Enhancing LLMs for Solving Competition-Level Algebra Problems Using Retrieval-Augmented Generation and Chain-of-Thought Prompting

Param Somane
Department of Computer Science and Engineering
University of California San Diego
psomane@ucsd.edu

Abstract

Mathematical problem-solving remains a significant challenge for large language models (LLMs), especially at the competition level. This paper evaluates and enhances the performance of LLMs in solving competition-level algebra problems by integrating Retrieval-Augmented Generation (RAG) and Chain-of-Thought (CoT) prompting techniques. I conduct experiments using open-source LLMs on a subset of challenging algebra problems from the MATH benchmark. My results demonstrate that combining CoT prompting with advanced models like DeepSeek-Prover significantly improves accuracy. I analyze the limitations of current models, discuss computational constraints, and propose future directions for leveraging retrieval mechanisms and formal methods to enhance mathematical reasoning in LLMs.

1 Introduction

Mathematical reasoning poses a significant challenge for large language models (LLMs). Despite advances in natural language processing (NLP), LLMs often struggle with complex mathematical problems, particularly at the competition-level (Hendrycks et al., 2021). This study aims to evaluate and enhance the capabilities of LLMs in solving competition-level mathematical problems by integrating Retrieval-Augmented Generation (RAG) and Chain-of-Thought (CoT) prompting.

Previously, I participated in the *AI Mathematical Olympiad - Progress Prize 1* Kaggle competition (Investments, 2024), which aimed to develop algorithms and models capable of solving challenging mathematical problems written in LaTeX format. During the competition, I utilized the DeepSeek-Math-7B-RL model (Shao et al., 2024), a specialized LLM fine-tuned for mathematical reasoning tasks. While this model showed promise, its performance on competition-level problems was limited, highlighting the challenges LLMs face in complex mathematical problem-solving.

This experience motivated me to explore methods to enhance LLMs' mathematical reasoning capabilities. In this study, I evaluate and compare different techniques and models, including the newer DeepSeek-Prover-V1.5-RL (Xin et al., 2024), to investigate whether integrating Retrieval-Augmented Generation (RAG) and Chain-of-Thought (CoT) prompting can improve the performance of LLMs on competition-level mathematical problems.

1.1 Contributions

My main contributions are:

- **Evaluation of LLMs on Competition-Level Math Problems**: I assess the performance of open-source LLMs on challenging algebra problems from the MATH dataset.
- **Integration of RAG and CoT**: I develop a unified framework combining Retrieval-Augmented Generation and Chain-of-Thought prompting to enhance mathematical reasoning.
- **Comprehensive Analysis**: I provide detailed analysis of the results, including error analysis, consistency evaluation, and the impact of computational constraints.
- **Future Directions**: I propose strategies for leveraging retrieval mechanisms, such as using ColBERT to retrieve relevant theorems, to further improve LLMs' mathematical problem-solving abilities.

2 Related Work

Mathematical reasoning remains a significant challenge for Large Language Models (LLMs). Early models like GPT-3 (Brown et al., 2020) demonstrated limitations in solving complex mathematical problems, especially those requiring multi-step reasoning. To address these challenges, Wei et al.

(2023) introduced *Chain-of-Thought* (CoT) prompting, encouraging models to generate intermediate reasoning steps, which significantly improved performance on mathematical reasoning tasks.

Building upon CoT, Wang et al. (2023) proposed *Self-Consistency Decoding*, enhancing reasoning by sampling multiple reasoning paths and selecting the most consistent answer. This method has shown to improve the reliability of LLM outputs in mathematical problem-solving.

Retrieval-Augmented Generation (RAG) (Lewis et al., 2021) combines retrieval mechanisms with generation models to provide external knowledge, aiming to enhance performance on knowledge-intensive tasks. The effectiveness of RAG, however, depends on the quality and relevance of the retrieved documents, as highlighted by Khattab and Zaharia (2020).

