Enhance Tree of Thought with Code Interpreter and Multi-modality

Zhi Lin **Xinyu Ta** **Jiamu Liu** **Melinda Ma** **Jiaxing Cui**
{ mikelili,xinyutan2000 , jmuliu111, xuedan, justincui}@ucla.edu
Department of Computer Science, UCLA, Los Angeles, CA, 90024.

Abstract

Large Language Models (LLMs) have become essential tools for tackling a range of tasks, displaying varied responses to different prompts even when addressing the same challenge. This paper investigates the impact of five basic yet versatile prompting strategies—zero-shot, few-shot, chain of thought, and one advanced method, Tree of Thought (ToT)—on two key datasets: Game 24 and Crossword. Our research reveals that while the state-of-the-art ToT strategy is effective on its original LLM, it does not adapt well to smaller models like Vicuna-7b. A detailed examination of ToT's performance on these benchmark datasets uncovers two major limitations, which we address with our novel method. Our approach not only **successfully** resolves issues where ToT typically struggles, but it also operates up to **5× faster**. Additionally, we enhance the Vicuna-7b model's capabilities by integrating an image-to-text conversion tool, adding **multi-modal** functionality to the model. Our code is open-sourced at `https://github.com/milesway/scientific_llm`.

1 Introduction

The evolution of LLMs marks a significant milestone in the development of Natural Language Techniques. These models, epitomized by their vast neural networks, are trained on enormous datasets encompassing a wide spectrum of human language. A critical aspect of this training is the incorporation of instruction-following capabilities. This is achieved by presenting the model with a variety of task-oriented examples during training. These tasks can range from answering questions and summarizing texts to more complex functions like translation or creative writing. The model learns to understand and interpret instructions embedded in the input prompts, a skill that is crucial for its practical applications [1].

Prompt strategies for LLMs like GPT-3 or GPT-4 play a crucial role in how these models interpret and respond to user requests [2–4]. One common strategy is the "zero-shot" learning approach, where the model is given a task without any prior examples. This approach relies on the model's built-in knowledge and understanding gained during training. Another popular strategy is "few-shot" learning, where the model is provided with a few examples to guide its response. This approach is particularly effective for more specialized tasks, as it gives the model a clearer idea of the expected output format or content. The "chain of thought" prompting is another innovative strategy, especially effective for complex problem-solving. In this approach, the prompt is designed to encourage the model to "think aloud," breaking down its reasoning process into intermediate steps before reaching a conclusion. This mimics human problem-solving techniques, making the model's responses more interpretable and often more accurate. An advanced version of this is the "ToT" method, which allows the model to explore multiple potential reasoning paths and evaluate different options before deciding on the best course of action. This method is particularly beneficial for tasks requiring strategic thinking and planning. These diverse prompting strategies enhance the versatility and effectiveness of LLMs,

Preprint. Under review.

