Adapted Promptbreeder: Self-Referential Self-Improvement via Prompt Evolution

Rajan Singh
Department of Computer Science and Engineering
University of Minnesota, Twin Cities
Minneapolis, MN, USA
sing0780@umn.edu

Mason Greseth
Department of Computer Science and Engineering
University of Minnesota, Twin Cities
Minneapolis, MN, USA
grese031@umn.edu

***Abstract*—The performance of Large Language Models (LLMs) is highly sensitive to the prompts they receive, yet prompt design remains largely manual and ad hoc. Promptbreeder is a recently proposed evolutionary framework that automates prompt optimization by using LLMs to mutate and evaluate both task prompts and mutation strategies. In this work, we extend the Promptbreeder framework to the domain of text simplification, a task that presents unique challenges due to its subjective nature and evaluation complexity. Under computational constraints, we implement a simplified version of Promptbreeder and explore its ability to improve prompt performance over multiple generations. Our results highlight both the potential and limitations of evolutionary prompt search in generation tasks, and suggest directions for more adaptive, scalable prompt optimization methods.**

I. Introduction

Large Language Models (LLMs) like GPT-4 have demonstrated remarkable performance across a range of natural language processing tasks. However, their effectiveness is often heavily influenced by the quality of the prompts they receive. Recent strategies such as Chain-of-Thought (CoT) [1] prompting and Automatic Prompt Engineering (APE) [2] have shown that even small prompt modifications can significantly impact model behavior. Despite these advances, the design of prompts remains largely manual and brittle, requiring substantial human intuition and task-specific tuning.

To address this limitation, evolutionary approaches such as **Promptbreeder** [3] have emerged as promising frameworks for *automated prompt optimization*. Promptbreeder introduces a self-referential evolutionary loop in which LLMs are not only used to evaluate and mutate task prompts, but also to evolve the very mutation prompts that control how mutations are generated. This leads to a bootstrapping mechanism capable of improving prompts over time with minimal human oversight.

While Promptbreeder has shown strong results in reasoning and classification tasks, its application to other NLP domains remains underexplored. In this project, we investigate the feasibility of applying Promptbreeder to the domain of **text simplification**, where the goal is to rewrite complex input text into a simpler and more accessible version while preserving meaning. Text simplification introduces unique challenges for prompt optimization due to its subjective evaluation metrics and the need for semantic fidelity.

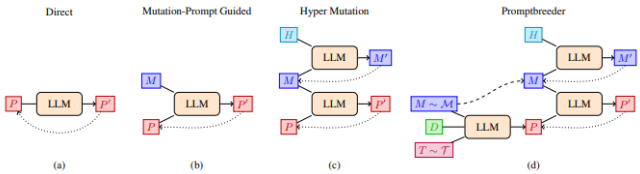

Figure 1: Promptbreeder structure

Our work involves implementing a lightweight version of Promptbreeder under compute constraints, analyzing its performance in the text simplification domain, and proposing modifications to the mutation strategy.

II. Problem Statement

Hand-crafted prompt strategies, such as Chain-of-Thought Prompting, significantly enhance the reasoning abilities of

Large Language Models (LLMs) but often remain suboptimal. There is a need for a more adaptive and automated approach to optimizing prompts for various domains. The challenge lies in developing a mechanism that can iteratively improve task prompts without manual intervention, ensuring enhanced performance across diverse reasoning and classification tasks. We are basing our strategy on the Promptbreeder mechanism and aim to extend it to a new domain while performing an analysis on the specifics of the framework while working under compute constraints.

III. Approach

The original Promptbreeder framework introduces a self-referential evolutionary algorithm in which a population of task-prompts and mutation-prompts are evolved over multiple generations to improve performance on a given task. A key innovation in Promptbreeder is its use of 9 distinct mutation operators, which fall into five broad categories: direct mutations, estimation-of-distribution mutations, hypermutations (i.e., mutations of mutation-prompts), Lamarckian mutations (from correct model outputs), and prompt crossover/context shuffling. These operators are applied stochastically during evolution and are designed to maintain diversity and avoid early convergence to suboptimal prompt strategies. The framework evaluates prompt fitness by using an LLM to solve a batch of tasks and then retains or mutates prompt units based on their relative performance.

