ProRefine: Inference-Time Prompt Refinement with Textual Feedback (Student Abstract)

Deepak Pandita[1, 2], Tharindu Cyril Weerasooriya[1], Ankit Shah[1], Isabelle Diana May-Xin Ng[1, 3], Christopher M. Homan[2], Wei Wei[1]

[1]Center for Advanced AI, Accenture
[2]Rochester Institute of Technology
[3]University of California, Berkeley
deepak@mail.rit.edu, t.weerasooriya@accenture.com

Abstract

Agentic workflows, where multiple AI agents collaborate to accomplish complex tasks like reasoning or planning, play a substantial role in many cutting-edge commercial applications. These workflows depend critically on the prompts used to provide the roles models play in such workflows. Poorly designed prompts that fail even slightly to guide individual agents can lead to sub-optimal performance that may snowball within a system of agents, limiting their reliability and scalability. To address this important problem of inference-time prompt optimization, we introduce ProRefine, an innovative inference-time optimization method that uses an agentic loop of LLMs to generate and apply textual feedback. ProRefine dynamically refines prompts for multi-step reasoning tasks without additional training or ground truth labels. Evaluated on five benchmark mathematical reasoning datasets, ProRefine significantly surpasses zero-shot Chain-of-Thought baselines by 3 to 37 percentage points. This approach not only boosts accuracy but also allows smaller models to approach the performance of their larger counterparts. This highlights its potential for building cost-effective and powerful hybrid AI systems, thereby democratizing access to high-performing AI.

Extended version — https://arxiv.org/abs/2506.05305
Code —
https://github.com/deepakpandita57/ProRefine_public

Introduction

Prompt is a key element in chain-of-thought (CoT) (Wei et al. 2022) based LLM reasoning. Prior work on prompt optimization often focuses on either *offline fine-tuning*, which requires extensive training data, or universal application of *largest, most capable models to every task*. This presents a practical dilemma in many real-world scenarios. Continuously fine-tuning is not always feasible, and relying exclusively on state-of-the-art models is often computationally prohibitive. A different approach is needed for scenarios that require *dynamic, on-the-fly repair* for specific and difficult queries where a standard prompt fails. This is particularly true in *resource-aware deployments*, where a smaller model may suffice for most tasks but requires enhancement for a small subset of critical queries. ***The goal, therefore, shifts from finding a single, universally optimal prompt to performing targeted, inference-time intervention.***

To address this need, we introduce **ProRefine** (Inference-time **Pro**mpt **Refine**ment with Textual Feedback), which builds upon CoT by adaptively improving prompts using feedback and an optimizer to refine prompts for the task-performing LLM. This workflow, motivated by the teacher-student framework (Torrey and Taylor 2013), where a teacher guides a student to perform a task by providing feedback at intermediate steps, but implemented via LLM interactions without pre-training, represents a novel approach to adaptive agentic reasoning. We explore policy optimization for aligning compound AI systems, drawing inspiration from TextGrad (Yuksekgonul et al. 2024) and policy gradient algorithms, such as PPO.

ProRefine

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 $i * k$ tokens to control the granularity of the feedback. In each iteration, LLM_{task} produces k more tokens, attempting to refine prior output while progressively continuing its response to the query.
- **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.

