ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

LRE Language Reasoning Education

Final Report

When Less is More: The Power of One Strategic Step in LLM Refinement

Semester Project

Piyushi Goyal
Legi: 23-956-915
Department of Computer Science

Advisor: Kumar Shridhar
Supervisor: Prof. Mrinmaya Sachan

October 21, 2024

Abstract

In this project, we explore the relationship between accuracy and time cost during inference in Chain-of-Thought (CoT) reasoning for Math Word Problems. Our approach builds on previous work that emphasizes the importance of generating a correct initial step in a CoT to improve final accuracy.

Rather than allowing a large language model (LLM) to generate the entire CoT in one go, we instead prompt it to generate only the first step. We then leverage the significance of this first step by introducing a second LLM—referred to as the "verifier"—tasked with assessing the correctness of that initial step. The verifier determines whether the generation should continue or stop. If the initial step is incorrect, we halt the generation and move on to the next question. Our experimental results support our hypothesis: this approach reduces the time needed to generate tokens, with only a minor drop in overall accuracy. These findings are particularly relevant for scenarios where a trade-off between accuracy and time is acceptable, as they enable significant time savings by skipping questions likely to result in incorrect answers.

Additionally, we conducted experiments to investigate the impact of verifier model size on its ability to both detect and correct an incorrect first step. In one experiment, the verifier not only identified incorrect steps but also replaced them with the correct ones, allowing the generation to proceed. While this approach incurred a higher time cost than the baseline, it led to improved overall accuracy. In another experiment, we focused on the subset of samples where the first step was rejected as incorrect. For these cases, we found that larger verifiers were more accurate and outperformed the baseline when paired with a smaller generator model. These findings further demonstrate the potential of our methodology, highlighting how delegating verification to another LLM can enhance performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

- *Rationale of the work*

  Large Language Models (LLMs) possess impressive capabilities, but they are prone to generating incorrect or nonsensical responses, known as hallucinations. Correcting these errors is crucial for improving output quality, yet determining when and how to intervene presents a significant challenge. In this project, we propose a solution by implementing systematic refinement processes that catch and correct errors early in the reasoning chain. By combining techniques such as Chain of Thought (CoT) reasoning and subanswer-based verification, we aim to enhance the model's ability to self-correct, particularly in multi-step reasoning tasks, leading to more reliable and accurate outcomes.

  However, while improving accuracy is important, managing time cost is equally critical. Excessive computational demands from multiple refinements can limit the practical scalability of LLMs. Therefore, it is essential to devise strategies that balance effective error correction with efficient token usage. Our approach addresses both these aspects, aiming to develop more reliable and cost-effective AI systems capable of autonomously refining their outputs.

- *Scope of the work*

  As discussed earlier, while LLMs can generate responses to a wide variety of questions, their accuracy is not always reliable. Introducing mechanisms to assess and refine these responses can improve overall outcomes. The Ask, Refine, and Trust (ART) model [19], uses sub-questioning to validate LLM outputs. However, this method often requires multiple subquestions, which can lead to higher token costs. On the other hand, the QuestCoT approach [6] focuses on starting with the most relevant sub-question, reducing unnecessary computations.

  In this project, we explored the potential of combining these two methodologies to optimize the efficiency of ART by generating only the first step instead of the entire Chain of Thought (CoT). To achieve this, we conducted a detailed analysis by integrating the QuestCoT principles into the ART framework. We then systematically compared the performance of this hybrid model against a baseline model that used Chain of Thought reasoning alone. While this project focuses on testing the model on the GSM8K dataset [2], future work could extend its application to other domains such as code generation, multiple-choice questions, and pure sciences, among others.

Chapter 2

Related Work

- *Most relevant work*

  Recently, there has been a surge of work in the field of self-refinement and reasoning capabilities in LLMs. The strategies employed for reasoning often form the backbone of self-refinement methods. In shaping our project and defining its scope, we consulted various works within the domain of self-reasoning in Large Language Models (LLMs). There are three primary approaches to self-reasoning: Chain-of-Thought (COT) [23], [9], [26]; subquestion decomposition [13], [16], [15], [18]; and Program of Thought [1]. The Program of Thought approach involves solving questions using coding, which is outside the scope of our project.

  Building on these reasoning techniques, researchers have begun developing various self-refinement strategies. Most LLM refinement techniques leverage one of these methods ([10], [24], [14], [5], [25]) or combine three steps in one [17]. In [17], a single LLM handles the entire process of generating, refining, and selecting prompts. Meanwhile, [3] introduces the idea of using subquestions to verify facts and mitigate hallucination. Similarly, [19] assigns the task of verifying the Chain of Thought (COT) generated by the main LLM to a smaller LM, which carries out verification using subquestions. The concept of distilling reasoning capabilities into smaller models has been explored in works like [20], [4], and [11]. [7] discusses post-hoc prompting as a way to enable LLMs to assess their own performance and accuracy. [5] further explores the difference between intrinsic and extrinsic reasoning in LLMs, focusing on the importance of both pre-hoc and post-hoc prompting.

  A key influence for our project was [6], which demonstrated that improving the first step in a reasoning chain significantly boosts final answer accuracy. These works collectively serve as the foundation for our research direction.

- *Focus of this project*

  The collection of related work helps identify the specific focus area within the broader field of self-reasoning in LLMs. [17] introduces SCREWS, a modular framework for reasoning with revisions. In this framework, the three modules—Sampling, Resampling, and Selection—are all managed by the same LLM. Building on this foundation, [19] delegates verification and refinement tasks to a smaller LM. This extension highlights the shift toward more sophisticated methodologies, emphasizing the potential of smaller LLMs in the reasoning process.

  In parallel, [6] offers an innovative approach where only one crucial sub-question is used at the start to determine if the LLM begins the reasoning process correctly, followed by a Chain of Thought. This strategy is informed by research showing that accuracy increases with each subsequent sub-question. By emphasizing accuracy from the outset, this method demonstrates the potential for significant performance improvements.