In our implementation, we replicate the core evolutionary framework of Promptbreeder but make two key changes. First, due to compute constraints and to better isolate the effect of operator diversity, we limit our mutation set to only three mutation operators:

First-order Hyper-Mutation: We prepend the hyper-mutation prompt — "Please summarize and improve the following instruction:" — to the best performing mutation prompt thus far, prompting the LLM to generate a new mutation prompt. This newly created mutation prompt is then applied to the task prompt of the unit. This setup allows us to assess the influence of the hyper-mutation mechanism by observing how its resulting mutation prompt impacts the quality of the evolved downstream task prompt.

Zero-order Hyper-Mutation: Here we concatenate the original problem description to a randomly sampled thinking style from our original list and give that to the LLM to create a new mutation prompt. This generates mutation prompts in a similar fashion to the initialization prompts.

Lamarckian Mutation: This approach represents a 'Lamarckian' mutation operator. In this method, we prompt the LLM with a correct chain of reasoning that leads to a valid answer using the template: *"I gave a friend an instruction and some advice. Here are the correct examples of his workings out: <<correct working out>>. The instruction was:"* This effectively reverses the process by inferring a task prompt from the provided solution. This operator is especially valuable when the original problem description is missing, inadequate, or ambiguous. To match the text simplification domain, the string was changed to *"I gave a friend this sentence <<complex sentence>> and some advice. Here is the correct example of him simplifying it: <<correct working out>>. The instruction was:"* The prompt then has context and this limits hallucinations.

By comparing performance across runs using just one mutation operator versus all three implemented operators, our goal is to test a central claim of the Promptbreeder paper: that mutation operator diversity is critical to the success of the framework. Specifically, we hypothesize that restricting mutation variety will reduce the system's ability to explore and escape local optima, leading to lower prompt quality and slower improvement over generations.

To evaluate this, we apply the framework to the text simplification domain—a departure from the original paper, which focused on reasoning and classification benchmarks. We adapt Promptbreeder to generate prompts that guide an LLM in simplifying complex input sentences. Fitness is measured using a lightweight reward function based on sentence-level metrics that reflect simplicity and preservation of meaning.

IV. Results and Analysis

In evaluating our framework, we identified key aspects of the framework and compared results with and without those elements. In Figure 2, the presence of hyper-mutation is shown to be imperative to the success of the framework. This is due to the need for exploration to find new, more successful mutation prompts. The increasing rate of change in the run without hyper-mutation shows the compounding effects of hallucinations. Since the LLM receives only the previous generation of prompts for each new generation, the poor mutation prompts will hurt the framework's performance additively over generations.

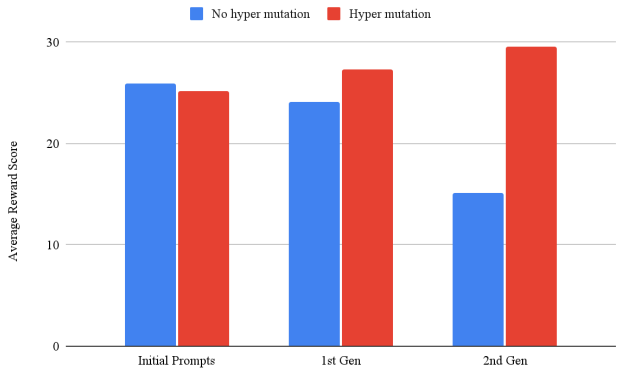

Figure 2: Average Reward Score Based on Presence of Hyper-Mutation

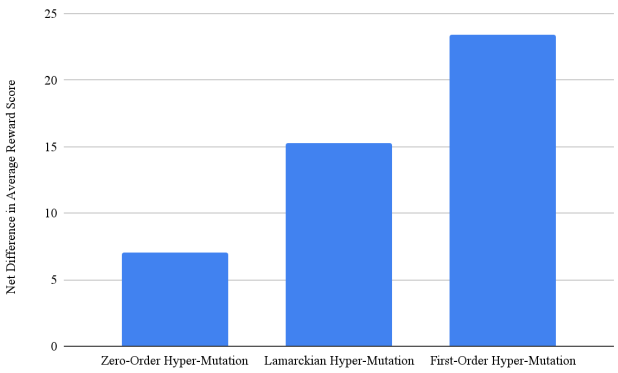

Figure 4: Net Difference in Average Reward Score Based on Hyper-Mutation Technique

