Query-Dependent Prompt Evaluation and Optimization with Offline Inverse RL

Hao Sun[*], **Alihan Hüyük, Mihaela van der Schaar**
DAMTP, University of Cambridge

Abstract

In this study, we aim to enhance the arithmetic reasoning ability of Large Language Models (LLMs) through zero-shot prompt optimization. We identify a previously overlooked objective of query dependency in such optimization and elucidate two ensuing challenges that impede the successful and economical design of prompt optimization techniques. We introduce Prompt-OIRL, which harnesses offline inverse reinforcement learning to draw insights from offline prompting demonstration data. Such data exists as by-products when diverse prompts are benchmarked on open-accessible datasets. With Prompt-OIRL, the query-dependent prompt optimization objective is achieved by first learning an offline reward model. This model can evaluate any query-prompt pairs without accessing LLMs. Subsequently, a best-of-N strategy is deployed to recommend the optimal prompt. Our experimental evaluations across various LLM scales and arithmetic reasoning datasets underscore both the efficacy and economic viability of the proposed approach.

1 Introduction

Recent advances in Large Language Models (LLMs), such as ChatGPT [1], underscore the value of aligning with human preferences [2]. That said, even for state-of-the-art general-purpose LLMs such as GPT-4, solving intricate problems such as *arithmetic reasoning* can still be challenging. Further unlocking LLMs' full potential for those complex tasks remains an open challenge.

Out of the many attempts, *prompting* — a natural language prefix or instruction that explains how to complete the task — stands out as a lightweight promising solution for eliciting the capabilities of LLMs without model parameter tuning [3]. While the advances in zero-shot prompting strategies [4] highlight the potential in finding effective query-independent solutions, its reliance on manual crafting efforts and the vast search space over natural language intensifies the difficulty in discovering effective prompts [5]. This study aims to augment the arithmetic reasoning capacities of LLMs, a capability considered crucial [6, 7]. While there is a wealth of both expert-devised and machine-generated prompts in this domain [4, 8, 9], our endeavor is to advance further via prompt optimization. Our approach begins with the articulation of an adjusted objective, subsequently highlighting two inherent challenges.

Adjusted Objective: Query-Dependent Prompt Optimization. In the literature, research on zero-shot prompting mainly focused on finding prompts that work better on a distributional level rather than an instance level. For instance, while multi-agent debate Liang et al. [10], Du et al. [11] can improve reasoning in general, there are cases where not using a prompt can be even better than using generally effective prompts. Motivated by the fact that no prompt is perfect for all queries, in this work, we set an adjusted objective of query-dependent prompt optimization, rather than a distributional-level prompt optimization.

[*]hs789@cam.ac.uk; An extended version of this paper is available at this link; Code is available at GitHub.

37th Conference on Neural Information Processing Systems Efficient Natural Language and Speech Processing Workshop (NeurIPS 2023).

Challenge 1: Inference Time Evaluation is Hard. A primary challenge arises in evaluating the effectiveness of prompts during inference, especially when the true answer to a query is not known a priori. For arithmetic reasoning, enumerating many *potentially effective* prompts and enquire language models with those prompted queries will only result in a batch of answers, yet determining which response is correct in the absence of a ground-truth label requires extra non-trivial effort. Additionally, the computational cost associated with this prompt enumeration process is substantial.

Challenge 2: Online Prompt Evaluation and Optimization is Expensive. On the other hand, searching for *potentially effective* prompts heavily relies on expensive online evaluation. In the literature, evaluating the effectiveness of a proposed prompt requires assessing its performance on multiple datasets and LLMs, to show its *distributional* superiority over the others. In the arithmetic reasoning task, using the GPT-3.5-turbo API [2] to evaluate a single prompt on a medium-sized dataset with 10k query-answer pairs will lead to an ≈$1 cost, yet learning from trials and errors often requires millions of interactions even for tasks with a few actions to choose from [12]. The vast action space of natural language further increases the prohibitive cost for such interactive search [13, 5].

Solution: Query-Dependent Prompt Evaluation and Optimization with Offline Inverse RL. Having witnessed the significant advancements achieved in expert-crafted and algorithm-discovered prompting in arithmetic reasoning [8, 14, 4], our work seeks a solution of combining existing human knowledge in prompting systematically, effectively, and cost-efficiently.

2 Query-Dependent Prompting Optimization

We first introduce preliminaries to provide a formal definition of the query-dependent prompting problem. We then introduce our solution.

Objective Given a dataset $\mathcal{D} = \{x^{(i)}, y^{*(i)})\}_{i\in[N]}$ of queries and their expected answers, the objective of *query-agnostic* zero-shot prompt optimization usually studied in the literature is to find the *distributional* optimal prompting strategy $\bar{\pi}^*$ that maximize the *expected* quality of answers w.r.t. metric r by feeding prompted queries $\pi(x)$ to the language model ℓ and getting answers $\hat{y} = \ell(\pi(x))$:

$$\bar{\pi}^* = \arg\max_{\pi} \mathbb{E}_{(x^{(i)},y^{*(i)})\sim\mathcal{D}} \left[r(y^{*(i)}, \ell(\pi(x^{(i)}))) \right], \tag{1}$$

Instead, in this work we meet the **Adjusted Objective** by using a query-dependent approach:

$$\pi^* = \arg\max_{\pi} r(y^{*(i)}, \ell(\pi(x^{(i)}))), \tag{2}$$

In a nutshell, Equation(1) seeks a single prompting strategy that achieves good performance on the dataset, yet Equation(2) seeks different prompts for different queries in the dataset. Clearly, π^* should be better than $\bar{\pi}^*$ in the sense

$$\mathbb{E}_{(x^{(i)},y^{*(i)})\sim\mathcal{D}} \left[r(y^{*(i)}, \ell(\pi^*(x^{(i)}))) \right] \geq \mathbb{E}_{(x^{(i)},y^{*(i)})\sim\mathcal{D}} \left[r(y^{*(i)}, \ell(\bar{\pi}^*(x^{(i)}))) \right]. \tag{3}$$

Step 1: Existence of Offline Prompt Demonstrations We start by emphasizing the presence and significance of prompt demonstrations used in benchmarking existing prompts. In the realm of research dedicated to improving the arithmetic reasoning capacities of LLMs, several zero-shot prompts have been proposed. Notable examples include CoT [4], APE [8], and a concurrent work OPRO [9], among others.

It's important to highlight that none of these prompts uniformly enhance the LLMs' arithmetic reasoning capabilities. Their efficacy is assessed at a *distributional level*, as per Equation(1). Different language models might exhibit varied preferences for prompts [9].

To benchmark the effectiveness of the proposed prompts, prior researchers have utilized standardized, openly accessible arithmetic reasoning datasets and reported the overall success rate of obtaining correct answers. Denoting different existing prompts with superscripts, e.g., $\pi^{(1)}(x) = (x)$ the no prompting, $\pi^{(2)}$ = CoT prompting,

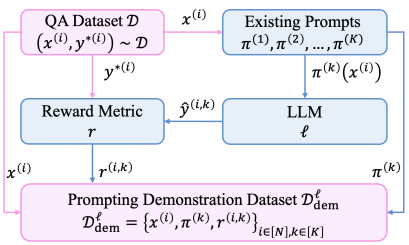

Figure 1: *The offline demonstration dataset is generated as a by-product of evaluating existing (query-agnostic) prompts.*

$\pi^{(3)}$ = APE prompting, and so on — when there are K prompts have been evaluated, the following demonstrations will be constructed as by-products:

$$\mathcal{D}^{\ell}_{\text{dem}} = \{x^{(i)}, \pi^{(k)}, r^{(i,k)} = r(y^{*(i)}, \ell(\pi^{(k)}(x^{(i)}))\}_{i\in[N],k\in[K]} \tag{4}$$

This process is visualized in Figure 1, depicting the generation of prompt-alignment demonstration datasets from evaluating existing prompts. Notably, the rewards in $\mathcal{D}^{\ell}_{\text{dem}}$ are influenced by ℓ.

Step 2: Offline Reward Modeling: Inverse RL without an Environment Upon initial inspection, it might seem that reward modeling is unnecessary given the reward metric. i.e., $r(y^*, \hat{y}) = \mathbb{1}\{\hat{y} = y^*\}$. However, a more thorough examination of this metric reveals two concerns that limit its broader utilization: first, the reward metric is a function over language model ℓ, making every single call of such a metric costly; second, the computation of the reward necessitates access to the ground truth answer y^*, which is always not available in inference (deployment) time.

To address those issues, we introduce a parameterized *proxy reward model* that is a function of query x and prompt π. We denote the proxy reward as $\Upsilon_\theta(x, \pi(x))$ with parameters θ. Notably, this proxy reward omits the language models ℓ and the ground-truth labels from its calculations. By design, its output should align with the true reward r. Therefore, we establish a supervised learning objective to minimize the discrepancies between the proxy reward model and the true reward.