  Our study identifies a unique intersection between self-refinement algorithms and the strategic use of sub-answers. This convergence forms the core of our investigation, allowing us to explore how these elements can be effectively combined to improve the performance of self-reasoning LLMs. In the following discussion, we will further examine the implications, differences, and similarities between these approaches, clarifying their contributions and exploring potential synergies.

Chapter 3

Materials and Methods

- *Methodology*

  As discussed in Chapter 2, the foundations of this project are built upon the frameworks proposed by [19] and [6]. In the former, the authors utilize a separate LLM, referred to as the 'Asker', which is trained to generate subquestions for a given Math World Problem (MWP). The primary model generates a Chain of Thought (COT) for the MWP, which, along with the original question, is passed to the Asker. The Asker then verifies the prediction by generating a set of subquestions. If it determines that any essential questions remain unanswered in the COT, it returns both the MWP and the subquestions to the main model, prompting it to generate a revised COT.

  [6] introduces a more streamlined approach with the 'QuestCOT' framework. Here, a smaller model is tasked with generating a subquestion solely for the first step of the solution process, followed by the COT. They demonstrate that when the first step is correct, the likelihood of arriving at an accurate final answer increases significantly.

  The main limitation of the approach in [19] is its reliance on multiple subquestions to verify the LLM's predictions, which can become prohibitively expensive in terms of output time. In response to this, our project adopts the methodology from [6] and restricts the verification process to only the first step, thereby eliminating the need to generate an entire COT upfront.

  To support this approach, we conducted an analysis of the GSM8K dataset [2] to assess how often the final answer is correct when the first step is accurate. Our findings confirm that the percentage of COTs yielding correct final answers after a correct first step surpasses that of the baseline case.

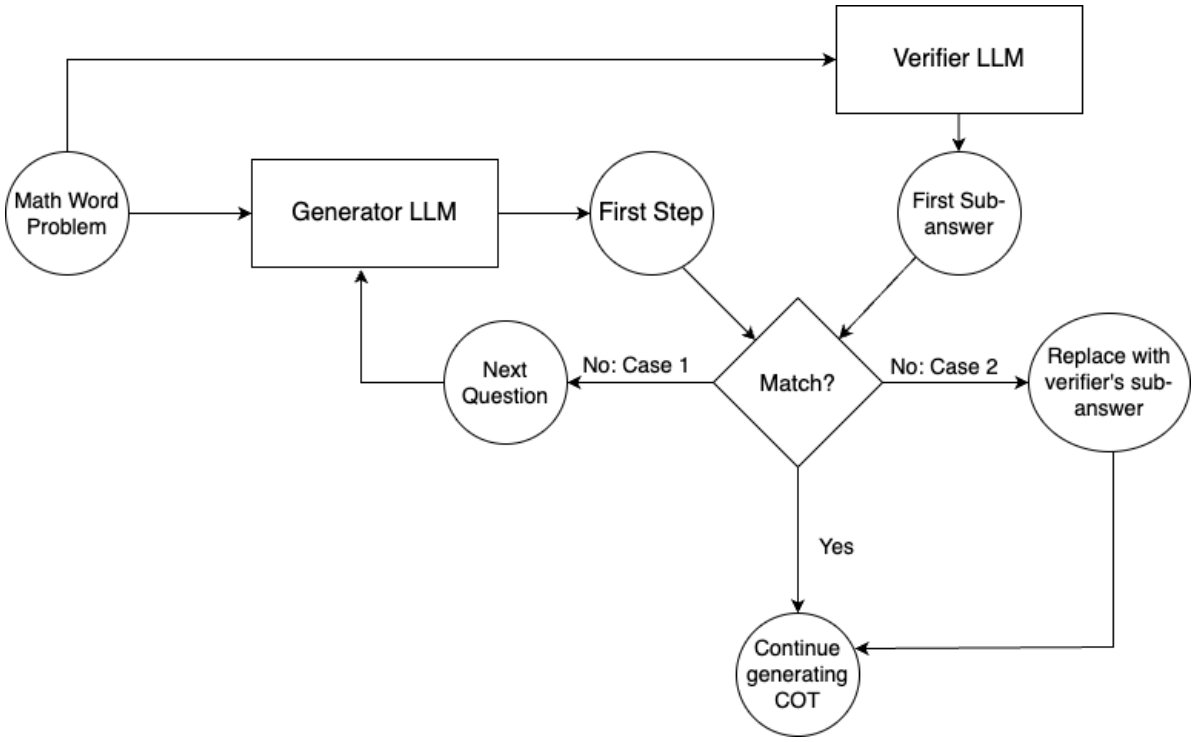

Figure 3.1: Methodology

Following this, we use the GSM8K dataset to train two models: a verifier and a generator. The verifier is trained on the Socratic set from GSM8K, where it is specifically instructed to generate only the first subquestion and its corresponding subanswer during inference. The generator, on the other hand, is trained on the COT set from GSM8K.

The methodology for integrating these models is outlined in Figure 3.1. The process begins when an MWP is given to the generator model, which outputs only the first step of the solution. This first step is then compared with the subanswer generated by the verifier, as depicted in the ”Match?” decision block in the flowchart.

We define our baseline as the scenario where the generator produces the complete COT when prompted with a question, and we calculate the associated output time cost and accuracy. To test our hypothesis, we modify the generator’s behavior by prompting it to generate only the first step of the solution before stopping. This first step is then compared to the subanswer provided by the verifier.

If the two outputs match (indicated by the ”Yes” path in the flowchart), the generator proceeds to generate the remainder of the COT. However, if there is no match, two alternative cases are considered, as shown in the ”No” branches of the flowchart. In Case 1, the generator skips the current question entirely, following the findings of [6], which indicate that questions with an incorrect first step are likely to result in an incorrect final answer. In Case 2, the first step generated by the verifier replaces the generator’s output, after which the generator continues generating the remaining steps starting from this verified first step.