The results in Figure 2 are based on first-order hyper-mutation and the change of the average reward score shows how hyper-mutation allows for the framework to deal with hallucinations. Without the diversity that hyper-mutation provides, the framework suffers from hallucinations that build on each other throughout the generations, resulting in a lower average reward score. In our preliminary results, we utilized the Google Gemini API to utilize the Gemini 1.5 Flash model, but we wanted to compare the performance of this model to GPT-3.5 Turbo. Those results are shown in Figure 3 and it is clear that OpenAI's GPT-3.5 Turbo performs better with this framework for the text simplification task.

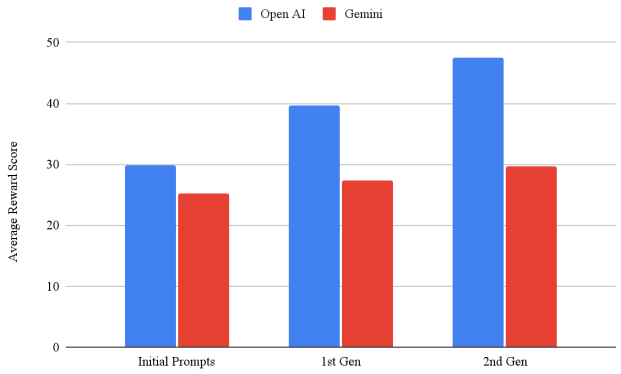

Figure 3: Average Reward Score Based on LLM Utilized

Based on the results of the LLM comparison, we utilized GPT-3.5 Turbo to get our final results. As described earlier, we applied three different hyper-mutation techniques and compared the results to discern which technique helped increase the average reward score the most. The results in Figure 4 show that first-order hyper-mutation is the best of the techniques by a wide margin.

Zero-order hyper-mutation struggles to improve the mutation prompt in a meaningful way because a random thinking style is not tailored to the specific task. It is quite simple, but it does not create enough exploration into the prompt space. Lamarckian hyper-mutation gives the LLM some information of previous correct ideas, but it still has limited exploration and does not give the context of the actual mutation prompt used. First-order hyper-mutation explores enough on top of exploiting previous success to find the natural ceiling of average reward scores. In text simplification, the reward score can only be optimized so much (simplifying the sentence without losing its original meaning) and, while the framework shows improvement over generations, the success of first-order hyper-mutations shows that exploration through mutation and hyper-mutation is the key aspect that leads to the framework's success.

V. Conclusion

In this project, we implemented a simplified version of the Promptbreeder framework and applied it to the task of text simplification. Our goal was to evaluate how well the framework performs under compute constraints and reduced mutation diversity, and to test the importance of specific design elements such as hyper-mutation. Through a series of controlled experiments, we found that first-order hyper-mutation plays a critical role in maintaining prompt quality across generations by combating compounding hallucinations and enabling effective exploration of the prompt space. Our results also show that GPT-3.5 Turbo outperforms Gemini 1.5 Flash in this setting, and that zero-order and Lamarckian hyper-mutations underperform compared to first-order techniques due to limited contextual awareness and task specificity.

While we did not implement the full range of mutation operations proposed in the original Promptbreeder paper, our findings support its central claim: mutation operator diversity is essential for sustained performance improvement. This work highlights both the promise and the sensitivity of self-referential prompt evolution systems and lays the groundwork for future improvements.

Acknowledgments

The work is accomplished as part of the CSCI 8980 AI for Sequential Decision Making class at the University of Minnesota, Twin Cities, and is a replication of the original Promptbreeder paper [3] developed by researchers at Google DeepMind.

We would like to extend a special thank you to Alex Slinger for providing us with his OpenAI API key which allowed us to scale our project further and improve results.

References

[1] Zhang, Zhuosheng, et al. "Automatic chain of thought prompting in large language models." *arXiv preprint arXiv:2210.03493* (2022).

[2] Zhou, Yongchao, et al. "Large language models are human-level prompt engineers." *The Eleventh International Conference on Learning Representations*. 2022.

[3] Fernando, Chrisantha, et al. "Promptbreeder: Self-referential self-improvement via prompt evolution." *arXiv preprint arXiv:2309.16797* (2023).