ProRefine: Inference-Time Prompt Refinement with Textual Feedback

Deepak Pandita[1,2], Tharindu Cyril Weerasooriya[1], Ankit Parag 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

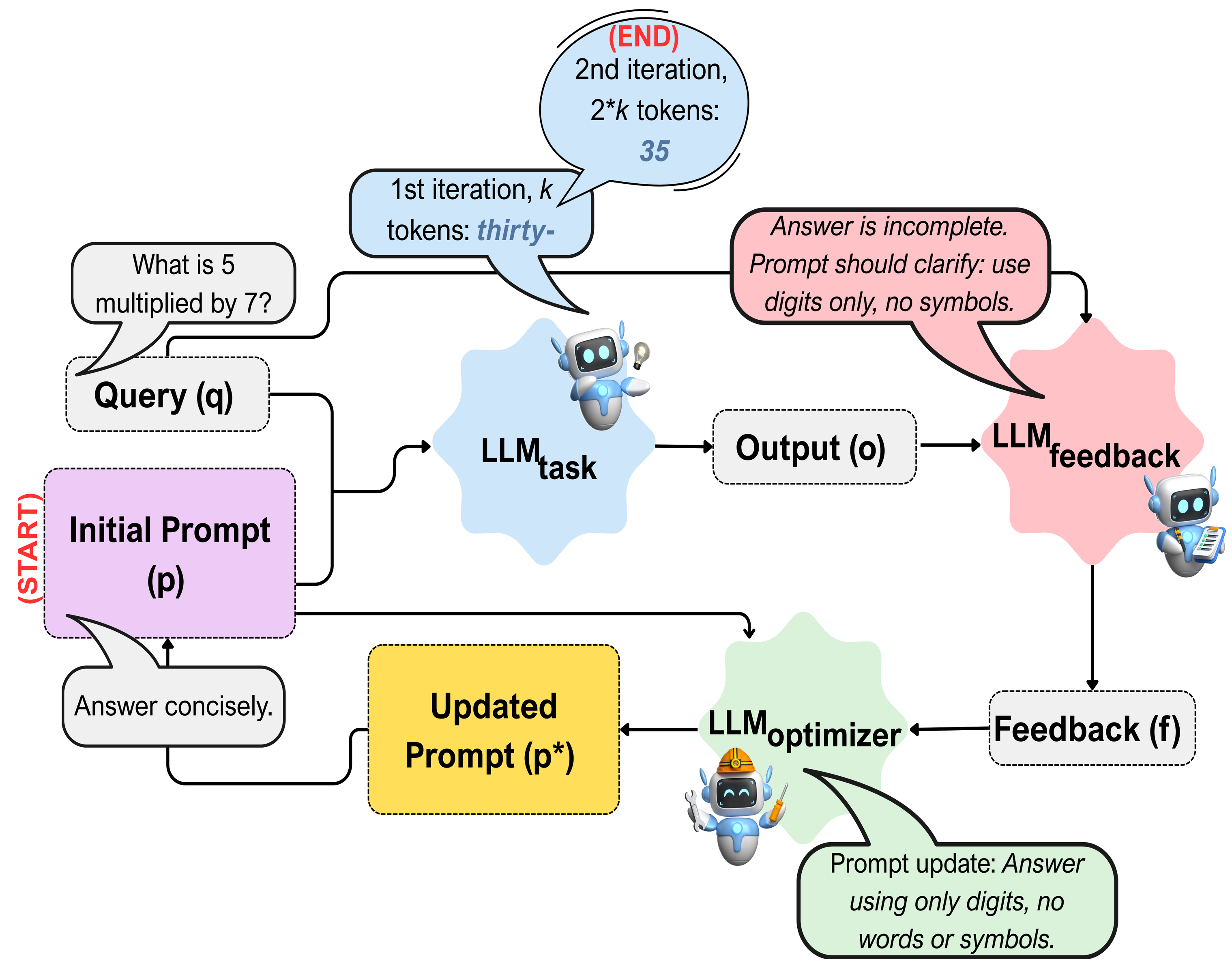
Introduction

Problem: Agentic workflows involving LLMs critically depend on high-quality prompts.

- Poorly designed prompts lead to sub-optimal performance and error propagation
- Reliance on continuous fine-tuning or exclusively using the largest, most expensive models is often computationally prohibitive

Solution: We introduce **ProRefine**, an innovative, task-agnostic method for *inference-time prompt optimization*.

- ProRefine dynamically refines prompts for multi-step reasoning tasks
- Uses an agentic loop of LLMs that generate and utilize textual feedback
- No additional training or ground-truth labels required

