ProRefine: Inference-time Prompt Refinement with Textual Feedback

Anonymous ACL submission

Abstract

Agentic workflows, where multiple AI agents collaborate to accomplish complex tasks, are becoming increasingly prevalent. However, these workflows often suffer from error propagation and sub-optimal performance, largely due to poorly designed prompts that fail to effectively guide individual agents. This is a critical problem because it limits the reliability and scalability of these powerful systems. We introduce ProRefine, an innovative inference-time prompt optimization method that leverages textual feedback from large language models (LLMs) to address this challenge. Without additional training or ground truth labels, ProRefine dynamically refines prompts for multi-step reasoning tasks. Evaluated on object counting, word sorting, and grade-school math problems, ProRefine significantly surpasses zero-shot Chain-of-Thought baselines by 3 to 43 percentage points. This approach not only boosts accuracy but also allows smaller models to match the performance of larger ones, highlighting its potential for efficient, scalable AI deployment, and democratizing access to high-performing AI.

1 Introduction

A critical condition for the advancement of artificial intelligence (AI) is that AI outputs align with human values (Feng et al., 2024). Human feedback, a scarce and subjective resource (Gray and Suri, 2019; Kahneman et al., 2021; Weerasooriya et al., 2023; Prabhakar et al., 2024), plays a vital role. LLMs, trained on vast datasets, utilize alignment techniques to generate more human-like and accurate responses (Rao et al., 2023; Sorensen et al., 2024; Kirk et al., 2024). Traditional methods employ Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017), with Proximal Policy Optimization (PPO) (Schulman et al., 2017) being a popular choice for fine-tuning LLMs.

Recent research has explored using LLMs as judges and evaluators (Zheng et al., 2023; Chiang and Lee, 2023; Wang et al., 2023; Liu et al., 2023; Bavaresco et al., 2024; Li et al., 2024). Building on use of LLMs to judge outputs, are *compound* systems that use LLMs to generate feedback / critiques (Saunders et al., 2022; Pryzant et al., 2023). Akyurek et al. (2023) and (Wadhwa et al., 2024) explore agentic frameworks to detect, critique, and refine tasks that require factual correctness. Another approach is to iteratively refine output using self-generated feedback (Madaan et al., 2023). TextGrad (Yuksekgonul et al., 2024) introduced automatic "differentiation" via text, using textual feedback to optimize the performance of individual components within a compound AI system. See additional related work in Appendix A.1.

Our work focuses on optimizing the *prompt*, a key element in chain-of-thought (CoT) (Wei et al., 2022) based LLM reasoning. CoT mimics human problem-solving by breaking down complex tasks into smaller, manageable steps. The system we introduce here, **ProRefine**, (Inference-time **Pro**mpt **Refine**ment with Textual Feedback) builds upon CoT by adaptively self-improving prompts using feedback ($LLM_{feedback}$) and an optimizer ($LLM_{optimizer}$) to refine prompts for the task-performing LLM (LLM_{task}). This process is motivated by the teacher-student framework (Torrey and Taylor, 2013) where a teacher agent guides a student agent to perform a task by providing feedback at intermediate steps. We explore policy optimization for aligning compound AI systems, drawing inspiration from TextGrad and policy gradient algorithms like PPO.

While prior work has explored prompt optimization (Shin et al., 2020; Deng et al., 2022; Yang et al., 2024; Dong et al., 2024), they all required there to be training examples in order to produce task-specific prompts. By contrast, ProRefine is task agnostic and requires no additional training

Query: I have a lettuce head, a stalk of celery, two yams, a cauliflower, a carrot, two onions, two potatoes, and three cabbages. How many vegetables do I have?

Answer: 13

Initial prompt: You will answer a reasoning question. Think step by step. The last line of your response should be of the following format: 'Answer: $VALUE' where VALUE is a numerical value.

