# Final Report – SELF-FILTER

Bowen Yang UNI: by2365

April 2024

## 1 Introduction

In this paper, I introduce SELF-FILTER, an innovative algorithm designed to enhance the performance of large language models (LLMs) in solving math word problems (MWPs). MWPs require models to integrate language understanding with numerical reasoning, a challenge that existing LLMs like GPT-3.5 and GPT-4 are already tackling in various domains such as coding and summarization. The key advancement of SELF-FILTER lies in its ability to generate multiple solution candidates and utilize a specialized agent, the PCA (Principal Candidate Agent), to select the most accurate answer automatically.

The SELF-FILTER pipeline is an exploratory project designed to enhance the capabilities of less expensive models like GPT-3.5 in solving MWPs. This initiative holds practical implications in educational technology by providing a more affordable tool for teaching and learning mathematics. While SELF-FILTER may not revolutionize the field, it seeks to make reliable problem-solving tools more accessible, thereby contributing to the democratization of education.

SELF-FILTER's methodology involves forcing LLMs to produce diverse answers through techniques like random few-shot prompting (R-FSP) and chain of thought (COT). Subsequently, PCA evaluates these answers to determine the most plausible solution. This process, although intensive, ensures that even less advanced LLMs can achieve and exceed their typical performance thresholds on MWPs, with the algorithm proving effective on higher-end models as well. For a detailed overview of the system's architecture and functionality, please refer to Figure 1 below. This work aims to provide the community with a robust, scalable solution that enhances the utility of LLMs across a range of applications.

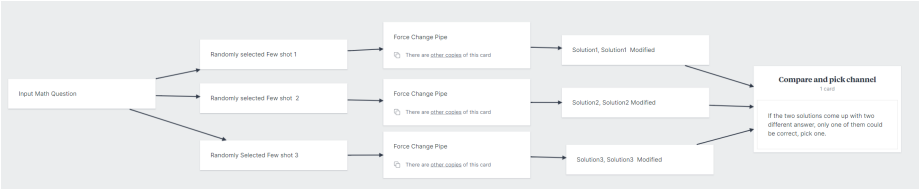


Figure 1: High-Level Architecture of SELF-FILTER

For detailed implementation of the project, please refer to: https://github.com/BWN133/SEFL-FILTER

## 2 Exploring Path

In this section, I will describe my exploration this semester into enhancing the reasoning skills of LLMs for solving elementary-level math word problems. I initially experimented with a Retrieval-Augmented Generation (RAG)-based [Lewis et al., 2020] approach, recognizing that math questions at the primary school level often follow predictable patterns and are relatively straightforward. I thought that by classifying problems into distinct categories, I could use similar questions as few-shot prompts to boost the model's efficiency. To test this idea, I developed a rudimentary categorizer to organize the GSM8K [Cobbe et al., 2021] dataset and then used categorized problems as prompts to see if performance improved. Unfortunately, the experiment showed no significant enhancements. I suspect this was due to the categorizer's lack of sophistication, which likely failed to provide relevant information. This realization led me to conclude that achieving sufficiently detailed categorization was nearly as complex as solving the math problems themselves, making this method impractical.

I then explored an alternative strategy aimed at learning from the model's past mistakes without resorting to reinforcement learning. I aimed to increase accuracy by assembling a collection of instances where the model struggled into some format and using those as learning aids for new problems. However, this approach also required precise categorization to align relevant insights with the questions at hand, ultimately proving to be impractical.

After exploring various methods, I shifted my focus towards guiding the thought processes of language models by employing specific prompts. Initially, I identified recurring errors made by GPT3.5-turbo to derive inspiration. Through this analysis, I made several notable observations:

1. Less powerful models struggle significantly more with backward logic compared to forward logic.

2. These models exhibit a limited capacity to recall important text-based information mentioned earlier in the conversation, particularly when minimal crucial data is provided. For instance, consider the question: "John

drives for 3 hours at a speed of 60 mph and then turns around because he realizes he forgot something very important at home. He tries to get home in 4 hours but spends the first 2 hours in standstill traffic. He spends the next half-hour driving at a speed of 30mph, before being able to drive the remaining time of the 4 hours going at 80 mph. How far is he from home at the end of those 4 hours?" from GSM8K [Cobbe et al., 2021], where GPT3.5 consistently misunderstands the implication of turning around and misinterprets the relationship between the computed distances.