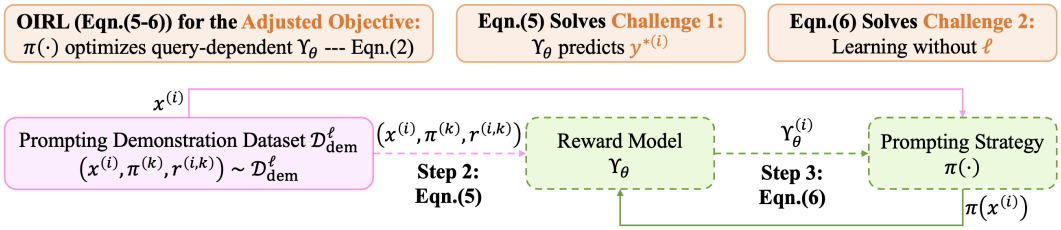

Figure 2: *Prompt-OIRL addresses the specified Objective and challenges*. It first learns a proxy reward model from the offline demonstration dataset we created in the last section. Such a learned reward model can be applied in inference time to evaluate prompts in a query-dependent manner without access to the language model, hence optimizing prompt w.r.t. such a proxy reward model solves all issues identified.

In the case of arithmetic reasoning tasks, the reward signal is binary, hence the proxy reward model can be trained as a classification task, predicting whether the prompt can result in a correct answer when fed to the language model. Specifically, we consider the Cross-Entropy loss given the demonstration data collected in the previous section:

$$\mathcal{L}_{\text{CE}}(\theta; \mathcal{D}^{\ell}_{\text{dem}}) = -\mathbb{E}_{i\in[N],k\sim[K]}\left[r^{(i,k)} \log \sigma\left(\Upsilon^{(i,k)}_\theta\right) + (1 - r^{(i,k)}) \log\left(1 - \sigma\left(\Upsilon^{(i,k)}_\theta\right)\right)\right] \tag{5}$$

where we use $\Upsilon^{(i,k)}_\theta$ to denote $\Upsilon_\theta(x^{(i)}, \pi^{(k)}(x^{(i)}))$ for conciseness, and σ is the sigmoid function. Different from the *online* reward signal provided by $r(y^*, \ell(x'))$ that requires expensive interactions with black-box LLMs, the proxy reward model $\Upsilon_\theta(x, \pi(x))$ can provide *offline* white-box feedback to prompts. Therefore, it can be more accessible in guiding the search for a better prompting strategy. Figure 2 illustrates how such a reward model can be learned and leveraged in providing feedback to prompting strategy optimization.

Step 3: Offline Prompt Optimization with the Learned Reward Model.

Given the learned reward model is an effective proxy of the performance evaluator for query-prompting strategy pairs, we would then be able to solve the prompt optimization problem of Euqation (2) with an alternative ***offline*** objective that is feasible to execute in inference-time — without the requirement of a language model ℓ and the non-accessible inference-time golden label y^*:

$$\pi^* = \arg\max_\pi \Upsilon_\theta(x, \pi(x)) \approx \arg\max_\pi r(y^*, \ell(\pi(x))) \tag{6}$$

In general, any policy optimization technique has the potential to be applied in solving Equation (6). To name a few examples, such a technique can be any of the previous approaches used in ***online*** prompt optimization like RL [5], beam search [14], evolution strategies [8].

3 Experiment

Tasks We use the tasks of MultiArith [15], GSM8K [16], SVAMP [17] in the arithmetic reasoning domain because they are widely studied in zero-shot prompting, and hence rich expert-crafted and machine-generated prompting knowledge is available. Leveraging such knowledge facilitates our offline data collection procedure. **LLMs** To demonstrate the general applicability of Prompt-OIRL, we experiment with datasets generated with LLMs at different abilities, scaling from the GPT-3.5-turbo model [1], to TigerBot-13B-chat [18], and LLaMA2-7B-chat [19]. All created offline demonstration datasets, including the query-prompt pairs, prompted answers from different LLMs, and the correctness of those answers will be released as a publicly accessible dataset. We defer detailed discussions on the usage of embeddings, data processing, and training details can be found in Appendix C.

Experiment Setup In the experiment, we use **BoTr Eqn.(1)** to denote the setting of selecting the best-of-training time prompts according to objective Eqn.(1). In this case, the prompt to be chosen in test or inference time should be the one that achieves the overall best training performance. We then consider the setting of **BoTr Eqn.(2)**, when Eqn.(2) becomes the objective. In such a setting, the proxy reward model will be used to select in the query-dependent best-of-N selection. Alternatively, we compare the LLM-confidence-based baseline in approaching Eqn.(2), which uses LLMs to reflect how confident they are for different prompted answers. **LLM Confidence** thence selects the most confident answers that correspond to training prompts according to confidence scores provided by LLMs. Finally, our method **Prompt-OIRL** selects the best prompt from both training time prompts and held-out prompts with regard to the learned reward model.

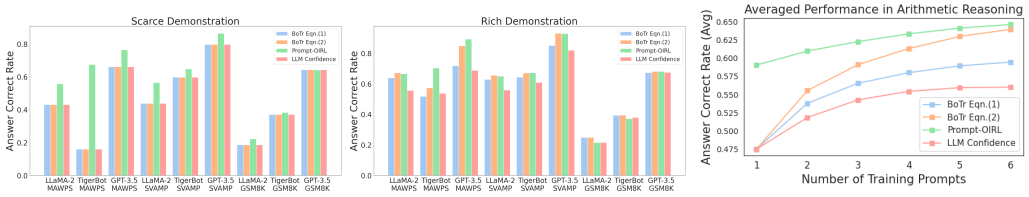

Figure 3: *First two panels: Performance of Prompt-OIRL under two typical settings.* On both settings with scarce and rich demonstration data, Prompt-OIRL achieves better performance. Last panel: The averaged performance of different methods under varying demonstration data availability in arithmetic reasoning

Results We present results on two typical settings in Figure 3 under different demonstration availability: (1) in the scarce demonstration setting (left panel) when $K = 1$, where only 1 prompt is available for reward model learning; and (2) in the rich demonstration setting (right panel) when $K = 5$, where more prompts demonstrations are available. We note when $K = 1$, there should be no difference between using Eqn.(1) and Eqn.(2) in the test-of-training strategy.

From the results, we can conclude: **(1).** in the scarce demonstration setting where demonstration data is collected with a single prompt, Prompt-OIRL significantly ($+24.3\%$) outperforms the best-of-training strategy; **(2).** in the rich demonstration settings where multiple prompts are available, Prompt-OIRL is significantly better ($+8.8\%$) than using a query-agnostic objective of Eqn.(1); **(3).** in the rich demonstration setting, Prompt-OIRL is slightly better ($+1.8\%$) than selecting the best training strategy according to the learned reward model — this is not surprising as the prompts we used for training are all tested as helpful themselves; **(4).** On both settings, Prompt-OIRL achieves significantly (by $+24.3\%$ and $+14.7\%$, separately) better results compared to the LLM confidence score-based baseline — without further interactions with the LLMs.

Additionally, we change the availability of prompts used for training and provide a performance comparison averaged over all tasks and LLMs we benchmarked in Figure 3. In general, using more prompts in training leads to better results. The results under different individual settings are provided in Appendix D.1

Take-Aways Prompt-OIRL fulfills the **Adjusted Objective** and achieves superior performance over the baselines under various demonstration data availability. When demonstration data is limited, Prompt-OIRL achieves a remarkably higher success rate in getting correct answers, and its performance can be further improved with an increase in data availability.

References

[1] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[2] R OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023.

[3] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[4] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

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

[6] OpenCompass. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`, 2023.

[7] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023.

[8] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

[9] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

[10] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

[11] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[13] Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E Gonzalez. Tempera: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[14] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023.

[15] Subhro Roy and Dan Roth. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*, 2016.

[16] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[17] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

[18] TigerResearch. Tigerbot: A cutting-edge foundation for your very own llm. `https://github.com/TigerResearch/TigerBot`, 2023.

[19] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[20] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[21] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[22] Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pages 2020–2027. PMLR, 2021.

[23] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[25] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[26] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[27] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[28] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

[29] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[30] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[31] Chi Zhang, Sanmukh Rao Kuppannagari, and Viktor Prasanna. Brac+: Going deeper with behavior regularized offline reinforcement learning. 2020.

[32] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[33] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[34] Daniel Jarrett, Alihan Hüyük, and Mihaela Van Der Schaar. Inverse decision modeling: Learning interpretable representations of behavior. In *International Conference on Machine Learning*, pages 4755–4771. PMLR, 2021.

[35] Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

[36] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[37] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[38] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[39] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[40] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34: 4028–4039, 2021.

[41] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[42] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[43] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[44] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pages 6818–6827. PMLR, 2019.

[45] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[46] Stefan Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

[47] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[48] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf`.

[49] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

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

[51] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[52] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

[53] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

[54] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[55] Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. Generated knowledge prompting for commonsense reasoning. *arXiv preprint arXiv:2110.08387*, 2021.

