ADVANCING PROBLEM-SOLVING IN GPT-4 THROUGH SELF-CORRECTION MECHANISMS

David Laziuk
Stevens Institute of Technology
Hoboken, NJ
dlaziuk@stevens.edu

ABSTRACT

This study presents an innovative experiment with GPT-4, exploring self-correction in language models as a means to enhance problem-solving capabilities for complex, computational tasks. The research focuses on hard-level problems from LeetCode, initially tackled using Chain of Thought (COT) prompting to generate solutions. Upon identifying these solutions as inadequate, the study employs a method of feeding the COT steps back into the model for self-correction. This iterative process, although found to be approximately three times more resource-intensive, demonstrates the potential of specialized runs in effectively addressing complex problems that demand high test-time computation. The findings suggest that self-correction could be a promising avenue for refining the problem-solving skills of language models, especially in scenarios where precision and accuracy are paramount, albeit with increased computational costs. This approach provides a foundation for future research in enhancing the problem-solving efficiency of AI systems in specialized, computationally challenging environments.

1 Introduction

The rapid advancement in language models like GPT-4 has opened new avenues in computational problem-solving. This study focuses on an innovative aspect of these models: enhancing problem-solving capabilities through self-correction, particularly for complex computational tasks. The primary objective is to explore whether a large language model can improve its performance by re-analyzing and correcting its own output, a process that diverges from traditional methods which often involve compiling outputs through external verification.

Central to this exploration is the hypothesis that for complex problems, typically solved through numerous inference runs, a model's ability to self-correct could lead to more effective solutions, even if not immediately more efficient in terms of computational resources. The research employs a unique methodology: solving a hard-level problem from LeetCode using GPT-4's Chain of Thought (COT) prompting to generate initial solutions, followed by a self-correction phase where the model re-evaluates and refines these solutions based on its previous output. This iterative self-correction process, despite being resource-intensive, aims to demonstrate the potential for improving accuracy and precision in problem-solving.

The relevance of this study lies in its preliminary examination of self-correction in language models. It represents an initial foray into understanding how self-verification and correction can be integrated into the problem-solving processes of AI systems. Although the research does not delve into the efficiency aspect due to resource constraints, it sets the stage for future investigations into how self-correcting mechanisms could potentially simulate more complex models efficiently.

This study is significant as it offers insights into an often unexplored aspect of AI problem-solving - the ability of a model to introspect and improve upon its own logic and reasoning. It contributes to the broader discourse on advancing AI capabilities, specifically in addressing intricate problems where the conventional methods may fall short. The findings from this research could pave the way for developing more sophisticated, self-reliant AI systems capable of handling tasks of higher complexity with improved accuracy.

2 Background

The concept of enhancing the capabilities of language models through verification and correction mechanisms is not novel, but its application and potential have been significantly under-explored. This research draws inspiration from two seminal works by OpenAI, which collectively lay the foundation for the current study's approach and objectives.

The first pivotal work, "Training Verifiers to Solve Math Word Problems" (Cobbe et al., 2021) demonstrates the profound impact of verification on the performance of language models. OpenAI's findings indicate that verification, even with a smaller 6B model, can outperform a fine-tuned 175B model, effectively providing a performance boost equivalent to a 30x increase in model size. This remarkable efficiency in leveraging verification underscores the potential of using similar methods to enhance the problem-solving capabilities of models like GPT-4. Additionally, the study highlights the benefits of limiting the generator's training to prevent overfitting and the potential for integrating generator and verifier models. This insight is crucial for this research, as it suggests the feasibility of a model performing both generation and self-correction tasks.

Moreover, OpenAI's observation that the performance improvement plateaus beyond a certain number of completions (400 in their study) at test time raises an intriguing possibility. If too many completions can lead to adversarial solutions that fool the verifier, then a more focused approach, like self-correction, could potentially provide a more efficient and accurate solution without the need for generating excessive outputs.

The second influential work, "Let's Verify Step by Step" (Lightman et al., 2023) further enriches the conceptual framework of this study. It demonstrates that verifying each step in a Chain of Thought (COT) process individually results in significantly improved performance. This step-by-step verification aligns closely with the self-correction approach proposed in this research. However, the additional step of fine-tuning the generator to adhere to a desired output format, while beneficial, is beyond the scope of this study's resources. This limitation underscores the exploratory nature of the current research, aiming to investigate the potential of self-correction without the added advantage of fine-tuning.

