KnowledgeFMath: 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

Abstract

We introduce KnowledgeFMath, 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, KnowledgeFMath includes 1,259 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 26 LLMs with different prompting strategies like Chain-of-Thought and Program-of-Thought. Our experimental results reveal that the current best-performing system (i.e., GPT-4 with CoT prompting) achieves only 56.6% accuracy, leaving substantial room for improvement. Moreover, while augmenting LLMs with external knowledge can improve their performance (e.g., from 33.5% to 47.1% for GPT-3.5) , their accuracy remains significantly lower than the estimated human expert performance of 92%. We believe that KnowledgeFMath can advance future research in the area of domain-specific knowledge retrieval and integration, particularly within the context of solving math reasoning problems.

github.com/yale-nlp/KnowledgeFMath

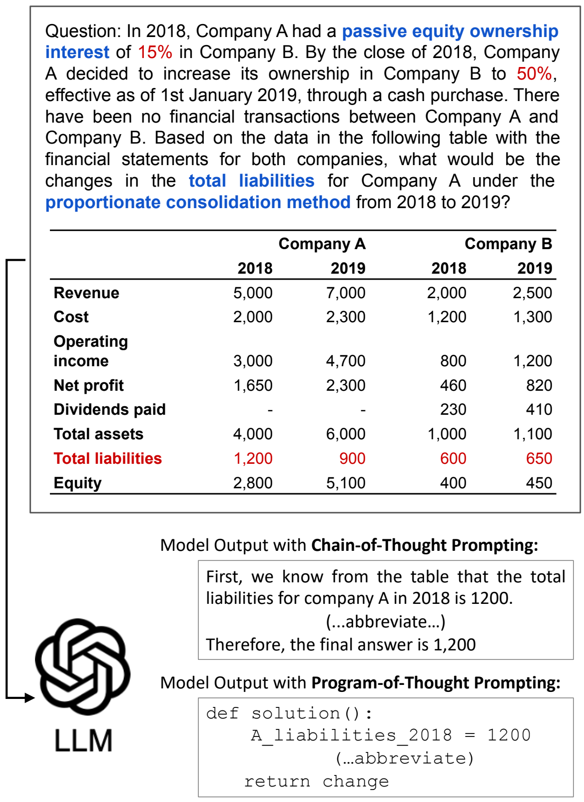

Figure 1: An example of KnowledgeFMath. To answer the given question, LLMs are required to comprehend specialized financial terms, such as "passive equity ownership interest" and "proportionate consolidation method". Additionally, they must interpret tabular data within the question and accurately identify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

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.

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 illus-

*Equal Contribution

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12841–12858
August 11-16, 2024

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
KnowledgeFMath (ours)	Finance	College	Internet+Expert	1,259	✓	✓	Python Program

Table 1: Comparison between KnowledgeFMath and existing **math reasoning** datasets. KnowledgeFMath 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*: 39.0% 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 KnowledgeFMath.

trated in Table 1, existing math reasoning benchmarks 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., 2023b; 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., 2022; Zhao et al., 2023b), which adds another layer of complexity to the knowledge-intensive problem-solving.

We introduce KnowledgeFMath, the first benchmark tailored for evaluating LLMs in the context of *Knowledge-intensive Math reasoning* in the *Finance domain*. The dataset contains 1,259 problems that cover a broad range of finance subareas (e.g., investment analysis, risk assessment, and financial forecasting), with 39.0% of the problems necessitating data interpretation over tabular data. Each problem is accompanied by detailed, expert-annotated solutions and explanations, 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 1,760 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, 26 model models from 14 organizations. Notably, this includes *math-specific* (Luo et al., 2023a), *code-based* (Xu et al., 2023; Luo et al., 2023b; Tunstall et al., 2023) LLMs, as well as *mixture of experts* (MoE) LLMs (Mistral.AI, 2023). Two prompting methods, Chain-of-Thought (CoT) (Wei et al., 2022) and Program-of-Thought (PoT) (Chen et al., 2023b), are adopted for experiments. Our experimental results indicate that all evaluated *open-source* LLMs scored below 24% in accuracy using various prompting methods, including CoT and PoT prompting. Proprietary models perform better, with GPT-4 significantly outperforming other LLMs, achieving an accuracy of 56.6% when applying CoT prompting. However, it still lags far behind human expert performance in the open-book setting, which stands at 92%. This significant gap between LLMs and human experts demonstrates the challenges of KnowledgeFMATH, highlighting 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 KnowledgeFMATH, 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-4) 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 KnowledgeFMATH

In this section, we describe the dataset construction process for KnowledgeFMATH. We begin by constructing a knowledge bank that includes well-formulated definitions of 1,760 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 1,760 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 process of 1) knowledge collection, 2) semi-automated knowledge formulation, and 3) knowledge bank update and maintenance in Appendix A.1. It is worth noting that this knowledge bank is versatile and can be applied to a variety of finance-relevant tasks for future research.

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.

2.2 KnowledgeFMATH 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., 2024), 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 1,259 questions, 674 are marked as having been adapted from existing resources, and 491 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 346 new inclusions and 83 revisions.

2.3 KnowledgeFMATH 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. Additionally, annotators are required to write detailed comments, making the code more readable and understandable. 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.

