FinMathBench: A Formula-Driven Benchmark for Evaluating LLMs' Math Reasoning Capabilities in Finance

Yi He, Ping Wang*, Shiqiang Xiong, Chao Chen, Haixiang Hu

Ant Group

{hy403929, luojing.wp, shiqiang.xsq, chixi.cc, zengxian}@antgroup.com

Abstract

Many existing financial math reasoning benchmarks suffer from data contamination and high manual construction costs. To address this, we propose a novel formula-driven approach to dynamically construct math reasoning benchmarks in finance. Our two-stage approach: (1) generates single-formula questions by LLMs using a "Mask-for-Solve" paradigm for ground truth answers, and (2) synthesizes multi-formula questions through hierarchical tree-based DAGs. Our approach ensures novelty (via LLMs' creativity) and controllability of difficulty (via DAG structure). Based on a self-constructed financial formula bank, we utilize the proposed method to build FinMathBench, the first formula-driven and fully LLM-generated benchmark aimed at assessing LLMs' math reasoning abilities in finance, containing 946 questions across 4 complexity levels. Evaluation results on 40 LLMs demonstrate significant accuracy drops in multi-formula questions, e.g., 72.9% (1-Formula) to 14.0% (4-Formula) for GPT-4o under Chain-of-Thought prompting. Three critical flaws of LLMs are also observed: poor direct calculation performance, bias toward frequently solved variables in formulas, and erroneous "correction" of valid but extreme financial values. These findings highlight gaps in current LLMs' domain-specific reasoning and underscore FinMathBench's value for advancing robust financial LLMs.

Datasets — https://github.com/ant-research/FinMathBench

Introduction

Large Language Models (LLMs) have exhibited remarkable proficiency in a range of tasks (OpenAI 2023; AI@Meta 2024), including math word problems (MWPs) (Lu et al. 2023a; Chen et al. 2023c) which demand both understanding various contexts with numerical data and reasoning over complex logics. Although LLMs have achieved significant progress in solving fundamental math problems (Wei et al. 2022; Chen et al. 2023a; Luo et al. 2023), there is still a gap in the practicality of LLMs in solving MWPs in specific domains such as medicine and finance (Yang, Liu, and Wang 2023; Xie et al. 2023; Zhao et al. 2024a), since math reasoning in these fields requires not only basic mathematical skills, but also domain-specific knowledge. Considering that there inherently exist many math reasoning scenarios (e.g., financial analysis, risk assessment and investment decision-making) in finance, this paper also focuses on finance.

It can be found from Table 1 that many benchmarks are derived from the Internet, existing textbooks, or actual examination questions, which highlights a common challenge faced by most benchmarks: **data contamination**. Data contamination refers to the potential overlap between the content of benchmarks and the vast corpora used in the training of LLMs. This overlap raises the debate of "Generalization vs. Memorization" (Magar and Schwartz 2022; Biderman et al. 2023). To mitigate the impact of data contamination, some benchmarks are entirely written by human experts (Chen et al. 2021; Zhu et al. 2021; Zhao et al. 2024a). However, this leads to significant economic and time costs, particularly when the benchmarks are of substantial size.

To address the challenges of data contamination and costly artificial construction, we introduce a formula-driven question generation and synthesis approach. Recognizing that financial mathematics fundamentally revolves around domain-specific formulas, our approach is anchored on a self-constructed financial formula bank. Specifically, our approach includes two complementary components: (1) The first component leverages LLMs to dynamically create questions rooted in one financial formula, with the ground truth answers being obtained through the "Mask-for-Solve" paradigm (see Figure 1). By randomly instantiating variables within predefined ranges and embedding decoy content, our method generates diverse, contextually rich questions while maintaining mathematical rigor. (2) The second component uses hierarchical tree-based Directed Acyclic Graphs (DAGs) to orchestrate pre-generated single-formula questions, thereby formulating multi-step reasoning questions as well as enabling precise scalability in question complexity by varying the number of interlinked formulas. Together, our method guarantees both novelty (via LLMs' creativity) and controllability (via DAG structure).