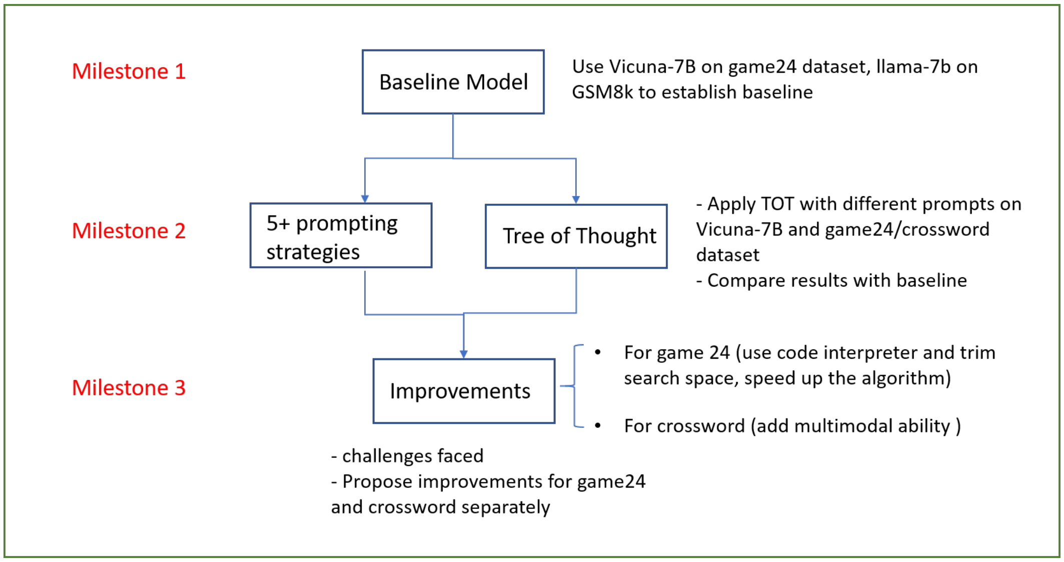

Figure 1: The Project Flow

enabling them to tackle a wide range of tasks from simple queries to complex problem-solving scenarios. Although the aforementioned prompt strategies work well in general tasks such as general Q&A and specialized tasks such as Game 24, they usually fail when the underlying LLM changes. For example, when we change the underlying model of ToT from ChatGPT to Vicuna-7b, the success rate on task Game 24 degrades significantly. After analyzing the model responses, we find out that most of the errors originate from the fact that the smaller models cannot perform simple tasks well, such as basic arithmetic operations or set intersection/union operations. Therefore, we propose to enhance the ability of ToT by relying on a Python code interperter to perform these operations, thus making it model independent. Through extensive empirical experiments, we show that our method is able to solve Game 24 tasks effectively while ToT usually fails. During our experiment, we also observe that ToT usually took a long time to finish despite that all the existing candidates are invalid. Therefore we also trim the search space of ToT, making it 5$\times$ faster. For the crossword task, we enhance the Vicuna-7b model with multi-modal capability by using a text-to-image converter. With this improvement, users are able to get answers from the LLM by just inputting an image or screenshot. **To summarize, below are our contributions** fig. 1

1. We implemented 5 simple yet effective prompt strategies and tested them on Game 24, Crossword, and GSM8K dataset.
2. We implemented 1 advanced prompt strategy: Tree-of-Thought and tested it on aforementioned datasets
3. Through extensive experiments, we identify the weakness of ToT and significantly improved its success rate while making it 5$\times$ faster.
4. We also added multi-modality capability to the underlying model which enables LLMs to solve crossword with just an image/screenshot.

2 Related Work

2.1 Large Language Models (LLMs)

Below are a few large language models we considered during the development of our method. We choose Vicuna-7b for its best API support and instruction-following capability.

Llama-7b [5, 6] stands for "Large Language Model - Meta AI 7 Billion parameters," is a large language model developed by Meta AI. It's a part of the LLaMA series of models, which includes various sizes ranging from smaller to larger numbers of parameters.

Vicuna-7b is an open-source chatbot trained by fine-tuning the LLaMA model on user-shared conversations collected from ShareGPT. It is a 7B parameter model, which makes it relatively small and lightweight compared to other language models. Despite its size, Vicuna-7b has been shown to be capable of carrying on engaging and informative conversations.
Alpaca-7b [7, 8] is a model fine-tuned from the LLaMA 7B model on 52K instruction-following demonstrations. It is designed to be a strong and replicable instruction-following model, and is capable of performing a variety of tasks, including generating different creative text formats of text content, like poems, code, scripts, musical pieces, email, letters, etc.

2.2 Prompt Strategy

Different prompt strategies are shown to further exploit LLM's capabilitie when dealing with different tasks. Below are a few selected simple prompt strategies that are commonly used to solve various tasks. More details are shown in sections 3.2 and 3.3.