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 2: Human evaluation over 200 samples of KnowledgeFMATH. 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 ≥ 4 to indicate the annotation quality of KnowledgeFMATH

Property	Value
Knowledge Bank	
# Knowledge Terms	1,760
Textual Definition Length (Median/Avg)	64.0 / 69.3
% *w.* Mathematical Definition	62.8%
KnowledgeFMATH Dataset	
Question Length (Median/Avg)	49.0 / 55.7
% Questions with Table	39.0 %
# Rows per Table (Median/Avg)	3.0 / 3.2
# Columns per Table (Median/Avg)	6.0 / 6.9
# Knowledge Terms per Example (Median/Avg)	2.5 / 2.4
# Math Operations in Python Solution (Median/Avg)	5.0 / 5.3
# Code Lines in Python Solution (Median/Avg)	5.0 / 6.1
# Comment Lines in Python Solution (Median/Avg)	3.0 / 3.5
Validation Set Size	259
Test Set Size	1,000

Table 3: Basic statistics of the constructed knowledge bank and KnowledgeFMATH dataset.

2.4 Data Quality Validation

We conduct a comprehensive validation protocol to ensure the high quality of the annotated data. For each annotated question, 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 and inter-evaluator agreements over 200 sampled examples. As illustrated in Table 2, KnowledgeFMATH has a high annotation quality.

2.5 Data Statistics and Dataset Release

Table 3 describes the basic statistics of KnowledgeFMATH, with topic-type distribution shown in Figure 4 in Appendix. We randomly divide the dataset into two subsets: *validation* and *test*. The *validation* set contains 259 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., 2024), the answer for the *test* set will not be publicly released. Instead, we will 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., 2023; Lu et al., 2024), the main evaluation of KnowledgeFMATH 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 KnowledgeFMATH, 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

3.1 Large Language Models

We evaluate following LLMs on KnowledgeFMATH:

Chain-of-Thought Prompting Method:

```
[system prompt]
You are a financial expert, you are supposed to to answer the given question. You need to output the answer in your final sentence like 'Therefore, the answer is ...'. The answer should be a numeric value.

[user input]
Question: {question}

Table: {table}

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

Program-of-Thought Prompting Method:

```
[system prompt]
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.

[user input]
Question: {question}

Table: {table}

Please generate a Python program to answer the given question.
```python
def solution( ):
```

Figure 3: Examples of *zero*-shot CoT and PoT prompts.

- **General**: GPT-3.5&4 (OpenAI, 2022, 2023a), Gemini-Pro (Google, 2023), Llama-2 (Touvron et al., 2023), Mistral (Jiang et al., 2023), MPT (Team, 2023), Falcon (Almazrouei et al., 2023), WizardLM (Luo et al., 2023b), Yi (01.AI, 2023), Baichuan (Yang et al., 2023a), Phi-1.5 (Li et al., 2023), and DeepSeek (DeepSeek, 2023).
- **Math-specific**: WizardMath (Luo et al., 2023a).
- **Code-based**: CodeLlama (Rozière et al., 2023), WizardCoder (Luo et al., 2023b), and Lemur (Xu et al., 2023).
- **Mixture of Experts (MoE)**: Mixtral of experts (Mistral.AI, 2023).

By default, we use `chat` or `instruct` versions for each model, when available, otherwise, we used their `base` version. Additionally, we select the most recent, largest, and best-performing checkpoint available as of paper submission (i.e, December 10th, 2023). All the model weights of evaluated open-sourced LLMs can be found at HuggingFace Model Hub[1]. The implementation details (i.e., LLM parameter setting, tabular data serialization, and final answer extraction and evaluation) are discussed in Appendix B.1.

3.2 Prompting Methods

Following recent LLM reasoning benchmark works (Lu et al., 2024; Chen et al., 2023c), we