Initial answer: 7

Intermediate model output (o): *To find the total number of vegetables, we need to count each type of vegetable*

Feedback (f): *Incorrect assumption. The task is to find the total number of individual vegetables, not just count the types. Instead, focus on counting the quantities of each vegetable and summing them up.*

Optimized prompt (p*): *You will answer a reasoning question. Think step by step. To find the total count of individual vegetables, count the quantities of each vegetable separately and sum them up. Then, provide the final numerical value. The answer should include the total count of individual vegetables, not just the types. Include the total count in the format \"There are X individual vegetables.\" The last line of your response should be of the following format: 'Answer: $VALUE' where VALUE is a numerical value.*

Final answer: 13

Figure 1: Example of ProRefine system in action, highlighting the utility of feedback in prompt optimization. More examples in Appendix (Figure 5).

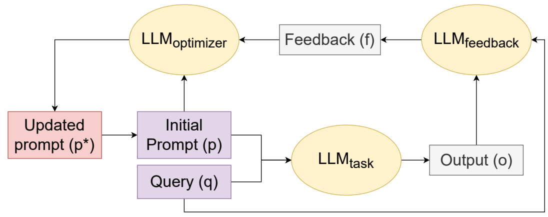

Figure 2: Overview of the ProRefine system, illustrating the iterative process of prompt optimization using feedback from LLMs in an agentic workflow.

or ground-truth labels. ProRefine is an inference-time optimization method that relies on the ability of LLMs to provide and act upon feedback for optimization, see Figure 1 for an example. ProRefine's ability to break complex tasks into smaller steps and self-improve using feedback makes it a practical solution for multi-step agentic workflows. This method is also suitable for black-box LLMs, where the internal parameters are inaccessible. To demonstrate the effectiveness of ProRefine, we evaluate our method across benchmark reasoning-based datasets.

Key Contributions:

- We propose a novel method - ProRefine for *inference-time* prompt optimization using textual feedback.
- We evaluate ProRefine on three datasets: object counting, word sorting, and grade-school math problem-solving and compare our method against CoT.

2 ProRefine

ProRefine is an inference-time prompt optimization algorithm that optimizes prompts by using textual feedback. ProRefine involves interactions between three LLMs:

$\boldsymbol{LLM_{task}}$: Executes the task based on the current prompt, generating the initial and subsequent outputs.

$\boldsymbol{LLM_{feedback}}$: A more capable model that critiques the LLM_{task}'s output, providing detailed feedback on improvements. This model needs to be strong enough to provide insightful and accurate critiques (Saunders et al., 2022; Bai et al., 2022).

$\boldsymbol{LLM_{optimizer}}$: Interprets the feedback and refines the prompt, aiming for coherent and task-focused improvements. This LLM is crucial for ensuring the prompt evolves effectively.

ProRefine (Algorithm 1) works as follows:

Algorithm 1: ProRefine

Input: Query: q, Initial prompt: p, tokens_per_step: k, max_steps: n, LLMs: LLM_{task}, $LLM_{feedback}$, $LLM_{optimizer}$

Output: Optimized prompt: p^*

$p^* = p$

for $i = 1$ **to** n **do**

  $o_i = LLM_{task}(p^*, q)$ `// Generate i * k tokens`

  $f_i = LLM_{feedback}(q, o_i)$ `// Get feedback`

  $p* = LLM_{optimizer}(p^*, f_i)$ `// Optimize the prompt`

  if $EOS\_token\ in\ o_i$ **then**

    break

return p^*

Initialization: Start with an initial prompt p for the task, a query q, and parameters defining the generation and optimization process (k tokens per step, n maximum steps).

Generation and Feedback Loop:

- **Generation:** Use LLM_{task} to generate an output based on the current prompt p^* and query q. This step is limited to k tokens to control the granularity of feedback.
- **Feedback:** $LLM_{feedback}$ evaluates the generated output o_i against the query q to provide textual feedback f_i. This feedback encapsulates how the output could be improved, focusing on aspects such as accuracy, relevance, or coherence.
- **Optimization:** $LLM_{optimizer}$ uses the feedback f_i to refine the prompt p^*. This step involves modifying the prompt to better align with the task requirements or to correct identified deficiencies in previous generations.