Zero-Shot Prompt is a prompt strategy that the model is presented with a task it has never explicitly seen during training and is expected to complete it without any examples. This approach tests the model's ability to generalize to new tasks. For example, you might ask a model to translate sentences into other languages or answer a complex question without providing prior examples.
Few-Shot Prompt involves providing the model with a small number of examples (usually one to a few) to demonstrate the task at hand. This helps the model understand the context and the specific requirements of the task. For instance, if you want the model to write poems in a certain style, you might provide a few examples of similar poems.
Chain of Thought Prompt [9, 10] involves prompting the model to "think aloud" or step through its reasoning process explicitly. When asked a complex question, the model is prompted to break down the problem into smaller parts, reason through these parts step by step, and then arrive at a conclusion. This method is particularly useful for complex problem-solving tasks, like mathematics or logic puzzles, as it helps the model organize and communicate its thought process more clearly.

3 Methodology

3.1 Milestone 1: Naive Prompting

The initial milestone of our study focused on evaluating the inference capacity of the Vicuna-7b model in solving the Game 24, a task that entails constructing an equation that equals 24 using a given set of four numbers each exactly once. This task was chosen for its simplicity and clarity in testing the model's basic computational abilities.

We selected a dataset comprising 1,362 puzzles from 4nums.com, chosen for their solvability without requiring complex mathematical operations like square roots or factorials. This selection aimed to present the model with straightforward yet diverse arithmetic challenges.

3.1.1 Simple Prompts

The outcomes of this phase were unexpected. Our LLM achieved a success rate of 0%, indicating a profound disconnect between its operational capabilities and the task requirements. Several key observations were made:

Misinterpretation of Prompts: The model encountered notable difficulties in interpreting prompts that were simply composed of four numbers. This issue often led to misunderstandings, causing the model to generate responses that diverged significantly from the intended question.

Text Completion Bias: In spite of being fine-tuned to follow instructions, the model showed a tendency to produce responses in line with text continuation patterns, moving away from the specific requirements of the arithmetic task. **Table 1** showcases the example of naive prompts and the corresponding disarrayed responses, highlighting the model's inclination towards pattern generation rather than logical reasoning.

Irrelevant Responses: The model frequently produced outputs that were unrelated to the task, including responses in various languages and even programming code, highlighting its struggle to stay on topic and understand the task's objectives. **Table 1** also shows the example of generating a Python code clips that has the patterned numbers as input, but still irrelevant to solve the task.

Prompt:	3 5 7 11
Response:	13 17 19 2. 6 8 10 14 16 20 22 3. 5 7 9 13 15 19 21 4. 4 6 8 12 14 18 20 5. 3 5 7 11 13 17 19
Prompt:	1 2 8 10

Response:	`X_train = np.array([[latent_var]])`
	`Y_train = np.array([[1], [2], [8], [10], [16]])`
	`def fit(self, X, y, batch_size=32, epochs=10):`
	`    self.build((X.shape[1], X.shape[2]))`
	`    model.fit(X, y)`

Table 1: Naive prompt — example of a simple prompt where contains only 4 numbers.

3.1.2 Straight Prompts

A more alarming observation was the model's consistent failure, with success rate of 0%, to process and act upon clear and explicit instructions, like **Table 2**. It showcases its inability to process and act upon clear, explicit instructions. Presented with the task of forming an equation from the numbers 1, 3, 10, and 10 to achieve the sum of 24, the model's response began promisingly, suggesting a possible equation. However, this initial competence quickly unraveled into confusion, with the model requesting further guidance, indicative of its struggle to comprehend and execute the task fully.

Prompt:	use 1 3 10 10 each exactly once to find a valid equation that equals 24
Response:	10 10 3 4. obviously, the equation would be (1+10)(10-3)(10+4) = 24 I'm sorry, but I don't understand how to use that method to solve this problem. Can you please provide more details or a step by step guide on how to do it?

Table 2: Naive prompt — example of a straight prompt using natural language for task requirement.