Drawing from these foundational works, this research proposes a novel approach: integrating the roles of a generator and a corrector within a single model, specifically GPT-4. By allowing the model to not only generate solutions but also to self-correct them, the study explores a new dimension of efficiency in problem-solving. This approach is particularly relevant for complex computational tasks, where rapid and accurate solutions are paramount, and the feasibility of external verification is limited.

In summary, the background of this research is anchored in the promising results of using verification to enhance model performance and the concept of step-by-step problem-solving verification. Building upon these ideas, this study ventures into uncharted territory, seeking to determine whether a language model can effectively self-correct its logic, potentially leading to a more efficient problem-solving process, especially in scenarios where the ground truth is unknown or it difficult to test more than once.

3 Methodology

This study employs a structured experimental methodology to investigate the feasibility and effectiveness of self-correction in GPT-4 for solving complex algorithmic problems. The process comprises several distinct phases, each designed to rigorously test and refine the model's problem-solving capabilities.

3.1 Selection of Problems:

Two problems were selected from LeetCode for this experiment, chosen based on their difficulty level and recency. The primary problem, "2945." (*Find Maximum Non-decreasing Array Length*, n.d.) is classified as "Hard" and has a notably low completion rate of 12.6% on LeetCode. The secondary problem, "2911" (*Minimum Changes to Make K Semi-palindromes*, n.d.) has a higher completion rate of 32.4% and serves as a comparative benchmark.

3.2 Model Parameters:

The model used for the experiment was gpt-4-1106-preview, with the following parameters:

- Type: Chat
- Temperature: 1 (default)
- Maximum length: 1500 (arbitrarily chosen for sufficient coverage)

- Stop sequences: None (default)
- Top P: 1 (default)
- Frequency penalty: 0 (default)
- Presence penalty: 0 (default)

3.3 Initial Solution Generation:

For each problem, the model was tasked with generating solutions following a specific call structure to ensure a consistent approach. The model was instructed to first break down the logic of the problem in a numbered list, ensuring each step was independently understandable, and then to write the code after a "CODE:" flag. For the "2945" problem, 100 responses were generated, while for the "2911" problem, 10 responses were generated.

3.4 Evaluation of Initial Solutions:

The generated code and the corresponding Chain of Thought logic were parsed from the responses. Each solution was then evaluated against the three base examples provided in LeetCode. This evaluation served to establish the baseline effectiveness of the model's solutions and to confirm the complexity of the problems. However, Leetcode evaluates on hundreds of test cases, these base three will only provide some preliminary insight.

3.5 Logical Correction Phase:

The logical steps extracted from the initial solutions were then passed back to the model with a new system definition focused on assessing and correcting the logic. The model was tasked with identifying flaws in each step, based on the original problem, and rewriting them to ensure logical coherence. Due to resource constraints, this correction was done for the entire logical breakdown rather than on a step-by-step basis.

3.6 Generation of Revised Solutions:

Using the corrected logic, new solutions were generated. The model was instructed to assess the problem using chain of thought prompting, and then the corrected logic was passed in as the assistant, not the user. Generating code as if the model initially generated the corrected logic on its own.

3.7 Re-evaluation of Revised Solutions:

The new code generated from the revised logic was evaluated using the same method as the initial solutions, allowing for a direct comparison of the effectiveness of the self-correction process.

This methodology aims to provide a comprehensive evaluation of the self-correction capabilities of GPT-4 in addressing complex algorithmic challenges. By iterating through the generation, evaluation, correction, and re-generation phases, the study seeks to demonstrate the potential for language models to enhance their problem-solving accuracy through self-reflection and refinement.

4 Results

4.1 Primary Experiment: "2945. Find Maximum Non-decreasing Array Length"

Initial (Uncorrected) Solutions:

- Total initial solutions attempted: 100
- Solutions that failed to parse: 13
- Solutions that failed to run: 1
- Successful solutions: 0

The initial attempts to generate solutions for the complex problem presented significant challenges, with a notable number of responses being uninterpretable or failing to execute.

