FinanceMATH: Knowledge-Intensive Math Reasoning in Finance Domains

Yilun Zhao[*1] **Hongjun Liu**[*2,3] **Yitao Long**[3]
Rui Zhang[4] **Chen Zhao**[2,3] **Arman Cohan**[1,5]

[1]Yale University [2]NYU Shanghai [3]New York University
[4]Penn State University [5]Allen Institute for AI

github.com/yale-nlp/FinanceMath
financemath-acl2024.github.io

Abstract

We introduce FinanceMATH, a novel benchmark designed to evaluate LLMs' capabilities in solving knowledge-intensive math reasoning problems. Compared to prior works, this study features three core advancements. First, FinanceMATH includes 1,200 problems with a hybrid of textual and tabular content. These problems require college-level knowledge in the finance domain for effective resolution. Second, we provide expert-annotated, detailed solution references in Python program format, ensuring a high-quality benchmark for LLM assessment. We also construct a finance-domain knowledge bank and investigate various knowledge integration strategies. Finally, we evaluate a wide spectrum of 51 LLMs with both Chain-of-Thought and Program-of-Thought prompting methods. Our experimental results reveal that the current best-performing system (*i.e.,* GPT-4o) achieves only 60.9% accuracy using CoT prompting, leaving substantial room for improvement. Moreover, while augmenting LLMs with external knowledge can improve model performance (*e.g.,* 47.5% → 54.5% for Gemini-1.5-Pro), their accuracy remains significantly lower than the estimated human expert performance of 92%. We believe that FinanceMATH can advance future research in the area of domain-specific knowledge retrieval and integration, particularly within the context of solving reasoning-intensive tasks.

1 Introduction

Large language models (LLMs) have been increasingly recognized for their potential for complex problem-solving in real-world scenarios (OpenAI, 2023a; Touvron et al., 2023; Jiang et al., 2023). Solving math reasoning problems has emerged as a key method for assessing LLMs' capabilities (Roy and Roth, 2015; Amini et al., 2019; Cobbe et al., 2021; Chen et al., 2023c), as it demands both understanding contextual information and reasoning over complex logics.

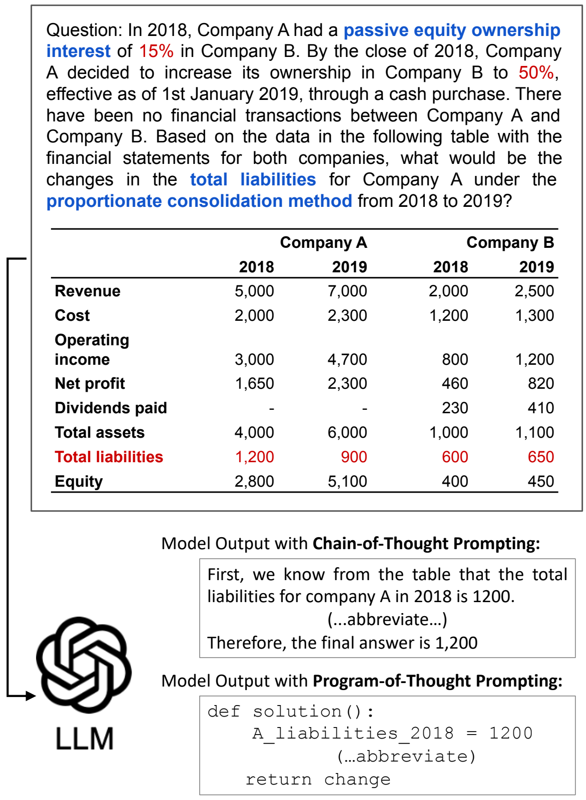

Figure 1: An example of FinanceMATH. To answer the given question, LLMs are required to comprehend specialized financial terms, such as "passive equity ownership interest" and "proportionate consolidation method". Additionally, they must interpret tabular data within the question and accurately identify question-relevant data points in the table.

Recent advancements in LLMs have led to remarkable progress in solving fundamental math problems (Wei et al., 2022; Lewkowycz et al., 2022; Chen et al., 2023b; Wang et al., 2023; Luo et al., 2023a; Azerbayev et al., 2024). However, as illustrated in Table 1, existing math reasoning bench-

[*]Equal Contribution

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12841–12858

Dataset	Domain	Level	Source	# Examples	Table Reasoning?	Knowledge-Intensive?	Solution Format
MAWPS (Koncel-Kedziorski et al., 2016)	Math	Elem. School	Generated	3,320	✗	✗	Text
ASDiv (Miao et al., 2020)	Math	Elem. School	Internet	2,305	✗	✗	Math Equation
SVAMP (Patel et al., 2021)	Math	Elem. School	ASDiv	1,000	✗	✗	Math Equation
Math23K (Wang et al., 2017)	Math	Elem. School	Internet	23,162	✗	✗	Math Equation
GSM8K (Cobbe et al., 2021)	Math	Middle School	CrowdSource	8,500	✗	✗	Text
MATH (Hendrycks et al., 2021)	Math	High School	Competition	12,500	✗	✗	Text
AQuA (Ling et al., 2017)	Math	College	GMAT, GRE	100,000	✗	✗	Text
MathQA (Amini et al., 2019)	Math	College	AQuA	100,000	✗	✗	Math Equation
MathQA-Python (Austin et al., 2021)	Math	College	AQuA	23,914	✗	✗	Python Program
MathVista (Lu et al., 2024)	Math	Elem. to College	Internet+Expert	6,141	Few	Few	Text
TabMWP (Lu et al., 2023)	Math	Middle School	Textbooks	38,431	✓	✗	Text
FinQA (Chen et al., 2021)	Finance	College	Expert	8,281	✓	✗	Math Program
TAT-QA (Zhu et al., 2021)	Finance	College	Expert	16, 552	✓	✗	Text
MultiHiertt (Zhao et al., 2022)	Finance	College	Expert	10,440	✓	✗	Math Equation
DocMath-Eval (Zhao et al., 2023a)	Finance	College	Expert	5,974	✓	Few	Python Program
TheoremQA (Chen et al., 2023c)	STEM	College	Internet+Expert	800	✗	✓	Text
FinanceMATH (ours)	Finance	College	Internet+Expert	1,200	✓	✓	Python Program

Table 1: Comparison between FinanceMATH and existing **math reasoning** datasets. FinanceMATH is distinguished by three unique characteristics: (1) *Knowledge-Intensive*: Problems necessitate domain-specific knowledge, complemented by a financial knowledge bank for research facilitation; (2) *Table Reasoning*: 40.2% of problems incorporate table information, requiring models to understand table structure as well as interpret and reason over tabular data; (3) *Expert Annotation*: Each problem is accompanied by a detailed, expert-annotated Python-formatted solution. Such solution annotation combines the explicitness of code execution with the descriptive power of natural language explanations in python comment format, offering a more effective and adaptable solution representation for complex math reasoning problems in FinanceMATH.

marks typically do not require specialized domain knowledge. This becomes a notable shortcoming when considering practical applications of LLMs. Measuring progress in specialized areas such as finance and healthcare typically involves addressing *domain-specific* and *knowledge-intensive* problems, which goes beyond the scope of general mathematical reasoning. Recognizing this gap in the existing benchmarks, we focus on the finance domain. We chose this domain because, as illustrated in Figure 1, it often involves scenarios requiring not only basic mathematical skills but also a deep understanding of financial concepts (Yang et al., 2023; Xie et al., 2023; Wu et al., 2023). Additionally, the finance domain frequently employs tables to represent data (Zhu et al., 2021; Chen et al., 2021; Zhao et al., 2022; Li et al., 2022b,a; Zhao et al., 2023b,d), which adds another layer of complexity to the knowledge-intensive problem-solving.

We introduce FinanceMATH, the first benchmark tailored for evaluating LLMs in the context of knowledge-intensive math reasoning in the Finance domain. The dataset contains 1,200 problems that cover a broad range of finance subareas, with 40.2% of the problems necessitating data interpretation over tabular data. Each problem is accompanied by expert-annotated, Python-formatted solutions, providing a comprehensive reference for evaluating the LLMs' performance. Additionally, we collect and release a comprehensive knowledge bank, which includes detailed definitions and explanations for 864 financial terms and concepts, facilitating future research on improving knowledge-intensive problem-solving through knowledge retrieval.

We evaluate a wide spectrum of open-source and proprietary LLMs, specifically, 51 model models from 16 organizations. Notably, this includes *math-specific* (Luo et al., 2023a; Shao et al., 2024; Ying et al., 2024), *code-based* (Guo et al., 2024; Luo et al., 2023b; AI@Mistral, 2024a; Lozhkov et al., 2024) LLMs, as well as *mixture of experts* (MoE) LLMs (Mistral.AI, 2023; Databricks, 2024). Two prompting methods, Chain-of-Thought (Wei et al., 2022) and Program-of-Thought (Chen et al., 2023b), are adopted for experiments.