Specialized models have been developed to push the limits of mathematical reasoning in open-source LLMs. Shao et al. (2024) introduced **DeepSeek-Math**, which leverages reinforcement learning and fine-tuning to improve mathematical problem-solving capabilities. More recently, Xin et al. (2024) presented **DeepSeek-Prover**, harnessing proof assistant feedback for reinforcement learning, showing significant improvements over previous models.

Integrating code execution with LLMs, Chen et al. (2023) proposed *Program-of-Thought* (PoT) prompting, which disentangles computation from reasoning by generating code as intermediate steps. This method leverages symbolic computation libraries like SymPy (Meurer et al., 2017) to enhance accuracy in numerical reasoning tasks.

My work builds upon these advancements by integrating Chain-of-Thought prompting, Self-Consistency Decoding, Retrieval-Augmented Generation, and code execution with SymPy to enhance mathematical problem-solving in LLMs. I specifically evaluate and compare the performance of DeepSeek-Math and DeepSeek-Prover models on competition-level mathematical problems, providing insights into their capabilities and limitations.

3 Dataset

3.1 Data Source

I utilized the MATH dataset (Hendrycks et al., 2021), focusing on the first 50 algebra problems with a difficulty level of 5 (hard). This subset provides a challenging benchmark for evaluating LLMs on competition-level mathematical reasoning.

3.2 Dataset Statistics

- **Total Problems**: 50
- **Average Problem Length**: 177 characters
- **Average Solution Length**: 1,054 characters
- **Category**: Algebra (Hard)

The MATH dataset (Hendrycks et al., 2021) contains competition-level mathematical problems ranging from high school to Olympiad level. Due to computational constraints, I focused on the first 50 algebra problems with a difficulty rating of 5 (hard), providing a challenging benchmark for evaluating LLMs' mathematical reasoning capabilities.

3.3 Data Preprocessing

The dataset was preprocessed to extract problem statements and final boxed answers. LaTeX formatting was converted to plain text where necessary. Problems were tokenized for input into the models, ensuring compatibility with tokenizer limitations.

4 Methodology

4.1 Conceptual Approach

My approach integrates Retrieval-Augmented Generation (RAG) and Chain-of-Thought (CoT) prompting to enhance the problem-solving capabilities of LLMs. By retrieving relevant context from mathematical textbooks, the model is provided with domain-specific knowledge. CoT prompting encourages the model to generate intermediate reasoning steps, improving logical consistency.

4.2 Baseline Methods

I implemented the following baseline methods:

- **Zero-Shot Prompting**: Directly prompting the model to solve the problem without additional context or reasoning steps.
- **Standard CoT Prompting**: Encouraging the model to think step-by-step to arrive at the solution.
- **Code Execution with SymPy**: Utilizing code generation capabilities with SymPy for symbolic computation.

4.3 Advanced Techniques

4.3.1 Retrieval-Augmented Generation (RAG)

I employed RAG to retrieve relevant documents from mathematical textbooks (Zhao, 2023), using ColBERT (Khattab and Zaharia, 2020) for efficient

similarity search. The retrieved documents provide additional context to the LLM during problem-solving.

4.3.2 Self-Consistency Decoding

I implemented self-consistency decoding (Wang et al., 2023), generating multiple reasoning paths and selecting the most consistent answer. This approach aims to improve the reliability of the model's solutions.

5 Implementation Details

5.1 Environment and Resources

Experiments were conducted using two NVIDIA Tesla T4 GPUs (16 GB VRAM each) provided via Kaggle's free GPU resources. Computational constraints limited the size of the dataset and the number of samples for self-consistency decoding.

5.2 Model and Libraries

I utilized the DeepSeek-Prover-V1.5-RL model (Xin et al., 2024), a specialized LLM fine-tuned for mathematical reasoning. For comparative analysis, I also evaluated the DeepSeek-Math-7b-RL model (Shao et al., 2024), which was previously employed during the Kaggle competition.