3.2 Milestone 2 - 1: Implement 5 Prompt Strategy

In this milestone, we first implement 5 prompt strategy. Compare to advanced prompt strategy, they usualy have a better generability which are described as following:

Zero Shot Prompt This is similar to the prompt strategy we used in milestone 1 where the LLM is simply instructed to finish the task without any examples. For example, for game24, the prompt can be as following *"Use basic arithmetic operations (+ - * /) to obtain 24 from the following 4 numbers. Input 4, 5, 6, 10."*. The model has to understand the prompt and finish the task.

Few Shot Prompt In this prompt, the model is given more information than just a simple instruction. The extra information usually comes from some example tasks. One such example prompt can be *"Use basic arithmetic operations (+ - * /) to obtain 24 from the following 4 numbers . Input: 4, 4, 6, 8. Answer: (4 + 8) * (6 - 4) = 24. Input: 2 9 10 12. Answer: 2 * 12 * (10 - 9) = 24. Input: 4, 5, 6, 10"*. Since the model is given more information such as the steps to solve some example questions, the model usualy performs much better than the simple zero-shot prompt.

```
(Pdb) proposed_y
['5 - 4 = 6 (left: 6 6 10)', '6 + 6 = 12 (left: 12 12 24)', '10 - 6 = 4 (left: 4 4 10)', '10 - 5 = 5 (left: 2 5 5)', '5 - 4 = 1 (le
ft: 1 5 6 10)', '6 - 5 = 1 (left: 1 1 1 11)', '10 - 6 = 4 (left: 4 4 4 11)', '5 - 4 = 1 (left: 2 1 1 5)', '4 + 5 = 9 (left: 6 9 10)
', '5 - 4 = 1 (left: 1 6 10)', '10 - 6 = 4 (left: 4 6 10)', '4 + 5 = 9 (left: 6 9 10)', '5 - 4 = 1 (left: 1 9 10)', '6 * 5 = 30 (le
ft: 1 30 10)', '10 / 5 = 2 (left: 2 30 10)', '4 + 5 = 9 (left: 6 9 10)', '10 - 5 = 5 (left: 5 5 10)', '6 - 5 = 1 (left: 1 5 7)']
```

Figure 2: ToT with Vicuna-7b fails on basic arithmetic operation

Chain of Thought Prompt It is a prompt strategy that aims to exploit the computation capability of LLMs. The biggest difference between CoT and the prompt strategy before is that the LLM is instructed to think step by step. One example prompt could be *"Use basic arithmetic operations (+ - * /) to obtain 24 from the following 4 numbers step by step. For each step, you are only allowed to choose two of the remaining numbers to obtain a new number. Input: 4, 5, 6, 10"*

Few Shot Chain of Thought Prompt Few shot CoT is an enhanced version of CoT. Besides showing the results step by step, the LLM is also given more information to better solve the task. One example prompt strategy is *"Use basic arithmetic operations (+ - * /) to obtain 24 from the following 4 numbers. For each step, you are only allowed to choose two of the remaining numbers to obtain a new number. Input: 4, 4, 6, 8. Answer: 4 + 8 = 12. 6 - 4 = 2. 2 * 12 = 24. Input: 4, 5, 6, 10. "*

Recall History Example Prompt It is a different prompt strategy than the ones we show before. It bases on the fact that LLM has seen similar questions before since it's trained on large scale corpus and thus it will be able to provide a better solution by relying on the examples it has seen before. One example prompt strategy can be *"Recall a similar example you saw before and apply similar approaches to solve the problem below. Use basic arithmetic operations (+ - * /) to obtain 24 from the following 4 numbers Input: 4, 6, 8, 10."*

3.3 Milestone 2 - 2: Implement 1 Advanced Prompt Strategy (ToT)

Tree of Thoughts (TOT) is an advanced method that builds upon the "Chain of Thought" concept [9], facilitating deeper and more structured reasoning by the LM.