Our experimental results demonstrate a significant gap between existing LLMs and human experts. Specifically, the current best-performing system (*i.e.,* GPT-4o) achieves only 60.9% accuracy with CoT prompting, which still lags far behind human expert performance in the open-book setting, which stands at 92%. These results highlight the challenges of FinanceMATH, underscoring the need for further advancements in LLMs for knowledge-intensive problem-solving capabilities. Next, we investigate how to integrate domain-specific knowledge to enhance the problem-solving capabilities of LLMs. We investigate various popular knowledge integration strategies and reveal that including question-relevant knowledge into the prompt can consistently improve LLMs' performance. This provides insights for future work to develop more advanced knowledge-augmented strategies to realize higher performance gains.

Our contributions are summarized below:

- We propose FinanceMATH, the first knowledge-intensive math reasoning benchmark in finance domains, aimed at evaluating LLMs' abilities in knowledge-intensive math reasoning.
- We conduct comprehensive evaluations using a diverse array of LLMs, uncovering a substantial performance gap between the best-performing LLM (*i.e.,* GPT-4o) and human experts.
- We present a detailed analysis on augmenting LLMs with various knowledge integration strategies. This provides valuable insights for future work in knowledge-intensive problem solving.

2 FinanceMATH Benchmark

In this section, we describe the dataset construction process for FinanceMATH. We begin by constructing a knowledge bank that includes well-formulated definitions of 864 financial terms. We then instruct expert annotators to use knowledge terms within the constructed knowledge bank to create knowledge-intensive questions with a hybrid of textual and tabular content.

2.1 Knowledge Bank Construction

We construct a knowledge bank that covers a wide range of 864 knowledge terms in the finance domain. It simplifies the creation of knowledge-intensive questions by annotators and enables the exploration of various topics within domain knowledge. The knowledge bank includes finance-domain-specific terms (*e.g.,* "exchange rate" and "net present value") collected from Wikipedia. Each knowledge term is accompanied with their corresponding *textual definitions* and, where applicable, *mathematical formulas* in python format. An example of included knowledge terms is illustrated in Figure 2. We detail the the main processes for knowledge bank construction as follows:

Knowledge Term:
Exchange Rate

Definition:
An exchange rate is the value or price of one country's currency in relation to another currency. It determines how much of one currency can be exchanged for another and can fluctuate regularly based on market conditions, import and export demand, inflation, and a host of other economic factors.

Mathematical Formula:
```
def exchange_rate(original_currency, new_currency):
        return original_currency / new_currency
```

Figure 2: An example of knowledge terms "Exchange Rate" included in the constructed knowledge bank.

Knowledge Collection To construct a knowledge bank, we first collect knowledge relevant to the finance domain from Wikipedia using "finance" and "economics" as key search terms. After collecting the raw financial data, we adopt comprehensive heuristics, embedding-based methods to remove duplicates. This procedure ensures the uniqueness of each knowledge term in our bank.

Automatic Knowledge Formulation To enhance the adaptability and usability of the knowledge bank, we incorporate a two-step automatic knowledge formulation process, making each piece of collected knowledge standardized and distilled into a clear, concise format. The primary motivation for using *automatic* knowledge formulation is cost efficiency and effectiveness. We have observed that GPT-3.5 models are adept at handling this straightforward task with minimal bias, as this process does not involve the addition of extraneous knowledge. We first prompt GPT-3.5 to reformulate the gathered information for each financial term into a concise, paragraph-long textual definition. Since some financial terms come with mathematical definitions, we address the issue of varied formula formats in the original sources (e.g., LaTeX and HTML). We instruct GPT-4 to transform these formulas into a unified python program format. Figure 2 illustrates an example of knowledge terms collected in the knowledge bank.

Knowledge Bank Update and Maintenance After formulating knowledge using LLMs, during the dataset annotation stage (Section 2.2), we dynamically update and maintain the constructed knowledge bank, incorporating new knowledge that, although not initially covered, is essential for answering the annotated questions. Additionally, we remove any duplicate entries identified by the annotators. We eventually collect 864 pieces of financial knowledge in the knowledge bank, with 57.4% of the terms including Python-formatted mathematical definitions.

2.2 FinanceMATH Question Annotation

For each financial term in the knowledge bank, we instruct annotators to create a corresponding math reasoning question, if applicable. The answer to the composed question should be a numeric value. The annotators are required to adhere to the following guidelines for a successful question annotation:

Question Annotation If the annotators choose to adapt questions from textbooks or the Internet instead of creating their own from scratch, they are asked to adhere to copyright and license regulations, avoiding data from sites prohibiting copy and redistribution. Furthermore, they are required not only to modify the surface-level description of the question but also to change the associated numeric values. In light of the emerging concerns about *data contamination* in LLMs (Shi et al., 2024; Deng et al., 2024a), we instruct annotators to conduct a Google search for each annotated question, ensuring that no similar question appears on the first page of the search results. Additionally, we recognize that many financial problems involve tables, as shown in Figure 1. Such tabular data plays a crucial role in thoroughly understanding financial problems, and it presents unique challenges for LLMs in terms of comprehension and interpretation. Therefore, we encourage and reward annotators to include tables that are relevant and accurately represent the data pertinent to the questions. Finally, out of 1,200 questions, 674 are marked as having been adapted from existing resources, and 482 are accompanied with tabular data.

Identifying Question-relevant Knowledge After a question is annotated, annotators must identify 1-3 key financial concepts for answering this question. They then search for each term in our constructed knowledge bank. If the term is included, they verify its context and details for relevance. If a term is absent or with low-quality definition, annotators receive a bonus for documenting the term, providing a brief explanation or definition and outlining its relevance to the problem. These identified terms are subsequently added or updated in the knowledge bank, resulting in a total of 123 new inclusions and 47 revisions.

2.3 FinanceMATH Solution Annotation

As illustrated in Table 1, existing math reasoning benchmarks typically represent solutions using text or mathematical equations. However, solutions in text format often lack the precision and unambiguous nature required for computational problem-solving. Solutions in mathematical equations are explicit, but less descriptive, as the semantic meaning associated with each numeric value in the equations can be ambiguous. Moreover, these two formats are less adaptable for use in automated systems due to variations in language and difficulties in semantic parsing and execution.

To overcome these limitations, we use Python programs, starting with "`def solution():`", to represent solutions. Such Python program combines the explicitness of code execution with the descriptive power of annotated comments, offering a more effective and adaptable solution representation for complex math reasoning problems. Specifically, annotators are required to first define variables with meaningful names at the beginning of the Python function. These variables correspond to the key elements or quantities mentioned in the textual or tabular content of questions. The annotators then proceed to write a sequence of Python statements that logically solve the problem, step by step. To ensure the accuracy and functionality of the Python-format solutions, our annotation interface automatically executes the Python function. This execution checks that the return type of the answer is either a float or an int and verifies that there are no execution errors.

2.4 Data Quality Validation

We conduct a comprehensive validation protocol to ensure the high quality of FinanceMATH. For each example, we first assign another annotator to validate whether: 1) the question is meaningful and grammatically correct, 2) the associated knowledge terms are accurately annotated and complete, 3) the Python-format solution is logically correct and easy to understand. Validators are asked to revise examples that do not meet these standards.

We also report the human evaluation scores over 200 randomlysampled examples. As illustrated in Table 5 in the Appendix, FinanceMATH has a high annotation quality.

Property	Value
Knowledge Bank	
# Knowledge Terms	864
Textual Definition Length (Median/Avg)	47.1 / 49.7
% *w.* Mathematical Definition	57.4%
FinanceMATH Dataset	
Question Length (Median/Avg)	54.0 / 61.8
% Questions with Table	482 (40.2%)
# Rows per Table (Median/Avg)	3.0 / 3.0
# Columns per Table (Median/Avg)	4.0 / 5.0
# Knowledge Terms per Example (Median/Avg)	2.5 / 2.4
# Math Operations in Python Solution (Median/Avg)	5.0 / 5.6
# Lines in Python Solution (Median/Avg)	6.0 / 6.6
Development Set Size	200
Test Set Size	1,000

Table 2: Basic statistics of the constructed knowledge bank and FinanceMATH dataset.

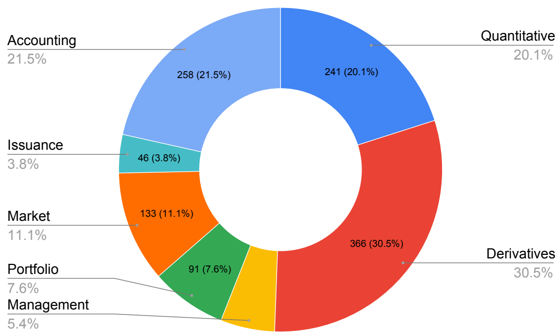

Figure 3: Topic distribution of FinanceMATH.