We then calculate the time cost and accuracy for both sub-cases to evaluate their effectiveness compared to the baseline.

Chapter 4

Experiments and Results

- *Experimental Design*

  We utilized the Unsloth library to download the models and applied LoRA for parameter-efficient fine-tuning [12]. For inference and validation, we employed VLLM [8], as it is one of the fastest frameworks for generation tasks. The experiments were conducted on Nvidia RTX GeForce 3090/4090 GPUs, each with 24GiB of RAM.

  The models used in our experiments include Gemma-2B and Gemma-7B from the Gemma family [21]. To further validate our findings, we also conducted experiments on Qwen2.5 models (0.5B, 1.5B, 3B, 7B) [22]. We explored all possible combinations of these models by pairing them within their respective families, with one model acting as the verifier and another as the generator. However, we restricted pairings to models within the same family for consistency.

  The GSM8K dataset [2] was used for both training and testing purposes throughout this project. The experimental setup followed the Unsloth notebooks as templates, and the trainer parameters did not require any modifications.

- *Results*

  In this section, we summarize the results for each model family using the accompanying tables and figures. The black dot corresponds to the baseline cost and accuracy of the generator model, which generates the entire COT for all samples in the test set. The colored dots represent the generator's performance when different-sized models serve as the verifier for the first step.

  As described in Chapter 3, we handle cases where the first step is incorrect in two ways: either by skipping the question (Correct FS) or by replacing the first step with that of the verifier (Replaced FS). On the Y-axis, we plot the accuracy of the final answer for both scenarios.

  The formula for time cost is:

$$\text{Time Cost} = (\text{time\_per\_token} \times \text{num\_generated\_tokens})_{\text{verifier}} + (\text{time\_per\_token} \times \text{num\_generated\_tokens})_{\text{generator}}$$

  We save the number of tokens that are generated for each pair and then substitute those in the above equation. To calculate the time per token, we note the tokens/sec value for each model on only one sample.
  For both cases (sections 4.1 and 4.2), the first term remains the same, while only the second term changes. The X-axis represents the total time cost calculated from this formula. Therefore, the colored dots illustrate the generator's accuracy and time cost for the final answer when different-sized verifiers are used.
  In section 4.3, we present an additional experiment in which we calculate the accuracy specifically for the rejected samples (i.e., those with an incorrect first step). We evaluate performance across all asker sizes and observe a notable increase in accuracy when the verifier is larger than the generator.

- ***Key Findings***
  The key observations from our study are summarized as follows:

- When the verifier is smaller than the generator, significant time savings are achieved by proceeding only with questions that have a correct first step.
- Replacing incorrect first steps with correct ones (as identified by the verifier), and then generating the remaining COT on all test samples, leads to a substantial increase in time cost, with only a marginal improvement in accuracy.
- For samples rejected due to an incorrect first step, when the first step is replaced using a verifier bigger than the generator, the accuracy of the final answer exceeds the baseline accuracy considerably for these same samples.

4.1 Correct FS

4.1.1 Qwen2.5

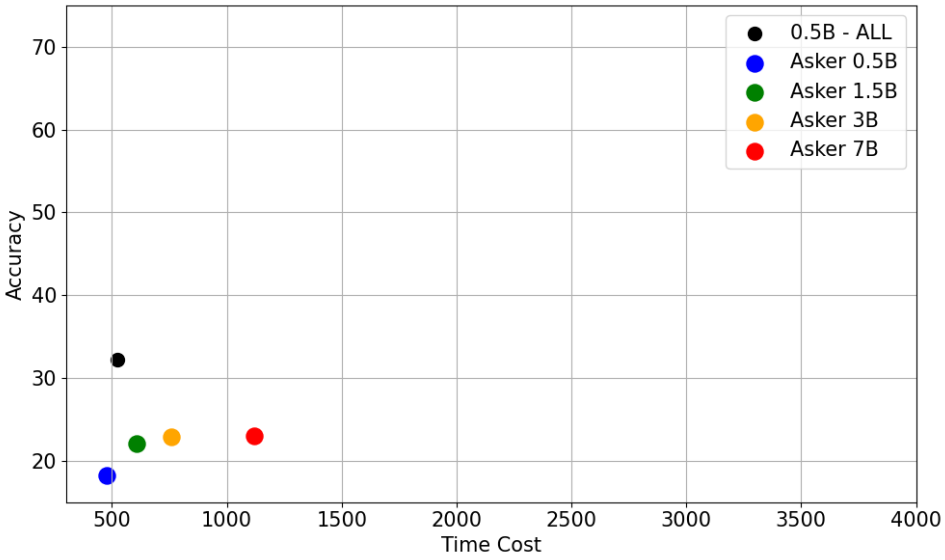

Figure 4.1: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples	(525, 32.22)
0.5B	(479, 18.27)
1.5B	(607, 22.06)
3B	(757, 22.89)
7B	(1119, 23.04)