Using the proposed approach, we construct **FinMathBench**, the first formula-driven and fully LLM-generated benchmark to evaluate financial math reasoning abilities of LLMs, containing 946 questions across 4 difficulty levels. While FinMathBench is implemented in Chinese (as a practical choice for our domain experts), the proposed approach itself is language-agnostic: the workflow could be adapted

*Corresponding author

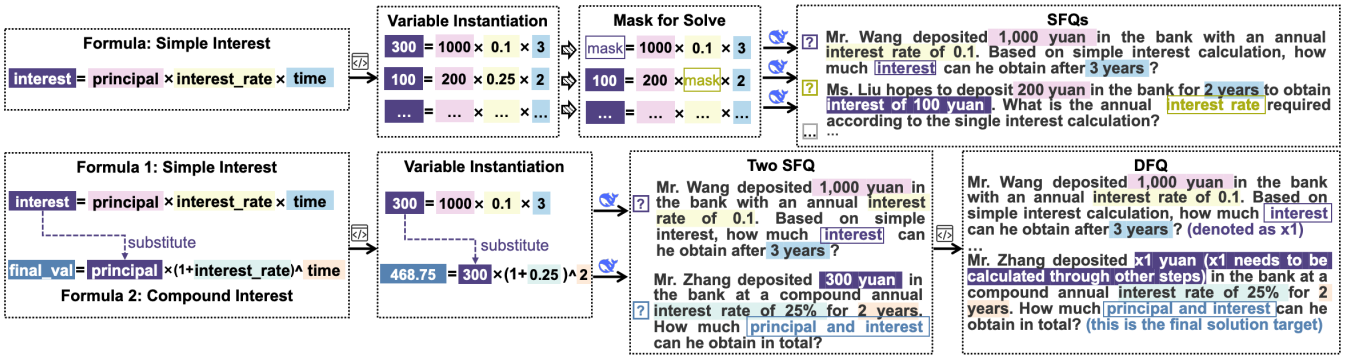

Figure 1: Illustration of the single-formula question generation and dual-formula question synthesis.

to other languages by replacing prompts with linguistically appropriate equivalents (e.g., English prompts for English benchmarks) and minor domain-specific adjustments.

We evaluate 40 LLMs, including finance-specific, math-specific, code-specific, mixture of experts, and general models. Two prompting methods, Chain-of-Thought (CoT) and Program-of-Thought (PoT), are adopted. The results demonstrate significant drops in accuracy as the number of formulas involved increases, e.g., GPT-4o drops from 72.9% (1-formula) to 14.0% (4-formula) under CoT. We reveal three critical observations: (1) most LLMs struggle with direct calculation, performing better in PoT than CoT prompting; (2) LLMs exhibit a bias towards frequently solved variables (vs. infrequent ones); (3) LLMs often mistakenly "correct" valid but extreme values to fit typical ranges, which would result in fatal errors during extreme market crises. These results highlight the challenges of FinMathBench, underscoring the need for further advancements of LLMs in finance.

Our main contributions are as follows:

- We propose a novel method that leverages LLMs to generate single-formula-based financial mathematical reasoning questions (with ground-truth answers) and a tree-based DAG algorithm for synthesizing questions involving multi-formula reasoning chains. Our method effectively avoids the issue of data contamination and allows fine-grained control over the difficulty of questions.
- We construct FinMathBench, the first formula-driven and fully LLM-generated benchmark for evaluating the math reasoning abilities of LLMs in finance. Additionally, we release the complementary self-constructed financial formula bank containing 148 cases.
- We conduct comprehensive evaluations on a wide range of 40 LLMs and identify three findings that could provide insights for further development of LLMs in finance.

Related Works

MWP Benchmark Table 1 lists some of the existing MWP benchmarks on general math (Cobbe et al. 2021; Lu et al. 2023b, 2024) and finance. In finance, FinQA (Chen et al. 2021) and TAT-QA (Zhu et al. 2021) develop text-and-table benchmarks based on publicly available earnings reports of listed companies. FinanceMATH (Zhao et al. 2024a) constructs knowledge-intensive questions based on a self-built financial term library. DocMath-Eval (Zhao et al. 2024b) is dedicated to math reasoning with very long contexts. Given the complexity of question construction, the above financial MWP benchmarks are all written by human experts.