[1]https://huggingface.co/models
```

evaluate two established prompting methods, with examples of prompt illustrated in Figure 3.

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 Experimental Results

4.1 Main Results

Table 4 illustrates the performance of the evaluated LLMs using CoT and PoT prompting methods on the test set of KnowledgeFMATH. From this, we draw the following conclusions:

GPT-* significantly outperforms other open-source LLMs Proprietary models demonstrate the best performance on KnowledgeFMATH. Notably, as illustrated in Table 4, GPT-4 significantly outperforms other LLMs, achieving an accuracy of 56.6% on KnowledgeFMATH with CoT. In contrast, open-source LLMs significantly lag behind. Furthermore, our case study in Table 7 shows that while GPT-* models are capable of performing complex mathematical calculations, other LLMs often fail to understand financial terms, leading to incorrect mathematical expressions. This highlights a critical need for future development efforts to close the performance gap.

Substantial Discrepancy in Performance Between Human Experts and LLMs Even the best-performing LLM, GPT-4, performs much worse than human experts. For instance, the accuracy of GPT-4 using the CoT prompting method stands at 56.6%, 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 tackling complex problem-solving tasks within specialized domains that are knowledge-intensive.

Analysis of Open-sourced LLMs Among open-source LLMs with CoT prompting, Mixtral MoE achieves the best performance, demonstrating the effectiveness of applying a *mixture of experts* framework. Moreover, WizardMath also performs well as it is further instruction-tuned to learn mathematical reasoning. Moreover, among open-source LLMs with PoT prompting methods, Lemur achieves better performance than its backbone (i.e., Llama-2), demonstrating the effectiveness of tuning LLMs on code-based tasks for enhanced reasoning and coding capabilities.

4.2 Program-of-Thought Analysis – LLMs' Ability to Generate Executable Programs

We observe that the PoT prompting method consistently improves performance over the CoT method in GPT-* models and code-based LLMs. In contrast, the performance of several general LLMs, such as Mistral and WizardLM degrades with PoT prompting. To better analyze the reasons for these differing performance outcomes, we examine the execution rate of each LLM under PoT prompting, measuring how many of the generated Python programs are executable. Figure 6 illustrates the relationship between execution rate and accuracy across different models. It demonstrates that the degraded performance when applying PoT prompting is attributable to the low execution rate. For instance, although WizardLM achieves competitive performance with CoT, it struggles to consistently generate executable Python solutions, leading to lower accuracy with the PoT prompting approach.

4.3 Case Study and Error Analysis

In Table 4, we observe that GPT-* models significantly outperform other LLMs. Notably, the latest GPT-4 version achieves an accuracy of 56.6% using CoT prompting, closely approaching the non-expert human-level performance in the close-book setting (i.e., 58%). To gain a deeper insight into the capabilities and limitations of GPT-* on our dataset, we conducted a comprehensive error analysis and case studies. This was based on 100 randomly sampled examples from KnowledgeFMATH *development* set where `GPT-3.5-1106` exhibited failures. We identify four common mistakes that the current LLMs are likely to make (i.e., *misinterpretation of required knowledge*, *computation error*, *table misunderstanding*, and *question misunderstanding*). We provide detailed examples and explanations for each error type in Table 6 in Appendix. Moreover, we also present case study of model output from various LLMs with CoT prompting, as shown in

Model	Size	Notes	Quantitative		Derivatives		Accounting		Management		Portfolio		Economics		Corporate		Avg.	
			CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	**CoT**	PoT
Close-book																		
Non-Expert																	58.0	
Expert																	73.0	
Open-book																		
Non-Expert																	84.0	
Expert																	92.0	
GPT-4-1106-preview	–	–	66.8	66.9	50.8	54.1	54.2	34.4	40.5	51.4	66.7	66.7	57.7	50.4	65.2	58.7	56.6	53.1
GPT-4-0613	–	–	58.2	67.7	43.1	50.8	43.1	32.1	36.5	50.0	44.1	68.8	43.1	51.1	56.5	63.0	46.3	52.1
GPT-3.5-1106	–	–	42.6	56.4	26.2	27.7	32.1	19.5	28.4	28.4	32.3	45.2	35.0	32.8	28.3	32.6	32.4	33.9
GPT-3.5-0613	–	–	37.5	49.0	17.7	24.4	24.4	17.2	21.6	24.3	16.1	39.8	27.7	29.2	17.4	26.1	24.3	29.6
Mixtral	8x7B	MoE	30.9	15.2	18.7	10.5	23.7	6.1	14.9	13.5	25.8	29.0	24.1	11.7	30.4	8.7	23.5	12.2
Deepseek	67B	–	29.7	19.5	20.5	15.4	22.5	22.2	20.3	13.5	24.7	21.6	23.4	7.3	17.4	10.1	23.3	16.7
Gemini-Pro	–	–	31.6	29.6	15.4	16.7	24.0	17.6	12.2	21.6	21.5	35.5	26.3	19.7	17.4	17.4	22.0	21.5
WizardMath	70B	Math	21.1	2.3	15.1	0.9	22.5	1.4	16.2	0.0	21.5	0.0	17.5	5.3	15.2	2.6	18.1	1.7
WizardLM	70B	–	23.8	11.3	12.8	10.5	14.5	7.3	16.2	8.1	17.2	14.0	16.8	11.7	13.0	13.0	16.4	10.3
Lemur	70B	Code-based	19.9	16.8	12.8	6.8	17.6	8.7	17.6	17.8	12.9	15.1	16.1	11.4	21.7	10.4	16.2	11.3
Llama 2	70B	–	19.5	10.5	13.6	6.9	12.6	8.4	8.1	8.1	14.0	8.6	17.5	11.0	17.4	13.0	14.9	8.8
Falcon	180B	–	14.1	6.5	13.3	2.5	11.1	2.3	9.5	3.2	8.6	6.5	14.6	5.3	10.9	0.0	12.5	3.8
Llama 2	13B	–	8.2	7.0	8.5	3.3	11.5	5.0	16.2	6.8	8.6	8.6	11.7	7.3	10.9	4.4	9.9	5.5
Yi	34B	–	10.6	3.5	6.4	2.8	9.9	3.1	8.1	1.4	2.2	5.4	12.4	1.5	13.0	0.0	8.7	2.9
Llama 2	7B	–	7.8	2.0	6.9	1.8	8.0	1.5	8.1	0.0	5.4	3.2	11.7	0.7	6.5	2.2	7.8	1.7
WizardCoder-Py	34B	Code-based	8.2	4.2	5.6	0.3	9.5	0.5	4.1	1.6	8.6	3.9	10.2	1.8	4.4	0.0	7.6	1.6
MPT	30B	–	7.4	3.5	6.7	1.0	5.7	1.5	4.1	0.0	4.3	1.1	12.4	3.7	6.5	2.2	6.9	1.9
Baichuan2	13B	–	8.2	3.9	6.4	2.1	8.0	1.2	5.4	0.0	2.2	1.1	8.0	2.2	4.4	2.2	6.8	2.1
Mistral	7B	–	10.2	5.1	4.9	1.9	8.0	2.8	2.7	1.6	7.5	2.6	5.1	4.4	4.4	2.6	6.7	3.0
Vicuna	33B	–	5.5	5.5	5.4	4.1	5.3	6.5	4.1	4.1	2.2	8.6	5.1	8.0	8.7	4.4	5.2	5.6
Phi-2	2.7B	–	9.4	3.5	2.1	1.5	3	1.2	6.8	4.1	2.2	3.2	5.8	1.5	2.2	0.0	4.9	2.1
CodeLlama	34B	Code-based	6.3	5.6	5.1	1.9	2.3	2.8	2.7	3.2	2.2	3.9	8.0	0.9	2.2	2.6	4.6	3.0
Llama 1	65B	–	3.5	1.6	2.3	1.0	4.6	0.4	0.0	0.0	5.4	0.0	2.9	1.5	6.5	2.2	3.3	1.0
CodeLlama	7B	Code-based	3.9	1.9	2.8	2.8	1.9	1.4	1.4	0.1	0.0	1.1	5.8	2.6	4.4	6.5	2.9	2.2
Phi-1_5	1.3B	–	3	0.0	2.3	0.3	2.7	0.0	1.4	1.4	0.0	0.0	0.7	2.2	0.0	0.0	1.9	0.4
Llama 1	7B	–	1.2	0.4	2.1	1.0	1.9	0.0	0.0	0.0	1.1	0.0	3.7	0.0	0.0	0.0	1.8	0.4

Table 4: Results of Chain-of-Thought and Program-of-Thought prompting on the *test* set of KnowledgeFMATH. 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.

Table 7 in the Appendix.