The implementation was carried out in Python using a variety of libraries and models from HuggingFace:

- **Hugging Face Transformers**: For loading and interacting with pre-trained LLMs.
- **Datasets**: For handling and processing the datasets.
- **Accelerate**: For efficient multi-GPU usage and model preparation.
- **SymPy** (Meurer et al., 2017): For symbolic mathematics and verifying the correctness of solutions.
- **FAISS** (Johnson et al., 2017): For efficient similarity search in high-dimensional spaces, utilized in the retrieval component.
- **Sentence-Transformers** (Reimers and Gurevych, 2019): For generating embeddings of text data.
- **ColBERT** (Khattab and Zaharia, 2020): Implemented via the `ragatouille` package for the retrieval mechanism in Retrieval-Augmented Generation.
- **RAGatouille**: For integrating retrieval and generation in a unified framework.
- **Pandas**: For data manipulation and analysis.
- **Google Generative AI API**: Accessed via the `google.generativeai` client for generating text embeddings of documents for RAG.

These libraries and models were essential in implementing the various components of my methodology, including retrieval mechanisms, symbolic computation, and data analysis.

5.3 Retrieval Mechanism

For RAG, I used the first 300 documents from the math textbooks dataset (Zhao, 2023). ColBERT (Khattab and Zaharia, 2020) was employed to index and retrieve relevant documents based on term similarity.

5.4 Prompt Templates

I designed several prompt templates for different methods, including zero-shot, standard CoT, and code execution prompts. For self-consistency decoding, I generated multiple answers using these prompts and selected the most frequent or consistent one.

6 Results and Analysis

6.1 Evaluation Metrics

Accuracy was calculated based on the correctness of the final numerical answer. Symbolic equivalence was checked using SymPy (Meurer et al., 2017). Consistency was measured as the proportion of identical answers generated during self-consistency decoding.

6.2 Performance Comparison

Method	Accuracy (%)
Zero-Shot Prompting [DSP]	58.0
Standard CoT Prompting [DSP]	**60.0**
Code Execution with SymPy [DSP]	52.0
RAG (DeepSeek-Prover)	56.0
RAG + CoT (DeepSeek-Prover)	50.0
Self-Consistency (3 Samples) [DSP]	58.0
Self-Consistency (5 Samples) [DSP]	48.0
Self-Consistency (3 Samples) [DSM]	22.0

Table 1: Performance comparison of different methods on the test set using DeepSeek-Prover [DSP] and DeepSeek-Math [DSM] models.

The DeepSeek-Math-7B-RL model, which I previously used during the Kaggle competition, achieved an accuracy of only **22%** using Self-Consistency Decoding with three samples. This performance is significantly lower compared to the DeepSeek-Prover-V1.5-RL model under the same conditions,

which achieved **58%** accuracy. The results indicate that DeepSeek-Prover substantially outperforms DeepSeek-Math on the selected set of hard algebra problems from the MATH dataset.

The underperformance of DeepSeek-Math may be attributed to its training focus and limitations in handling the complexity of competition-level problems. In contrast, DeepSeek-Prover incorporates proof assistant feedback and advanced reinforcement learning techniques, enhancing its ability to understand and solve complex mathematical problems more effectively.

The Standard CoT Prompting method achieved the highest accuracy of 60%, indicating its effectiveness in enhancing mathematical reasoning. Combining RAG with CoT did not yield significant improvements, possibly due to computational constraints limiting the retrieval of more relevant documents.

The superior performance of the Standard CoT Prompting method suggests that encouraging explicit reasoning steps is crucial for solving complex mathematical problems. DeepSeek-Prover's strong baseline performance indicates that model architecture and training methodologies play a significant role in mathematical reasoning capabilities.