Table 4.1: Generator 0.5B performance with different verifiers

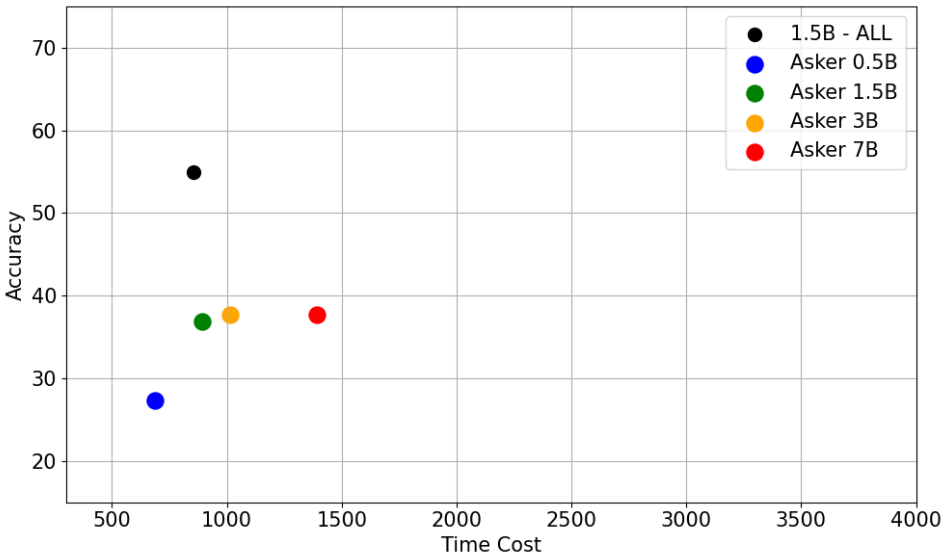

Figure 4.2: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples	(853, 54.96)
0.5B	(687, 27.36)
1.5B	(891, 36.92)
3B	(1014, 37.68)
7B	(1392, 37.68)

Table 4.2: Generator 1.5B performance with different verifiers

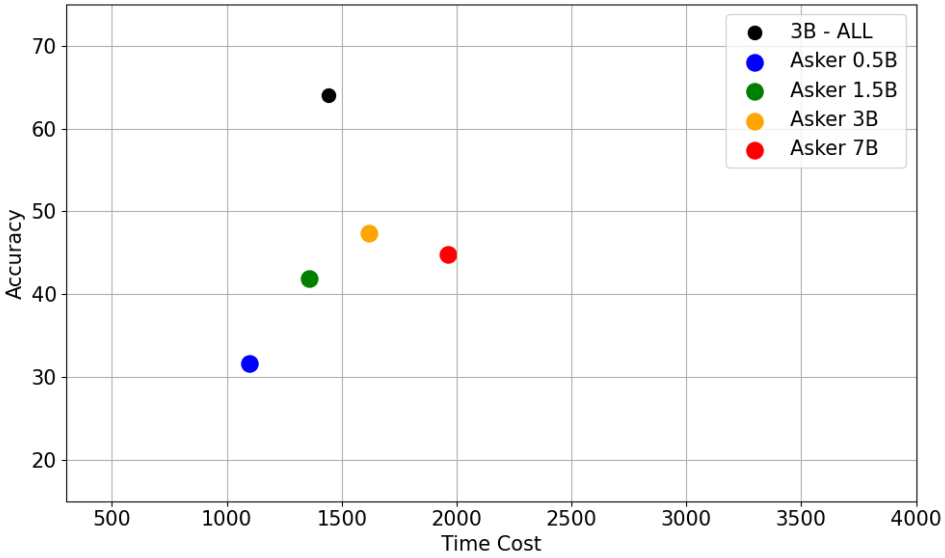

Figure 4.3: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples	(1444, 64.06)
0.5B	(1098, 31.61)
1.5B	(1360, 41.93)
3B	(1620, 47.38)
7B	(1963, 44.80)

Table 4.3: Generator 3B performance with different verifiers

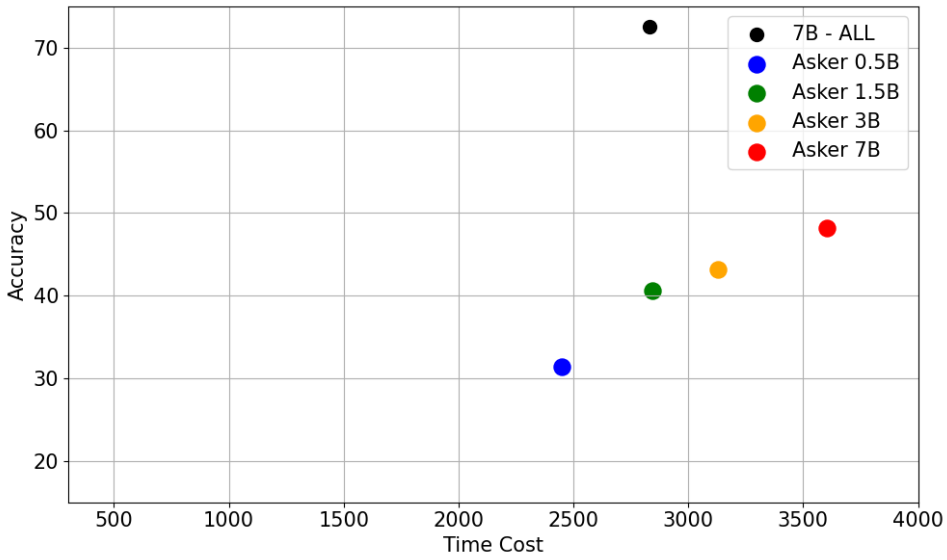

Figure 4.4: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples	(2833, 72.55)
0.5B	(2449, 31.46)
1.5B	(2845, 40.56)
3B	(3130, 43.13)
7B	(3605, 48.21)