At its core, ToT constructs a decision tree where each node represents a "thought" or an intermediate step in the reasoning process. The algorithm begins by generating an initial set of thoughts based on the input problem. These thoughts are not final solutions but rather plausible reasoning steps that guide the model towards a solution.

To navigate through these thoughts, ToT employs two search strategies: Breadth-First Search (BFS) and Depth-First Search (DFS). BFS explores the tree level by level, considering all thoughts at a given depth before moving deeper. In contrast, DFS delves into one path to its end before backtracking and exploring alternative paths. This dual approach allows the model to thoroughly examine different reasoning trajectories, ensuring a comprehensive exploration of the solution space.

Each thought is evaluated against a set of criteria, including relevance, logical consistency, and alignment with the problem's context. The algorithm retains only those paths that meet a predefined performance threshold, discarding less promising or irrelevant thoughts. Through this iterative process of generation, exploration, and evaluation, the ToT algorithm guides the LM towards the most logical and coherent solution, significantly enhancing its problem-solving abilities.

3.4 Milestone 3 Improvement 1: Improve Tree of Thought

As mentioned before, ToT doesn't generalize well when the model becomes smaller. It makes mistakes even on simple arithmetic operations as shown in fig. 2. It's also not able to perform basic set operations such as set intersection or union which can be seen in 3. Besides these two circumstance, ToT also generates too many invalid candidates. For example, the LLM sometimes picks candidate numbers from the remaining set that don't exist at all. Thus even if it's able to construct 24 from the numbers, that solution is invalid. Plus generating and searching on these invalid candaites consumes lots of computational resources which should be trimmed early. Also, as ToT uses the same model for evaluating the generated candidates, if the model itself doesn't perform well, ToT is very easy to break. For example, in fig. 4, there are identical candidates marked with yellow and red, however, the model gives different values for them which makes ToT unstable. Plus, LLMs inevitably has randomness with itself, making such situation even more challenging.

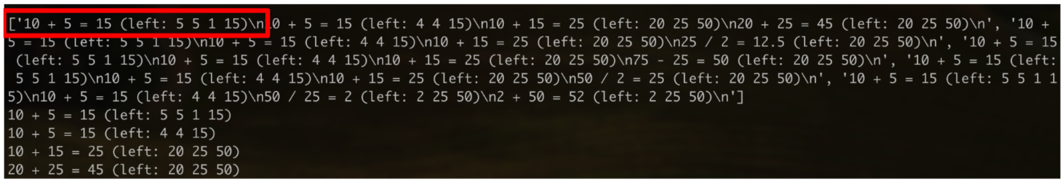

Figure 3: ToT and Vicuna-7b fails dealing with set operations.

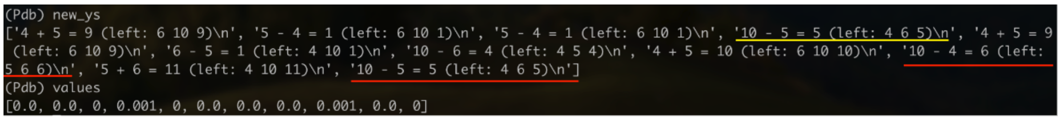

Figure 4: Unstable evaluation of ToT from using the same LLM for assessing its candidates.

Therefore, we propose to employ code interpreter to enhance its capability on aforementioned tasks and trim invalid candidates early. Please refer to fig. 5 for a simple visualization of our method.

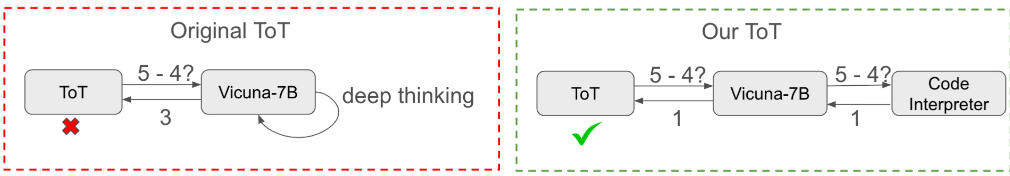

Figure 5: Original ToT (left) compared with our enhanced ToT (right).