3. The limited real-world understanding of less powerful models significantly impacts their ability to solve math word problems. For example, in another question in GSM8k [Cobbe et al., 2021] the question: "Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150 percent. How much profit did he make?", GPT3.5 frequently errs by incorrectly including the $50,000 repair cost in the profit calculation.

Following these insights, I reviewed numerous scholarly articles and experimented with several techniques, including but not limited to:

1. Instructing LLMs to summarize the meaning of each quantity in an MWP and then solve the problem. No improvements were observed, as LLMs continued to overlook crucial information.

2. Directing LLMs to both summarize the meanings and relationships of each quantity before solving the problem. This method also failed to show improvements, as LLMs continued to misinterpret relationships during the extraction phase.

3. Having LLMs rephrase the question before attempting to solve it, which was ineffective as LLMs often altered the meaning of the problem, likely because they did not understand it initially.

4. Prompting LLMs to process MWP sentence by sentence, storing information deemed important from each sentence. Solving the problem when the last sentence was provided did not work, as the reduced content hindered the attention mechanism's ability to focus appropriately.

5. Employing quantities in the question to generate variables initially, then generating Python code to solve the problem. LLMs persisted in their original errors by writing incorrect code.

6. Requesting LLMs to simplify the question's description before solving it based on the simplified question. Often, essential information was omitted as LLMs failed to recognize its importance.

After evaluating the aforementioned methods, I concluded that this particular approach is impractical. Firstly, it is challenging to devise a logical structure that universally applies to all types of questions. Secondly, machines process

information differently than humans, and attempts to mimic human thought processes do not typically enhance their performance and may sometimes impair it.

Despite the lack of success in my previous efforts, I have identified some beneficial characteristics of GPT-3.5-turbo, which are listed below:

1. The simpler and shorter the proposed task, the better the performance of the model.

2. Different few-shot prompts lead LLMs to generate varied content. My hypothesis is that few-shot prompts do not teach LLMs new information but rather activate specific neuronal pathways within the model.

Based on the principles outlined above, I have developed an algorithm called SELF-FILTER. This algorithm leverages multiple agents to enhance the performance of smaller LLMs on MWPs.

# 3 SELF-FILTER

The architecture of SELF-FILTER is straightforward and comprises two distinct stages: the candidate generation stage and the filtering stage.

During the candidate generation stage, the objective is to produce as many unique solutions as possible. Given that GPT-3.5 has considerably weaker logical capabilities compared to GPT-4 and is prone to various errors, I suspect that encouraging GPT-3.5 to generate numerous unique solutions will enhance the likelihood of obtaining at least one correct answer. Initially, $n$ separate agents are assigned to solve the given MWP. Each agent receives three distinct random few-shot prompts along with the question. To maximize the probability of generating diverse solutions, completely different few-shot examples are provided to each agent. After all $n$ agents complete their reasoning, $n$ solutions are produced, each including a reasoning field from the agent's COT reasoning and an answer field with the agent's final answer. Each output is then sent to another set of $n$ enhancement agents, which are instructed to assume the proposed solution is incorrect and to propose alternative solutions. Consequently, a total of $2n$ solutions are generated and moved to the filtering stage.

For the filtering stage, I designed an Arena match-like filtering algorithm. Initially, a Pick Correct Agent (PCA) is defined, which receives two solution candidates with differing answers and selects one as the correct answer. PCA is limited to comparing only two candidate solutions at a time, as I discovered that providing more candidates simultaneously reduces the model's accuracy. Repetitive answers among the candidate solutions are eliminated, and the unique solution candidates are subjected to PCA in an Arena Match format. In this format, the PCA-selected correct solution remains on stage, while the others are excluded. Ultimately, the single solution that remains on stage is output as the final result.