Table 4.4: Generator 7B performance with different verifiers

4.1.2 Gemma

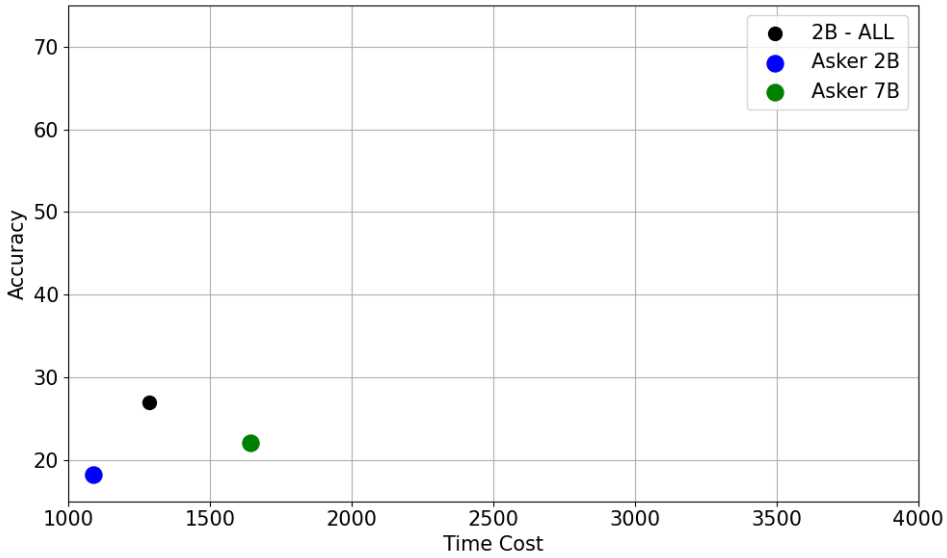

Figure 4.5: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples	(1286, 26.99)
2B	(1088, 18.27)
7B	(1644, 22.06)

Table 4.5: Generator 2B performance with different verifiers

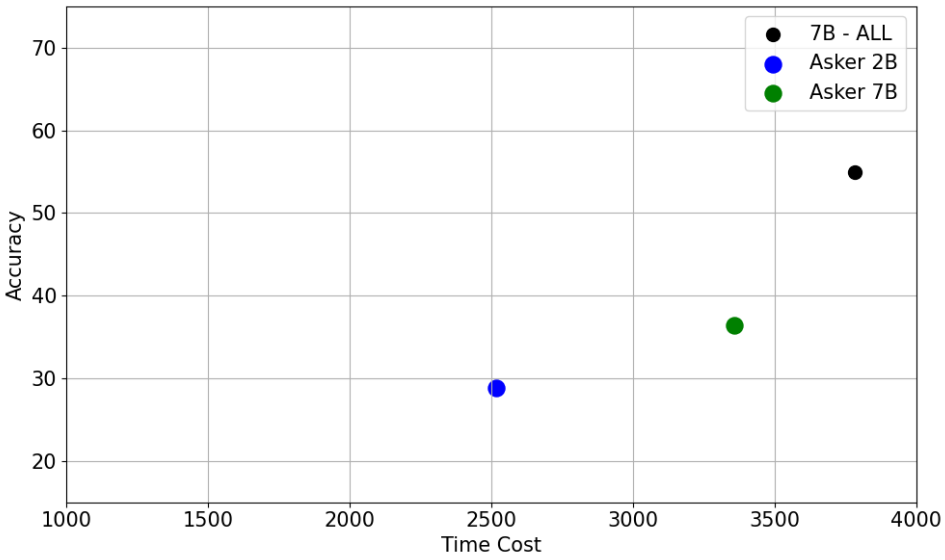

Figure 4.6: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples	(3784, 54.96)
2B	(2518, 28.88)
7B	(3358, 36.46)

Table 4.6: Generator 7B performance with different verifiers

We can summarize the following for this subsection:

- For Qwen, a clear trend emerges across all generator sizes: the lowest time cost is consistently achieved with the 0.5B model.
- Overall, the results align with expectations. There is a noticeable decline in accuracy, but this is accompanied by a substantial reduction in time cost when compared to the baseline.
- As the generator size increases, the accuracy gap widens. The table demonstrates this increase in multiples of 10, with the difference between the highest accuracy using a verifier and the baseline ranging from approximately 10 to 30.
- For Gemma, the outcomes are more favorable. The accuracy gap remains relatively small while achieving a significant drop in time cost, particularly with the 2B model. However, for the 7B model, the accuracy gap increases to around 20, although this is still an improvement over Qwen.

4.2 Replaced FS

4.2.1 Qwen2.5

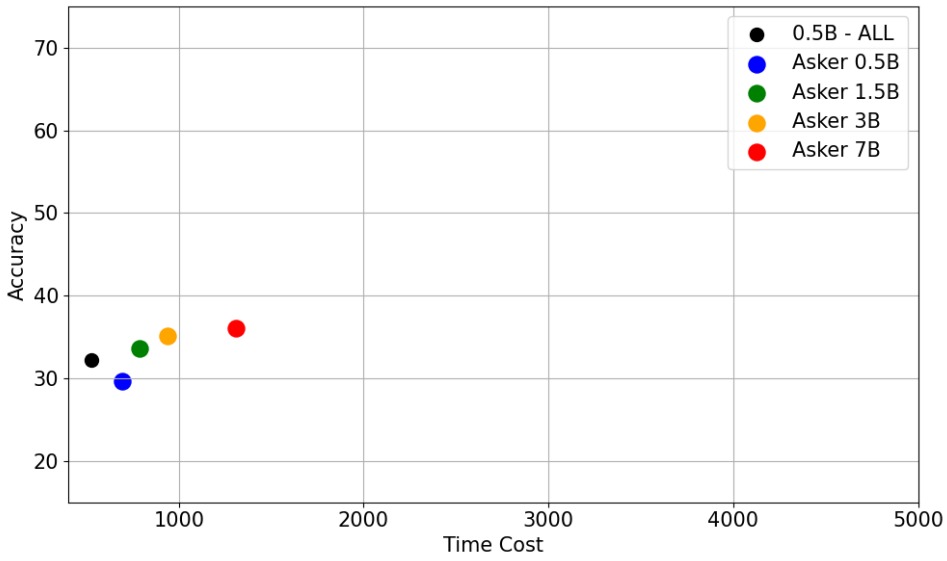

Figure 4.7: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples Original	(525, 32.22)
0.5B	(693, 29.64)
1.5B	(788, 33.58)
3B	(937, 35.17)
7B	(1310, 36.08)