Termination: The process iterates until either the maximum number of steps n is reached or an end-of-sequence (EOS) token is detected in the output, indicating the completion of the task.

2.1 Implementation Details

The granularity and duration of the optimization process are controlled by parameters k (tokens per step) and n (maximum steps). These parameters can be tuned based on task complexity and desired output quality.

2.2 Improvements

We propose two modifications aimed at increasing the efficiency and performance of ProRefine. Currently, ProRefine is executed on all queries sent to LLM_{task}, even when LLM_{task} produces a correct output. To optimize this process, we suggest the following two approaches:

First, we propose using an LLM to evaluate the output generated for each query. This evaluation LLM, while not perfect and lacking ground truth, serves to assess whether the output produced by LLM_{task} is correct. If the evaluation LLM deems the output incorrect, ProRefine is executed, otherwise, the output is used as is. We refer to this method as *ProRefine (LM)*.

Second, in cases where a "perfect" evaluator exists such as with many mathematical problems, where verifying a candidate solution is easier than deriving the correct answer, we can leverage this evaluator to judge the output of LLM_{task}. To simulate this scenario, we compare the output of LLM_{task} against the ground-truth. We refer to this method as *ProRefine (GT)*.

3 Experiments and Evaluation

3.1 Data

We evaluate ProRefine on three reasoning tasks, each of which involves multi-step reasoning, making them suitable for evaluating prompt optimization in agentic workflows: object counting and word sorting from the BIG-Bench Hard benchmark (100 test examples each; (Srivastava et al., 2023)), and grade-school math problem-solving from GSM8K (1319 test examples; (Cobbe et al., 2021)). We use the same dataset splits and evaluation as Yuksekgonul et al. (2024). Refer to Appendix Figure 4 for the system and task prompts we use for each dataset.

3.2 Results

We report results with three models - *Llama3.2-1B-instruct*, *Llama3.2-3B-instruct*, and *Llama3.1-8B-instruct* (Meta, 2024) for LLM_{task}. The prompts are optimized using Algorithm 1, with *Llama3.1-70B-instruct* used for feedback generation, prompt optimization, and evaluation. We select the values of hyperparameters $k = 10$ and $n = 25$ to control the granularity of feedback and duration of optimization. We compare the performance of our method against the zero-shot Chain-of-Thought (CoT) baseline and report test accuracy with 95% confidence interval. The results are shown in Figure 3. For complete tabular results, refer to Appendix Table 1.

Our results show that ProRefine is able to significantly improve the performance of all the LLM_{task} models as compared to the zero-shot CoT baseline. For object counting, the performance is improved by $3-21$ percentage points with significant improvement observed for *Llama3.2-3B-instruct* and *Llama3.1-8B-instruct*. For word sorting, the performance gain ranges from $8-43$ percentage points with significant results observed for all the models. For GSM8K, the maximum performance gain is 27.5 percentage points with significant improvement observed for all the models however a slight performance drop (0.012) is also observed for ProRefine with *Llama3.2-3B-instruct*. It is noteworthy that the number of significant results increase with model size.