6.3 Comparison with DeepSeek-AI's Reported Performance

DeepSeek-AI reported that their DeepSeek-Math-RL 7B model achieves **36.2%** accuracy on the MATH dataset (Shao et al., 2024), while the DeepSeek-Prover-V1.5-RL model significantly improves performance on formal theorem-proving tasks (Xin et al., 2024). In my experiments, the DeepSeek-Math model achieved an accuracy of **22%** on a subset of 50 hard algebra problems, which is lower than the reported overall performance.

This discrepancy could be due to several factors:

- **Dataset Selection**: My evaluation focused on the hardest problems (difficulty level 5) in the algebra category, which may present greater challenges than the overall dataset used in DeepSeek-AI's evaluations.
- **Computational Constraints**: Limited computational resources restricted the number of samples and the extent of self-consistency decoding, potentially affecting the model's ability to arrive at correct solutions.
- **Model Updates**: The DeepSeek-Prover model benefits from recent advancements, including proof assistant feedback and enhanced reinforcement learning strategies, leading to superior performance compared to DeepSeek-Math.
- **Prompt Engineering Differences**: The structure and phrasing of prompts used in my experiments may differ from those employed by DeepSeek-AI during their evaluations. Effective prompt engineering is crucial for eliciting accurate and coherent responses from LLMs. Variations in prompt templates, instructions, and context provided can lead to significant differences in model performance.

The significant performance gap underscores the importance of specialized training techniques and the integration of formal methods to improve LLMs' mathematical reasoning capabilities.

6.4 Analysis of Correct Extracted Answers

Method	Avg. Correct Answers
Self-Consistency (5 Samples)	2.14
Self-Consistency (3 Samples)	1.50
RAG + Self-Consistency (3 Samples)	1.42
Zero-Shot Prompting	0.58
Standard CoT Prompting	0.60
Code Execution with SymPy	0.52
RAG	0.56
RAG + CoT	0.50

Table 2: Average number of correct extracted answers by method for DeepSeek-Prover.

Methods utilizing self-consistency decoding with more samples generated a higher average number of correct extracted answers, suggesting that multiple reasoning paths can improve solution accuracy. However, increasing the number of samples from 3 to 5 did not lead to better overall accuracy, indicating diminishing returns.

6.5 Examples of Errors

Here are a few problems where

Problem: *An airplane climbs 100 feet during the first second after takeoff. In each succeeding second it climbs 100 feet more than it climbed during the previous second. How many seconds does it take for the plane to reach an altitude of 12,000 feet above its takeoff height?*

Correct Answer: 15

LLM's Incorrect Answer: 120

Step-by-Step Analysis

The LLM incorrectly used the formula for the n-th term of an arithmetic sequence to represent the total altitude:

$$a_n = a_1 + (n-1)d, \quad \text{setting} \quad a_n = 12,000.$$

This formula calculates the distance climbed in the n-th second, not the cumulative altitude. The correct approach is to use the sum of the first n terms:

$$S_n = \frac{n}{2}\left[2a_1 + (n-1)d\right], \quad \text{with} \quad S_n = 12,000.$$

Solving this equation yields $n = 15$, the correct number of seconds.

Problem: *Let*

$$f(x) = \begin{cases} k(x) & \text{if } x > 3, \\ x^2 - 6x + 12 & \text{if } x \leq 3. \end{cases}$$

Find the function $k(x)$ such that f is its own inverse.

Correct Answer: $k(x) = -\sqrt{x-3} + 3$

LLM's Incorrect Answer: $k(x) = 3 + \sqrt{x-3}$

Step-by-Step Analysis

The LLM correctly identified that $k(x)$ should be the inverse function of $x^2 - 6x + 12$ for $x > 3$. However, it chose the positive square root

$$k(x) = 3 + \sqrt{x-3}.$$

Since f must be its own inverse, and f is defined for $x \leq 3$ with $f(x) \geq 3$, the correct inverse should be

$$k(x) = 3 - \sqrt{x-3},$$

to ensure the outputs map back to the domain $x \leq 3$. The LLM's choice of the positive root leads to outputs outside the intended domain.