Table 4.7: Generator 0.5B performance with different verifiers

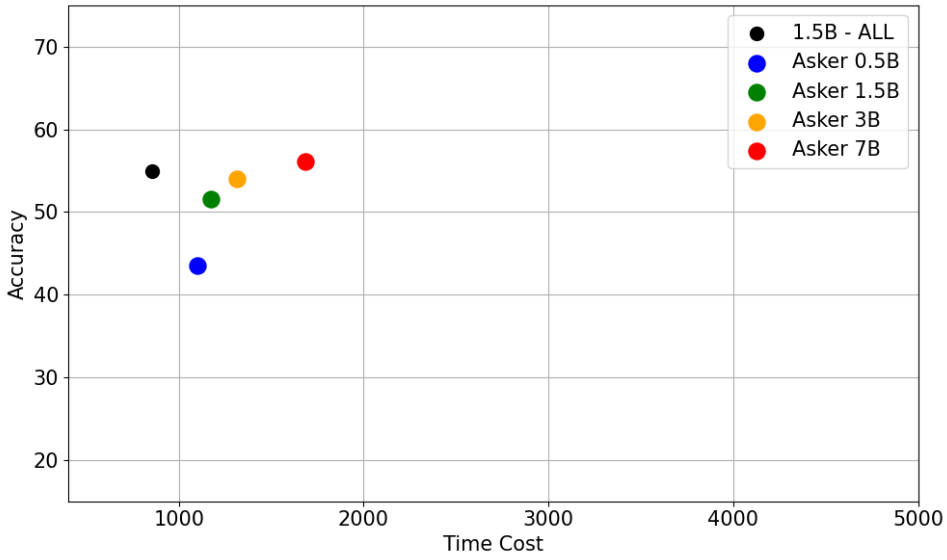

Figure 4.8: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples Original	(853, 54.96)
0.5B	(1102, 43.51)
1.5B	(1172, 51.55)
3B	(1314, 53.98)
7B	(1685, 56.10)

Table 4.8: Generator 1.5B performance with different verifiers

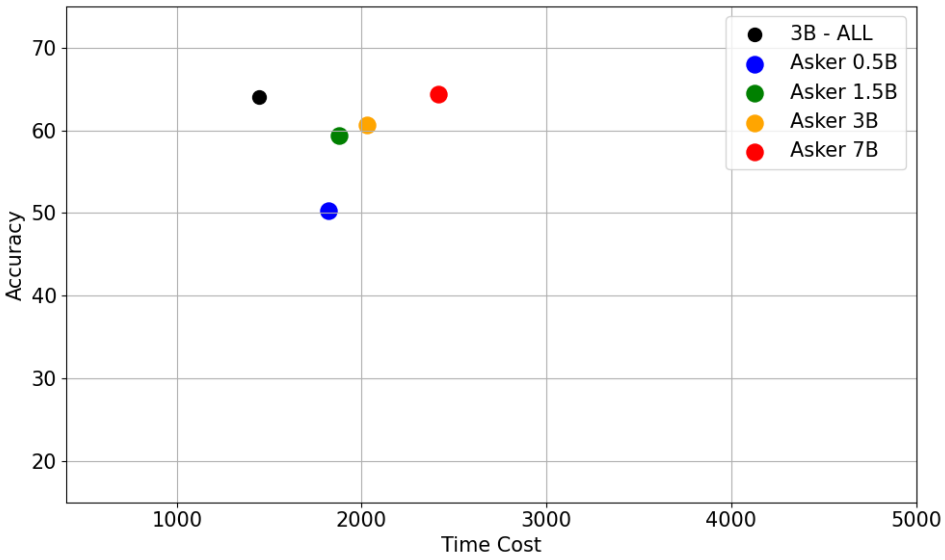

Figure 4.9: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples Original	(1444, 64.06)
0.5B	(1818, 50.34)
1.5B	(1878, 59.36)
3B	(2030, 60.65)
7B	(2416, 64.44)

Table 4.9: Generator 3B performance with different verifiers

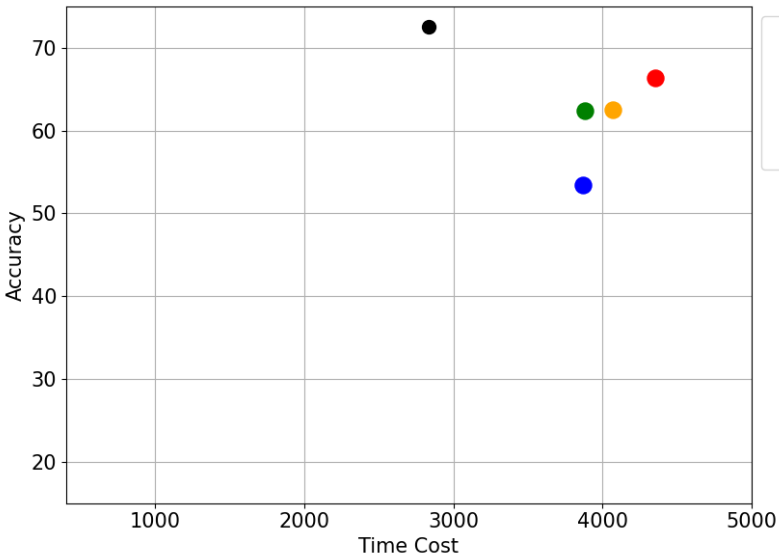

Figure 4.10: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples Original	(2833, 72.55)
0.5B	(3867, 53.44)
1.5B	(3882, 62.39)
3B	(4071, 62.47)
7B	(4355, 66.33)