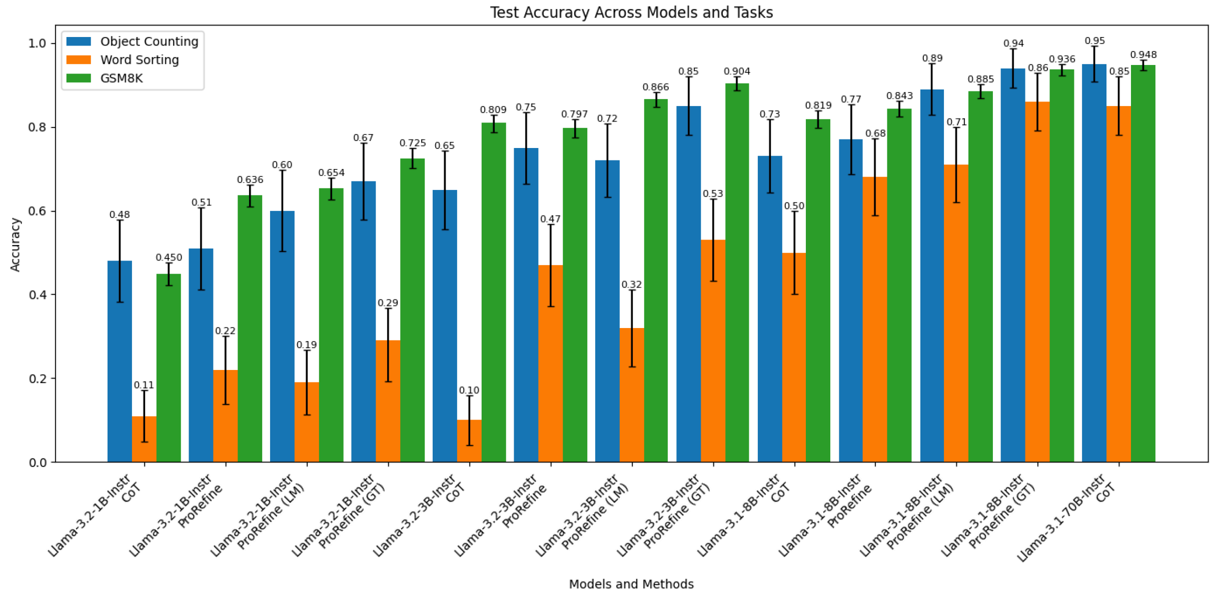

Figure 3: Test Accuracy [with 95% confidence interval] across different models and datasets. *Llama3.1-70B-instruct* is used for feedback generation, prompt optimization, and evaluation. Full table of results in Appendix (Table 1).

We notice that ProRefine is able to improve the performance of lesser capable models *Llama3.2-3B-instruct* and *Llama3.1-8B-instruct* to that of more capable ones *Llama3.1-8B-instruct* and *Llama3.1-70B-instruct* respectively. We also experiment using ProRefine with a comparatively smaller model (*Llama3.1-8B-instruct*) for $LLM_{feedback}$ and $LLM_{optimizer}$. However, it did not yield substantial improvement in all scenarios and in fact decreased the performance in some cases. This is consistent with the findings of the prior work that smaller models are deficient in understanding and incorporating feedback (Saunders et al., 2022; Bai et al., 2022).

4 Discussion

The results demonstrate that ProRefine is a broadly applicable method for improving LLM performance at inference-time. The "performance gap bridging" effect is particularly significant, suggesting that ProRefine can be a compute-effective alternative to simply scaling up model size, which could prove critical in resource-constrained settings.

The largest performance gains are observed on the word sorting task, indicating that tasks requiring more complex reasoning or manipulation of intermediate outputs benefit the most from ProRefine's iterative refinement. The mixed results when using a smaller model for $LLM_{feedback}$ illustrate the importance of "knowledge asymmetry," i.e., that the feedback model needs to be "sufficiently capable" of providing useful critiques.

ProRefine builds on TextGrad (Yuksekgonul et al., 2024), but differs crucially in that ProRefine runs at inference-time. Moreover, TextGrad is a supervised fine-tuning method, while ProRefine requires no training data, making its data requirements less onerous. ProRefine also relates to other prompt optimization techniques (Shin et al., 2020; Deng et al., 2022; Yang et al., 2024; Dong et al., 2024), but combines in a novel fashion inference-time operations with LLM-generated feedback.

ProRefine can offer a degree of interpretability by exposing the outputs from $LLM_{feedback}$, allowing insights into the model's reasoning process. Although evaluated on reasoning and math tasks, ProRefine is general and applicable to other tasks.

5 Conclusion

We introduced ProRefine, a novel, practical, and *inference-time* prompt optimization method for agentic workflows. ProRefine leverages LLM-generated textual feedback to dynamically refine prompts, leading to significant performance improvements without requiring additional training or ground-truth labels. It is particularly effective for multi-step reasoning tasks and can bridge the performance gap between smaller and larger LLMs. The *inference-time* nature makes it readily deployable and adaptable, contributing to more efficient, transparent, and accessible AI systems.