Problem: *Dr. Jones lives in a country with a progressive tax system. That is, he does not pay any taxes on the first $20,000 in income he makes, he pays 5% taxes on the next $25,000, he pays 10% taxes on the next $35,000, he pays 15% on the next $50,000, and he pays 20% on every additional dollar onwards. If Dr. Jones pays $10,000 in taxes, then how much income does he make?*

Correct Answer: $115,000

LLM's Incorrect Answer: $98,750

Step-by-Step Analysis

The LLM incorrectly assumed that Dr. Jones's income fell into the highest tax bracket (20%), leading to an illogical negative taxable income when setting up the equation. Here's the step-by-step correct approach Calculate taxes for each bracket

First \$20,000: $0\% \Rightarrow \$0$
Next \$25,000: $5\% \Rightarrow 0.05 \times 25,000 = \$1,250$
Next \$35,000: $10\% \Rightarrow 0.10 \times 35,000 = \$3,500$
Next \$50,000: $15\% \Rightarrow 0.15 \times 50,000 = \$7,500$

Total taxes up to \$130,000:

$$\$1,250 + \$3,500 + \$7,500 = \$12,250$$

Since Dr. Jones pays \$10,000 in taxes, which is less than \$12,250, his income does not reach the top of the 15% bracket. Therefore, his income falls within the 15% tax bracket.

Set up the equation

$$\begin{aligned} \text{Taxes}_{total} &= \$1,250 + \$3,500 + 0.15(x - \$80,000) \\ &= \$10,000 \end{aligned}$$

Solve for x

$$\begin{aligned} 0.15(x - 80,000) &= 10,000 - 1,250 - 3,500 \\ &= 5,250 \\ \therefore x - 80,000 &= \frac{5,250}{0.15} = 35,000 \\ \therefore x &= 80,000 + 35,000 \\ &= \$115,000 \end{aligned}$$

Thus, Dr. Jones's income is \$115,000.

Problem: *What is the domain of the real-valued function*

$$q(x) = \frac{\sqrt{x}}{\sqrt{1-x^2}} \ ?$$

Express your answer as an interval or as a union of intervals.

Correct Answer: $[0, 1)$

LLM's Incorrect Answer: $[0, 1]$

Step-by-Step Analysis

The LLM correctly identified the conditions

$$x \geq 0, \quad 1 - x^2 > 0 \implies x^2 < 1.$$

Combining these gives $0 \leq x < 1$. However, it included $x = 1$ in the domain, despite the denominator $\sqrt{1-x^2}$ becoming zero at $x = 1$, which is undefined. This oversight led to the incorrect domain $[0, 1]$.

Problem: *Billy shoots an arrow from 10 feet above the ground. The height of this arrow can be expressed by the equation $h = 10 - 23t - 10t^2$, where t is time in seconds since the arrow was shot. If the center of a target is raised 5 feet off the ground, in how many seconds must the arrow reach the target in order for Billy to hit the bull's-eye?*

Correct Answer: $\frac{1}{5}$

LLM's Incorrect Answer: $\frac{-23 + \sqrt{329}}{20}$

Step-by-Step Analysis

The LLM correctly set up the quadratic equation:

$$5 = 10 - 23t - 10t^2 \implies 10t^2 + 23t - 5 = 0.$$

Applying the quadratic formula, it calculated:

$$t = \frac{-23 \pm \sqrt{23^2 - 4 \cdot 10 \cdot (-5)}}{2 \cdot 10} = \frac{-23 \pm \sqrt{529 + 200}}{20} = \frac{-23 \pm \sqrt{729}}{20}.$$

However, $\sqrt{729} = 27$, so:

$$t = \frac{-23 + 27}{20} = \frac{4}{20} = \frac{1}{5}.$$

The LLM mistakenly calculated $\sqrt{329}$ instead of $\sqrt{729}$, leading to an incorrect expression for t.