[56] Jieyi Long. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

[57] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[58] Dave Hulbert. Tree of knowledge: Tok aka tree of knowledge dataset for large language models llm. `https://github.com/dave1010/tree-of-thought-prompting`, 2023.

[59] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[60] Xiaonan Li and Xipeng Qiu. Mot: Pre-thinking and recalling enable chatgpt to self-improve with memory-of-thoughts. *arXiv preprint arXiv:2305.05181*, 2023.

[61] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

[62] Ryan Greene, Ted Sanders, Lilian Weng, and Arvind Neelakantan. New and improved embedding model, 2022.

[63] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[64] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[65] Rui Wang, Hongru Wang, Fei Mi, Yi Chen, Ruifeng Xu, and Kam-Fai Wong. Self-critique prompting with large language models for inductive instructions. *arXiv preprint arXiv:2305.13733*, 2023.

[66] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Ethics Statement This paper introduces the pioneering application of offline inverse reinforcement learning to query-dependent prompting, enhancing the arithmetic reasoning capabilities of LLMs. Yet, deploying such a method inappropriately could yield undesirable results. Additionally, the method's dependence on LLMs and prompting demonstration data underscores potential concerns about data privacy and consent, especially as it leans on future offline datasets for human-centered tasks.

Reproducibility Statement All code and offline demonstration datasets will be released upon acceptance. For the seek of full reproducibility, we provide supplemental experiment details in Appendix C, and attached example code in the supplementary material of the submission.

Appendix: Table of Contents

A Extended Discussion on Prompt-OIRL and RLHF

A.1 Connections and Difference from a Descriptive Perspective

To better understand our contribution, it would be helpful to link and contrast Prompt-OIRL with the framework of reinforcement learning from human feedback (RLHF) or AI feedback (RLAIF). In general, the success of RLHF and RLAIF motivates our idea of integrating human expertise in prompting, and Prompt-OIRL can be interpreted as a special type of prompt policy learning from ***offline*** AI feedback, which differentiates from existing literature.

The fact that Prompt-OIRL first learns a reward model and then performs policy optimization using the reward model makes the method related to RLHF and RLAIF. From such a perspective, Prompt-OIRL treats prompt evaluation and optimization as a process of *aligning prompting strategies with the preferences of the LLMs*, contrasting to aligning LLM responses with human preferences. In analogy with the RLHF that improves the performance of LLMs with regard to human preferences, we hypothesize that such a perspective could lead to more effective prompts while avoiding the need for random exploration of the prompt space, which is computationally expensive.

That said, there are several key differences that make Prompt-OIRL distinguishable from the existing RLHF and RLAIF literature. RLHF (or RLAIF) often includes the following steps: 1. sampling from pre-trained language models to generate dialogue data; 2. human (or AI) raters are then asked to rank those generated data according to given criteria such as harmless and helpful; 3. a preference model is trained on the ranked dataset, as proxy to the human (or AI) raters; 4. the language model is fine-tuned with the learned preference model using reinforcement learning.

By contrast, we would like to highlight the differences in objective, non-interactive offline learning problem setting, and optimization procedure flexibility: First, Objective: The objective of Prompt-OIRL is to evaluate and optimize prompting strategies, rather than enhance LLMs' alignment to non-differentiable objectives. Importantly, the learned reward model in Prompt-OIRL can work in isolation as a prompt evaluator. Second, Non-Interactive Offline Setting: Prompt-OIRL works in a purely offline setting, rather than assuming access to human or AI raters to *actively* generate feedback. Third, the optimization procedure in Prompt-OIRL is highly flexible, which is not necessary to be a language model as in RLHF and RLAIF. For instance, the prompt optimization process can collaborate with a human prompting engineer, who proposes potential prompting strategies and selects the best according to the learned reward model.

A.2 Prompt-OIRL and RLHF with the Formal RL Language.

Reinforcement Learning and Offline-RL In Reinforcement Learning (RL), an agent learns through interacting with an environment and receiving feedback in the form of rewards [20]. The fundamental objective of RL is to find a policy, which is a mapping from states to actions, that maximizes the expected cumulative reward over time.

Such a learning paradigm can be formally represented using the Markov Decision Processes (MDPs), where decisions are made in discrete time steps, and each decision affects the state of the environment in the subsequent step.

Formally, we denote the MDP as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0, \gamma\}$, where $\mathcal{S} \subset \mathbb{R}^d$ denotes the d-dim state space, $\mathcal{A}$ is the action space. Broadly, the *environment* includes $\mathcal{T}$ and $\mathcal{R}$, the former denotes the transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ that controls transitions between states, and the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ provides feedback. $\rho_0 = p(s_0) \in \Delta(\mathcal{S})$ denotes the initial state distribution. γ is the discount factor that trades off between short-term and long-term returns.

Online RL considers the problem of learning a policy $\pi \in \Pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$, such that the expected cumulative reward $\mathbb{E}_{a_t \sim \pi, s_{t+1} \sim \mathcal{T}, s_0 \sim \rho_0} \sum_{t=0}^{T} \gamma^t r_t(s_t, a_t)$ in the MDP is maximized. In the online RL setting, an agent learns through trial and error. It actively interacts with the environments — including both transition dynamics $\mathcal{T}$ and the reward function $\mathcal{R}$. And optimize its polity through either on-policy [21, 12, 22] or off-policy algorithms [23–26].

At each time step t, an agent observes a state s_t from the environment and selects an action a_t according to its policy π. Upon taking the action, the agent receives a reward r_t and transit to a new state s_{t+1}. The agent's objective is to maximize its expected return.

Figure 4: Pictorial illustration of Online RL: an agent actively interacts with the environment, which is composed of the dynamics model that controls transition, and the reward model that provides feedback.

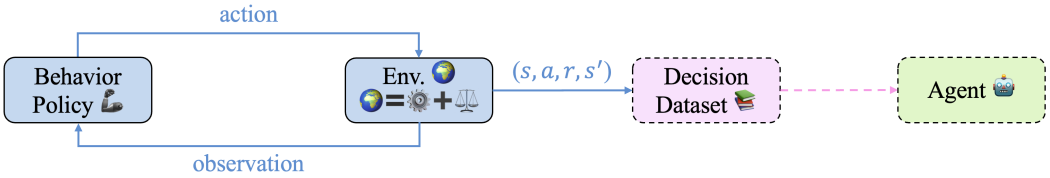

Figure 5: Pictorial illustration of Offline RL: a decision dataset is collected through interactive logs between a behavior agent and the environment. Such offline dataset is then used to optimized a parameterized policy — the learnable agent.

Offline RL, also known as batch-RL or data-driven RL, focuses on learning optimal policies from a fixed dataset of interaction data without further interaction with the environment. This approach is particularly relevant in scenarios where online interaction is expensive, risky, or impractical. The primary challenge in Offline RL is that the dataset may not sufficiently cover important regions of the state-action space, which can lead to extrapolation errors and suboptimal policies [27].

Offline RL algorithms aim to leverage this fixed dataset to derive a policy that would perform well if deployed in the real environment. Techniques often employed in Offline RL include constraining the policy to only consider data in the dataset, regularizing against drastic changes from a behavior policy, and employing uncertainty estimation to avoid regions where the dataset provides limited information [28, 29, 28, 30]. In general, such a dataset can either be generated by rolling out an expert that generates high-quality solutions to the task [27, 31, 32] or a non-expert that executes sub-optimal behaviors [32, 29, 28, 33, 34] or be a mixture of both [35].

That said, the research on existing offline-RL mainly focuses on application on dense-reward tasks like robotics control [32, 36]. The optimization for reward on the trajectory level such as the case in LLM researchwhere responses may only be evaluated on the sentence-level, rather than on a token-wise level — is relatively underexplored.

Imitation Learning, Behavior Clone, and Inverse RL Imitation learning and learning from demonstrations are widely studied in the field of reinforcement learning and robotic learning [37–40], aiming at learning policies with a batch of expert trajectories. Working as the most straightforward solution, Behavior Cloning (BC) [41, 42] optimizes a policy through supervised learning, yet may suffer from compounding error or instability in learning from sub-optimal demonstrations [43, 44]. It is shown in [45] that data quality is of great importance in the success of BC. It is worth noting that, while imitation learning assumes the underlying reward mechanism is unknown, the learning from demonstration literature always uses the demonstrations as a warm-start for reward-sparse tasks like robotics manipulation [46, 47].

LLM Alignment with Human Preference The task of LLM alignment under human preference can be framed as RL, yet the challenge of online learning. See Figure 8 for a more detailed explanation.

In practice, collecting a limited number of human feedbacks and saving them as a decision dataset is feasible. In such cases, the LLM alignment as online RL is turned into an offline RL problem. As we have discussed above, offline RL also suffers from difficulties in learning. However, there is a unique property in the LLM alignment problem that enables Imitation Learning with those alignments (human-feedback) data.