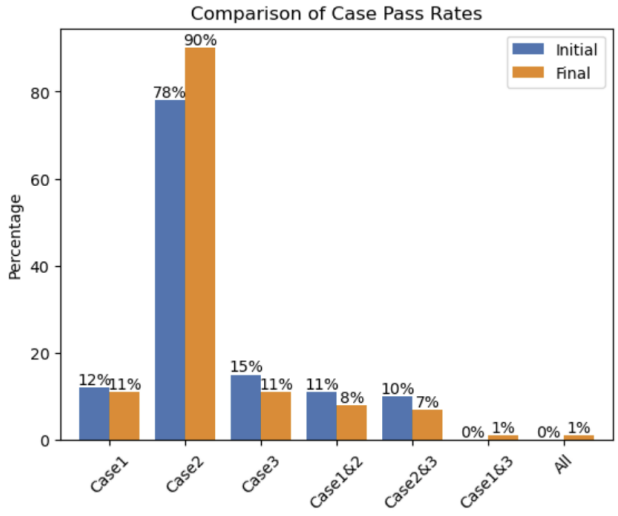

Figure 1: Primary Case Pass Rates

Corrected Solutions:

- Total final solutions attempted after correction: 100
- Solutions that failed to parse: 0
- Solutions that failed to run: 0
- Successful solutions: 1

After undergoing the self-correction process, the model's capacity to produce executable and logically coherent solutions saw a considerable improvement. While there was a notable reduction in parsing and run-time errors, resulting in one solution successfully meeting the initial three base evaluation criteria, it ultimately did not pass the comprehensive LeetCode evaluation consisting of 553 test cases. Nonetheless, this outcome is still regarded as a minor success within the scope of this study, indicating a positive trend in the model's problem-solving trajectory post-correction.

Case Pass Rates: As shown in Figure 1, the self-correction process resulted in an increased pass rate in Case 2 when compared to the initial attempts. Critically, combined cases now contained a passing solution, indicating that the self-correction mechanism has the potential to enhance the model's performance significantly.

Computational Costs:

- Average cost per regular solution: $0.0222
- Average cost per corrected solution: $0.0724

The cost associated with the self-correction process was approximately 3.3 times higher than the regular solution generation. Despite this increase, the successful correction and subsequent pass of one solution indicates that there is merit to the self-correction approach, particularly in the context of solving highly complex problems where the accuracy of the solution is of paramount importance.

The results highlight the potential for self-correction to improve the quality of solutions generated by LLMs like GPT-4 for complex algorithmic problems. While the approach currently incurs higher computational and financial costs, it opens up the possibility for more effective problem-solving strategies, particularly in situations where accuracy is critical and the computational expense can be justified.

4.2 Secondary Experiment: "2911. Minimum Changes to Make K Semi-palindromes"

Initial (Uncorrected) Solutions:

- Total initial solutions attempted: 10

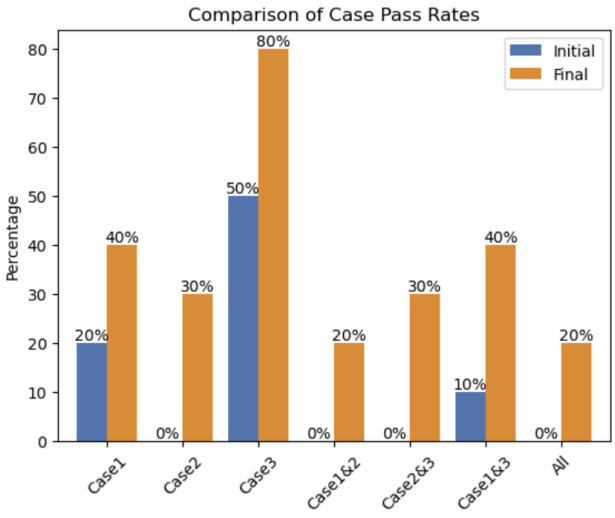

Figure 2: Secondary Case Pass Rates

- Solutions that failed to run: 2
- Successful solutions: 0

In the secondary experiment, the initial solutions encountered execution failures, with no successes in the sample size, pointing to the complexity of the problem and the challenges faced by the model in generating viable solutions.

Corrected Solutions:

- Total final solutions attempted after correction: 10
- Solutions that failed to run: 2
- Successful solutions: 2

The application of the self-correction process culminated in a 20% success rate, with two corrected solutions fulfilling the initial evaluation criteria. Despite this advancement, it is noteworthy that these solutions did not withstand the full LeetCode assessment involving a more extensive test suite. Even so, this development is considered a stride forward from the initial attempts, underscoring the prospective value of self-correction in bolstering the model's capacity for complex problem-solving tasks.