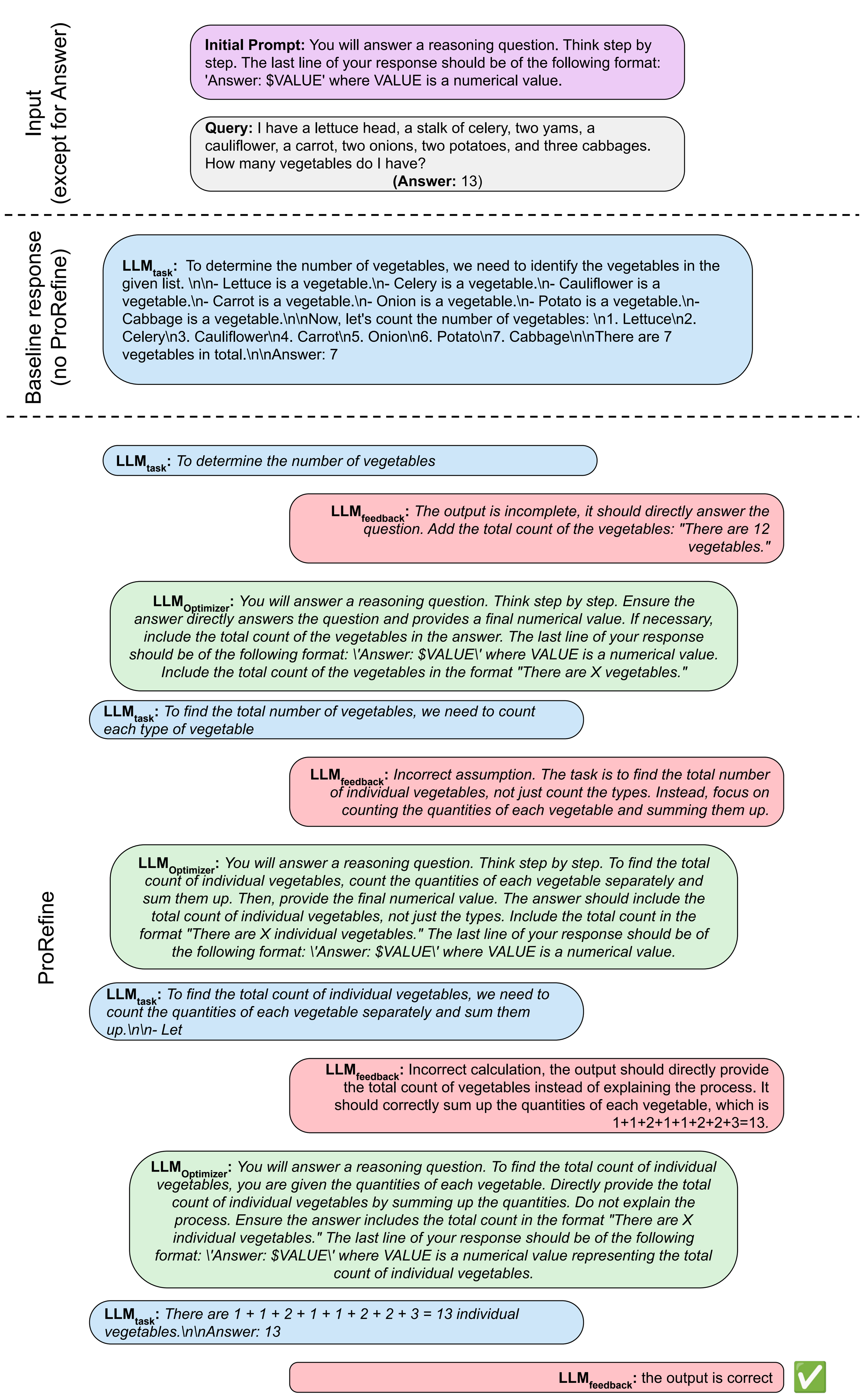


Overview of the ProRefine system, illustrating the iterative process of prompt optimization using feedback from LLMs. In each iteration, LLM_{task} extends its output by k tokens, enabling step-by-step feedback from $LLM_{optimizer}$ to progressively refine the prompt.

ProRefine

ProRefine extends the **Chain-of-Thought (CoT)** prompting approach by introducing a refinement loop with three specialized LLM agents.

Agent Role	Function	Benefit
LLM_{task}	The primary model generating the response. Extends its output by k tokens in each iteration.	Enables *step-by-step reasoning* and provides intermediate steps for critique.
$LLM_{feedback}$	Generates *textual feedback* for the output of LLM_{task}.	Provides *fine-grained, localized feedback* directly on the reasoning chain.
$LLM_{optimizer}$	Refines and updates the initial prompt for LLM_{task} based on the feedback.	Allows for *dynamic correction* before errors propagate through the reasoning chain.



Results

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]

Test Accuracy with 95% confidence intervals across five benchmark datasets and models. * and ** denote statistically significant improvements over one or two baseline methods, respectively. Results in bold indicate the highest accuracy for a dataset-method combination. † demonstrates the upper bound potential of the optimization loop and the impact of verifier quality. Llama-3.1-70B-instruct is employed for feedback generation, prompt optimization, and evaluation.

Key Findings

- ProRefine achieved significant performance gains ranging from 3 to 37 percentage points over CoT baselines
- Performance improvements scale with model size
- ProRefine allows smaller LLMs to approach the zero-shot performance of their larger counterparts
- High-quality verifier is crucial for improving task performance at test-time

Conclusion and Future Work

ProRefine offers a practical solution for multi-step agentic workflows by providing an on-demand "expert intervention" via the feedback loop.

- **Robustness:** The inference-time optimization process prevents errors from compounding
- **Versatility:** Suitable for black-box LLMs with API only access
- **Interpretability:** The textual feedback steps generated by $LLM_{feedback}$ offer insights into the reasoning correction process

Future Work

- Extend to tool-using agents
- Adaptive stopping criteria

References

- Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems (Ling et al., ACL 2017)
- Are NLP Models really able to Solve Simple Math Word Problems? (Patel et al., NAACL 2021)
- Beyond the imitation game: Quantifying and extrapolating the capabilities of language models (Srivastava et al., TMLR 2023)
- Chain-of-thought prompting elicits reasoning in large language models (Wei et al., NeurIPS 2022)
- Textgrad: Automatic" differentiation" via text (Yuksekgonul et al., arXiv preprint 2024)