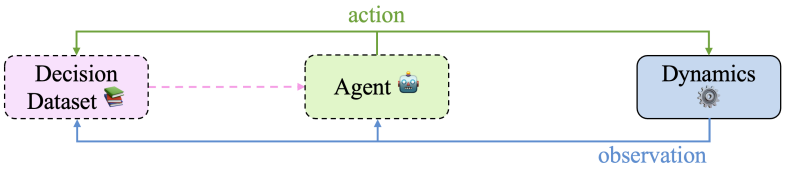

Figure 6: Pictorial illustration of IL: a dynamics model is always needed to obtain observations. The decision dataset can be used to provide a reference for the desired behavior. Learning (the pink dashed line) can be achieved by minimizing the behavior divergence between the decision dataset and the agent.

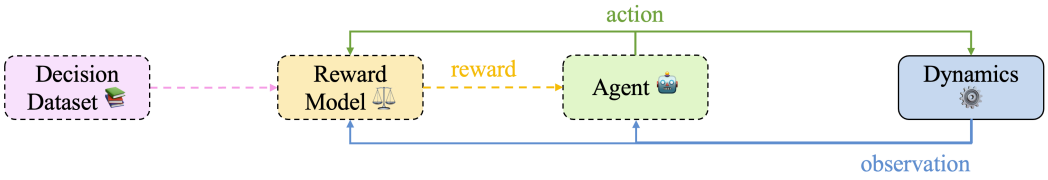

Figure 7: Pictorial illustration of IRL: Inverse RL first learns a reward model, with which the imitation learning now becomes an "online" RL problem — the accessible dynamics provide observations, and the learned reward model provides feedback.

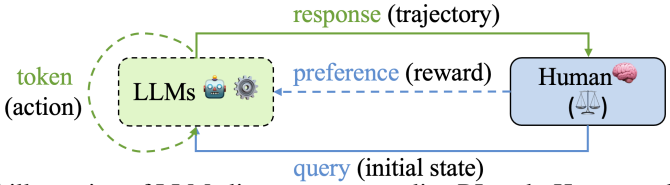

Figure 8: Pictorial illustration of LLM alignment as an online RL task. However, this is only the ideal case: it is normally unaffordable to always keep humans in the loop. Human is the reward model, and the LLMs themselves act as a dynamics model.

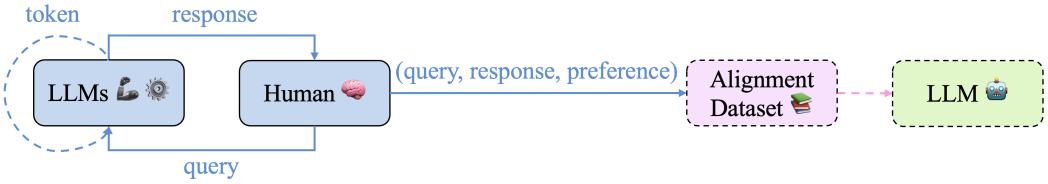

Figure 9: Pictorial illustration of LLM alignment as an offline RL task: The alignment dataset is collected through a limited number of interactions between humans and LLMs.

RLHF: Solving an Offline-RL Task with Online Inverse RL We highlight that RLHF solves the offline RL task of aligning human preferences through an offline dataset with online IRL. Figure 10 and Figure 11 illustrates such an interpretation.

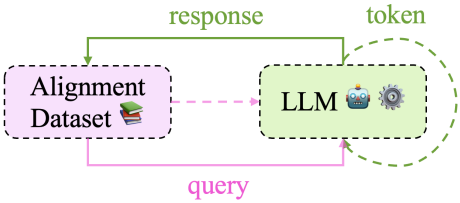

Figure 10: Pictorial illustration of RLHF as IL: as LLM itself acts as the dynamics model, the offline RL problem of LLM alignment with human preference becomes an imitation learning problem.

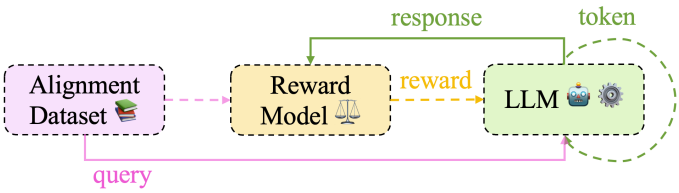

Figure 11: Pictorial illustration of RLHF as Online IRL: Approaching the imitation learning problem through IRL, RLHF first learns a reward model, with which it has both components in an environment — the dynamics model (itself) and the reward model (learned). Therefore, RLHF can apply conventional online RL algorithms like PPO as its solver.

Prompting as Policy Learning: Alignment with LLM Preference While prompt optimization is by nature an RL task and can be approached by learning from trial and error, randomly exploring the action space of natural language is infeasible. An analogy is the task of LLM alignment with human feedback: where an LLM agent interacts with human annotators to collect feedback on its responses and seek for better alignment with the human annotators' preference. In such a task, it is also unrealistic and infeasible to always put humans in the loop. Therefore, techniques of RLHF are proposed to seek the solution when only an offline alignment dataset is available.

On the other hand, prompting optimization is indeed an LLM-centric type of alignment, as opposed to human-centric alignment: it aligns the prompter's prompting strategies according to LLMs' preferences and feedback. It faces the same type of challenge that always keeping LLM in the loop during optimization is not feasible. Figure 12 provides a pictorial illustration of how prompting can be interpreted as an (expensive) online RL task.

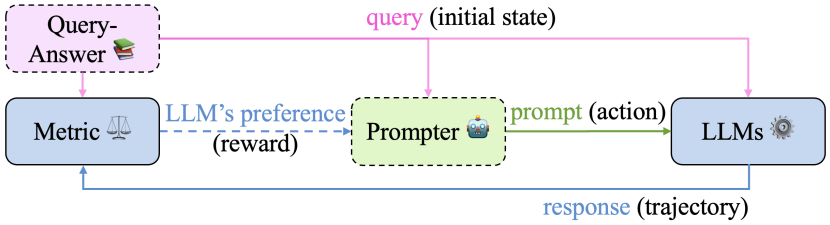

Figure 12: Pictorial illustration of prompting as an (expensive) online RL task: the reward calculation requires extensive usage of the LLM, which is expensive in practice.

Prompt-OIRL: Solving an Offline RL Task with Offline Inverse RL The key observation and claim we would emphasize is that we can interpret Prompt-OIRL as solving the offline **RL** of aligning prompting strategies from LLMs' feedback through offline **IRL**.

Offline Data as By-Products In the main text, we have highlighted the existence and importance of prompt demonstrations generated in benchmarking existing prompts. For instance, in the research on enhancing the arithmetic reasoning abilities of LLMs, multiple zero-shot prompts are proposed, such as the CoT [4], APE [8], and more recently OPRO [9], etc.

In order to benchmark the effectiveness of those proposed prompts, previous researchers use standardized open-accessible arithmetic reasoning datasets and report the overall successful rate of getting correct answers. Figure 13 provides a pictorial illustration of how the evaluation of existing prompts generates prompt-alignment demonstration datasets as its by-products.

Solving Offline RL by Offline Inverse RL The key observation of such a problem is that zero-shot prompting are single-step decision-making problem. Therefore, even without the transition dynamics models (LLMs), we are still able to conduct an inverse RL in an offline manner. Prompt-OIRL circumvents the difficulty in evaluating prompts using offline reward modeling. Figure 14 provides a pictorial illustration of such an interpretation.

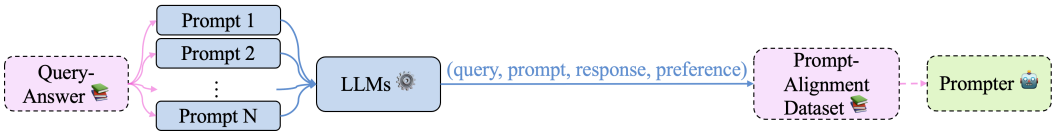

Figure 13: Pictorial illustration of prompting as an offline RL task.

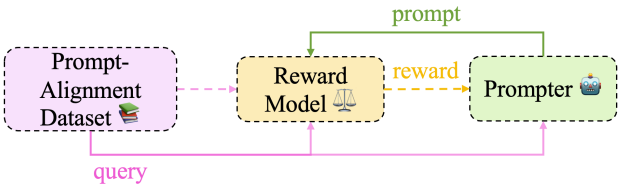

Figure 14: Pictorial illustration of prompting as an Offline IRL task: the offline prompt demonstration dataset is used to learn a reward model, as a function of query and input, to avoid the requirement of an LLM during evaluation. Prompt-OIRL then leverages such a learned reward model for prompt optimization in a purely offline manner.

Conclusive Remark RLHF solves the problem of *learning from offline LLM alignment dataset according to human feedback* by turning the problem from offline RL into an online IRL, and then an online RL after learning the reward model. Prompt-OIRL solves the problem of *learning from offline prompting dataset according to LLM feedback* by turning the problem from offline RL into an offline IRL problem, to get rid of the dependency of LLMs in evaluation and optimization.