Limitations

This work has the following limitations that we acknowledge has potential for future explorations:

- **Computational Cost and Scalability:** ProRefine's iterative process increases computational cost compared to single-shot prompting. Scaling to substantially larger or more complex tasks presents both computational and methodological challenges.
- **Generalizability:** Evaluation is currently limited to reasoning and mathematical tasks. Further research is needed to assess performance across a broader range of NLP tasks and domains.
- **Model Dependence and Feedback Quality:** ProRefine relies on a large, capable LLM for feedback, which may limit accessibility. The quality and potential biases of this feedback are crucial factors impacting performance and require further investigation.
- **Evaluation and Bias:** Using an LLM for evaluation introduces potential biases. More comprehensive human evaluations and robust methods for mitigating evaluator bias are needed.
- **Hyperparameter and Feedback Loop:** Current approach requires manual hyperparameter tuning and iterative prompting might cause issues, and automatic ways needs further research.

References

Afra Feyza Akyurek, Ekin Akyurek, Ashwin Kalyan, Peter Clark, Derry Tanti Wijaya, and Niket Tandon. 2023. RL4F: Generating natural language feedback with reinforcement learning for repairing model outputs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7716–7733, Toronto, Canada. Association for Computational Linguistics.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, André F. T. Martins, Philipp Mondorf, Vera Neplenbroek, Sandro Pezzelle, Barbara Plank, David Schlangen, Alessandro Suglia, Aditya K Surikuchi, Ece Takmaz, and Alberto Testoni. 2024. Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks. *Preprint*, arXiv:2406.18403.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yihong Dong, Kangcheng Luo, Xue Jiang, Zhi Jin, and Ge Li. 2024. PACE: Improving prompt with actor-critic editing for large language model. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7304–7323, Bangkok, Thailand. Association for Computational Linguistics.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Yicheng Feng, Yuxuan Wang, Jiazheng Liu, Sipeng Zheng, and Zongqing Lu. 2024. LLaMA-rider: Spurring large language models to explore the open world. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4705–4724, Mexico City, Mexico. Association for Computational Linguistics.

Mary L. Gray and Siddharth Suri. 2019. *Ghost work: how to stop Silicon Valley from building a new global underclass*. Houghton Mifflin Harcourt, Boston.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Daniel Kahneman, Olivier Sibony, and Cass R. Sunstein. 2021. *Noise: a flaw in human judgment*, first edition edition. Little, Brown Spark, New York. OCLC: on1249942231.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*.

Hannah Rose Kirk, Alexander Whitefield, Paul Röttger, Andrew Bean, Katerina Margatina, Juan Ciro, Rafael Mosquera, Max Bartolo, Adina Williams, He He, Bertie Vidgen, and Scott A. Hale. 2024. The PRISM Alignment Dataset: What Participatory, Representative and Individualised Human Feedback Reveals About the Subjective and Multicultural Alignment of Large Language Models. *arXiv preprint*. ArXiv:2404.16019 [cs].

Zhen Li, Xiaohan Xu, Tao Shen, Can Xu, Jia-Chen Gu, Yuxuan Lai, Chongyang Tao, and Shuai Ma. 2024. Leveraging large language models for NLG evaluation: Advances and challenges. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16028–16045, Miami, Florida, USA. Association for Computational Linguistics.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

Maitrey Mehta, Valentina Pyatkin, and Vivek Srikumar. 2024. Promptly predicting structures: The return of inference. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 112–130.

Meta. 2024. llama-models/models/llama3_2/MODEL_card.md at main · meta-llama/llama-models.

Akshara Prabhakar, Thomas L. Griffiths, and R. Thomas McCoy. 2024. Deciphering the factors influencing the efficacy of chain-of-thought: Probability, memorization, and noisy reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3710–3724, Miami, Florida, USA. Association for Computational Linguistics.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. 2024. Recursive introspection: Teaching language model agents how to self-improve. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Leonardo Ranaldi and Andrè Freitas. 2024. Self-refine instruction-tuning for aligning reasoning in language models. *arXiv preprint arXiv:2405.00402*.