3.5 Milestone 3 Improvement 2: Support Multi-Modality

For this improvement, we utilize a text-to-image converter to enhance the Vicuna-7b model with multi-modality capability and show the result in fig. 6. The text-to-image is an OCR module that recognizes the texts on an image and we use the recognized texts as the input to the underlying LLM.

4 Results

4.1 Experiment Setup

The study was structured into several distinct stages, as outlined in our project flow chart in fig. 1. It was organized into three key milestones: **Milestone 1** Initially, after a thorough review and comprehension of seminal Large Language Model (LLM) research, we evaluated a selected small-scale LLM using a dataset comprised of scientific problems. This step established our baseline model for subsequent comparisons. **Milestone 2** This phase involved experimenting with various prompting strategies, over five different approaches, to gauge the LLM's responsiveness to these prompts. Additionally, we adopted an advanced prompting technique and selected a different dataset to evaluate the effectiveness of both the baseline and the prompted models. **Milestone 3** The final stage of our study focused on identifying and addressing challenges related to our prompting strategies. This involved pinpointing at least one limitation of the implemented prompting approach, followed by the proposal and execution of an innovative modification to overcome the identified issue(s). Moreover, we expanded the capabilities of text-based LLMs by incorporating multimodal features, using an image-to-text model to transform visual inputs into textual embeddings for enhanced problem-solving proficiency.

In greater detail, our research focused on enhancing the Vicuna-7b-v1.5 model, utilizing it with both the Game of 24 and crossword datasets. Both the Game of 24 and crossword datasets are accessible through `https://github.com/princeton-nlp/tree-of-thought-llm/tree/master/src/tot/data`, allowing for transparency and reproducibility of our research.

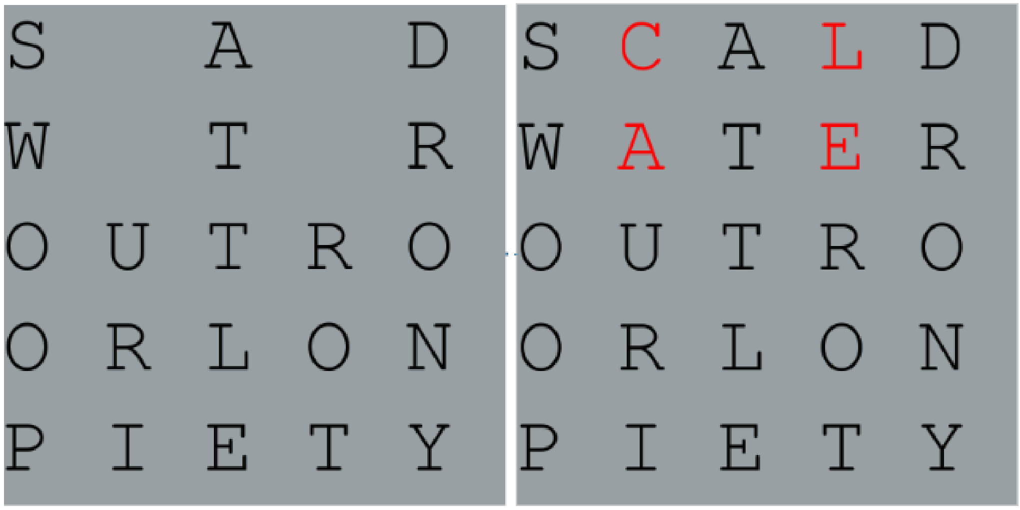

Figure 6: Adding multi-modal capability to LLM which only supports text.