B Extended Related Work

B.1 Learning from Human Expertise and Imitation Learning

Recent advances in Large Language Models (LLMs) have shown impressive performance as general-purpose agents, especially with the remarkable success of ChatGPT where human knowledge is injected via *reinforcement learning from human feedback* (RLHF) [48, 1, 2]. During the training of those models, human feedback is leveraged to reduce the harmfulness and at the same time improve the instruction-following ability of models.

Imitation learning and learning from demonstrations are widely studied in the field of reinforcement learning and robotic learning [37–40], aiming at learning policies with a batch of expert trajectories. Working as the most straightforward solution, Behavior Cloning (BC) [41, 42] optimizes a policy through supervised learning, yet may suffer from compounding error or instability in learning from sub-optimal demonstrations [43, 44]. It is shown in [45] that data quality is of great importance in the success of BC. It is worth noting that, while imitation learning assumes the underlying reward mechanism is unknown, the learning from demonstration literature always uses the demonstrations as a warm-start for reward-sparse tasks like robotics manipulation [46, 47].

In prompt generation tasks, the demonstrations are always imperfect, T-REX [49] extrapolates beyond a learned reward function to achieve super-demonstration performance. We build our work based on T-REX to address the difficulty of learning from imperfect demonstration. We also benchmark the performance of BC, with a specific consideration on the quality of demonstrations [44, 45].

B.2 Zero-Shot and Few-Shot Prompting

The ability of Zero-shot prompting emerges in the language models trained on large amounts of data like GPT-3 and GPT-4 [1, 2]. And it was shown in [50] that instruction-fine-tuning improves the zero-shot learning ability of language models.

Notwithstanding the impressive zero-shot performance exhibited by large language models, these models often exhibit suboptimal performance in executing more complex tasks under a zero-shot setting. Leveraging few-shot prompting presents a viable approach for facilitating in-context learning [51, 52].

This technique necessitates the inclusion of demonstrations within the prompt, effectively guiding the model toward enhanced performance. These demonstrations act as conditioning mechanisms for succeeding examples, leading the model to generate better responses.

B.3 Chain-of-Thought Prompting

In some more challenging tasks like complex arithmetic, commonsense, and symbolic reasoning tasks, the chain-of-thought (CoT) prompting is shown to be more effective in helping the language models to get correct answers [3]. CoT includes additional reasoning steps in the few-shot prompting examples. [4] further introduces zero-shot CoT, showing that adding task-agnostic instruction can improve the model performance in specific tasks. In [53], Auto-CoT combines the universality of zero-shot CoT and the capability of original CoT driven by demonstrations and proposes to automatically construct demonstrations based on clustering and diversity-based sampling that are beneficial for CoT reasoning.

B.4 Other Prompting Strategies

[54] further improves the few-shot CoT method by sampling multiple diverse reasoning paths and marginalizing those paths, choosing the most consistent answers among all sampled reasoning paths.

The Generated Knowledge Prompting [55] improves commonsense reasoning by incorporating knowledge or information related to the questions to make more accurate predictions. Tree-of-thoughts (ToT) methods [56, 57] combines tree-based planning methods with reasoning skills of language models, and solves hard reasoning problems step by step via multiple round conversations. [58] also put forward a related idea that leverages multiple thoughts of a language model in a single prompt. Memory and Retrieval Augmented Generation (RAG) [59], that is able to combine parametric memory and non-parametric memory like Wikipedia in completing knowledge-intensive tasks. MoT [60]: Pre-thinking based on the external unlabeled dataset and then recalling the related knowledge during inference. In the concurrent work of Yang et al. [9], LLMs are used as optimizers to solve a variety of optimization problems, including prompt optimization. It is worth noting the most important two differences are in the offline nature and the query-dependant prompting strategy of our method.

Table 1: Prompt-OIRL differentiates from existing literature on prompt optimization by (1) considering the **Adjusted Objective** and optimizing **query-dependent prompt**; (2) being able to perform **offline prompt evaluation** to address **Challenge 1**; (3) optimizing prompt in the **offline setting** without access to the LLMs to address **Challenge 2**; (4) **utilizing existing expert knowledge** to reduce the difficulty in RL; (5) generating **human-readable prompts**; (6) working on the most general **natural language prompt space**; (7) **free from gradient** information of LLMs; (8) using offline inverse reinforcement learning as the solver of the problem.

Method	(1) Query Dependent Prompt	(2) Offline Prompt Evaluation	(3) Offline Prompt Optimization	(4) Expert Knowledge Inspired	(5) Human Readable Prompt	(6) Prompt Space	(7) LLM Gradient Free	(8) Solver Used
Soft-Prompt	✓	✗	✗	✗	✗	Embeddings	✗	Gradient-Guided Search
APO	✗	✗	✗	✓	✓	$\mathcal{X} = \mathcal{V}^\infty$	✓	Beam Search
APE	✗	✗	✗	✗	✓	$\mathcal{X} = \mathcal{V}^\infty$	✗	Evolution Strategy
TEMPERA	✓	✗	✗	✓	✓	Edit	✓	RL
RLPrompt	✗	✗	✗	✗	✗	$\{\mathcal{V}^2, \mathcal{V}^5\}$	✓	RL
Prompt-OIRL (ours)	✓	✓	✓	✓	✓	$\mathcal{X} = \mathcal{V}^\infty$	✓	Offline Inverse-RL

Embeddings: the language of LMs; Edit: including operations like swap, delete, etc.; $\mathcal{V}$ is the vocabulary, and the superscript over it denotes the length of prompts. $\mathcal{V}^\infty$ denotes the natural language space — the most general interpretable format.

C Supplemental Experiment Details

C.1 General Setups

we present empirical evidence illustrating the efficacy of Prompt-OIRL in addressing the previously highlighted challenges and meeting the revised objective. We first outline the general experimental setups.

Tasks We use the tasks of MultiArith [15], GSM8K [16], SVAMP [17] in the arithmetic reasoning domain because they are widely studied in zero-shot prompting, and hence rich expert-crafted and machine-generated prompting knowledge is available. Leveraging such knowledge facilitates our offline data collection procedure.

Table 2: prompts used in offline training dataset collection.

No.	Effective Prompts Discovered by Experts and Algorithms	Explanation
1	"The answer is:"	direct prompting
2	"Let's think step by step:"	zero-shot CoT [4]
3	"Let's work this out in a step by step way to be sure we have the right answer:"	APE discovered [8]
4	"First, decompose the question into several sub-questions that need to be solved, and then solve each question step by step:"	Least-to-most [61]
5	"Imagine three different experts are answering this question. All experts will write down 1 step of their thinking, and then share it with the group. Then all experts will go on to the next step, etc. If any expert realizes they're wrong at any point then they leave."	Tree-of-thought [58]
6	"3 experts are discussing the question, trying to solve it step by step, and make sure the result is correct:"	multi-agent debate [10]

Prompts As shown in Table 2, we consider 6 existing zero-shot prompts in arithmetic reasoning tasks. We construct our offline demonstration dataset using interaction logs between LLMs and queries prompted by those strategies.

LLMs To demonstrate the general applicability of Prompt-OIRL, we experiment with datasets generated with LLMs at different abilities, scaling from the GPT-3.5-turbo model [1], to TigerBot-13B-chat [18], and LLaMA2-7B-chat [19]. All created offline demonstration datasets, including the query-prompt pairs, prompted answers from different LLMs, and the correctness of those answers will be released as a publicly accessible dataset. We defer detailed discussions on the usage of embeddings, data processing, and training details can be found in Appendix C.

C.2 Offline Data Processing and Embeddings

For LLM ℓ and task k, we re-organize the offline demonstration dataset using the embedding function $\mathcal{E} : \mathcal{V}^{\infty} \mapsto \mathbb{R}^{1536}$, which maps a sequence of natural language context to a fix-length vector of size 1536 [62]. Therefore, using $e_x^{(i)} = \mathcal{E}(x^{(i)}), e_\pi^{(j)} = \mathcal{E}(\pi^{(j)}), r^{(ij)} = \mathbb{1}\{y^* = \ell(\pi^{(j)}(x^{(i)}))\}$ to denote the embeddings and reward instantiation, our demonstration dataset Equation (4) in implementation can be expressed as follows:

1. **Training Data:** $\mathcal{D}_{\ell,k}^{(\text{train})} = \{e_x^{(i)}, e_\pi^{(j)}, r^{(ij)}\}_{i\in\{1,\ldots,N\},j\in\{1,\ldots,K\}}$
2. **Test Data on Query:** $\mathcal{D}_{\ell,k}^{(\text{test q})} = \{e_x^{(i)}, e_\pi^{(j)}, r^{(ij)}\}_{i\in\{N+1,\ldots,N+M\},j\in\{1,\ldots,K\}}$
3. **Test Data on Prompts:** $\mathcal{D}_{\ell,k}^{(\text{test p})} = \{e_x^{(i)}, e_\pi^{(j)}, r^{(ij)}\}_{i\in\{N+1,\ldots,N+M\},j\in\{K+1,\ldots,K+P\}}$

The GSM8K task contains 7473 queries with golden answers for training and 1319 held-out queries with golden answers for testing; the SVAMP task contains 19690 examples, which are split into a training query-answer set of size 15000 and a testing query-answer set of size 4690; the MAWPS task contains 7685 examples, which are split into a training query-answer set of size 6000 and a testing query-answer set of size 1685. Therefore, for each prompting strategy, there are 7473, 15000, and 6000 demonstrative examples that can be collected through the interactive logs with a certain language model for the three tasks that can be used for training, respectively.