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 knowledge augmentation in LLMs to solve 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 5 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. Therefore, 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 Ada Embedding[2] 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 knowledge definitions that are not entirely accurate (Chen et al., 2023a; Peng et al., 2023). To address this issue of unfaithfulness, we harness the power of external knowledge retrieval

[2] `https://platform.openai.com/docs/guides/embeddings`, we use the `text-embedding-ada-002` version.

Setting	Llama-2-70B	GPT-3.5$_{1106}$
wo. knowledge augmentation	14.3	33.5
LLM as Knowledge Base	13.9 (-0.4)	34.4 (+0.9)
BM25 ($n = 3$)		
Vanilla Retrieval	13.9 (-0.4)	35.1 (+1.6)
LLM as Retrieval Re-Ranker	16.2 (+1.9)	37.1 (+3.6)
LLM-instructed Retrieval	16.2 (+1.9)	40.5 (+7.0)
OpenAI Ada Embed. ($n = 3$)		
Vanilla Retrieval	14.7 (+0.4)	37.1 (+3.6)
LLM as Retrieval Re-Ranker	16.6 (+2.3)	39.7 (+6.2)
LLM-instructed Retrieval	17.0 (+2.7)	41.3 (+7.8)
OpenAI Ada Embed. ($n = 5$)		
Vanilla Retrieval	14.7 (+0.4)	36.9 (+3.0)
LLM as Retrieval Re-Ranker	17.8 (+3.5)	40.5 (+7.0)
LLM-instructed Retrieval	18.9 (+4.6)	41.3 (+7.8)
Oracle	25.1 (+10.8)	47.1 (+13.6)

Table 5: Results of CoT prompting approach under different knowledge augmentation settings on the *development* set of KnowledgeFMATH.

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 Experimental Results

As illustrated in Table 5, improving the question-relevance of incorporated knowledge can consistently improve the LLMs' performance. Specifically, LLMs equipped with retrieved knowledge from Ada Embedding consistently outperform those using retrieved knowledge from BM25. This is due to the more advanced capabilities of the Ada Embedding-based retriever. 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), GPT-3.5 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. Table 8 and Table 9 in Appendix C present a case study on effectiveness of various knowledge integration strategies.

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. 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. Contemporary to our work, MMMU (Yue et al., 2023) and MathVista (Lu et al., 2024) include examples that require complex visual reasoning in expert domains.

7 Conclusion

This paper introduces KnowledgeFMATH, aimed at assessing LLMs in knowledge-intensive math reasoning. Our comprehensive evaluations of 26 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 indicate that integrating domain-specific knowledge can improve LLMs' problem-solving abilities. We believe this research provides valuable insights for future work in advancing LLMs in complex problem-solving within expert domains.

Limitations

In this work, we propose KnowledgeFMATH 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 (Appendix B.1) is still not perfect. In some cases, this methods fails to locate the answer, leading to the reported accuracy being an approximate lower bound. (2) In our experiment, we regard tables in the question as textual input (Appendix B.1). 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., 2023; 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. However, we believe that training on finance data can help improve LLMs' capabilities in solving knowledge-intensive financial problems.

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.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Maitha Alhammadi, Mazzotta Daniele, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of language models: Towards open frontier models.

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.

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.

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*.

DeepSeek. 2023. Deepseek llm: Let there be answers. `https://github.com/deepseek-ai/DeepSeek-LLM`.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2024. 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*.

Google. 2023. Gemini.

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.

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.

Moxin Li, Fuli Feng, Hanwang Zhang, Xiangnan He, Fengbin Zhu, and Tat-Seng Chua. 2022. 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.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: **phi-1.5** technical report. *arXiv preprint arXiv:2309.05463*.

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.

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.

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.

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.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code.

Oscar Sainz, Jon Ander 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.

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.

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.

MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-03-28.

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.

Lewis Tunstall, Nathan Lambert, Nazneen Rajani, Edward Beeching, Teven Le Scao, Leandro von Werra, Sheon Han, Philipp Schmid, and Alexander Rush. 2023. Creating a coding assistant with starcoder. *Hugging Face Blog*. Https://huggingface.co/blog/starchat.

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, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2023. Pixiu: A large language model, instruction data and evaluation benchmark for finance. *ArXiv*, abs/2306.05443.

Yiheng Xu, Hongjin Su, Chen Xing, Boyu Mi, Qian Liu, Weijia Shi, Binyuan Hui, Fan Zhou, Yitao Liu, Tianbao Xie, Zhoujun Cheng, Siheng Zhao, Lingpeng Kong, Bailin Wang, Caiming Xiong, and Tao Yu. 2023. Lemur: Harmonizing natural language and code for language agents.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023a. Baichuan 2: Open large-scale language models.

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

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi.

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.

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 KnowledgeFMATH Dataset

A.1 Knowledge Bank Construction

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.

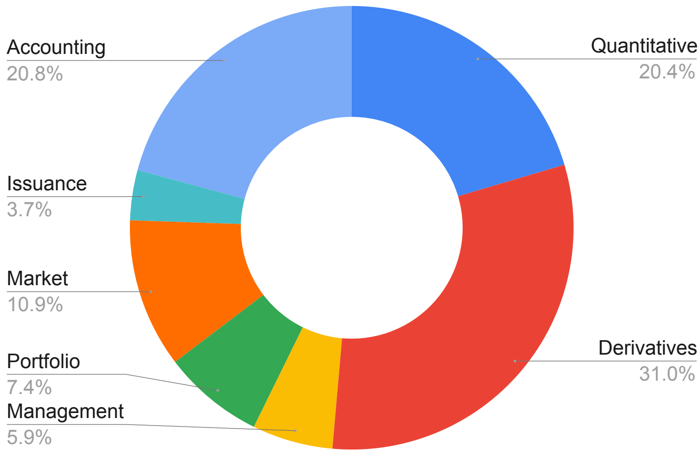

Figure 4: Topic distribution of KnowledgeFMATH.

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-* 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 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 1,760 pieces of financial knowledge in the knowledge bank, with 52% of the terms including Python-formatted mathematical definitions.

B Experiment Setup

B.1 Implementation Details

LLM Experiment 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. For questions involving tabular data, we converted the tables into Markdown format for model input.

Final Answer Extraction For LLM with CoT prompting, we adopt the answer extraction pipeline from 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-2 can effectively understand such table representations.

C Experimental Results and Analysis

Chain-of-Thought Prompting Method:

```
[system prompt]
You are a financial expert, you are supposed to to answer the given question. You need to output the answer in your final sentence like 'Therefore, the answer is ...'. The answer should be a numeric value.

[user input]
Relevant Knowledge:
{knowledge}

Question: {question}

Table: {table}

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

Program-of-Thought Prompting Method:

````
[system prompt]
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.

[user input]
Relevant Knowledge:
{knowledge}

Question: {question}

Table: {table}

Please generate a Python program to answer the given question.
```python
def solution( ):
````

Figure 5: Examples of knowledge-augmented CoT Prompts we used in Section 5.
````

Error Type (# Appear)	Representative Question	Representative Cot Result	Ground-truth Python-format Solution	Explanation

Misinterpretation of Required Knowledge (42/100)	What is the immediate ZAR/HKD exchange rate given by a broker?
	`Ratio      Spot rate`
	`—          —`
	`CNY/HKD    0.8422`
	`CNY/ZAR    0.9149`
	`CNY/SEK    1.0218`
		...The ZAR/CNY spot rate is 0.9149 and the HKD/CNY spot rate is 0.8422. Let's calculate: 0.9149 / 0.8422 = 1.0863**(wrong formula)** Therefore, the answer is 1.0863.	def solution():
			CNY_ZAR = 0.9149
			CNY_HKD = 0.8422
			ZAR_HKD = (1/CNY_ZAR) * CNY_HKD
			return ZAR_HKD
				For the given problem, the formula chosen for the solution is incorrect and not the correct formula for the corresponding financial knowledge.

Error in Complex Computation (24/100)	The Novartis Company noted a notable rise in its profitability, which led to a material increase in its credit score. Consequently, the market demanded a 100 basis point tighter spread to Gilts on Novartis's 8-year bond. In case the bond's altered duration is 6.0 and its convexity is 55.0, what would be the effect of this change on the return?	The change in spread here is -100 basis points (or -1.0 when we express this as a percent). We can plug this into the formula along with the given duration and convexity values: Change in Return = -6.0 (-1.0) + 0.5 * 55.0 * (-1.0)**2. Solving the above equation gives us an increase in return of 3 percent. Therefore, the answer is 3.**(wrong answer)**	def solution(): modified_duration = 6.0 delta_spread = -0.01 convexity = 55.0 return_impact = -(modified_duration * delta_spread) + 0.5 * convexity * (delta_spread**2) return return_impact	The calculation process is correct, but the final result given is incorrect.

Table Misunderstanding (12/100)	A specialist compiled the financial information for a Real Estate Investment Trust. Using the income statement, what was the Real Estate Investment Trust's share price at the end of 2017 according to the Price / FFO multiple?
	`Income Statement    End of 2017`
	`—                   —`
	`Total rental income        450`
	`Total property expenses    228`
	`Net operating income       222`
	`Other expenses              22`
	`General expenses            28`
	`Depreciation                45`
	`Net interest expenses       35`
	`Net income                  92`
	`Weighted average shares    118`
	`Earnings per share        0.78`
	`Price/FFO                  13x`
		...Given the net income and the adjustments we need to make, we can calculate FFO as follows: FFO = Net Income + Depreciation and Amortization + Other Expenses (Losses on Sale of Properties) FFO = 35 (Net Income) **(wrong number for Net Income)** + 45 (Depreciation) + 22 (Losses on sale of properties) = 102 US$ million...We were given Price/FFO multiple as 13x and the weighted average shares as 92 million**(wrong number for weighted average shares)**. Using this we can calculate the share price: Share Price = 13 * (102 / 92) Share Price = 13 * (1.1086) Share Price = 14.41 (approx.)**(wrong answer)**	def solution():
			multiple = 13
			net_income = 92
			depreciation_and_amortization = 45
			loss_from_property_disposal = 22
			shares_outstanding = 118
			FFO = net_income
			+ depreciation_and_amortization
			+ loss_from_property_disposal
			FFO_per_share = FFO / shares_outstanding
			stock_price = multiple * FFO_per_share
			return stock_price
				The inclusion of the table led to a deviation in the selection of the formula, resulting in the use of an incorrect formula during the calculation. Additionally, an incorrect number was chosen in one of the steps of the calculation due to an error in reading data from the table, causing the inserted number to be inconsistent with the requirements of the formula.