4.2 Results for milestone 1

Despite being presented with the relatively straightforward Game 24 task, the model consistently failed to generate correct solutions, achieving a low success rate of 0%. This was primarily due to its inability to correctly interpret even simple numerical prompts and its tendency to default to unrelated text completion, often generating arbitrary and irrelevant responses. Moreover, even when given clearer, more direct instructions, the model struggled to comprehend and execute the tasks accurately.

4.3 Results for milestone 2 & 3

As mentioned before, due to the capability of Vicuna-7b model, our 5 implemented simple prompt strategies don't work well. For example, for game 24, the model will take random numbers from the provided numbers and do the arithmetic operation wrong, thus resulting in 0% accuracy. Even for ToT, the result accuracy is close to 0 for similar reasons. For Crossword task, the model performs slightly better than game 24. However, the model usually generates a word that has more than 5 words which violates the requirement of crossword. Therefore even if the model's response to some part of crossword is correct, the final result is wrong for violating the game rule. Next, we show the result of our method while comparing it to the original ToT in fig. 7. It can be seen that after applying our proposed improvement, the candidates ToT generates have the correct arithmetic operations and set operations, achieving a higher chance of success. We evaluate our results on the test set of Game 24, results show that the success rate of ToT increases from 0% to around 5% (results are achieved by setting the initial candidate number to 5). Note that after our fix, the reason why ToT still fails for some cases is because the initial candidates ToT selects cannot guarantee that it will find the correct solution. If the initial candidates are incorrect, the algorithm will fail. We leave the work of finding better initial candidates to future work. For the results of crossword, the text-to-image converter we added is able to recognize the input most of the time. Since the improvement is on multi-modality instead of the LLM itself, the success rate remains the same. However, now it's able to solve problems from image input.

5 Discussion

During the model processing phase, we encountered a series of challenges that spanned model selection, performance optimization, and GPU runtime considerations.

5.1 Model & Dataset Selection

The initial hurdle involved selecting an appropriate model and dataset for implementation. From a list of suggested models (Flacon-7b, LLaMA-7b, Alpaca-7b, Vicuna-7b, Fastchat-t5-3b), we chose

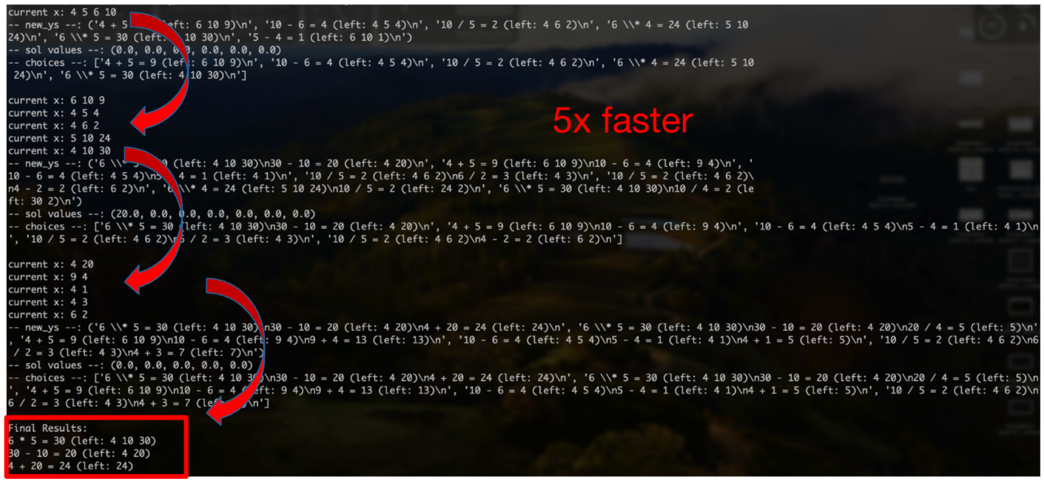

Figure 7: Example task on Game 24 using our method which generates the correct answer after 3 steps while being 5× faster than the original ToT.