2.5 Data Statistics and Dataset Release

Table 2 describes the basic statistics of FinanceMATH, with topic-type distribution shown in Figure 3. We randomly divide the dataset into two subsets: *development* and *test*. The *development* set contains 200 examples and is intended for model development validation. The *test* set comprises the remaining 1,000 examples and is designed for standard evaluation. To prevent data contamination (Shi et al., 2024; Sainz et al., 2023; Deng et al., 2024b), the answer for the *test* set will not be publicly released. Instead, we develop and maintain an online evaluation platform, allowing researchers to evaluate models and participate in a leaderboard. Following recent LLM reasoning benchmarks (Chen et al., 2023c; Yue et al., 2024; Lu et al., 2024), the main evaluation of FinanceMATH is conducted under a *zero-shot* setting on the *test* set to assess LLMs' capabilities to generate accurate answers without fine-tuning or few-shot demonstrations on our benchmark.

2.6 Human-level Performance Evaluation

To provide a rough but informative estimate of human-level performance by non-experts and experts on FinanceMATH, we randomly sampled 50 examples from the *validation* set. We enroll two experts, both with the CFA license, and two non-experts to individually solve these questions.

We first evaluate their performance in a *closed-book* setting, where the evaluators do not have access to the internet or textbooks and are required to finish the 50 questions within three hours. The non-expert evaluators achieve accuracy of 54% and 62% (average 58%), and the expert evaluators achieve accuracy of 76% and 70% (average 73%).

We then transition to an *open-book* setting, where the evaluators are asked to use the internet and textbooks to correct their initial errors. This setting is designed to assess how external knowledge resources could enhance human problem-solving abilities and accuracy. The non-expert evaluators improved their accuracy to 86% and 82% (average 84%). Similarly, the expert evaluators improved the accuracy to 94% and 90% (average 92%).

3 Evaluated Systems

This section discusses the investigated LLMs and prompting methods in our work.

3.1 Large Language Models

We evaluate following LLMs on FinanceMATH:

- **General**: GPT-3.5&4 (OpenAI, 2022, 2023a, 2024), Gemini-1.5 (Gemini, 2024), Claude-3&3.5 (Anthropic, 2024), Llama-2&3&3.1 (Touvron et al., 2023; AI@Meta, 2024), Mistral (Jiang et al., 2023), Phi-3 (Abdin et al., 2024), Gemma-1&2 (Team et al., 2024), WizardLM-2 (Xu et al., 2023), Yi-1.5 (01.AI, 2023), Qwen-2 (Bai et al., 2023), Command R+ (Cohere, 2024b), Aya (Cohere, 2024a), and GLM-4 (GLM et al., 2024).
- **Math-specific**: WizardMath (Luo et al., 2023a), DeepSeek-Math (Shao et al., 2024), Mathtral (AI@Mistral, 2024b), and InternLM-Math (Ying et al., 2024).
- **Code-based**: DeepSeek-Coder-V1 (Guo et al., 2024), WizardCoder (Luo et al., 2023b), Codestral (AI@Mistral, 2024a), DeepSeek-Coder-V2 (also MoE architecture, DeepSeek-AI (2024)), and StarCoder2 (Lozhkov et al., 2024).

Chain-of-Thought Prompt

```
[System Input]:
```
You are a financial expert, you are supposed to answer the given question. You need to first think through the problem step by step, documenting each necessary step. Then you are required to conclude your response with the final answer in your last sentence as "Therefore, the answer is {final answer}". The final answer should be a numeric value.

```
[User Input]:
```
Question: {question}

Let's think step by step to answer the given question.

Figure 4: Example of *zero*-shot CoT prompt used.

- **Mixture of Experts (MoE)**: Mixtral (Mistral.AI, 2023), WizardLM-2 (MoE, Xu et al. (2023)), DeepSeek-V2 (DeepSeek-AI, 2024), and DBRX (Databricks, 2024).

We select the most recent checkpoint available as of August 1, 2024. The details of each evaluated model, including the exact model version we used, are presented in Table 7 in Appendix. The experiments for open-sourced LLMs were conducted using `vLLM` framework (Kwon et al., 2023). For all the experiments, we set temperature as 1.0, Top P as 1.0, and maximum output length as 512.

3.2 Prompting Methods

Following Chen et al. (2023c) and Lu et al. (2024), we evaluate two established prompting methods, with examples of prompt illustrated in Figure 4 and Figure 6 in the Appendix, respectively.

Chain-of-Thought The CoT method (Wei et al., 2022; Kojima et al., 2022) instructs the LLMs to articulate a step-by-step reasoning process. This leads to a detailed explanation that culminates in the final answer.

Program-of-Thought Different from CoT, the PoT method (Chen et al., 2023b) disentangles computation from the reasoning process by prompting the LLMs to generate a structured program to represent the reasoning process. The final answer is then derived by executing the generated program with an external calculator.

4 Experiments

4.1 Experiment Setup

Final Answer Extraction For LLM with CoT prompting, we adopt the answer extraction pipeline from Lu et al. (2024) and Chen et al. (2023c) to identify the final answer from the model's output. For LLM with PoT prompting, we first extract the generated python solution from the model's output. If this python solution is executable, we execute it to obtain the final answer. Once we obtain the final answer from model's output, we compare it with the ground-truth answer for accuracy measurement.

Tabular Data Serialization Following previous work on table-relevant tasks (Chen, 2023; Zhao et al., 2023c), we use Markdown format to present tabular data in math reasoning problems. In our preliminary study, we discovered that GPT-* and Llama-3 models can effectively understand such table representations.

4.2 Main Results

Table 3 and Table 8 in Appendix illustrate the performance of the evaluated LLMs using CoT and PoT prompting methods on the FinanceMATH test and development sets, respectively.

The experimental results demonstrate that FinanceMATH poses significant challenges to current LLMs. Even the best-performing LLM, GPT-4o, performs much worse than human experts. Specifically, the accuracy of GPT-4o using the CoT prompting method stands at 60.9%, falling short of the 92% accuracy achieved by expert evaluators in the open-book setting. This gap highlights the critical need for further advancements in LLMs, especially in complex problem solving within specialized domains that are knowledge-intensive.

Open-source LLMs still significantly lag behind the most advanced versions of the three major families of proprietary LLMs. However, the two DeepSeek-V2 models are an exception. They achieve performance levels close to those of the best-performing proprietary models. This indicates the potential of open-source LLMs to close the performance gap with proprietary models in the near future, given continued innovation and community collaboration. Additionally, the proprietary LLMs and code-specific models typically achieve comparable or better performance when using PoT prompting compared to CoT prompting. For math-specific LLMs, InternLM2-Math-Plus surpasses its

