruct TAPR	0.00	0.06	0.07	**62.20**
LLaMA 3.2 3B-instruct TAPR + Selection	0.00	0.06	0.07	59.10

Table 14
NQ dataset full results, including other metrics such as exact-match and including results from the Qwen models

short inputs in Natural Questions and GSM8K. Furthermore, for full-prompt rewriting with the selection mechanism, we generate only 3 candidate rewrites per example to reduce computational overhead.

As seen in Table 15, we observe that full-prompt rewriting generally results in weaker performance compared to optimizing a single instruction for the entire dataset. This may be due to the loss of important information during the rewriting process, even though we tried preserving the original prompt within the full input that the Task LLM receives. Nevertheless, sample-level full-prompt rewriting can offer greater robustness. If a single rewritten prompt is suboptimal, it only affects one sample, whereas a poor general prompt can negatively impact all predictions for a dataset or task. As it is a more complex rewriting task than rewriting one prompt per dataset, we hypothesize that a more capable LLM might be more suitable for this purpose.

TAPR LLM	Rewriting Variant	NQ ↑	GSM8K ↑
Baseline prompt	–	**56.30**	82.38
Phi-4-mini	Base Model	49.70	83.00
Phi-4-mini	TAPR	53.95	77.60
Phi-4-mini	TAPR + Selection	53.16	78.20
Qwen3-4B	Base Model	53.71	81.30
Qwen3-4B	TAPR	49.65	79.50
Qwen3-4B	TAPR + Selection	51.44	**84.05**
LLaMA-3.2-3B	Base Model	53.15	79.00
LLaMA-3.2-3B	TAPR	51.10	79.60
LLaMA-3.2-3B	TAPR + Selection	49.35	77.70

Table 15
Comparison of full-prompt rewriting accuracy across the NQ and GSM8K datasets. Values are accuracy percentages measured by GPT-4o-mini, reflecting answer correctness for each prompt variant.

E. Experimental Results - Alternative Task LLMs

To assess the generality of our method, we evaluate TAPR performance with alternative Task LLMs, specifically Phi-4-mini-instruct and GPT-4o-mini. Results are presented in Table 16. They show how TAPR can work with other Task LLMs, most noticeably GSM8K performance with Phi-4-mini-instruct as the Task LLM goes from about 6% accuracy to around 35% with TAPR prompts. With a more capable Task LLM in GPT-4o-mini, the best results also come from the TAPR + Selection variant.

TAPR LLM	Task LLM	Variant	NQ ↑	GSM8K ↑
Baseline	Phi-4-mini	–	31.10	18.90
Phi-4-mini	Phi-4-mini	Base Model	**33.80**	6.10
Phi-4-mini	Phi-4-mini	TAPR	31.70	34.90
Phi-4-mini	Phi-4-mini	TAPR + Sel.	32.90	**35.00**
Baseline	GPT-4o-mini	–	63.16	77.56
LLaMA-3.2-3B	GPT-4o-mini	Base Model	64.23	59.00
LLaMA-3.2-3B	GPT-4o-mini	TAPR	64.00	86.40
LLaMA-3.2-3B	GPT-4o-mini	TAPR + Sel.	**64.50**	**88.40**

Table 16
Prompt rewriting performance with alternative Task LLMs (Accuracy %). Results are shown for NQ and GSM8K.

F. Experimental Results - Generalization

To assess how task-aware our method is, we show results on how training on one dataset relates to performance on other datasets. Figure 4 illustrate cross-dataset generalization through heatmaps, where each cell reports accuracy when a TAPR is trained on the row's dataset and evaluated on the column's dataset. We see that while training on the specific dataset mostly leads to the best performance, prompt rewriting ability does often transfer from one dataset to another, with training on NQ and GSM8K datasets leading to the most gains. In Table 17 we report results from training on multiple datasets, and see that while it is less effective than single dataset training, it still leads to some gains on non-summarization datasets.

TAPR LLM	Variant	NQ	Hotpot	CNN/DM	Sci	GSM
Baseline	Phi-4-mini	56.30	53.20	3.373	3.182	82.38
Phi-4-mini-instruct	Base Model	48.80	53.20	**3.782**	**3.802**	11.91
Phi-4-mini-instruct	Multi TAPR	**57.90**	**56.30**	3.362	3.068	**82.70**

Table 17
Performance of Phi-4-mini-instruct under TAPR training across multiple tasks. "Base Model" reports the untrained prompt performance. "Multi-dataset TAPR" corresponds to training on SciTLDR, GSM8K, and NQ jointly, with evaluation performed independently on each benchmark. Question answering and GSM8K results are reported as accuracy percentages, while CNN/DM and SciTLDR summarization scores are reported on a 1–5 scale. Best results within each column are shown in bold.

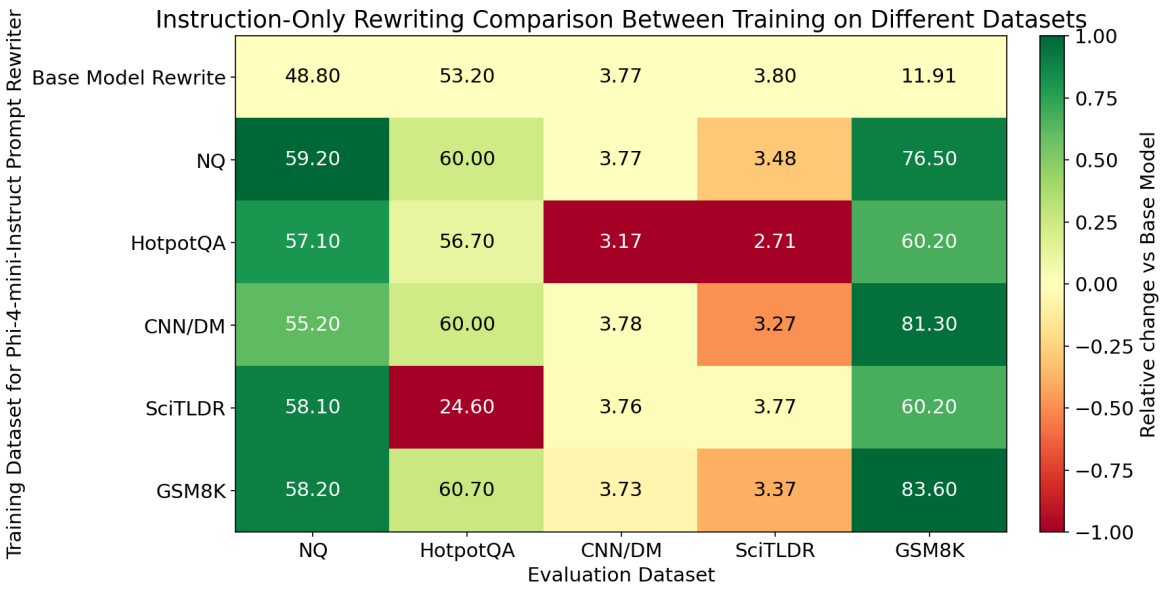

Figure 4: Cross-dataset generalization of TAPR. Each cell reports the accuracy when Phi-4-mini-instruct is trained on the dataset that the row is named after, and evaluated on the dataset that the column is named after. Summarization scores remain on a 1–5 scale, while QA and GSM8K are percentages. Cell shading encodes change relative to the base prompt (top row), with green indicating higher accuracy and red indicating lower, and each column's colormap is scaled independently.