Question Misunderstanding (7/100)	Three months ago, a supervisor initiated a receive-fixed and pay-equity swap. The annualized fixed interest is 3% and the equity index was at 100 at the time of the swap. The expiry of the swap is one year with a quarterly reset, and the nominal amount is valued at $100 million. The present spot rates are as stated: At what equity index level would this equity swap value stand at zero?
	`Maturity Years    Value Factor`
	`——-               ——-`
	`0.25              0.997506`
	`0.5               0.992556`
	`0.75              0.985222`
	`1                 0.975610`
		Given the fixed interest rate is annualized at 3%, we can calculate the quarterly interest payment as (frac3%4 = 0.75%) per quarter. On a nominal amount of $100 million, the fixed payment per quarter would be (100,000,000 times 0.75% = $750,000)...we calculate the present value (PV) of the fixed side like this:(PV_fixed = $750,000 times (0.992556 + 0.985222 + 0.975610))(PV_fixed = $750,000 times 2.953388 = $2,215,041)...Now let's calculate the present value (PV) of the equity side: if the equity index grows to (E), then ((E - 100) / 100) would be the growth rate of the equity side of the swap.**(not correct answer to the question**	def solution():
			fixed_rate = 0.03
			nominal_amount = 100000000
			current_spot_rates = [0.997506, 0.992556, 0.985222]
			number_of_days = 90
			denominator = 360
			value_fixed_leg = fixed_rate
			* (number_of_days / denominator)
			* nominal_amount
			* sum(current_spot_rates)
			+ (nominal_amount
			* current_spot_rates[-1])
			equity_index_price = value_fixed_leg / nominal_amount
			return equity_index_price
				Failed to correctly information from the table, therefore misunderstanding the question that needed to be answered, leading to the selection of a formula that calculates another financial value, not the parameter asked in the question.

Other case (15/100)				

Table 6: Case study on `GPT-3.5-turbo-1106`'s failure cases.

GPT-4
Output: Firstly, under the proportionate consolidation method, we consider the portion of assets, liabilities, revenue, and expenses we proportionately own in relation to the company we invested in. In the case of PZ, they increased their stake in T-Internet from 15% to 50% in 2019. Therefore, for 2019, the proportionate consolidation of liabilities for company PZ would be its own liabilities plus 50% of T-Internet's liabilities. From the figures, we have PZ's total liabilities for 2019 as 900 and T-Internet's total liabilities for 2019 as 650. Calculating 50% of T-Internet's liabilities: 0.50 * 650 = 325 **(right formula)**. To get the total liabilities under the proportionate consolidation method, we add PZ's own liabilities to its 50% share of T-Internet's liabilities.Doing this, we have: 900 (PZ's liabilities) + 325 (50% of T-Internet's liabilities) = 1225 **(right formula)**. Therefore, the answer is 1225 **(right answer)**.
GPT-3.5
Output: To determine the total liabilities for company PZ under the proportionate consolidation method for 2019, we need to use the formula: Total liabilities = PZ liabilities + (T-internet liabilities * ownership ratio) In 2018, PZ had a passive equity ownership interest of 15% in the T-internet company. So, for 2018, the ownership ratio is 15%. In 2019, PZ increased its ownership to 50% as of 1st January 2019. So, for 2019, the ownership ratio is 50%. From the given table, we can find the liabilities for PZ in 2018 and 2019: PZ liabilities in 2018 = 1200 PZ liabilities in 2019 = 900 We also have the liabilities for the T-internet company in 2018 and 2019: T-internet liabilities in 2018 = 600 T-internet liabilities in 2019 = 650 Using the formula and the given values, we can calculate the total liabilities for company PZ under the proportionate consolidation method for 2019: Total liabilities = 900 + (650 * 50%) **(right formula)** Therefore, the answer is 1225 **(right answer)**.
Llama-2-70B
Output: 1. First, we need to calculate the total liabilities for company PZ in 2019 under the proportionate consolidation method. We can do this by using the formula: Total liabilities = PZ liabilities + (T-internet liabilities * ownership ratio) We know that PZ liabilities in 2019 were 1,200,000 (from the table). We also know that T-internet liabilities in 2019 were 900,000 (from the table). The ownership ratio is 50%, which means that PZ owns half of T-internet's shares. So, we can plug these values into the formula: Total liabilities = 1,200,000 + (900,000 * 0.5) Total liabilities = 1,200,000 + 450,000 Total liabilities = 1,650,000 **(wrong formula)** 2. Next, we need to add the dividends paid by T-internet in 2019, which were 230,000 (from the table). We add this to the total liabilities we calculated above: Total liabilities = 1,650,000 + 230,000 Total liabilities = 1,880,000 3. Finally, we need to convert the answer to the proportionate consolidation method, which means we need to subtract the portion of T-internet's liabilities that are not owned by PZ. We can do this by multiplying T-internet's liabilities by the ownership ratio and subtracting the result from the total liabilities: Total liabilities = 1,880,000 - (900,000 * 0.5) Total liabilities = 1,880,000 - 450,000 Total liabilities = 1,430,000 **(wrong formula)**
WizardMath
Output: Step 1: Determine the cost of acquiring the 35% stake in T-internet. We need to calculate the cost of purchasing the additional 35% (50% - 15%) equity stake in T-internet. We can do this by multiplying the total book value of T-internet's equity by the percentage of the stake acquired. Cost of acquiring 35% stake = (35/100) * Equity value of T-internet = (35/100) * $450,000,000 =$157,500,000 **(wrong formula)** Step 2: Calculate the total consolidated liabilities for 2019. Since there have been no transactions between the two companies, the consolidated liabilities for both companies will be the sum of their individual liabilities. Total consolidated liabilities (2019) = PZ's liabilities (2019) + T-internet's liabilities (2019) Step 3: Find the total liabilities of PZ and T-internet for 2019. We can calculate this by adding the liabilities of both companies as reported in their individual financial statements for 2019. PZ's total liabilities for 2019 amount to $900,000, and T-internet's total liabilities for 2019 amount to $650,000. Total consolidated liabilities (2019) = PZ's liabilities + T-internet's liabilities = $900,000 + $650,000 = $1,550,000 **(wrong formula)** Therefore, the total liabilities for company PZ under the proportionate consolidation method for 2019 is $1,550,000. The answer is: the total liabilities for company PZ under the proportionate consolidation method for 2019 is $1,550,000 **(wrong answer)**.
Mistral-7B
Output: To determine the total liabilities for company PZ under the proportionate consolidation method for 2019, we need to calculate 50% of the total liabilities of the T-internet company as of January 1, 2019. We can find this information in the table provided. Therefore, the answer is: 450 (which is 50% of the total liabilities of the T-internet company, as of January 1, 2019) **(wrong answer)**.