Model	Size	Notes	Portfolio		Derivatives		Accounting		Quantitative		Corporate		Management		Economics		Avg.	
			PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	**CoT**
Close-book																		
Expert																	73.0	
Non-Expert																	58.0	
Open-book																		
Expert																	92.0	
Non-Expert																	84.0	
Proprietary LLMs																		
GPT-4o			75.0	45.8	58.8	55.4	60.3	69.4	82.9	66.3	77.8	69.4	62.5	58.9	67.0	56.9	67.0	60.9
Claude-3.5-Sonnet			73.6	55.6	54.1	51.8	66.7	69.9	75.6	63.9	72.2	72.2	64.3	58.9	62.4	60.6	64.8	60.6
Claude-3-Opus			66.7	56.9	53.5	45.2	62.1	59.8	79.5	64.9	72.2	83.3	51.8	46.4	59.6	45.0	62.9	54.7
GPT-4-Turbo			59.7	38.9	49.8	42.2	50.7	64.8	72.2	56.6	61.1	50.0	57.1	44.6	50.5	47.7	56.2	50.9
Gemini-1.5-Pro			68.1	50.0	53.1	30.7	56.6	55.2	69.8	57.6	58.3	63.9	51.8	55.4	50.5	44.0	58.2	47.0
GPT-4o-Mini			65.3	36.1	46.9	29.7	48.4	47.5	69.3	46.8	50.0	38.9	57.1	41.1	51.4	45.9	54.3	40.3
Gemini-1.5-Flash			69.4	33.3	43.6	28.7	52.0	48.9	67.8	49.8	58.3	61.1	50.0	37.5	47.7	34.9	53.6	40.1
Claude-3-Sonnet			59.7	37.5	37.0	28.4	48.0	43.4	66.8	48.8	47.2	55.6	48.2	33.9	48.6	35.8	49.4	38.6
Claude-3-Haiku			34.7	31.9	19.8	26.4	33.8	43.4	44.9	41.5	41.7	44.4	25.0	33.9	36.7	30.3	32.0	35.1
GPT-3.5-Turbo			47.2	25.0	24.4	16.5	29.2	29.2	51.2	33.2	27.8	22.2	37.5	21.4	32.1	23.8	34.3	24.6
Open-source LLMs																		
DeepSeek-V2	236B	MoE	72.2	43.1	49.8	46.5	56.6	53.9	77.6	68.8	61.1	63.9	57.1	44.6	59.6	56.9	60.5	54.1
DeepSeek-Coder-V2	236B	Code	65.3	44.4	50.2	43.6	58.9	56.6	77.1	67.3	52.8	72.2	51.8	50.0	57.8	53.2	59.7	53.8
Llama-3.1	405B		76.4	50.0	48.8	34.6	58.0	56.6	76.6	52.7	69.4	55.6	57.1	41.1	54.1	47.7	60.3	46.8
Llama-3.1	70B		62.5	38.9	39.3	29.7	47.5	53.0	65.8	50.7	61.1	58.3	44.6	44.6	44.0	36.7	49.8	42.4
Mistral-Large	123B		59.7	36.1	44.9	29.4	49.8	48.0	75.6	47.3	63.9	36.1	50.0	33.9	52.3	44.0	55.1	39.7
Qwen2	72B		41.7	30.6	30.7	23.4	38.8	48.0	52.7	42.0	44.4	50.0	39.3	33.9	38.5	36.7	39.6	36.1
Llama-3	70B		56.9	36.1	39.9	23.4	46.6	43.4	65.8	42.9	58.3	47.2	51.8	39.3	44.0	34.9	49.7	35.7
Phi-3-Medium	14B		40.3	31.9	31.4	23.1	36.1	47.0	54.2	42.4	41.7	50.0	39.3	32.1	35.8	28.4	39.0	35.0
Mixtral-8x22B	141B	MoE	15.3	31.9	4.6	23.4	9.6	35.2	22.9	38.0	5.6	36.1	16.1	33.9	3.7	30.3	10.8	31.4
DeepSeek-Coder-V2-Lite	16B	Code	38.9	36.1	19.1	21.1	23.3	31.5	49.3	42.4	33.3	30.6	26.8	26.8	26.6	26.6	29.4	30.1
Gemma-2	9B		30.6	34.7	19.1	19.8	31.5	34.7	43.4	37.6	25.0	33.3	33.9	25.0	27.5	26.6	29.6	29.3
Yi-1.5	9B		23.6	29.2	14.8	14.8	18.3	35.2	20.0	40.0	16.7	27.8	10.7	25.0	18.4	30.3	17.5	28.2
Yi-1.5	34B		19.4	25.0	16.2	16.8	18.3	35.2	23.9	36.6	19.4	19.4	23.2	28.6	18.4	28.4	19.2	27.5
WizardLM-2	141B	MoE	30.6	23.6	18.5	17.2	26.0	33.3	40.0	32.2	30.6	30.6	25.0	33.9	25.7	29.4	27.0	27.0
Phi-3-Mini	3B		27.8	22.2	12.9	13.5	28.3	31.0	38.0	35.1	27.8	19.4	28.6	32.1	16.5	20.2	24.3	24.4
Mistral-Nemo	12B		31.9	25.0	13.5	13.2	12.8	25.1	32.7	31.2	16.7	27.8	14.3	23.2	19.3	24.8	19.4	22.7
DBRX	132B	MoE	12.5	20.8	11.2	14.5	13.2	27.4	26.3	27.3	11.1	22.2	16.1	28.6	17.4	21.1	15.8	22.2
DeepSeek-Math	7B	Math	0.0	22.2	0.3	12.2	0.9	19.6	2.4	30.7	0.0	30.6	1.8	26.8	0.9	22.9	1.0	21.0
Qwen2	7B		8.3	18.1	5.3	12.9	4.6	24.7	19.5	29.3	5.6	16.7	14.3	23.2	6.4	18.4	8.9	20.5
GLM-4	9B		27.8	20.8	13.2	11.6	25.1	25.6	36.1	23.4	22.2	25.0	19.6	28.6	21.1	19.3	23.1	20.0
C4AI Command R+	104B		2.8	22.2	1.3	10.2	1.4	23.3	5.4	22.9	2.8	25.0	0.0	14.3	1.8	22.9	2.3	18.7
Mixtral-8x7B-v0.1	46B	MoE	0.0	22.2	0.0	11.9	0.0	16.4	1.5	24.9	0.0	13.9	0.0	17.9	0.0	21.1	0.3	17.7
Llama-3.1	8B		22.2	18.1	12.9	10.2	14.6	14.2	35.1	28.3	19.4	16.7	21.4	19.6	16.5	22.0	19.6	17.4
Mathstral	7B	Math	18.1	18.1	7.9	9.6	11.0	18.3	26.8	24.9	13.9	8.3	19.6	19.6	14.7	16.5	14.8	16.5
Codestral	22B	Code	41.7	13.9	16.8	9.9	23.7	16.0	54.2	25.8	19.4	11.1	42.9	16.1	26.6	19.3	30.4	16.2
Llama-3	8B		13.9	11.1	12.2	7.6	17.8	18.3	26.3	20.0	16.7	16.7	17.9	14.3	19.3	15.6	17.7	14.3
WizardLM-2	7B		23.6	8.3	6.6	6.9	11.4	18.7	22.0	19.0	11.1	8.3	16.1	8.9	12.8	14.7	13.4	13.1
WizardMath	7B	Math	5.6	11.1	5.0	6.3	10.5	16.9	18.5	12.7	5.6	11.1	7.1	21.4	9.2	15.6	9.6	12.3
DeepSeek-V2-Lite	16B	MoE	5.6	13.9	1.0	6.3	2.7	14.2	7.3	16.1	2.8	11.1	3.6	8.9	3.7	15.6	3.5	11.9
Mistral-v0.3	7B		1.4	13.9	1.3	4.6	1.4	15.1	6.8	15.6	2.8	8.3	0.0	10.7	2.8	11.9	2.6	11.1
Aya-23	35B		0.0	8.3	0.3	7.9	0.0	13.7	1.0	12.7	0.0	11.1	0.0	10.7	0.0	10.1	0.3	10.7
InternLM2-Math-Plus	7B	Math	8.3	16.7	3.0	4.6	5.9	9.1	14.2	19.5	0.0	8.3	12.5	12.5	10.1	8.3	7.5	10.5
Llama-2	70B		13.9	8.3	3.6	6.6	14.2	14.2	12.2	12.2	5.6	13.9	8.9	5.4	10.1	11.9	9.5	10.3
InternLM2	7B		5.6	6.9	4.6	4.0	6.4	11.9	16.1	15.6	5.6	2.8	12.5	7.1	5.5	10.1	8.0	9.1
StarCoder2	15B	Code	29.2	2.8	12.5	4.3	11.9	9.6	35.6	15.6	11.1	2.8	16.1	12.5	20.2	8.3	19.3	8.5
Gemma-1	7B		2.8	5.6	1.0	3.6	1.8	10.5	2.9	7.8	0.0	5.6	1.8	7.1	4.6	11.0	2.1	7.2
WizardCoder	33B	Code	19.4	4.2	5.0	2.6	6.8	5.5	37.1	10.7	8.3	5.6	21.4	3.6	11.9	9.2	14.8	5.9
DeepSeek-Coder-V1	33B	Code	12.5	4.2	2.0	3.0	6.4	5.0	15.6	8.3	5.6	5.6	10.7	7.1	8.3	5.5	7.8	5.2
Llama-2	7B		4.2	0.0	1.0	1.6	2.3	5.5	2.9	5.4	5.6	11.1	0.0	3.6	2.8	9.2	2.2	4.4
Aya-23	8B		1.4	1.4	0.3	2.3	0.0	4.6	0.0	5.8	0.0	2.8	0.0	0.0	0.0	8.3	0.2	4.0
Gemma-1	2B		0.0	0.0	1.3	2.0	3.2	6.4	5.8	5.4	8.3	0.0	0.0	0.0	1.8	5.5	2.8	3.7

Table 3: Results of Chain-of-Thought and Program-of-Thought prompting on the *test* set of FinanceMATH. We use average Accuracy using CoT prompting as the ranking indicator of model performance. Numbers underscored indicate that models with PoT prompting achieves better results than with CoT prompting.

backbone in CoT, improving from 9.1% to 10.5%. This demonstrates the effectiveness of instruction-tuning in enhancing math reasoning.