# 4 Comparison

There are two works worth mentioning that have conceptual similarities to my own. Since all works mentioned below utilize different models from mine and I cannot test my system across all these models for cost concern, this section will focus more on theoretical analysis rather than providing quantitative proof.

In [Madaan et al., 2023], SELF-REFINE is introduced, which takes an input sequence, generates an initial output, provides feedback on the output, and refines the output based on the feedback. I suspect this approach is particularly effective when models tend to make "careless" mistakes, which are errors unlikely to be repeated in nine out of ten instances. From my experiments, it appears lots of errors made by less capable models, such as GPT-3.5, are not "careless" mistakes. There is a high likelihood that the model will simply overlook the mistake during the feedback stage and repeat the same error.

[Wang et al., 2023] introduces a concept called self-consistency, which initially generates a set of responses and then uses a majority vote to select the correct response. Personally, I believe that a majority vote is not an ideal solution-picking strategy for lower-level models like GPT-3.5. Theoretically, a majority vote would be effective if the model achieves over 50 percent accuracy on all problems. Although GPT-3.5 has demonstrated the ability to exceed 50 percent accuracy on the entire GSM8K dataset [Cobbe et al., 2021], this does not guarantee that it will consistently achieve this level of accuracy on each individual problem. However, the ablation study mentioned in the section below will disprove my theory, showing that self-consistency remains an exceptionally effective algorithm.

# 5 Experiments

For all experiments described below, I am utilizing the first hundred data points from the GSM8K dataset [Cobbe et al., 2021] as the benchmark due to cost concerns. Since the performance of LLMs can vary, I will conduct each experiment three times and use the average result for analysis.

To evaluate the effectiveness of the SELF-FILTER algorithm compared to the traditional Chain of Thought (COT) method, I use the GSM8K dataset [Cobbe et al., 2021] as a benchmark and compare the performance of my SELF-FILTER system with $n = 3$ agents against a traditional COT system using 7-shot prompting. Both systems utilize GPT-3.5-turbo as the base model. A significant improvement has been observed with SELF-FILTER.

Subsequent ablation studies were conducted using the same data set to further dissect what contributes most to the effectiveness of SELF-FILTER.

Firstly, I examined the impact of random few-shot prompts on the number of unique solutions generated. I developed a No-Random version of the SELF-FILTER pipeline, which does not use random few-shot samples. Instead, all $n$ agents use the same set of few-shot examples for each question, and the results are then compared.

Secondly, I explored the effect of the enhancement pipeline. I implemented a No-Enhancement version that directly employs $2n$ agents accepting random few-shot examples without using enhancement agents and compared its performance with the original SELF-FILTER pipeline.

# 6 Results

## 6.1 Experimental Results

The outcomes of all experiments are analyzed across various metrics:

1. Accuracy: the proportion of correct answers identified by the system for the specific dataset.
2. Average number of candidates generated: the count of unique solutions proposed during the solution generation stage. This metric is recorded to assess the diversity of solution generation.
3. Correct solution generation rate: the frequency at which at least one correct solution is proposed among all attempts.
4. PCA Accuracy: the success rate of the PCA in choosing the correct answer from among those questions that have differing candidate solutions, with at least one correct solution being present.
5. Majority Vote Accuracy: the effectiveness of employing a majority vote strategy with solutions generated during the candidate generation stage to determine the final response.

For detailed numerical results, please refer to Figure 2.

| | Accuracy | Correct Solution Generate Rate | Average Unique Solutions Generated | PCA Accuracy | Best Accuracy | Majority Vote Accuracy |
|---|---|---|---|---|---|---|
| COT | 76.00% | 76.00% | 1 | N/A | 76.00% | 76.00% |
| SELF-FILTER | 83.33% | 93.67% | 1.383 | 54.79% | 85.00% | 87.67% |
| No-Rand | 81.33% | 93.67% | 1.397 | 53.15% | 83.00% | 86.67% |
| No-Enhance | 81.67% | 93.00% | 1.393 | 43.80% | 85.00% | 87.00% |

Figure 2: Experiment Results

## 6.2 Cost

For a 3-agent SELF-FILTER pipeline, there is an average of approximately 8 to 9 calls to the GPT-3.5-turbo API. Considering that GPT-4 is about a hundred times more expensive than GPT-3.5, using SELF-FILTER with GPT-3.5 remains roughly ten times cheaper than employing GPT-4.