Abhinav Sukumar Rao, Aditi Khandelwal, Kumar Tanmay, Utkarsh Agarwal, and Monojit Choudhury. 2023. Ethical reasoning over moral alignment: A case and framework for in-context ethical policies in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13370–13388, Singapore. Association for Computational Linguistics.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *arXiv preprint*. ArXiv:2206.05802 [cs].

Timo Schick, Jane A. Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2023. PEER: A collaborative language model. In *The Eleventh International Conference on Learning Representations*.

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

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Axel Sorensen, Siyao Peng, Barbara Plank, and Rob Van Der Goot. 2024. EEVEE: An easy annotation tool for natural language processing. In *Proceedings of The 18th Linguistic Annotation Workshop (LAW-XVIII)*, pages 216–221, St. Julians, Malta. Association for Computational Linguistics.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Lisa Torrey and Matthew Taylor. 2013. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1053–1060.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*.

Manya Wadhwa, Xinyu Zhao, Junyi Jessy Li, and Greg Durrett. 2024. Learning to refine with fine-grained natural language feedback. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12281–12308, Miami, Florida, USA. Association for Computational Linguistics.

Jiaan Wang, Yunlong Liang, Fandong Meng, Zengkui Sun, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023. Is ChatGPT a good NLG evaluator? a preliminary study. In *Proceedings of the 4th New Frontiers in Summarization Workshop*, pages 1–11, Singapore. Association for Computational Linguistics.

Tharindu Cyril Weerasooriya, Sujan Dutta, Tharindu Ranasinghe, Marcos Zampieri, Christopher M. Homan, and Ashiqur R. KhudaBukhsh. 2023. Vicarious Offense and Noise Audit of Offensive Speech Classifiers: Unifying Human and Machine Disagreement on What is Offensive. *arXiv preprint*. ArXiv:2301.12534 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. *Preprint*, arXiv:2309.03409.

Kevin Yang, Yuandong Tian, Nanyun Peng, and Dan Klein. 2022. Re3: Generating longer stories with recursive reprompting and revision. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4393–4479, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

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

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. 2024. Agent-as-a-judge: Evaluate agents with agents. *Preprint*, arXiv:2410.10934.

A Appendix

Table 1 shows the complete results for ProRefine discussed in Section 3.2. Table 2 shows results for ProRefine with/out chat history. Using the entire chat history in each step degrades the performance in most cases.

Figure 4 shows the system prompts used for tasks and LLMs. Figure 5 shows an example where the optimized prompt is unable to guide the model to produce the right output.

A.1 Related Work

ProRefine draws inspiration from and contributes to several interconnected research areas. A crucial field is prompt engineering and optimization, where early work focused on manual prompt crafting (Wei et al., 2022), while recent efforts explore automatic optimization. Methods like AutoPrompt (Shin et al., 2020) and RLPrompt (Deng et al., 2022) use gradient-based search and reinforcement learning, respectively, but typically require training data. Other approaches leverage LLMs themselves for prompt generation (Yang et al., 2022; Pryzant et al., 2023; Zhou et al., 2022; Yang et al., 2024;