4.3 Error Analysis

To gain a deeper insight into the capabilities and limitations of open-source LLMs on our dataset, we conduct a comprehensive error analysis and case studies. The error analysis is based on 50 sampled failure cases of Llama-3-70B from the *development* set. We choose the Llama-3 model as the focus since many open-source models are developed using it as the backbone. We identify three common mistakes of current LLMs: **(1) Misinterpretation of Required Knowledge** (27/50): the model fails to accurately identify and interpret

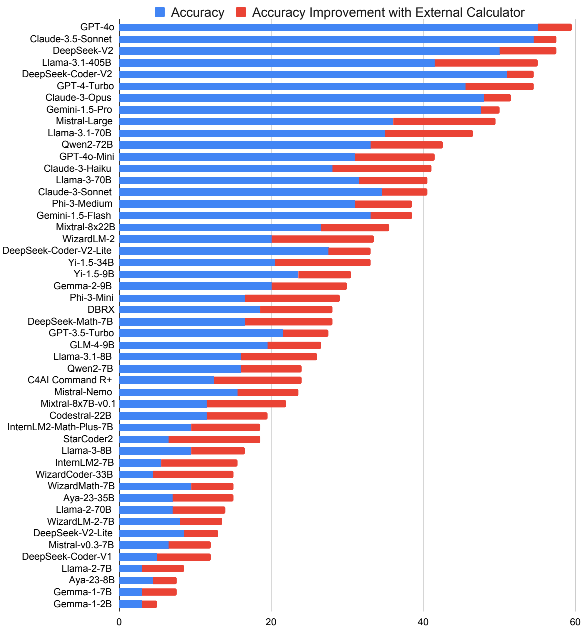

Figure 5: Calibrated results of Chain-of-Thought prompting on the ***development*** set with an external calculator for math computation. Performing complex math computations correctly is still challenging for LLMs, especially open-source ones.

the domain-specific knowledge needed to answer a question correctly, leading to incorrect responses. Table 6 in Appendix illustrates an error example. **(2) Incorrect Math Computation** (19/50): the mathematical computation in the intermediate or final step is incorrect, although the reasoning process is correct. **(3) Table Misunderstanding** (3/50): The model misinterprets the data within complex-structure tables.

To separate computational abilities from final accuracy, we employed an external calculator (Inaba et al., 2023) for CoT outputs. Specifically, we used GPT-3.5-Turbo to extract single-line math expressions from the models' textual responses and executed these expressions to obtain the final answers. Figure 5 illustrates the calibrated results of LLM CoT performance with an external calculator. It demonstrates that performing complex math computations correctly is still challenging for LLMs, especially open-source ones.

4.4 Program-of-Thought Analysis

To better analyze the PoT prompting methods, we examine the execution rate of each LLM under PoT prompting, measuring how many of the generated Python programs are executable. Figure 8 in the Appendix illustrates the relationship between execution rate and accuracy across different models. It demonstrates that for models unable to consistently generate executable programs (*i.e.,* models with an execution rate < 60%), their degraded performance when applying PoT prompting is attributable to the low execution rate. For instance, although Mistral-8$\times$22B achieves competitive performance with CoT, it struggles to consistently generate executable Python solutions, leading to lower accuracy with the PoT prompting approach. Conversely, for LLMs capable of generating executable programs (i.e., models with an execution rate > 80%), the final answer accuracy is mainly attributed to the reasoning capabilities of the models.

5 Knowledge Augmentation Analysis

In this section, we provide a comprehensive analysis to understand the performance of LLMs and the quality of knowledge incorporated into the input context, aiming to provide insights for future work on solving knowledge-intensive tasks.

5.1 Evaluated Knowledge-Augmented Method

We develop and evaluate various knowledge-augmented approaches. For each setting, we include the definition of question-relevant knowledge terms within the prompts (Figure 7 in Appendix).

- **Oracle:** To investigate the headroom in knowledge augmentation, we use an oracle setting, where the *ground-truth* knowledge terms associated with the question (Section 2.2) are included.
- **LLM as Knowledge Base:** Recent work (Petroni et al., 2019; Kang et al., 2023) demonstrates that LLMs themselves can effectively serve as knowledge bases. This approach is particularly valuable in scenarios where an external knowledge base is unavailable. We prompt LLMs to first identify the financial terms required to answer the question. They then generate definitions of each identified knowledge term using the inherent data memorization capabilities.
- **Knowledge Retrieval:** We use the question as the retrieval query to the constructed knowledge bank. We investigate 1) BM25 as sparse retriever and 2) OpenAI Text Embedding V3 Large as dense retriever to retrieve the top-n question-relevant knowledge terms from knowledge bank.
- **LLM-Instructed Knowledge Retrieval:** While the method of using "LLM as Knowledge Base" can effectively identify the knowledge required to answer a question, it is likely to produce

Setting	Llama-3-70B	Gemini-1.5-Pro
wo. knowledge augmentation	31.5	47.5
LLM as Knowledge Base	29.0 (-2.5)	48.5 (+1.0)
BM25 ($n = 3$)		
Vanilla Retrieval	30.0 (-1.5)	44.5 (-3.0)
LLM as Retrieval Re-Ranker	32.0 (+0.5)	49.0 (+1.5)
LLM-instructed Retrieval	32.5 (+1.0)	48.0 (+0.5)
OpenAI Embedding-3-L ($n = 3$)		
Vanilla Retrieval	32.5 (+1.0)	49.0 (+1.5)
LLM as Retrieval Re-Ranker	33.5 (+2.0)	50.5 (+3.0)
LLM-instructed Retrieval	33.5 (+2.0)	52.0 (+4.5)
Oracle	37.5 (+6.0)	54.5 (+7.0)

Table 4: Results of Chain-of-Thought prompting approach under different knowledge augmentation settings on the *development* set of FinanceMATH.

knowledge definitions that are not entirely accurate (Chen et al., 2023a; Peng et al., 2023). To address this unfaithfulness issue, we harness the power of external knowledge retrieval for obtaining more trustworthy knowledge definitions. Specifically, instead of using the original question as the retrieval query, we utilize each knowledge term along with its definition generated from the "LLM as Knowledge Base". This approach provides a more informative and semantically similar basis for knowledge retrieval.

- **LLM as Retrieval Re-Ranker:** Recent studies have demonstrated LLMs' competitive capabilities in re-ranking retrieved candidates to output a more precise list (Sun et al., 2023). Therefore, in this setting, we first use retriever in "Knowledge Retrieval" to retrieve top-$3n$ candidates. Subsequently, we prompt LLMs to select top-n most relevant knowledge terms from this candidate set.

5.2 Knowledge Augmentation Results

As illustrated in Table 4, improving the question-relevance of incorporated knowledge can consistently improve the LLMs' performance. Specifically, LLMs equipped with retrieved knowledge from OpenAI Text Embedding consistently outperform those using retrieved knowledge from BM25, due to the more advanced retrieval capabilities of the former. Among different LLM-aided retrieval strategies, *LLM-Instructed Knowledge Retrieval* achieves the best performance, demonstrating the effectiveness of using *refined* queries for knowledge retrieval. Nevertheless, it is worth noting that even when incorporated with the ground-truth knowledge (*i.e.,* the oracle setting), Gemini-1.5-Pro still performs much worse than human experts in close-book setting (*i.e.,* 92.0%). This highlights the need for future work on developing more advanced domain-specific knowledge integration methods.

6 Related Work

The development of general-purpose intelligent systems is significantly dependent on the foundational aspect of mathematical reasoning, a topic that has garnered considerable attention in the academic community. As illustrated in Table 1, researchers have proposed a wide spectrum of math reasoning datasets that cater to a variety of educational levels, ranging from elementary school to college (Koncel-Kedziorski et al., 2016; Wang et al., 2017; Amini et al., 2019; Miao et al., 2020; Patel et al., 2021; Cobbe et al., 2021; Hendrycks et al., 2021; Austin et al., 2021; Lu et al., 2023). However, these math reasoning benchmarks typically do not require specialized domain knowledge, a notable shortcoming when considering the practical applications of LLMs. Therefore, recent work has investigated the LLMs' capabilities in knowledge-intensive problem solving. For example, Chen et al. (2023c) collected a theorem-driven question-answering dataset, designed to evaluate AI models' ability to apply theorems in solving challenging science problems. MMMU (Yue et al., 2024) and MathVista (Lu et al., 2024) include examples that require complex multimodal reasoning in expert domains. Different from this recent work, which focuses on benchmarking LLM performance, our work also constructs a finance-domain knowledge bank, investigating various knowledge integration strategies to enhance knowledge-intensive problem solving. Moreover, FinanceMATH also requires LLMs to understand and interpret tabular data in expert domains to solve the problems.

7 Conclusion

This paper introduces FinanceMATH, a benchmark aimed at assessing LLMs in knowledge-intensive math reasoning. Our comprehensive evaluations of 51 LLMs, using both CoT and PoT prompting methods, identify significant areas where LLMs need to enhance their specialized knowledge for complex problem-solving in expert domains. Additionally, our knowledge augmentation analysis indicates that integrating domain-specific knowledge can improve LLMs' problem-solving abilities. We believe this research provides valuable insights into advancing LLMs within expert domains.

Limitations

In this work, we propose FinanceMATH and conduct comprehensive analysis of different LLMs' capabilities in solving knowledge-intensive math reasoning problems in finance domains. However, there are still some limitations: (1) Our method for extracting final answer from model output is still not perfect. In some cases, this methods fails to locate the answer, leading to the reported accuracy being an approximate lower bound. Moreover, as the extracted answer can be in a different format than the ground truth, we apply rule-based methods to measure the exact match between the two values, which could introduce around 2% errors based on our case studies. (2) In our experiment, we regard tables in the question as textual input. However, in real-world scenarios, tabular data might appear as images, where people cannot obtain its textual content directly. In these cases, OCR tools to extract table content (Du et al., 2020) or LLMs with vision capabilities (OpenAI, 2023b; Yue et al., 2024; Lu et al., 2024) may be required. (3) Due to computational resource constraints, we do not tune LLMs on a large-scale finance-domain data ourselves (Xie et al., 2023, 2024). However, we believe that training on finance data can help improve knowledge-intensive problem solving in finance domains.