Our processed offline datasets and code for processing will be provided as open-source assets to facilitate future studies.

C.3 Reward Modeling: Implementation Matters

In our experiments, we find the gradient boosting methods [63, 64] are significantly better than using neural networks in reward modeling — in terms of both computational efficiency and performance.

We demonstrate the effectiveness using a minimalist approach by directly applying the XGBoost models and leaving further investigation on model selection to future work. That said, to exclude some potential alternatives — we have explored using fully connected neural networks as the reward model or using a ranked dataset mimicking the conventional RLHF setting, yet both of those choices perform worse than our presented implementation.

To enhance replicability, we use the following hyper-parameters for the gradient boosting model [63] in all experiment settings:

```
param = {'max_depth': 10, 'eta': 0.001, 'objective': 'binary:logistic'
    }
```

Further investigation on hyper-parameter tuning will lead to further improvement, especially in the pursuance of higher precision values in imbalanced prompt demonstrate datasets. (e.g., when using smaller LLMs like LLaMA-7B for the more challenging tasks like GSM8k.)

C.4 Held-Out Test Prompts

The following 10 prompts are used as a held-out test set for offline prompt evaluation. To enable the performance evaluation on held-out prompts, we collect interactive logs on each language model for 10 held-out novel prompting strategies. We will release the corresponding promoted answers on test queries for those held-out prompts as part of our offline dataset.

1. Approaching this logically, the steps to find the answer are:
2. Let's break this down into manageable steps to fully understand the problem:
3. Consider this as a puzzle, each piece contributing to the final answer. Let's place each piece, one by one:
4. Three scholars are analyzing this query from various perspectives, working collaboratively to build a comprehensive answer. Each contributes a step:
5. Let's solve this like a detective would solve a mystery, gathering clues and building up to the final solution step by step:
6. Imagine we're navigating a maze; each decision brings us closer to the center. Let's map our route:
7. Envision a round table meeting of expert problem solvers. Each participant suggests a step, building towards a consensus answer:
8. Like an architect constructing a building, let's design our answer carefully, layer by layer:
9. As if we are assembling a complex machine, let's put it together piece by piece:
10. Three wise philosophers are debating this question, each contributing a different aspect of the answer. Let's follow their discourse:

In order to further verify the generalization ability of the learned reward model, we extend to more held-out prompts. We use GPT-4 to generate those prompts through in-context learning. Specifically, we provided the human-crafted 6 prompting strategies to GPT-4, and asked GPT-4 to generate other potential prompting strategies. Our original chat history and prompts are available in the following anonymous link: `https://chat.openai.com/share/2c7652d2-2f96-48e8-b34f-efa4b15f8a61`.

1. Like a bee pollinating flowers, let's gather the essence from each point:
2. Picture this as a chain. Each link strengthens the whole:
3. As a librarian categorizes books, let's sort the information accordingly:
4. This is like unfolding origami to understand each crease and fold:
5. Imagine it as an echo in a valley, every sound has an origin and meaning:
6. Let's approach this like a chef creating a new recipe, ingredient by ingredient:
7. As a cartographer maps terrains, let's chart the nuances and details:
8. Like an athlete training for an event, every exercise has a purpose:

9. Think of it as wind chimes, each note contributing to the melody:
10. It's like flying a kite. Every tug and adjustment affects its flight:
11. Imagine we're in a lab, every test and observation is crucial for the conclusion:
12. Like a blacksmith forging metal, let's shape our understanding with precision:
13. As a botanist identifies plants, let's classify each detail:
14. This is like a radio tuning to different frequencies, let's find the right wavelength:
15. Imagine it's a theater play. Scene by scene, we unfold the plot:
16. Let's dive into this like a marine biologist exploring the ocean's depths:
17. As an electrician wires a circuit, each connection powers the system:
18. This is like a gardener pruning a plant, every cut makes it flourish:
19. Think of it as the gears of a bicycle, each turn propelling us forward:
20. Like a mountain climber uses tools, let's leverage each resource for understanding:
21. As a composer orchestrates a symphony, let's integrate each instrument:
22. This is like a pot of stew, simmering to meld the flavors:
23. Imagine building a sandcastle, each grain matters for the structure:
24. It's like peeling an onion, layer by layer, revealing the core:
25. Like a captain steering a ship, each decision adjusts our course:
26. As a mason lays bricks, every placement supports the structure:
27. This is like a game of chess, anticipating each move ahead:
28. Let's tackle this as a mechanic examines an engine, part by part:
29. Like a bird building a nest, every twig and feather matters:
30. Imagine crafting a sculpture, every detail brings it to life:
31. Like a student taking notes, we'll highlight the key points:
32. It's like sifting for gold, discarding the dirt to find the nuggets:
33. Let's weave through this as a spider spins its intricate web:
34. As a farmer tills the soil, every effort nurtures the crop:
35. Like a director shoots a film, each scene tells a part of the story:
36. Think of it as a flashlight in the dark, illuminating step by step:
37. Like a butterfly metamorphosing, we'll transition through stages of understanding:
38. Let's construct our insights as an engineer designs machinery:
39. Imagine it as the flow of a river, every bend and ripple shapes the course:
40. It's like a hawk soaring in the sky, observing details from a vantage point:
41. Like a detective following leads, every hint brings us closer to the truth:
42. As a tailor sews a garment, every stitch defines the shape:
43. Think of it as blending flavors in a dish, balancing to perfection:
44. Like a maestro leading an orchestra, every cue creates harmony:
45. Let's mold our comprehension as a child shapes clay into forms:
46. Imagine this as a marathon; every mile, every step takes us closer to the finish:
47. It's like mining for gems; we dig deep, assessing each discovery:
48. Let's examine this as a doctor diagnoses symptoms to understand the ailment:
49. Think of it as a tree growing, each branch reaching out to new insights:
50. Like a beekeeper tends the hive, let's understand each aspect of the colony:
51. As a carpenter joins wood, every angle and joint is crucial:
52. This is like creating a mosaic, each shard contributes to the whole:

53. Let's distill our thoughts as a brewer crafts a fine beverage:
54. Imagine as if we're setting a trap, each component is pivotal:
55. It's like a locksmith crafting a key, every cut unlocking understanding:
56. Like a painter blending colors, let's mix our ideas for clarity:
57. Let's tread as an explorer maps uncharted territory, noting every landmark:
58. Think of this as a puzzle box, each mechanism revealing deeper layers:
59. Like an artisan molds pottery, our hands shape the outcome:
60. Let's embark on this quest as a knight faces challenges, overcoming each obstacle:
61. Imagine we're piecing together an ancient manuscript; each fragment reveals more:
62. This is like building a bridge. Each segment must connect perfectly. Let's build:
63. Picture a gardener planting seeds. Each step, from soil preparation to watering, matters:
64. Think of this as weaving a tapestry. Every thread adds to the whole image:
65. Let's explore this like an astronaut discovering a new planet, detail by detail:
66. Envision this as a mosaic. Each tile contributes to the final artwork:
67. Like navigating a ship through a storm, each decision is vital. Let's chart the course:
68. It's as if we're restoring a masterpiece painting. Layer by layer, let's uncover:
69. Imagine being an archaeologist, unearthing an artifact. Bit by bit, we reveal:
70. Think of this as a dance, each move flowing into the next. Let's choreograph:
71. As a detective collecting clues for a case, let's unravel the mystery:
72. Like a novelist crafting a story, let's develop our narrative chapter by chapter:
73. Envision assembling a jigsaw puzzle, connecting pieces to see the whole picture:
74. Imagine sculpting from a block of marble; every chisel matters. Let's sculpt:
75. As a chemist mixing solutions, each ingredient alters the result. Let's mix:
76. This is like tuning an instrument. Every adjustment leads to harmony:
77. Like an actor rehearsing a script, let's understand our lines and their meaning:
78. Think of it as navigating a map. Each route we explore gives more insight:
79. As if we're crafting a potion, each herb and element has a purpose. Brew with me:
80. Let's approach this like a mathematician proving a theorem, step by logical step:
81. Picture us on a safari, observing every species. Detail by detail, we document:
82. Like setting up dominoes for a cascade, each placement is crucial. Let's set up:
83. Imagine you're a jeweler, evaluating a gem. Every facet reflects light differently:
84. Like a historian deciphering an old text, let's understand its context:
85. Think of this as a musical composition. Each note leads to the next movement:
86. It's like setting a table for a grand feast. Every detail adds to the ambiance:
87. As a photographer captures moments, let's focus on each element:
88. Imagine we're tailoring a suit. Each stitch, cut, and measurement counts:
89. This is like assembling a watch; every gear and spring is vital. Let's assemble:
90. Envision ourselves as geologists, studying layers of rock. Layer by layer, we analyze:
91. Like drafting an architectural blueprint, every line and measurement matters:
92. As if we're brewing a perfect cup of tea, each ingredient and timing is crucial:
93. This is like a relay race. Each leg of the race builds upon the last:
94. Think of it as mixing colors for a painting. Every shade adds depth and nuance:
95. It's like being a conductor, ensuring each instrument plays its part:
96. Imagine decoding an encrypted message. Every symbol has a meaning:

97. Like planning a trip, each destination and route makes the journey:
98. This is like lighting a sequence of candles. Each one illuminates more:
99. Envision it as a waterfall, each drop contributing to the flow:
100. Like a potter shaping clay, let's mold our understanding step by step:

C.5 Code, Hardware and Training Time

Our code, as well as the offline datasets, will be released as open-accessible. **During the review process, we provide source code in the supplementary material.** We highlight the Prompt-OIRL can be reproduced within a few hours using a single laptop using CPU. With our implementation, conducting OIRL for the GSM8k takes 50 minutes on a MacBookAir with an 8-core M2 chip, and takes only 5 minutes on a server with 16(out of 64)-core AMD 3995WX CPUs.

C.6 Addressing Challenge 1 through Offline Prompt Evaluation

Experiment Setup In this section, we study the effectiveness of the learned reward model by verifying its generalization ability from two perspectives: (1) with **seen-prompt + unseen-query**: we evaluate the effectiveness of the learned reward model with training-time prompts on **held-out queries** and (2) with **unseen-prompt + unseen-query**: we evaluate the effectiveness of the learned reward model with **held-out prompts** on held-out queries. To enable the performance evaluation on held-out prompts, we collect interactive logs on each language model for 10 held-out prompts. Details of those held-out prompts are provided in Appendix C.4.

We benchmark our approach against the *language models for self-criticism* (**LMSC**) baseline [65]. In this method, LLMs are tasked with verifying the accuracy of the answer, given both the query and the prompted response. Throughout our experiments, we gauge performance using **held-out test queries**. In our subsequent results, the notation **Ours (Q)** refers to evaluation outcomes employing training time prompts (with "Q" representing test queries), while **Ours (P)** refers to evaluations conducted on held-out prompts (with "P" representing test prompts). The same notations are applied to LMSC as LMSC (Q) and LMSC (P). Moreover, we compare results varying prompt demonstration data availabilities. Specifically, we change the number of prompts, represented as K, that is used in the training of the reward model. The result we report for K is averaged over combinations of $\binom{6}{K}$. Accuracy and precision are used as the evaluation metric.

Table 3: Effectiveness of the learned reward model is demonstrated through a comparison with LLM-based self-critic. Accuracies and precisions in predicting whether correct answers can be obtained on held-out queries are reported as an evaluation metric. *Higher is better.*

Task	Method	LLaMA2-7B-Chat		TigerBot-13B-Chat		GPT-3.5-turbo	
		Acc.	Precision	Acc.	Precision	Acc.	Precision
MAWPS	LMSC (Q)	0.47	0.433	0.776	0.175	0.662	0.664
	Ours (Q) K = 1	0.784	0.621	0.952	0.593	0.96	0.965
	Ours (Q) K = 5	0.795	0.632	0.95	0.593	0.96	0.965
	LMSC (P)	0.457	0.441	0.414	0.692	0.661	0.661
	Ours (P) K = 1	0.621	0.569	0.467	0.737	0.767	0.82
	Ours (P) K = 5	0.658	0.595	0.496	0.795	0.803	0.825
SVAMP	LMSC (Q)	0.474	0.441	0.434	0.735	0.801	0.805
	Ours (Q) K = 1	0.791	0.646	0.735	0.764	0.964	0.975
	Ours (Q) K = 5	0.8	0.65	0.738	0.764	0.962	0.97
	LMSC (P)	0.464	0.447	0.421	0.739	0.795	0.796
	Ours (P) K = 1	0.634	0.602	0.685	0.741	0.824	0.883
	Ours (P) K = 5	0.655	0.593	0.707	0.751	0.861	0.885
GSM8K	LMSC (Q)	0.205	0.187	0.635	0.537	0.662	0.656
	Ours (Q) K = 1	0.773	0.289	0.642	0.53	0.659	0.663
	Ours (Q) K = 5	0.726	0.301	0.646	0.542	0.656	0.657
	LMSC (P)	0.207	0.195	0.636	0.515	0.653	0.648
	Ours (P) K = 1	0.747	0.269	0.631	0.516	0.653	0.658
	Ours (P) K = 5	0.647	0.246	0.643	0.54	0.648	0.648

Results Table 3 showcases the results across the 3 tasks employing the 3 LLMs. From the results, we can deduce: **(1).** On held-out queries, we find the learned reward model is able to accurately predict

whether a certain prompt can lead to a correct answer. This holds true for both training prompts and held-out test prompts. **(2).** The prediction accuracy and precision of the learned reward model are significantly better than the LMSC baseline, demonstrating the superiority of leveraging the learned model rather than the LM itself in evaluating prompts and prompted answers. **(3).** Incorporating more prompts into the training dataset generally improves performance, particularly when evaluating held-out prompts. For more comprehensive results, please refer to Appendix D.2.

Take-Aways Challenge 1 can be effectively solved by Prompt-OIRL, which learns a reward model that is able to accurately evaluate prompts in an offline manner. The learned reward model is effective in predicting whether a given prompt can solve a given query without access to the language model. The prediction accuracy and precision are in general significantly better than the self-critic baseline, in which online interactions with LLMs are needed.

C.7 Addressing Challenge 2: Cost-Efficient Prompt Optimization

Experiment Setup As previously argued, utilizing the proxy reward model for prompt optimization, as opposed to directly using language models, would result in significantly reduced costs. Indeed, the expenses associated with inference time can be even more expensive [66]. In this section, we emphasize the cost-efficiency advantage of Prompt-OIRL during the inference time prompt optimization.

Prompt-OIRL evaluates prompts through its offline reward model and obviates the need for extra LLM interactions during inference. Picture a scenario where, for each query at inference, there exist K potential prompts to assess and select. With Prompt-OIRL, the optimal prompt is chosen without LLM interaction, ensuring that only the chosen prompt undergoes inference with the LLM to produce a singular response. Conversely, evaluating prompts via LLMs — by gauging their confidence or dissecting the nuances of various answers — demands extensive interaction prior to settling on a choice. We provide a detailed cost analysis of these methods, outlining the monetary implications of employing the GPT-3.5-turbo and TigerBot-13B-Chat APIs. We disclose the inference time, quantified in GPU hours for LLaMA2-7B-chat, which operated locally on an NVIDIA A4000 GPU.

Results Figure 15 illustrates the inference time costs associated with various LLMs and prompt selection strategies. Employing Prompt-OIRL for prompt optimization proves to be substantially more cost-efficient than methods reliant on LLMs. In the latter approach, all prompts need to be processed to obtain distinct prompted answers, necessitating K interactions with the LLMs. Following this, the LLMs are queried to verify the correctness of those answers and provide their confidence scores, incurring an additional K interactions. In contrast, Prompt-OIRL utilizes the offline reward model to pinpoint the most suitable prompt and only forwards the chosen prompt to the LLM for processing.

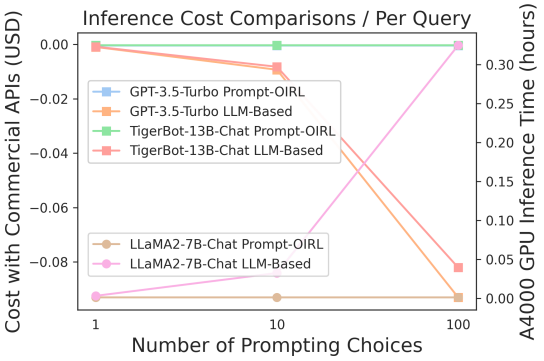

Figure 15: *The inference cost of different methods for a single query under different numbers of prompt choices.*

Take-Aways Challenge 2 is effectively mitigated by Prompt-OIRL. Prompt-OIRL permits offline evaluation and optimization of prompts through the learned reward model. Yet the LLM-Based methods rely on additional LLM interactions to evaluate prompted answers. Consequently, Prompt-OIRL offers a distinct advantage in terms of cost efficiency.

D Detailed Experiment Results

D.1 Improving Arithmetic Reasoning: How good are the optimized prompts?