Table 4.10: Generator 7B performance with different verifiers

4.2.2 Gemma

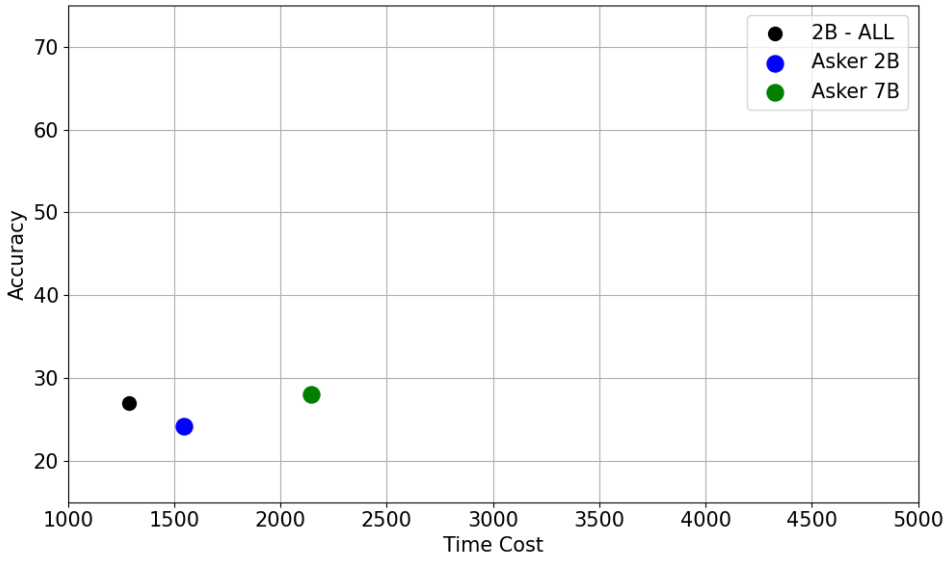

Figure 4.11: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples	(1286, 26.99)
2B	(1547, 24.18)
7B	(2145, 28.05)

Table 4.11: Generator 2B performance with different verifiers

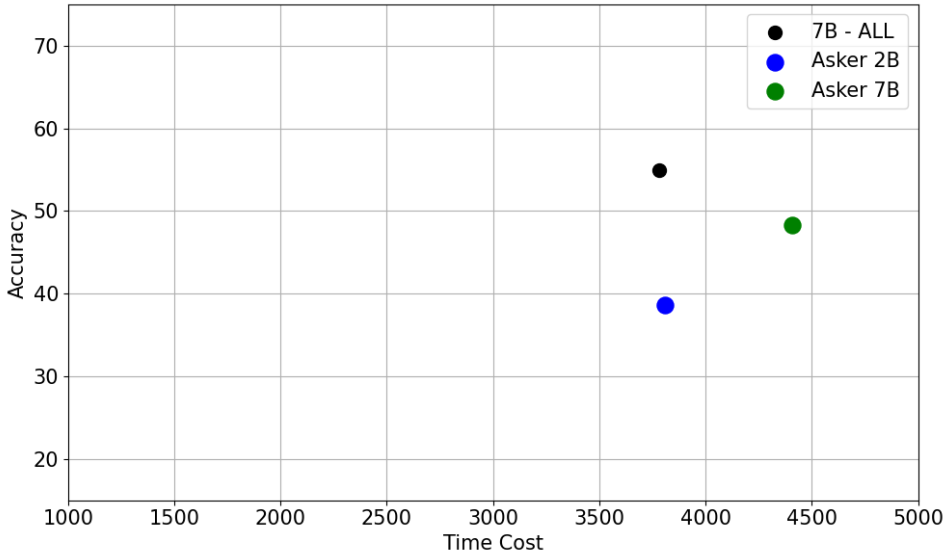

Figure 4.12: Accuracy vs Time Cost

Model	Time (secs), Acc
All Samples	(3784, 54.96)
2B	(3808, 38.66)
7B	(4408, 48.36)

Table 4.12: Generator 7B performance with different verifiers

To summarize all the plots in section 4.2:

- A consistent trend across all Qwen models is that the time cost is higher than the baseline, regardless of model size.
- For all generator sizes except 7B, the 7B verifier demonstrates higher accuracy than the baseline. However, whether this marginal accuracy gain justifies the corresponding increase in time cost is questionable.
- In the case of Gemma, the results are more stable. The time costs remain slightly below the baseline, while accuracy also shows improvement, albeit modest.

4.3 Accuracy of rejected samples

4.3.1 Qwen2.5

		0.5B Verifier	1.5B Verifier	3B Verifier	7B Verifier
0.5B Generator	**Baseline**	0.223 (163/729)	0.179 (113/633)	0.167 (102/610)	0.170 (108/634)
	Replaced	0.200 (146/729)	0.232 (147/633)	0.256 (156/610)	0.267 (169/634)
1.5B Generator	**Baseline**	0.464 (363/782)	0.386 (203/526)	0.393 (217/552)	0.397 (220/554)
	Replaced	0.271 (212/782)	0.335 (176/526)	0.373 (206/552)	0.430 (238/554)
3B Generator	**Baseline**	0.574 (430/749)	0.520 (293/564)	0.494 (219/443)	0.519 (255/491)
	Replaced	0.321 (240/749)	0.404 (228/564)	0.400 (177/443)	0.519 (255/491)
7B Generator	**Baseline**	0.667 (496/744)	0.619 (361/583)	0.623 (317/509)	0.578 (247/427)
	Replaced	0.387 (288/744)	0.497 (290/583)	0.507 (258/509)	0.569 (243/427)

Table 4.13: Accuracy of rejected samples

4.3.2 Gemma

		2B Verifier	7B Verifier
2B Generator	**Baseline**	0.18 (128/703)	0.179 (133/743)
	Replaced	0.122 (86/703)	0.179 (133/743)
7B Generator	**Baseline**	0.47 (332/705)	0.428 (247/577)
	Replaced	0.177 (127/705)	0.282 (163/577)

Table 4.14: Accuracy of rejected samples

- For each generator model, the baseline sub-row corresponds to the case where the model generates the entire COT. Then we pick the same samples as in the replaced set and find the accuracy of the final answer. The replaced sub-row shows the accuracy of the generator model when the first step is replaced by the verifier in the corresponding column. In each cell, the fraction represents the number of samples with correct final answer out of the total incorrect samples.
- It is evident that accuracy exceeds the baseline when the verifier is larger than the generator, indicating that the verifier is stronger.
- This highlights a tradeoff between time cost and accuracy. For larger verifiers, the time cost increases, but there is also a significant improvement in accuracy.
- Another noteworthy observation here is that if the generator is bigger than the verifier, then in some cases the verifier tends to incorrectly mark false positives, i.e., correct FS of generators are marked wrong because they don't match with that of the verifier but since the latter is weaker, there's a high chance the verifier's first step is wrong.