Acknowledgement

We are grateful for the compute support provided by Microsoft Research's Accelerate Foundation Models Research (AFMR) program. We would also like to thank the anonymous reviewers and area chairs for constructive discussions and feedback. Hongjun Liu and Chen Zhao are supported by Shanghai Frontiers Science Center of Artificial Intelligence and Deep Learning, NYU Shanghai.

References

01.AI. 2023. Yi: Open-source llm release.

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Eric Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xia Song, Masahiro Tanaka, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone.

AI@Meta. 2024. The llama 3 herd of models.

AI@Mistral. 2024a. Codestral: Hello, world!

AI@Mistral. 2024b. Mathstral model card.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Anthropic. 2024. Introducing the next generation of claude.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Liang Chen, Yang Deng, Yatao Bian, Zeyu Qin, Bingzhe Wu, Tat-Seng Chua, and Kam-Fai Wong. 2023a. Beyond factuality: A comprehensive evaluation of large language models as knowledge generators. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Wenhu Chen. 2023. Large language models are few(1)-shot table reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130, Dubrovnik, Croatia. Association for Computational Linguistics.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023b. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023c. TheoremQA: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7889–7901, Singapore. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Cohere. 2024a. Cohere for ai launches aya 23, 8 and 35 billion parameter open weights release.

Cohere. 2024b. Introducing command r+: A scalable llm built for business.

Databricks. 2024. Introducing dbrx: A new state-of-the-art open llm.

DeepSeek-AI. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2024a. Investigating data contamination in modern benchmarks for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8706–8719, Mexico City, Mexico. Association for Computational Linguistics.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2024b. Investigating data contamination in modern benchmarks for large language models.

Yuning Du, Chenxia Li, Ruoyu Guo, Xiaoting Yin, Weiwei Liu, Jun Zhou, Yifan Bai, Zilin Yu, Yehua Yang, Qingqing Dang, et al. 2020. Pp-ocr: A practical ultra lightweight ocr system. *arXiv preprint arXiv:2009.09941*.

Gemini. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.

Team GLM, :, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence.

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.

Tatsuro Inaba, Hirokazu Kiyomaru, Fei Cheng, and Sadao Kurohashi. 2023. MultiTool-CoT: GPT-3 can use multiple external tools with chain of thought prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1522–1532, Toronto, Canada. Association for Computational Linguistics.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. 2023. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

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

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Chenying Li, Wenbo Ye, and Yilun Zhao. 2022a. FinMath: Injecting a tree-structured solver for question answering over financial reports. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6147–6152, Marseille, France. European Language Resources Association.

Moxin Li, Fuli Feng, Hanwang Zhang, Xiangnan He, Fengbin Zhu, and Tat-Seng Chua. 2022b. Learning to imagine: Integrating counterfactual thinking in neural discrete reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 57–69, Dublin, Ireland. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone, Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian McAuley, Han Hu, Torsten Scholak, Sebastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, Mostofa Patwary, Nima Tajbakhsh, Yacine Jernite, Carlos Muñoz Ferrandis, Lingming Zhang, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2024. Starcoder 2 and the stack v2: The next generation.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. Mathvista: Evaluating math reasoning in visual contexts with gpt-4v, bard, and other large multimodal models. In *The Twelfth International Conference on Learning Representations*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *The Eleventh International Conference on Learning Representations*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023a. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023b. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

Mistral.AI. 2023. Mixtral of experts: A high quality sparse mixture-of-experts.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue.

OpenAI. 2023a. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

OpenAI. 2023b. Gpt-4v(ision) system card.

OpenAI. 2024. Hello gpt-4o.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10776–10787, Singapore. Association for Computational Linguistics.

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.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2024. Detecting pretraining data from large language models. In *The Twelfth International Conference on Learning Representations*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, Singapore. Association for Computational Linguistics.

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. 2024. Gemma: Open models based on gemini research and technology.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *ArXiv*, abs/2303.17564.

Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, Yijing Xu, Haoqiang Kang, Ziyan Kuang, Chenhan Yuan, Kailai Yang, Zheheng Luo, Tianlin Zhang, Zhiwei Liu, Guojun Xiong, Zhiyang Deng, Yuechen Jiang, Zhiyuan Yao, Haohang Li, Yangyang Yu, Gang Hu, Jiajia Huang, Xiao-Yang Liu, Alejandro Lopez-Lira, Benyou Wang, Yanzhao Lai, Hao Wang, Min Peng, Sophia Ananiadou, and Jimin Huang. 2024. Finben: A holistic financial benchmark for large language models.

Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2023. PIXIU: A comprehensive benchmark, instruction dataset and large language model for finance. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, Yudong Wang, Zijian Wu, Shuaibin Li, Fengzhe Zhou, Hongwei Liu, Songyang Zhang, Wenwei Zhang, Hang Yan, Xipeng Qiu, Jiayu Wang, Kai Chen, and Dahua Lin. 2024. Internlm-math: Open math large language models toward verifiable reasoning.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. 2024. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland. Association for Computational Linguistics.

Yilun Zhao, Yitao Long, Hongjun Liu, Linyong Nan, Lyuhao Chen, Ryo Kamoi, Yixin Liu, Xiangru Tang, Rui Zhang, and Arman Cohan. 2023a. Docmath-eval: Evaluating numerical reasoning capabilities of llms in understanding long documents with tabular data.

Yilun Zhao, Zhenting Qi, Linyong Nan, Boyu Mi, Yixin Liu, Weijin Zou, Simeng Han, Ruizhe Chen, Xiangru Tang, Yumo Xu, Dragomir Radev, and Arman Cohan. 2023b. QTSumm: Query-focused summarization over tabular data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1157–1172, Singapore. Association for Computational Linguistics.

Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. 2023c. Investigating table-to-text generation capabilities of large language models in real-world information seeking scenarios. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 160–175, Singapore. Association for Computational Linguistics.

Yilun Zhao, Chen Zhao, Linyong Nan, Zhenting Qi, Wenlin Zhang, Xiangru Tang, Boyu Mi, and Dragomir Radev. 2023d. RobuT: A systematic study of table QA robustness against human-annotated adversarial perturbations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6064–6081, Toronto, Canada. Association for Computational Linguistics.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

A Appendix

Annotation Quality	**%S $\geq$ 4**
Question Fluency	98.0
Question Correctness	95.3
Knowledge Relevance	94.1
Textual Definition Fluency	93.0
Textual Definition Correctness	94.7
Math Formula Correctness	88.0
Final Answer Correctness	98.0
Python Solution Correctness	96.0
Variable Name Meaningfulness	87.7
Comment Comprehensiveness	83.8

Table 5: Human evaluation over 200 samples of FinanceMATH. Three internal evaluators were asked to rate the samples on a scale of 1 to 5 individually. We report percent of samples that have an average score $\geq$ 4 to indicate the annotation quality of FinanceMATH.

Program-of-Thought Prompt Used

[System Input]:

You are a financial expert, you are supposed to generate a Python program to answer the given question. The returned value of the program is supposed to be the answer. Here is an example of the Python program:

```
```python
def solution():
    # Define variables name and value
    ...

    # Do math calculation
    ...

    # return answer
    return answer
```
```

[User Input]:

Question: {question}

Generate a Python program to answer the given question. Continue your output:

```
```python
def solution():
    # Define variables name and value
```

Figure 6: Example of *zero*-shot PoT prompt used.

Chain-of-Thought Prompt

[System Input]:

You are a financial expert, you are supposed to answer the given question. You need to first think through the problem step by step, documenting each necessary step. Then you are required to conclude your response with the final answer in your last sentence as "Therefore, the answer is {final answer}". The final answer should be a numeric value.

[User Input]:

Question: {question}

Relevant Knowledge:
{knowledge}

Let's think step by step to answer the given question.

Figure 7: Example of *zero*-shot CoT prompt used for knowledge augmentation analysis in Section 5.

[validation-69]
What is the gross domestic product for country B in 2015, according to the provided data?

Account Item	($ trillions)
Consumption	20.0
Capital consumption allowance	1.9
Government spending	4.2
Imports	2.2
Gross private domestic investment	6.0
Exports	1.8

Model Output
(..abbreviate..)
Net Investment = Gross private domestic investment - Capital consumption allowance = 6.0 - 1.9 = 4.1 trillion

Net Exports = Exports - Imports = 1.8 - 2.2 = -0.4 trillion (note: negative value indicates a trade deficit)

We can plug in the values into the GDP formula: GDP = Consumption + Net Investment + Government Spending + Net Exports = 20.0 + 4.1 + 4.2 - 0.4 = 28.9 trillion

Error Explanation
Misinterpretation of the GDP components: The model incorrectly subtracts the capital consumption allowance to derive net investment, but GDP should use gross investment.
Additionally, the math computation in the last step is incorrect

Table 6: Example of knowledge misinterpretation error made by Llama-3-70B.
```