Table 7: Case study of different LLMs with CoT prompting methods.

Question:
"Assuming a 30% marginal tax rate and an additional 5% risk premium for shares as opposed to bonds, what is the cost of equity using the bond-yield-plus-risk-premium method for a 20-year, $1,000 non-callable bond with 8% annual coupons that is presently valued at $1,105.94?"

Ground Truth Knowledge:
Bond-yield-plus-risk-premium method
Definition:
"The bond-yield-plus-risk-premium method is a method used to estimate the cost of equity. It involves taking the yield on a company's debt, which represents the risk investors are willing to take for lending money to the company, and adding a risk premium that represents the extra risk of investing in the company's equity. This method is based on the assumption that equity holders require a higher return than bondholders due to the higher risk associated with equity investments."
Formula:
```
def Bond_yield_plus_risk_premium_method(risk_premium, yield_to_maturity):
    # Define constants
    risk_premium = risk_premium
    yield_to_maturity = yield_to_maturity
    # Calculate cost of equity
    cost_of_equity = yield_to_maturity + risk_premium
    return cost_of_equity
```
Yield to maturity:
Definition:
"Yield to maturity (YTM) is the total return anticipated on a bond if it is held until it matures. YTM is expressed as an annual percentage rate (APR). It takes into account both the interest payments you receive each year, usually semi-annually, and any capital gain you may receive if you hold the bond until maturity and it is redeemed at a price above or below its purchase price."
Formula:
```
def Yield_to_maturity(FV, P, t, C):
    '''
    FV: Face value of the bond
    P: Price of the bond
    t: Time to maturity
    C: Coupon payment
    '''
    import sympy as sp
    newline    r = sp.symbols('r')
    # Formula for bond price
    price_eq = C*(1 - (1 + r)**-t)/r + FV/(1 + r)**t - P
    # Equation solution
    yield_to_maturity = sp.solve(price_eq, r)[1]
    return yield_to_maturity
```
Risk premium:
Definition: "A risk premium refers to the expected additional return on an investment that an investor requires as compensation for investing in a risky asset, over and above the risk-free rate. It is essentially the reward for taking on a higher level of risk."
Formula:
```
def Risk_premium(bond_yield, risk_premium):
    # Calculate cost of equity
    cost_of_equity = bond_yield + risk_premium
    return cost_of_equity
```

Python Solution:
```
def solution():
    import numpy_financial as npf
    # Given values
    FV = -1000 # Future Value
    PV = 1105.94 # Present Value
    N = 20 # Number of periods
    PMT = -80 # Payment per period
    # Calculating the interest rate
    i = npf.rate(N, PMT, PV, FV)
    # Converting to percentage
    i_percent = i * 100
    # Define constants
    risk_premium = 5
    yield_to_maturity = i_percent
    # Calculate cost of equity
    cost_of_equity = yield_to_maturity + risk_premium
    return cost_of_equity
```

Table 8: Original question used in the case study of knowledge integration 9. We also present its ground-truth knowledge definitions and Python-format solution.