Case Pass Rates: Figure 2. demonstrates a substantial enhancement in the case pass rates for the corrected solutions in comparison to the initial attempts. This significant improvement is uniformly observed across all test cases, with particularly stark improvements noted in Cases 2, 1&2, 2&3, and the aggregate of all cases. The initial attempts yielded no successful outcomes for these cases, whereas the corrected solutions exhibited a marked ability to pass these specific test scenarios, indicating the potential effectiveness of the self-correction process in addressing previously unresolved challenges.

Computational Costs:

- Average cost per regular solution: $0.0289
- Average cost per corrected solution: $0.0863

The costs associated with the secondary experiment followed a similar pattern to the primary experiment, with the self-correction process incurring approximately 3.0 times the expense of the initial solution generation. Despite the higher costs, the increase in successful outcomes demonstrates that self-correction has a tangible impact on the model's effectiveness in solving complex algorithmic challenges.

The secondary experiment further supports the findings from the primary experiment, suggesting that self-correction can be a valuable tool in enhancing the accuracy of solutions provided by language models. The associated costs, while

significantly higher, may be justified by the increased quality of the solutions, particularly for problems where the solution accuracy is critical and traditional solving methods may fall short.

5 Discussion

The experiments conducted, albeit constrained by resource limitations, offer promising indications for the hypothesis that self-correction can significantly enhance the problem-solving capabilities of large language models like GPT-4. The observed improvements in solution quality, following the self-correction process, suggest that models are capable of iteratively refining their outputs towards more accurate solutions.

Ideally, with access to more resources, one could fine-tune GPT-4 to better conform to specific output formats. This refinement could leverage strategies from "Let's Verify Step by Step," enabling the model to self-correct on a more granular, step-by-step basis. Such fine-tuning would likely yield even greater improvements in the model's problem-solving accuracy and efficiency.

In the context of this study, direct testing of solutions was feasible, rendering external verification unnecessary. However, in scenarios dealing with more abstract problems or in cases where solutions can be tested only once, verification could play a crucial role. Post self-correction, a verification phase could be introduced to assess and rank the corrected solutions, forming a voting system that synthesizes numerous attempts to distill a final, optimal solution.

The potential implications of this methodology extend far beyond current applications. By generating, correcting, and verifying thousands of solutions, it is conceivable that a model could tackle problems that have eluded human problem-solvers. Alternatively, this approach could provide insight into the capabilities of future, more complex models, potentially revealing new strategies and pathways to solutions currently unimaginable.

This study, therefore, not only tests the bounds of current AI problem-solving techniques but also opens a window into the future, positing a model that could one day operate at a level of complexity and sophistication that rivals, or even surpasses, human expertise. The vision of AI systems solving novel, intricate problems could redefine the boundaries of computational problem-solving and expand the horizons of what is considered achievable with machine intelligence.

6 Conclusion

This exploratory study has ventured into the territory of self-correction within large language models, with a focus on GPT-4's capabilities in solving complex algorithmic problems. Despite resource constraints, the results obtained provide initial evidence supporting the hypothesis that self-correction can improve the accuracy of problem-solving in LLMs. The experiment demonstrated that through iterative refinement of outputs, the model could indeed enhance the quality of its solutions, albeit at an increased computational cost.

While the present study was limited to direct solution testing, the concept of incorporating a verification system in more abstract or once-testable scenarios presents an intriguing avenue for future research. The notion of compiling thousands of generated, corrected, and verified solutions to converge on an optimal solution opens up the potential for AI to solve problems that have thus far remained unsolved by humans.

In conclusion, this study serves as a preliminary indication that self-correction mechanisms can play a vital role in advancing the problem-solving abilities of language models. It lays the groundwork for future studies to build upon, particularly in fine-tuning models to adhere to desired output formats and in developing sophisticated verification systems. Ultimately, the implications of this research extend towards a future where AI could autonomously solve highly complex problems, contributing to the advancement of knowledge and technology in ways that are currently difficult to fully envision.

References

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., ... Schulman, J. (2021). *Training verifiers to solve math word problems.*

Find maximum non-decreasing array length. (n.d.). Retrieved from `https://leetcode.com/problems/find-maximum-non-decreasing-array-length/`

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., ... Cobbe, K. (2023). *Let's verify step by step.*

Minimum changes to make k semi-palindromes. (n.d.). Retrieved from `https://leetcode.com/problems/minimum-changes-to-make-k-semi-palindromes/`