Dataset	Method	Llama-3.2 1B-it	Llama-3.2 3B-it	Llama-3.1 8B-it
Object Counting	CoT	0.48 [0.382, 0.578]	0.65 [0.556, 0.744]	0.73 [0.643, 0.817]
	TextGrad	**0.62** [0.524, 0.716]	0.73 [0.643, 0.817]	0.86 [0.792, 0.928]
	ProRefine (no verifier)	0.51 [0.412, 0.608]	**0.75** [0.665, 0.835]	0.77 [0.687, 0.853]
	ProRefine (verifier)	0.6 [0.503, 0.696]	0.72 [0.632, 0.808]	**0.89*** [0.839, 0.959]
	†ProRefine (optimal verifier)	0.67 [0.577, 0.763]	0.85* [0.780, 0.920]	0.94* [0.893, 0.987]
Word Sorting	CoT	0.11 [0.048, 0.172]	0.10 [0.041, 0.159]	0.50 [0.401, 0.598]
	TextGrad	**0.33*** [0.237, 0.423]	**0.61*** [0.514, 0.706]	0.69* [0.599, 0.781]
	ProRefine (no verifier)	0.22 [0.138, 0.302]	0.47* [0.372, 0.568]	0.68 [0.595, 0.779]
	ProRefine (verifier)	0.19 [0.113, 0.267]	0.32* [0.228, 0.412]	**0.71*** [0.621, 0.799]
	†ProRefine (optimal verifier)	0.29* [0.192, 0.368]	0.53* [0.432, 0.628]	0.86** [0.792, 0.928]
GSM8K	CoT	0.450 [0.423, 0.476]	0.809 [0.787, 0.829]	0.819 [0.797, 0.839]
	TextGrad	0.463 [0.436, 0.489]	0.801 [0.779, 0.822]	0.864* [0.845, 0.882]
	ProRefine (no verifier)	0.636** [0.610, 0.662]	0.797 [0.774, 0.818]	0.843 [0.823, 0.863
	ProRefine (verifier)	**0.654**** [0.627, 0.678]	**0.866**** [0.847, 0.883]	**0.885*** [0.868, 0.902]
	†ProRefine (optimal verifier)	0.725** [0.701, 0.749]	0.904** [0.888, 0.920]	0.936** [0.922, 0.949]
SVAMP	CoT	0.689 [0.66, 0.718]	0.869 [0.848, 0.890]	0.854 [0.832 , 0.876]
	TextGrad	0.684 [0.655, 0.713]	0.861 [0.840, 0.882]	0.84 [0.817, 0.863]
	ProRefine (no verifier)	0.774** [0.748, 0.800]	0.878 [0.858, 0.898]	0.877 [0.857, 0.897]
	ProRefine (verifier)	**0.808**** [0.784, 0.832]	**0.896** [0.877, 0.915]	**0.893*** [0.874, 0.912]
	†ProRefine (optimal verifier)	0.861** [0.840, 0.882]	0.925** [0.909, 0.941]	0.938** [0.923, 0.953]
AQUARAT	CoT	0.259 [0.202, 0.31]	**0.563** [0.498, 0.620]	0.586 [0.522, 0.643]
	TextGrad	**0.311** [0.250, 0.364]	0.524 [0.462 , 0.585]	0.559 [0.494, 0.616]
	ProRefine (no verifier)	0.205 [0.151, 0.250]	0.343 [0.284, 0.401]	0.398 [0.337 , 0.458]
	ProRefine (verifier)	0.268 [0.209, 0.318]	0.551 [0.486 , 0.608]	**0.606** [0.542, 0.663]
	†ProRefine (optimal verifier)	0.354 [0.292, 0.409]	0.598 [0.538, 0.659]	0.657 [0.595, 0.712]

Table 1: Test Accuracy with 95% confidence intervals across five benchmark datasets and models. * and ** denote statistically significant improvements over one or two baseline methods, respectively. Bold indicates the highest accuracy for a dataset-method combination. † demonstrates the upper bound potential of the optimization loop and the impact of verifier quality. *Llama3.1-70B-instruct* is employed for feedback generation, prompt optimization, and evaluation.

Unifying Verifier and Feedback: At inference time, verifiers play a crucial role in judging model outputs (Cobbe et al. 2021; Lightman et al. 2024; Snell et al. 2024). We employ the *Llama3.1-70B-instruct* model to function as both the feedback mechanism ($LLM_{feedback}$) and the verifier for simplicity. We manage these roles through separate API calls, each with a role-defining prompt. The verifier's function is to evaluate the initial output generated by LLM_{task} for each query. If the verifier assesses the output to be incorrect, the refinement process is triggered; otherwise, the output is used as is. This also saves computation on answers that are already correct.