Model	Organization	Size	Notes	Source
GPT-4-Turbo	OpenAI	–		gpt-4o-2024-05-13
GPT-4o	OpenAI	–		gpt-4-turbo-2024-04-09
GPT-3.5-Turbo	OpenAI	–		gpt-3.5-turbo-0125
Claude-3.5-Sonnet	Anthropic	–		claude-3-5-sonnet-20240620
Claude-3-Opus	Anthropic	–		claude-3-opus-20240229
Claude-3-Sonnet	Anthropic	–		claude-3-sonnet-20240229
Claude-3-Haiku	Anthropic	–		claude-3-haiku-20240307
Gemini-1.5-Pro	Google	–		gemini-1.5-pro
Gemini-1.5-Flash	Google	–		gemini-1.5-flash
Qwen2	Alibaba	7 & 72B		Qwen/Qwen2-*B-Instruct
Llama-2	Meta	7 & 70B		meta-llama/Llama-2-*b-chat-hf
Llama-3	Meta	8 & 70B		meta-llama/Meta-Llama-3-*B-Instruct
Llama-3.1	Meta	8 & 70B & 405B		meta-llama/Meta-Llama-3.1-*B-Instruct
Gemma-1	Google	2 & 7B		google/gemma-b-it
Gemma-2	Google	9B		google/gemma-2-9b-it
Mistral-v0.3	Mixtral AI	7B		mistralai/Mistral-7B-Instruct-v0.3
Mistral-Nemo	Mixtral AI	12B		mistralai/Mistral-Nemo-Instruct-2407
Mistral-Large	Mixtral AI	123B		mistralai/Mistral-Large-Instruct-2407
Mathstral	Mixtral AI	7B	Math-Specific	mistralai/Mathstral-7B-v0.1
Mixtral	Mixtral AI	46 & 141B	MoE	mistralai/Mixtral-Instruct-v0.1
Codestral	Mixtral AI	22B	Code-Specific	mistralai/Codestral-22B-v0.1
DeepSeek-Math	DeepSeek	7B	Math-Specific	deepseek-ai/deepseek-math-7b-instruct
DeepSeek-Coder-V1	DeepSeek	33B	Code-Specific	deepseek-ai/deepseek-coder-33b-instruct
DeepSeek-V2	DeepSeek	16 & 236B	MoE	deepseek-ai/DeepSeek-V2-Lite-Chat. We use the official API provided by DeepSeek for deepseek-ai/DeepSeek-V2-Chat
DeepSeek-Coder-V2	DeepSeek	16 & 236B	MoE	deepseek-ai/DeepSeek-Coder-V2-Lite-Instruct. We use the official API provided by DeepSeek for deepseek-ai/DeepSeek-Coder-V2-Instruct
Yi-1.5	01 AI	9 & 34B		01-ai/Yi-1.5-34B-Chat
Phi-3-Medium	Microsoft	14B		microsoft/Phi-3-medium-4k-instruct
Phi-3-Mini	Microsoft	3B		microsoft/Phi-3-mini-4k-instruct
GLM-4	THUDM	9B		THUDM/glm-4-9b-chat
DBRX	Databricks	132B	MoE	databricks/dbrx-instruct
C4AI Command R+	Cohere	104B		CohereForAI/c4ai-command-r-plus
InternLM2	InternLM	7B		internlm/internlm2-chat-7b
InternLM2-Math-Plus	InternLM	7B	Math-Specific	internlm/internlm2-math-plus-7b
WizardLM-2	WizardLM Team	7B		lucyknada/microsoft_WizardLM-2-7B
WizardMath	WizardLM Team	7B	Math-Specific	WizardLMTeam/WizardMath-7B-V1.1
WizardCoder	WizardLM Team	33B	Code-Specific	WizardLMTeam/WizardCoder-33B-V1.1
WizardLM-2 (MoE)	WizardLM Team	141B	MoE	alpindale/WizardLM-2-8x22B
Aya-23	Cohere	8 & 35B		CohereForAI/aya-23-*B
StarCoder2	BigCode	15B	Code-Specific	bigcode/starcoder2-15b-instruct-v0.1

Table 7: Details of the organization and model source (*i.e.,* model version for proprietary models, and Huggingface model name for open-source models) for the LLMs evaluated in FinanceMATH.