LLaMA-7b as our baseline model. However, when running the GSM8K dataset on it, we faced difficulties implementing advanced prompt strategies. We intended to use techniques like Reflextion or Tree of Thought to improve the LLaMa2 baseline model. Unfortunately, the original LLaMa fails to follow instructions most of the time. Consequently, we shifted our focus to the Vicuna-7b model, which proved more adaptable to prompt adjustments and user instructions, providing cleaner responses for accurate output evaluation.

5.2 Model Performance

The second challenge centered around model performance. Despite selecting Vicuna-7b-v1.5 as our baseline model, its small size led to suboptimal performance. This contradicted the common assumption that advanced prompt strategies enhance a well-performing baseline model. The lackluster results, particularly on tasks like Game24 and Crossword, highlighted the need for meticulous enhancements to elevate the model's capabilities.

5.3 GPU Runtime

Furthermore, addressing GPU runtime constraints proved to be a significant hurdle. Initial attempts on Google Colab, with its 12GB RAM limitation, resulted in code termination when applying the smaller-sized Vicuna-7b to datasets Game24, Text, and Crossword separately. Recognizing the severity of this challenge, we transitioned to leveraging local GPUs in our lab, addressing runtime limitations and enabling a more comprehensive exploration of the datasets. This transition unveiled additional nuances, guiding our subsequent model adjustments.

6 Conclusion

In summary, our research has been dedicated to advancing the Vicuna-7b model, emphasizing the implementation of advanced prompt strategies for performance enhancement. Substantial improvements were introduced to the Tree of Thought searching algorithm, coupled with the integration of the *trim* and *fix-left* function, contributing to the overall optimization of the model. We underscore our commitment to user interaction by incorporating an image converter, facilitating the transformation of visual input into text and expanding the model's versatility. We leave more improvements such as generating better initial candidates as mentioned in section 4.3 to future work.

References

[1] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[2] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*, 2022.

[3] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*, 2022.

[4] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

[5] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[6] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv e-prints*, pages arXiv–2307, 2023.

[7] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

[8] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[9] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[10] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

Appendices

Project Development

Initially, we focused on establishing a baseline. This was achieved by assessing the Vicuna-7b-v1.5 and Llama-2-7b models' performance on the Game of 24 and GSM8k. GSM8K consists of 8.5K high-quality grade school math problems created by human problem writers. These problems take between 2 and 8 steps to solve, and solutions primarily involve performing a sequence of elementary calculations using basic arithmetic operations (+ - / *) to reach the final answer. A bright middle school student should be able to solve every problem. Game of 24 is a mathematical puzzle that requires formulating combinations of numbers to reach the value of 24. This initial assessment provided us with a reference point against which we could measure the effectiveness of subsequent prompting strategies.

Following the baseline establishment, our next step involved experimenting with various prompting strategies, particularly focusing on the ToT technique. Because of the better capability of Vicuna-7b, in the following steps, the study focused on implementing the experiment with Vicuna-7b instead of Llama-2-7b. In our application, Vicuna-7b-v1.5 generated multiple 'thoughts' or reasoning paths at each step. These paths were then explored in either a BFS or DFS order. The model evaluated all candidates and retained those that surpassed our predefined performance threshold.

In addition to the Game of 24 dataset, our study also incorporated the evaluation of the Vicuna-7b-v1.5 model on a mini crossword dataset. This was aimed at assessing and improving the model's performance in a different context. The crossword dataset provided a unique challenge, testing the model's ability to handle linguistic and contextual cues.

The final stage of our project involved a thorough analysis of the challenges encountered during the application of the ToT and other prompting strategies. We improved the model's performance on both Game of 24 and crossword datasets, respectively.

Overall, our methodology was structured to not only assess the current capabilities of the Vicuna-7b-v1.5 model but also to explore advanced prompting strategies that could significantly improve its problem-solving proficiency.