### 6.3 Result Analysis

While the accuracy of SELF-FILTER appears promising outperforming the traditional COT method by an additional 7 percent, the ablation studies suggest that this method may be less effective than anticipated.

Firstly, random few-shot prompting and enhancement prompting surprisingly do not lead to an increase in the number of unique solution candidates. According to the results, both the No-Random SELF-FILTER and No-Enhancement SELF-FILTER generated a greater average number of unique candidates compared to the regular SELF-FILTER pipeline. In my opinion, this indicates that LLMs are primarily learning the patterns within the few-shot prompts rather than the actual content, making both random few-shot and enhancement pipelines ineffective.

Secondly, although SELF-FILTER generates fewer unique candidates, its accuracy surpasses accuracy of the other two versions. There is no clear evidence to support a positive correlation between the number of unique solutions and the rate of correct solution generation.

These results disproves my previous assumption that random few-shot prompts and enhancement pipelines would help generate more unique solutions, thereby increasing the rate of producing the correct response.

Third, the majority vote accuracy significantly outperforms the accuracy of SELF-FILTER. While PCA achieves accuracies above 50 percent in some tests, it remains substantially inferior to the simple use of a majority vote method. This inefficiency is unacceptable as PCA could require up to 2n additional API calls compare to majority vote. A potential reason for phenomena mentioned above is that PCA only considers unique answers during the comparison stage. For example, if five out of six solutions suggest that the final answer is 8, and one suggests 9, PCA will only compare these two distinct answers (8 and 9), treating the repeated correct answers as redundant. This approach indicates that PCA might be overlooking valuable consensus information that could otherwise enhance decision accuracy. To improve PCA, it could be beneficial to adjust the comparison algorithm to consider the frequency of candidate answers before elimination, thus preserving a form of majority wisdom while still reducing the number of comparisons.

## 7 Conclusion

Although SELF-FILTER proved less effective than anticipated, the filtering technique may still hold value, as there are potential ways to enhance its efficacy by combining it with a majority vote. For instance, it could be feasible to employ multiple filtering agents on a few candidates and subsequently conduct a majority vote among these agents to achieve better results.

### 7.1 Self-Evaluation

Taking this class has been quite a journey for me. From delivering my first 40-minute presentation to innovating on my own, I have learned a great deal. Overall, I have two major takeaways from this class.

Firstly, my ability to read academic papers has significantly improved. Prior to this semester, I did not read many papers as I was engaged in regular coursework and worked as a software engineer. The start of the semester was challenging because I was unsure what and how to read. However, as the semester progressed, I began to master the essential techniques of paper reading and became quite efficient at it. To date, I have read over a hundred papers across three seminar classes this semester.

Secondly, I have gained substantial experience in innovation during this class. Innovation proved to be a much more challenging task than I had anticipated. It is often a disheartening process, filled with numerous failures and disappointments. This is typical, as there are no set guidelines or "correct answers" in innovative endeavors. My major takeaways from engaging in this task are as follows:

1. Adopting a first-principles approach to thinking is extremely beneficial.
2. It is crucial to balance time spent exploring (reading papers, articles, etc.), brainstorming, and conducting experiments. We must continually test the intuitions that form the foundation of our theories.
3. Failing is an inevitable part of the process, but failing to understand why you failed is far worse.

## References

[Cobbe et al., 2021] Cobbe, K., Kosaraju, V., Bavarian, M., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems. *arXiv*, abs/2110.14168.

[Lewis et al., 2020] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv*, abs/2005.11401.

[Madaan et al., 2023] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. (2023). Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*.

[Wang et al., 2023] Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. (2023). Self-consistency improves

chain of thought reasoning in language models. In *Proceedings of the International Conference on Learning Representations*.