To quantify the verifier's impact, we analyze three distinct scenarios: *ProRefine (verifier)*, our standard approach which employs $LLM_{feedback}$ to guide refinement; *ProRefine (no verifier)*, wherein the refinement process operates without verification; and *ProRefine (optimal verifier)*, guided by a perfect verifier (simulated using ground-truth labels).

Experiments and Results

We evaluate ProRefine on five reasoning tasks - object counting and word sorting from the BIG-Bench Hard benchmark (Srivastava et al. 2023), grade-school math problem-solving from GSM8K (Cobbe et al. 2021), math word problems from SVAMP (Patel, Bhattamishra, and Goyal 2021), and algebraic word problems from AQUARAT (Ling et al. 2017).

We experiment with three models - *Llama3.2-1B-instruct*, *Llama3.2-3B-instruct*, and *Llama3.1-8B-instruct* for LLM_{task}. *Llama3.1-70B-instruct* is used for feedback generation, prompt optimization, and evaluation. We select the values of hyperparameters $k = 10$ and $n = 25$.

We compare ProRefine against the zero-shot Chain-of-Thought (CoT) baseline and TextGrad (Yuksekgonul et al. 2024). It is essential to remember that TextGrad is a supervised fine-tuning method that utilizes both the training and validation sets.

Our results (Table 1) demonstrate that ProRefine significantly improves LLM_{task} performance over the zero-shot CoT baseline in all but one experiment, and it outperforms TextGrad in 11 out of 15 cases overall. For *Llama3.2-1B-instruct* model, ProRefine can significantly outperform CoT and TextGrad on 2 out of 5 datasets. For *Llama3.2-3B-instruct* model, ProRefine can outperform CoT and TextGrad on 3 out of 5 datasets with one significant result. For *Llama3.1-8B-instruct* model, ProRefine can outperform CoT and TextGrad on all 5 datasets with 4 significant results.

Using ProRefine with an optimal verifier significantly improves performance for all tasks, achieving the best results in 13 out of 15 cases, highlighting the critical role of verifier quality. Notably, the number of significant improvements increases with larger model sizes. We also observe that ProRefine enables smaller models, such as *Llama3.2-3B-instruct* and *Llama3.1-8B-instruct*, to approach the zero-shot performance of larger models like *Llama3.1-8B-instruct* and *Llama3.1-70B-instruct*, respectively.

References

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2024. Let's Verify Step by Step. In *The Twelfth International Conference on Learning Representations*.

Ling, W.; Yogatama, D.; Dyer, C.; and Blunsom, P. 2017. Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems. In Barzilay, R.; and Kan, M.-Y., eds., *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 158–167. Vancouver, Canada: Association for Computational Linguistics.

Patel, A.; Bhattamishra, S.; and Goyal, N. 2021. Are NLP Models really able to Solve Simple Math Word Problems? In Toutanova, K.; Rumshisky, A.; Zettlemoyer, L.; Hakkani-Tur, D.; Beltagy, I.; Bethard, S.; Cotterell, R.; Chakraborty, T.; and Zhou, Y., eds., *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2080–2094. Online: Association for Computational Linguistics.

Snell, C.; Lee, J.; Xu, K.; and Kumar, A. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. ArXiv:2408.03314 [cs].

Srivastava, A.; Rastogi, A.; Rao, A.; Shoeb, A. A. M.; Abid, A.; Fisch, A.; Brown, A. R.; Santoro, A.; Gupta, A.; Garriga-Alonso, A.; et al. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Torrey, L.; and Taylor, M. 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*, 1053–1060.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; ichter, b.; Xia, F.; Chi, E.; Le, Q. V.; and Zhou, D. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 24824–24837. Curran Associates, Inc.

Yuksekgonul, M.; Bianchi, F.; Boen, J.; Liu, S.; Huang, Z.; Guestrin, C.; and Zou, J. 2024. TextGrad: Automatic" Differentiation" via Text. *arXiv preprint arXiv:2406.07496*.