Model	**Method**	**Object Counting**	**Word Sorting**	**GSM8K**
Llama-3.2 1B-Instruct	CoT	0.48 [0.382, 0.578]	0.11 [0.048, 0.172]	0.450 [0.423, 0.476]
	ProRefine	0.51 [0.412, 0.608]	0.22 [0.138, 0.302]	0.636* [0.610, 0.662]
	ProRefine (LM)	0.6 [0.503, 0.696]	0.19 [0.113, 0.267]	0.654* [0.627, 0.678]
	ProRefine (GT)	**0.67** [0.577, 0.763]	**0.29*** [0.192, 0.368]	**0.725***** [0.701, 0.749]
Llama-3.2 3B-Instruct	CoT	0.65 [0.556, 0.744]	0.10 [0.041, 0.159]	0.809 [0.787, 0.829]
	ProRefine	0.75 [0.665, 0.835]	0.47* [0.372, 0.568]	0.797 [0.774, 0.818]
	ProRefine (LM)	0.72 [0.632, 0.808]	0.32* [0.228, 0.412]	0.866** [0.847, 0.883]
	ProRefine (GT)	**0.85*** [0.780, 0.920]	**0.53**** [0.432, 0.628]	**0.904***** [0.888, 0.920]
Llama-3.1 8B-Instruct	CoT	0.73 [0.643, 0.817]	0.50 [0.401, 0.598]	0.819 [0.797, 0.839]
	ProRefine	0.77 [0.687, 0.853]	0.68 [0.595, 0.779]	0.843 [0.823, 0.863]
	ProRefine (LM)	0.89* [0.839, 0.959]	0.71* [0.621, 0.799]	0.885** [0.868, 0.902]
	ProRefine (GT)	**0.94**** [0.893, 0.987]	**0.86**** [0.792, 0.928]	**0.936***** [0.922, 0.949]
Llama-3.1-70B-Instruct	CoT	**0.95** [0.907, 0.993]	**0.85** [0.780, 0.920]	**0.948** [0.936, 0.960]

Table 1: Test Accuracy [with 95% confidence interval] across different models and datasets. *, **, and *** indicate that the result is significantly better than the lowest result, the second lowest, and the third lowest result respectively. Results in bold indicate the highest accuracy for a model-dataset combination. *Llama3.1-70B-instruct* is used for feedback generation, prompt optimization, and evaluation.

Model	**Method**	**Object Counting**	**Word Sorting**	**GSM8K**
Llama-3.2 1B-Instruct	ProRefine (LM)	0.6	0.19	0.654
	ProRefine (LM) with chat history	0.6	0.15	0.530
	ProRefine (GT)	0.67	0.29	0.725
	ProRefine (GT) with chat history	0.63	0.22	0.569
Llama-3.2 3B-Instruct	ProRefine (LM)	0.72	0.32	0.866
	ProRefine (LM) with chat history	0.71	0.36	0.820
	ProRefine (GT)	0.85	0.53	0.904
	ProRefine (GT) with chat history	0.8	0.54	0.850
Llama-3.1 8B-Instruct	ProRefine (LM)	0.89	0.71	0.885
	ProRefine (LM) with chat history	0.89	0.71	0.874
	ProRefine (GT)	0.94	0.86	0.936
	ProRefine (GT) with chat history	0.93	0.78	0.909

Table 2: Test Accuracy across different models and datasets. ProRefine (LM) and ProRefine (GT) with/out chat history. Initial responses are evaluated using *Llama3.1-70B-instruct*. *Llama3.1-70B-instruct* is also used for feedback generation, prompt optimization, and evaluation.

Mehta et al., 2024). ProRefine distinguishes itself by operating at inference time without training data or ground truth labels, extending the concept of Chain-of-Thought prompting (Wei et al., 2022) by iteratively refining the prompt.

Closely related is the growing body of work on LLMs as judges and evaluators (Zheng et al., 2023; Bavaresco et al., 2024; Chiang and Lee, 2023; Wang et al., 2023; Liu et al., 2023; Li et al., 2024; Verga et al., 2024; Zhuge et al., 2024). ProRefine utilizes this concept, employing an LLM to provide feedback (a form of evaluation), but crucially uses this feedback for optimization.

The idea of LLM self-improvement and iterative refinement is also highly relevant. Self-Refine (Madaan et al., 2023) is a prominent example, where an LLM generates both output and feedback, using the latter for refinement. Other works explore self-critiquing (Saunders et al., 2022) and reinforcement learning for critique generation (RL4F) (Akyurek et al., 2023), along with various feedback and refinement mechanisms (Wadhwa et al., 2024; Schick et al., 2023; Qu et al., 2024; Ranaldi and Freitas, 2024; Dong et al., 2024; Khattab et al., 2024). ProRefine shares this spirit but focuses specifically on prompt refinement, making it suitable for agentic workflows and black-box LLMs, unlike direct output modification.