Chapter 5

Findings

In this section we discuss some important findings from the results and possible directions of future research:

- *Interpreting the results*

  Time cost is a widespread issue that many are trying to mitigate, and this work can be applied in scenarios where a slight reduction in accuracy is acceptable for a significant decrease in output time cost. Ultimately, the user must determine the trade-off between time cost and accuracy. Our research assists users by demonstrating that they can avoid unnecessary time costs for questions where the model starts off incorrectly. Instead of continuing, they can skip these questions, knowing that the likelihood of ending with an incorrect final answer is high, as shown in [6]. Furthermore, we show that replacing the first step of the COT with one generated by the verifier can enhance accuracy, but this approach is not cost-effective, which again presents a trade-off scenario. We have also shown that the accuracy of rejected samples increases when the verifier is bigger than the generator and that smaller verifiers might also be incorrect.

- *Future directions*

  Another important observation in the Gemma Correct FS (4.1) case is the smaller gap in accuracy when using the 2B model compared to the 7B model. This raises the possibility that smaller models might perform better in terms of accuracy in certain contexts, while larger models may not always provide a proportional accuracy benefit. This could lead to further exploration of whether smaller models should be preferred as the main model in terms of accuracy, and larger ones in other contexts.

  Another potential extension of our experiment involves verifying each individual step in the Chain of Thought (COT) reasoning process. By identifying and replacing incorrect steps with those generated by the verifier, we could potentially improve overall accuracy while keeping time costs manageable. This approach would provide a more granular control over the reasoning process, potentially correcting errors early before they propagate, leading to more accurate final answers.

  Additionally, another promising direction is to guide the large language model (LLM) using pairs of subquestions and subanswers in gradual steps. Currently, models often attempt to resolve subanswers directly, but guiding them through a series of smaller, well-structured subquestions could lead to more focused reasoning and improved accuracy. This method might help in breaking down complex tasks into simpler parts, where each subquestion prompts the model to think more critically before moving to the next step. Although, this will involve more time cost than the baseline method of only COT. Exploring these directions could offer deeper insights into improving both the efficiency and accuracy of the LLMs.

Chapter 6

Conclusion

To conclude, this project highlights several promising future directions. One key avenue for further exploration is improving the reasoning process by verifying each individual step and correcting errors as they arise, rather than just evaluating the final output. Another valuable approach is scaling these techniques to larger models to assess their effectiveness in broader contexts. These incremental advancements represent meaningful progress in developing self-reasoning capabilities in large language models (LLMs).

The pipeline proposed in this paper offers considerable potential for improvement, especially if we can maximize the accuracy of the verifier. An optimized verifier could not only ensure correctness but also enhance the performance of the generator. By refining the verifier's capabilities, we can establish a more integrated system where the verifier actively contributes to guiding the generator, creating a more efficient and accurate output. This work is an essential step toward the development of more autonomous and self-correcting LLMs, capable of handling increasingly complex tasks with greater precision.

Bibliography

[1] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2023.

[2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[3] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models, 2023.

[4] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes, 2023.

[5] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2024.

[6] Kushal Jain, Niket Tandon, and Kumar Shridhar. Well begun is half done: Importance of starting right in multi-step math reasoning, 2024.

[7] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022.

[8] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[9] Qing Lyu, Kumar Shridhar, Chaitanya Malaviya, Li Zhang, Yanai Elazar, Niket Tandon, Marianna Apidianaki, Mrinmaya Sachan, and Chris Callison-Burch. Calibrating large language models with sample consistency, 2024.

[10] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.

[11] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1773–1781, Toronto, Canada, July 2023. Association for Computational Linguistics.

[12] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

[13] Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy, July 2019. Association for Computational Linguistics.

[14] Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. Refiner: Reasoning feedback on intermediate representations, 2024.

[15] Ethan Perez, Patrick Lewis, Wen tau Yih, Kyunghyun Cho, and Douwe Kiela. Unsupervised question decomposition for question answering, 2020.

[16] Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Sam McCandlish, Sheer El Showk, Tamera Lanham, Tim Maxwell, Venkatesa Chandrasekaran, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Question decomposition improves the faithfulness of model-generated reasoning, 2023.

[17] Kumar Shridhar, Harsh Jhamtani, Hao Fang, Benjamin Van Durme, Jason Eisner, and Patrick Xia. Screws: A modular framework for reasoning with revisions, 2023.

[18] Kumar Shridhar, Jakub Macina, Mennatallah El-Assady, Tanmay Sinha, Manu Kapur, and Mrinmaya Sachan. Automatic generation of socratic subquestions for teaching math word problems. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4136–4149, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[19] Kumar Shridhar, Koustuv Sinha, Andrew Cohen, Tianlu Wang, Ping Yu, Ram Pasunuru, Mrinmaya Sachan, Jason Weston, and Asli Celikyilmaz. The art of llm refinement: Ask, refine, and trust, 2023.

[20] Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling reasoning capabilities into smaller language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073, Toronto, Canada, July 2023. Association for Computational Linguistics.

[21] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology, 2024.

[22] Qwen Team. Qwen2.5: A party of foundation models, September 2024.

[23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[24] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*, 2023.

[25] Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. Answering questions by meta-reasoning over multiple chains of thought, 2024.

[26] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.