We provide detailed performance comparisons on all datasets and LLMs we used in the experiment. Averaging the performance will result in Figure 3. In most cases, using Prompt-OIRL consistently

achieves superior performance, especially when the number of training prompts is limited. Additionally, it is not surprising that in some cases, when the number of training prompts is sufficient, selecting the best prompt during training, rather than seeking further optimization on held-out prompts can be the best strategy.

That said, we can also observe some fail cases of Prompt-OIRL, when the weaker LLMs are trying to solve the most challenging tasks. In those cases, the difficulties mainly result from the inability to accurately evaluate prompts — for more details, please refer to the following section.

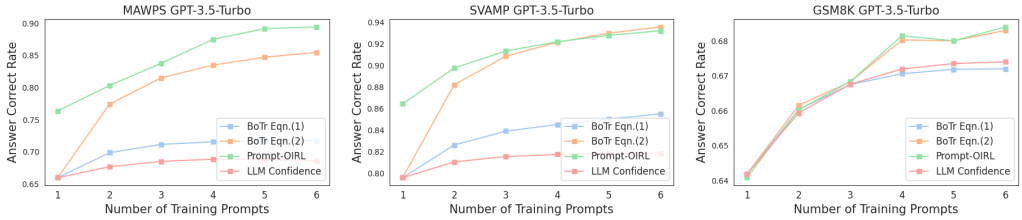

Figure 16: Prompting Success Rate on GPT 3.5 Turbo

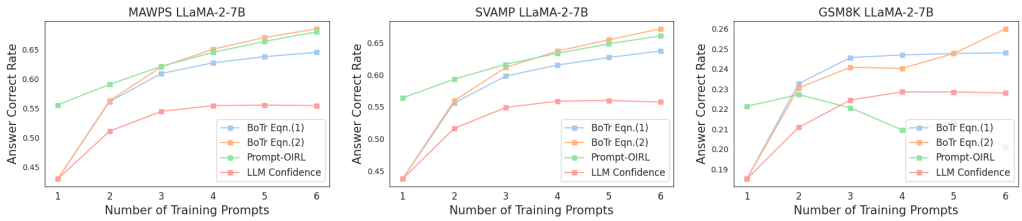

Figure 17: Prompting Success Rate on LLaMA-2-7B-Chat

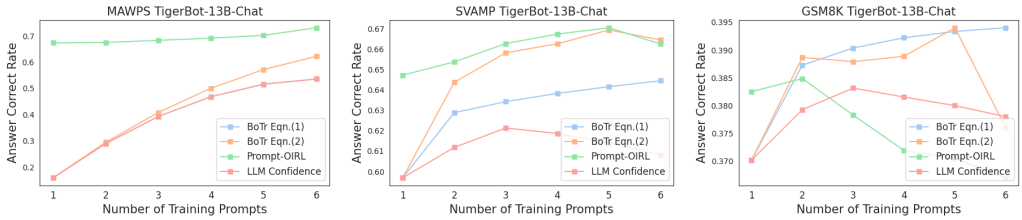

Figure 18: Prompting Success Rate on TigerBot-13B-Chat

D.2 Reward Modeling: Accuracy and Precision.

Accuracy We present the accuracy of different methods on different models and datasets in Figure 19 - Figure 21. The performance of the learned reward model consistently achieves higher prediction accuracy than using LLMs as critics. The prediction accuracy does not change much as a function of the number of training prompts.

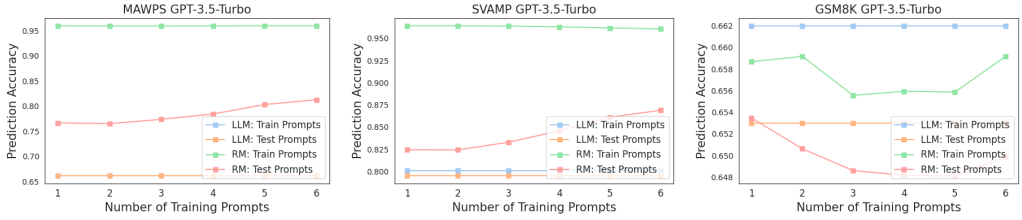

Figure 19: Accuracy on GPT 3.5 Turbo

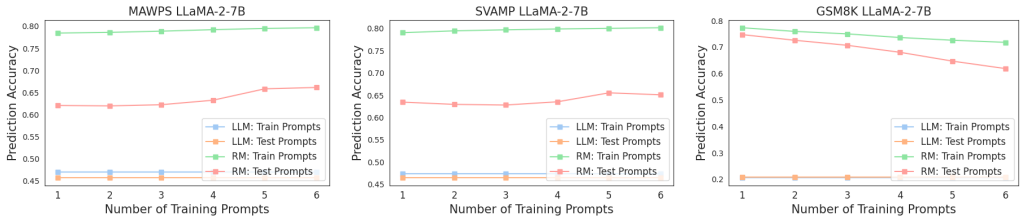

Figure 20: Accuracy on LLaMA-2-7B-Chat

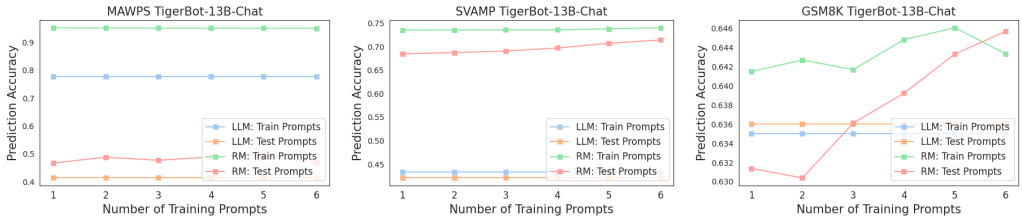

Figure 21: Accuracy on TigerBot-13B-Chat

Precision We present the precision of different methods on different models and datasets in Figure 22 - Figure 24. In most cases, using the learned reward model achieves significantly higher precision than using LLMs, meaning there is a higher probability that the predicted well-performing prompt is able to get a correct answer — the basis of our choice of using precision as the metric.

However, the reward model suffers difficulty when predicting the performance of LLMs with lower capabilities in arithmetic reasoning: when LLaMA-2-7B-Chat and the TigerBot-13B-Chat are facing the most challenging GSM8K tasks, their answers to the queries will be wrong in most cases. Achieving a high precision — in such cases — becomes more challenging. In this work, we set hyper-parameters uniformly to enhance reproducibility and disclose the superiority of the proposed method with minimal engineering optimization. Yet in practice, such an issue can potentially be further alleviated through e.g., sub-sampling or re-balancing the training dataset.

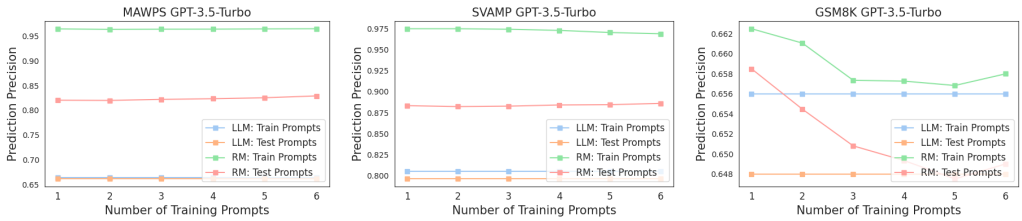

Figure 22: Precision on GPT 3.5 Turbo

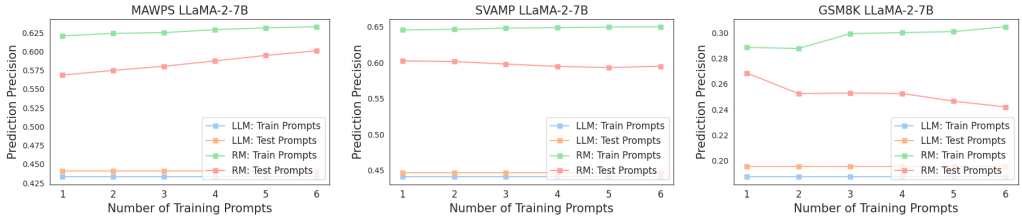

Figure 23: Precision on LLaMA-2-7B-Chat

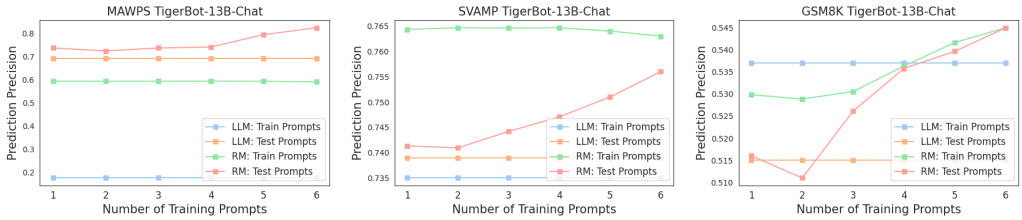

Figure 24: Precision on TigerBot-13B-Chat