ProRefine builds upon the core ideas in TextGrad (Yuksekgonul et al., 2024), which uses textual feedback for optimization within compound AI systems. However, ProRefine applies this to inference-

BBH Task Prompt: "You will answer a reasoning question. Think step by step. The last line of your response should be of the following format: 'Answer: $VALUE' where VALUE is a numerical value."

GSM8K Task Prompt: "You will answer a mathemetical reasoning question. Think step by step. The last line of your response should be of the following format: 'Answer: $VALUE' where VALUE is a numerical value."

Feedback system prompt: """You are a smart language model that evaluates the output of a language model for a given input.\n

You do not propose new output, only evaluate the given output critically, think step-by-step, and give very concise feedback to improve the output.

Ensure your feedback is correct and factual.\n

This is very important, if the output is correct do not provide any feedback, respond with 'the output is correct'.\n

Give your response by sending the feedback only. The text you send will be used directly.\n\n"""

Optimization system prompt: """You are part of an optimization system that improves the input prompt given to a large language model.\n

You are given a feedback for the model output. Your goal is to use this feedback to improve the input prompt.

The feedback may be noisy, identify what is important and what is correct. Pay attention to the constraints mentioned in the input prompt.\n

This is very important. You MUST make sure that the improved prompt does not deviate substantially from the input prompt and is generalizable for the task.

If the input prompt cannot be improved further, your response should be the input prompt as is.\n

Think step-by-step and send the improved prompt between tags <IMPROVED_PROMPT> improved prompt </IMPROVED_PROMPT>.

The text you send between the tags will be used directly to replace the system prompt for a large language model.\n\n
"""

Figure 4: System prompts for tasks and LLMs.

Query: Sort the following words alphabetically: List: sweatshirt bone ia wagoneer doleful homeobox hindustan convergent

Answer: bone convergent doleful hindustan homeobox ia sweatshirt wagoneer

Initial answer: To sort the words alphabetically, we need to compare each word and arrange them in order from A to Z.\n\nHere's the step-by-step process: *<skipped output here for space>* Now, let's arrange the words in alphabetical order:\n\n1. bone\n2. convergent\n3. doleful\n4. homeobox\n5. hindustan\n6. ia\n7. list\n8. sweatshirt\n9. wagoneer\n\nAnswer: 9

Last step model output: *Answer: bone, convergent, doleful, homeobox, hindustan, ia, list, sweatshirt, wagoneer*

Feedback: *\"List\" should not be included in the sorted output as it was part of the original instruction. The corrected output is: bone, convergent, doleful, hindustan, homeobox, ia, sweatshirt, wagoneer*

Optimized prompt: *<EM>IMPROVED_PROMPT</EM>\n\nYou are given a list of words and are asked to sort them in alphabetical order. Simply arrange the words in alphabetical order and provide the sorted list in a comma-separated format, excluding any words that were part of the original instruction. Ensure that the response includes every word from the original list in the correct order. Explicitly state the final answer in the format: 'Answer: [comma-separated sorted list of words]', without including the original instruction words.\n\n</EM>*

Final answer: "Answer: bone, convergent, doleful, homeobox, hindustan, ia, list, sweatshirt, wagoneer"

Figure 5: Example illustrating a case where the optimized prompt is unable to guide the model to produce the right output.

time prompt optimization, a key distinction. Finally, ProRefine belongs to the broader category of inference-time methods (Snell et al., 2024), that improve LLMs without weight modification, such as multi-agent debate (Du et al., 2024). Unlike these, ProRefine targets the prompt itself. Recent advances in understanding reasoning in LLMs (Jaech et al., 2024; Guo et al., 2025) also provide context to our prompt improvement.

In essence, ProRefine uniquely integrates inference-time operation, LLM-generated feedback, and prompt optimization, targeting multi-step reasoning in agentic workflows, addressing limitations of prior work related to training data and black-box model applicability.

A.2 Computing Environment

We run all of our experiments on compute nodes with a single Nvidia A100 GPU (80GB), 24 core processor, and 220GB RAM. For model's `generate` function we set the following parameters: `num_return_sequences=1, do_sample=False, top_p=None, temperature=None`.