Wo. knowledge augmentation **Output:** "First, we need to calculate the yield to maturity (YTM) of the bond. The YTM is the rate of return anticipated on a bond if it is held until maturity and all payments are made as scheduled. To calculate the YTM, we need to use the present value of the bond, the annual coupons, and the number of years to maturity. Present value of the bond = $1,105.94 Annual coupons = 8% of $1,000 = $80 Number of years to maturity = 20 Using these values, we can calculate the YTM using financial formulas or by using an online calculator. Assuming the YTM is calculated to be 6%, we can now calculate the cost of equity using the bond-yield-plus-risk-premium method. Cost of equity = YTM + risk premium YTM = 6% **(Wrong calculated result for yield to maturity)** Risk premium for shares = 5% Cost of equity = 6% + 5% = 11% Therefore, the answer is 11%."
BM25 as retriever **Output:** "To calculate the cost of equity using the bond-yield-plus-risk-premium method, we need to add the yield on the bond (representing the risk investors are willing to take) to the risk premium for shares. First, we need to calculate the yield on the bond. The bond has an 8% annual coupon rate and is currently valued at $1,105.94. Using the coupon payments formula, we can calculate the total coupon payments over the lifespan of the bond:Coupon payments = Coupon_Payments(n=20, pmt=1000*0.08, i_y=(1105.94/1000)-1) Next, we need to calculate the yield to maturity on the bond. The yield to maturity can be obtained using the spot yield formula:Yield to maturity = Spot_yield(current_price=1105.94, previous_price=1000)Now we can calculate the yield on the bond by dividing the total coupon payments by the present value of the bond:Yield on bond = Coupon payments / Present value of bond. Finally, we can calculate the cost of equity by adding the yield on the bond to the risk premium for shares:Cost of equity = Yield on bond + risk premium. Therefore, the answer is the cost of equity calculated using the bond-yield-plus-risk-premium method." **(Not give the direct answer to the question)**
OpenAI Ada Embedding as retriever **Output:** "Step 1: Calculate the yield to maturity of the bondTo calculate the yield to maturity, we need to use the present value formula for a bond. Given that the present value is $1,105.94, the annual coupon is 8%, and the bond has 20 years until maturity, we can calculate the yield to maturity using financial software or a financial calculator. Let's assume the yield to maturity is 7%.Step 2: Calculate the risk premiumGiven that there is an additional 5% risk premium for shares as opposed to bonds, the risk premium is 5%.Step 3: Use the bond-yield-plus-risk-premium method formulaUsing the formula of the bond-yield-plus-risk-premium method, we can calculate the cost of equity as follows:cost_of_equity = yield_to_maturity + risk_premiumcost_of_equity = 7% + 5% cost_of_equity = 12% Therefore, the cost of equity using the bond-yield-plus-risk-premium method is 12%." **(Right answer)**
LLM as Knowledge Base **Output:** "To determine the cost of equity using the bond-yield-plus-risk-premium method, we need to find the yield on comparable bonds and add the risk premium. First, let's calculate the yield on the bond. The bond has a face value of $1,000 and is currently valued at $1,105.94. The coupon rate is 8% and the bond has a 20-year maturity. Using the present value formula, we can calculate the yield:$1,105.94 = ($80 / (1 + yield)1̂) + ($80 / (1 + yield)2̂) + ... + ($80 + $1,000 / (1 + yield)2̂0) Simplifying this equation, we can solve for the yield using a financial calculator or software:Yield = 6.5%**(Wrong calculated result for yield to maturity)** Now, let's calculate the cost of equity using the bond-yield-plus-risk-premium method. We add the risk premium of 5% to the yield we calculated:Cost of Equity = Yield + Risk Premium = 6.5% + 5% = 11.5% Therefore, the answer is 11.5%."

Table 9: Case study of knowledge integration.

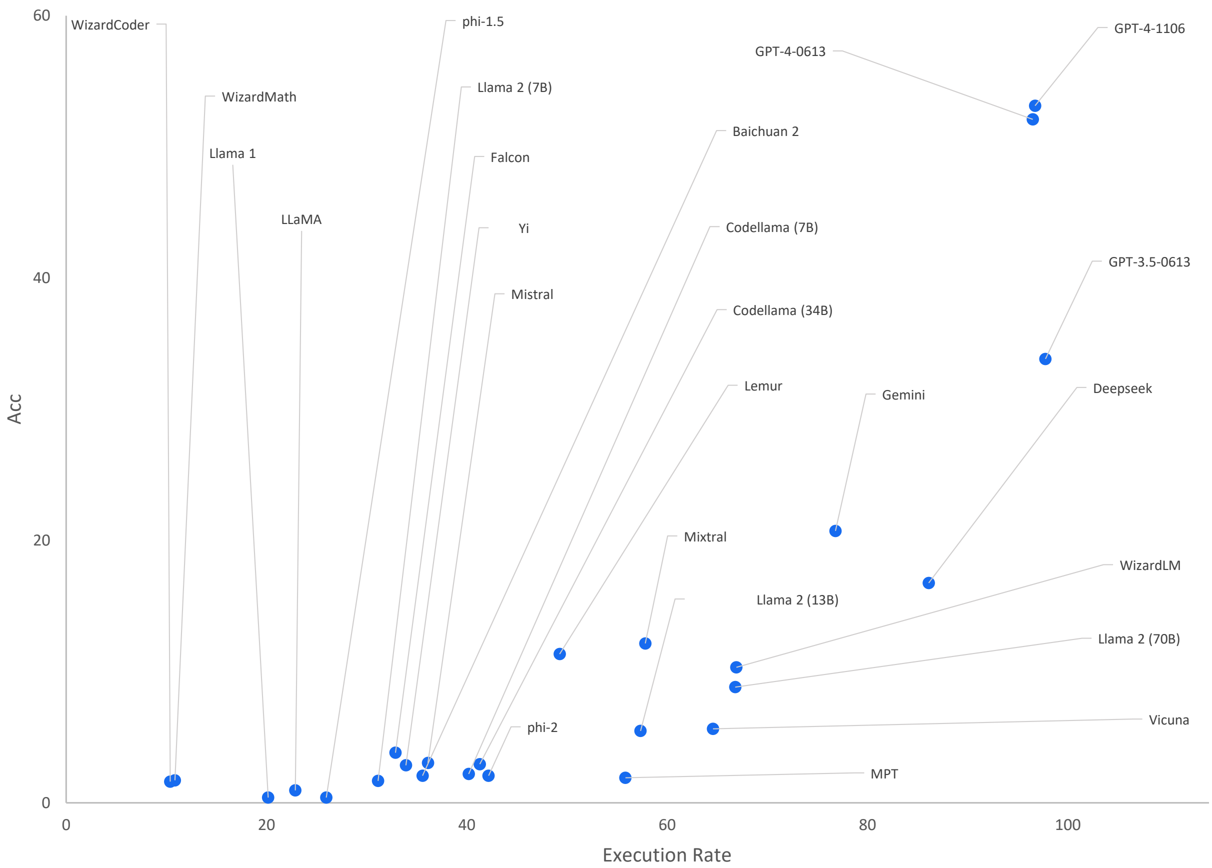

Figure 6: Relationship between execution rate and accuracy across different models. The degraded performance when applying PoT prompting is attributable to the low execution rate.