Model	Size	Notes	Accounting		Economics		Derivatives		Quantitative		Management		Portfolio		Corporate		Avg.	
			PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	**CoT**
Close-book																		
Expert																	73.0	
Non-Expert																	58.0	
Open-book																		
Expert																	92.0	
Non-Expert																	84.0	
Proprietary LLMs																		
GPT-4o			75.0	45.8	58.8	55.4	60.3	69.4	82.9	66.3	77.8	69.4	62.5	58.9	67.0	56.9	67.0	60.9
Claude-3.5-Sonnet			73.6	55.6	54.1	51.8	66.7	69.9	75.6	63.9	72.2	72.2	64.3	58.9	62.4	60.6	64.8	60.6
Claude-3-Opus			66.7	56.9	53.5	45.2	62.1	59.8	79.5	64.9	72.2	83.3	51.8	46.4	59.6	45.0	62.9	54.7
GPT-4-Turbo			59.7	38.9	49.8	42.2	50.7	64.8	72.2	56.6	61.1	50.0	57.1	44.6	50.5	47.7	56.2	50.9
Gemini-1.5-Pro			68.1	50.0	53.1	30.7	56.6	55.2	69.8	57.6	58.3	63.9	51.8	55.4	50.5	44.0	58.2	47.0
GPT-4o-Mini			65.3	36.1	46.9	29.7	48.4	47.5	69.3	46.8	50.0	38.9	57.1	41.1	51.4	45.9	54.3	40.3
Gemini-1.5-Flash			69.4	33.3	43.6	28.7	52.0	48.9	67.8	49.8	58.3	61.1	50.0	37.5	47.7	34.9	53.6	40.1
Claude-3-Sonnet			59.7	37.5	37.0	28.4	48.0	43.4	66.8	48.8	47.2	55.6	48.2	33.9	48.6	35.8	49.4	38.6
Claude-3-Haiku			34.7	31.9	19.8	26.4	33.8	43.4	44.9	41.5	41.7	44.4	25.0	33.9	36.7	30.3	32.0	35.1
GPT-3.5-Turbo			47.2	25.0	24.4	16.5	29.2	29.2	51.2	33.2	27.8	22.2	37.5	21.4	32.1	23.8	34.3	24.6
Open-source LLMs																		
DeepSeek-Coder-V2	236B	Code	38.5	41.0	62.5	66.7	47.6	46.0	75.0	61.1	33.3	44.4	79.0	52.6	60.0	50.0	55.5	51.0
DeepSeek-V2	236B	MoE	38.5	46.2	66.7	79.2	47.6	44.4	75.0	52.8	44.4	33.3	73.7	47.4	30.0	40.0	54.5	50.0
Llama-3.1	405B		41.0	51.3	54.2	54.2	46.0	33.3	80.6	47.2	33.3	11.1	68.4	31.6	40.0	50.0	53.5	41.5
Mistral-Large	123B		38.5	51.3	50.0	45.8	38.1	30.2	77.8	36.1	44.4	22.2	73.7	31.6	40.0	10.0	50.5	36.0
Llama-3.1	70B		25.6	38.5	41.7	50.0	33.3	19.0	66.7	47.2	55.6	22.2	68.4	47.4	30.0	30.0	43.0	35.0
Qwen2	72B		23.1	33.3	29.2	45.8	23.8	22.2	41.7	47.2	22.2	22.2	68.4	31.6	10.0	30.0	31.0	33.0
Llama-3	70B		33.3	38.5	37.5	45.8	33.3	22.2	66.7	33.3	33.3	22.2	68.4	31.6	30.0	30.0	43.0	31.5
Phi-3-Medium	14B		20.5	28.2	37.5	50.0	25.4	19.0	55.6	41.7	22.2	22.2	52.6	42.1	40.0	20.0	34.5	31.0
DeepSeek-Coder-V2-Lite	16B	Code	23.1	28.2	25.0	29.2	27.0	23.8	50.0	33.3	11.1	22.2	42.1	31.6	10.0	20.0	30.0	27.5
Mixtral-8x22B	141B	MoE	7.7	28.2	4.2	45.8	0.0	19.0	25.0	30.6	11.1	11.1	21.0	26.3	10.0	20.0	9.5	26.5
Yi-1.5	9B		10.3	30.8	20.8	25.0	11.1	14.3	36.1	30.6	33.3	22.2	26.3	26.3	10.0	20.0	19.0	23.5
Yi-1.5	34B		10.3	25.6	12.5	25.0	14.3	9.5	30.6	33.3	0.0	22.2	26.3	21.0	10.0	10.0	16.5	20.5
WizardLM-2	141B	MoE	15.4	28.2	41.7	29.2	17.5	11.1	52.8	19.4	0.0	11.1	47.4	15.8	10.0	40.0	28.0	20.0
Gemma-2	9B		23.1	20.5	25.0	29.2	20.6	11.1	36.1	33.3	11.1	0.0	42.1	26.3	10.0	10.0	25.5	20.0
GLM-4	9B		18.0	18.0	25.0	37.5	14.3	17.5	27.8	13.9	22.2	0.0	26.3	21.0	10.0	30.0	20.0	19.5
DBRX	132B	MoE	12.8	28.2	16.7	20.8	4.8	9.5	13.9	16.7	11.1	33.3	21.0	21.0	0.0	20.0	11.0	18.5
Phi-3-Mini	3B		18.0	23.1	25.0	25.0	11.1	7.9	30.6	16.7	0.0	22.2	31.6	15.8	20.0	20.0	19.5	16.5
DeepSeek-Math	7B	Math	0.0	18.0	0.0	20.8	0.0	12.7	2.8	13.9	0.0	11.1	0.0	31.6	0.0	10.0	0.5	16.5
Mathstral	7B	Math	18.0	23.1	20.8	25.0	11.1	11.1	27.8	19.4	11.1	11.1	36.8	15.8	0.0	0.0	18.5	16.5
Llama-3.1	8B		23.1	12.8	20.8	33.3	7.9	6.4	38.9	27.8	11.1	11.1	31.6	15.8	20.0	10.0	21.0	16.0
Qwen2	7B		7.7	28.2	8.3	12.5	6.4	12.7	8.3	16.7	0.0	11.1	10.5	15.8	0.0	0.0	7.0	16.0
Mistral-Nemo	12B		5.1	20.5	12.5	25.0	4.8	9.5	41.7	16.7	11.1	11.1	47.4	21.0	10.0	0.0	17.0	15.5
C4AI Command R+	104B		5.1	18.0	4.2	25.0	0.0	7.9	5.6	11.1	0.0	0.0	5.3	15.8	10.0	0.0	3.5	12.5
Codestral	22B	Code	18.0	7.7	25.0	20.8	17.5	9.5	50.0	16.7	22.2	0.0	42.1	15.8	10.0	0.0	26.5	11.5
Mixtral-8x7B-v0.1	46B	MoE	0.0	10.3	0.0	16.7	0.0	6.4	0.0	22.2	0.0	0.0	0.0	15.8	0.0	0.0	0.0	11.5
Llama-3	8B		18.0	10.3	20.8	0.0	11.1	7.9	25.0	13.9	11.1	11.1	26.3	21.0	0.0	0.0	17.0	9.5
WizardMath	7B	Math	7.7	10.3	12.5	16.7	4.8	7.9	13.9	5.6	0.0	11.1	15.8	10.5	0.0	10.0	8.5	9.5
InternLM2-Math-Plus	7B	Math	12.8	7.7	8.3	16.7	6.4	7.9	13.9	8.3	11.1	0.0	21.0	10.5	0.0	20.0	10.5	9.5
DeepSeek-V2-Lite	16B	MoE	7.7	10.3	4.2	8.3	1.6	9.5	2.8	5.6	0.0	0.0	10.5	10.5	0.0	10.0	4.0	8.5
WizardLM-2	7B		12.8	5.1	16.7	12.5	6.4	6.4	25.0	11.1	0.0	0.0	21.0	15.8	0.0	0.0	13.0	8.0
Llama-2	70B		15.4	7.7	16.7	8.3	4.8	4.8	11.1	5.6	0.0	0.0	15.8	21.0	10.0	0.0	10.5	7.0
Aya-23	35B		0.0	5.1	0.0	12.5	0.0	6.4	0.0	5.6	0.0	0.0	0.0	15.8	0.0	0.0	0.0	7.0
Mistral-v0.3	7B		0.0	12.8	0.0	8.3	4.8	3.2	2.8	5.6	0.0	0.0	5.3	10.5	0.0	0.0	2.5	6.5
StarCoder2	15B	Code	5.1	10.3	8.3	8.3	7.9	7.9	36.1	2.8	33.3	0.0	52.6	0.0	0.0	10.0	17.5	6.5
InternLM2	7B		7.7	7.7	16.7	4.2	3.2	7.9	11.1	2.8	0.0	0.0	26.3	5.3	0.0	0.0	9.0	5.5
DeepSeek-Coder-V1	33B	Code	2.6	10.3	8.3	4.2	3.2	3.2	13.9	8.3	0.0	0.0	21.0	0.0	0.0	0.0	7.0	5.0
WizardCoder	33B	Code	18.0	2.6	20.8	8.3	6.4	4.8	27.8	5.6	0.0	11.1	21.0	0.0	10.0	0.0	15.5	4.5
Aya-23	8B		0.0	10.3	0.0	8.3	0.0	3.2	0.0	0.0	0.0	0.0	0.0	5.3	0.0	0.0	0.0	4.5
Llama-2	7B		2.6	2.6	4.2	0.0	1.6	6.4	2.8	0.0	0.0	0.0	5.3	0.0	0.0	10.0	2.5	3.0
Gemma-1	2B		5.1	2.6	4.2	4.2	1.6	6.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	3.0
Gemma-1	7B		5.1	5.1	8.3	4.2	1.6	4.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5	3.0

Table 8: Results of Chain-of-Thought and Program-of-Thought prompting on the *development* set of FinanceMATH. We select the most recent version as of July 5, 2024, for each model. We use average Accuracy using CoT prompting as the ranking indicator of model performance. Numbers underscored indicate that models with PoT prompting achieves better results than with CoT prompting.

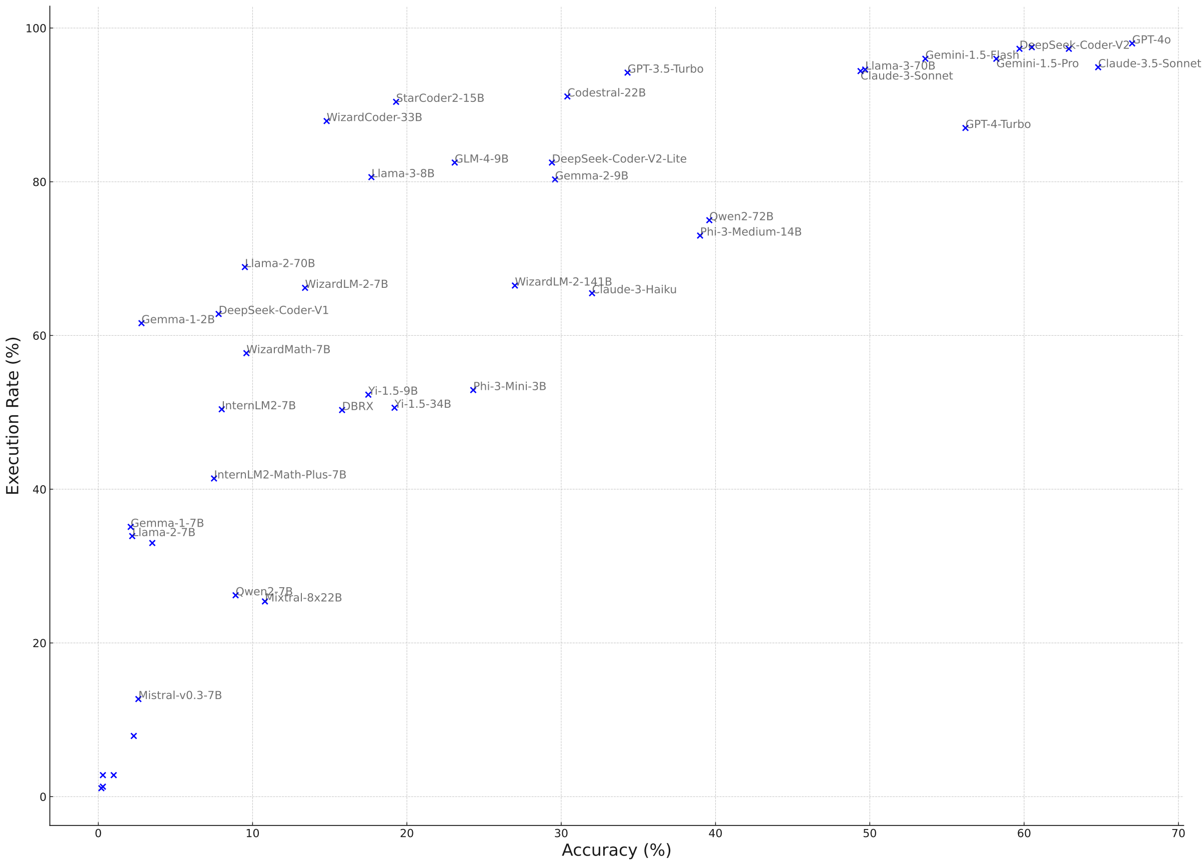

Figure 8: Relationship between execution rate and accuracy across different LLMs with PoT prompting on test set.