Problem: *If the function $j(x)$ is defined only on domain $[-1, 2]$, and is defined on this domain by the formula $j(x) = 2x^2 + 1$, then what is the range of $j(x)$? Express your answer as an interval or as a union of intervals.*

Correct Answer: $[1, 9]$

LLM's Incorrect Answer: $[3, 9]$

Step-by-Step Analysis

The LLM evaluated $j(x)$ at the endpoints:

$$j(-1) = 2(-1)^2 + 1 = 3, \quad j(2) = 2(2)^2 + 1 = 9.$$

It incorrectly assumed that the minimum value occurs at $x = -1$. However, the function $j(x)$ is minimized at $x = 0$ within the domain $[-1, 2]$

$$j(0) = 2(0)^2 + 1 = 1.$$

Thus, the correct range is $[1, 9]$.

6.5.1 Interpretation of Results

The Standard CoT Prompting method's success suggests that encouraging step-by-step reasoning enhances the LLM's problem-solving capabilities. The limited improvement from RAG indicates that merely adding external knowledge without sufficient computational resources may not be effective. The results also highlight that the number of correct extracted answers does not always directly translate to higher accuracy, as seen with the Self-Consistency Decoding (5 Samples) method.

7 Discussion

7.1 Limitations

Computational constraints significantly impacted my experiments. The use of only 300 documents for RAG and a small number of samples for self-consistency decoding limited the potential benefits of these methods. The limited VRAM of the GPUs restricted the size of the models and the depth of the self-consistency decoding.

7.2 Future Work

I propose leveraging more computational resources to:

- **Enhance Retrieval Mechanisms**: Segregate textbooks into individual theorems and concepts to retrieve more relevant context using ColBERT (Khattab and Zaharia, 2020). This could involve indexing mathematical theorems and definitions to provide precise information during problem-solving.
- **Integrate Formal Methods**: Combine LLMs with formal theorem provers to validate solutions, similar to approaches in (Gou et al., 2024).
- **Expand Dataset**: Use a larger dataset with diverse mathematical fields, including number theory and geometry, to assess the generalizability of the methods.
- **Scaling Computational Resources**: Leveraging more powerful hardware to test the impact of increased sample sizes in self-consistency decoding and larger retrieval corpora.

8 Conclusion

This study demonstrates that integrating CoT prompting enhances the mathematical problem-solving capabilities of LLMs. While RAG did not significantly improve performance under computational constraints, self-consistency decoding shows

promise when appropriately balanced with resource limitations. The DeepSeek-Prover model outperforms previous models, indicating that integrating formal methods and advanced reinforcement learning strategies is a promising direction for future research. Future work should focus on improving retrieval mechanisms, leveraging larger computational resources, and exploring the integration of formal mathematical knowledge into LLM reasoning processes.

9 Code Availability

The code and resources used in this study are publicly available at the following GitHub repository: https://github.com/ChestnutKurisu/CSE256_Final_Project_FA24. Readers are encouraged to access the repository for detailed implementations, datasets, and additional resources related to this research.

10 Acknowledgments

OpenAI's ChatGPT (model gpt-4o) was utilized to generate code, assist in adding comments and docstrings to Python functions, to help debug code errors, and assist in LaTeXformatting of tables and citations for typing up this report. The outputs from this AI model were modified with major changes to align with assignment requirements and ensure correctness. I actively reviewed, tested, and adjusted the generated code and explanations to reflect my own understanding.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

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

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset.

XTX Investments. 2024. Ai mathematical olympiad - progress prize 1. https://kaggle.com/competitions/ai-mathematical-olympiad-prize. Kaggle Competition.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, and Sean Vig. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. 2024. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search.

Wenting Zhao. 2023. Math textbooks dataset. https://huggingface.co/datasets/wentingzhao/math-textbooks. Dataset accessed on Mon 25 Mar 2024.