Data Contamination As the training corpus of LLMs expands, existing benchmarks face the challenge of data contamination (Magar and Schwartz 2022; Biderman et al. 2023). To alleviate data contamination, some benchmarks are entirely written by human experts with strict post-checking (e.g., FinanceMATH requires that the first page of Google search should not have questions similar to the constructed ones). Although there have been dynamic benchmarks like DynaBench (Kiela et al. 2021) and DynaBoard (Ma et al. 2021) to eliminate data contamination, they rely on expensive and tedious crowd-sourcing to collect data. DyVal (Zhu et al. 2024) uses the structural advantages of directed acyclic graphs to dynamically generate questions. However, it is only suitable for basic reasoning tasks and struggles with domain-specific applications.

FinMathBench

Formula Collection

To systematically collect mathematical formulas in finance, we first define seven core financial categories: accounting, insurance, fund, futures, banking, securities, and others. For each category, we design a targeted prompt (see our Github) to prompt DeepSeek-R1 (DeepSeek-AI et al. 2025) in web-enhanced mode to retrieve relevant financial formulas. We then ask two experts to review and deduplicate the collected formulas as well as supplement any omitted formulas. This rigorous process results in a final collection of $N = 148$ distinct financial formulas, with each formula being assigned to the most relevant category, as shown in Table 2.

Formula Annotation

Let $\mathcal{F} = \{f_1, f_2, \ldots, f_N\}$ denote the collected formulas, where each formula $f_i \in \mathcal{F}$ is defined as a four-tuple:

$$f_i = (\mathcal{V}_i, \mathcal{S}_i, \mathcal{C}_i, \mathcal{Q}_i)\,. \tag{1}$$

Here, $\mathcal{V}_i = \{v_{i1}, v_{i2}, \ldots, v_{iK_i}\}$ represents the variables in f_i, where each $v_{ij} \in \mathcal{V}_i$ has the following attributes:

Dataset	Domain	Language	Source	# Examples	Table Reasoning?
Math23K (Wang, Liu, and Shi 2017)	Math	Chinese	Internet	23,162	No
GSM8K (Cobbe et al. 2021)	Math	English	CrowdSource	8,500	No
MATH (Hendrycks et al. 2021)	Math	English	Competition	12,500	No
AQuA (Ling et al. 2017)	Math	English	GMAT/GRE	100,000	No
MathQA (Amini et al. 2019)	Math	English	AQuA	100,000	No
MathQA-Python (Austin et al. 2021)	Math	English	AQuA	23,914	No
MathVista (Lu et al. 2024)	Math	English	Internet+Expert	6,141	Few
TabMWP (Lu et al. 2023b)	Math	English	Textbooks	38,431	Yes
TheoremQA (Chen et al. 2023c)	STEM	English	Internet+Expert	800	No
FinQA (Chen et al. 2021)	Finance	English	Expert	8,281	Yes
TAT-QA (Zhu et al. 2021)	Finance	English	Expert	16,552	Yes
MultiHiertt (Zhao et al. 2022)	Finance	English	Expert	10,440	Yes
DocMath-Eval (Zhao et al. 2024b)	Finance	English	Expert	5,974	Few
FinanceMATH (Zhao et al. 2024a)	Finance	English	Internet+Expert	1,200	Yes
FinMathBench (ours)	Finance	Chinese	Generated+Synthesized	946	Yes

Table 1: Comparison between FinMathBench and existing math reasoning benchmarks.

- $\texttt{unit}(v_{ij})$: variable unit, e.g., yuan / day / year,
- $\texttt{range}(v_{ij})$: value range, e.g., $(0, 1]$ / $[100, 100000]$,
- $\texttt{isInteger}(v_{ij}) \in \{\text{True, False}\}$: mark whether the value should be an integer,
- $\texttt{isList}(v_{ij}) \in \{\text{True, False}\}$: mark whether the variable is a list type or a common numeric type,
- $\texttt{isFreqCal}(v_{ij}) \in \{\text{True, False}\}$: mark whether v_{ij} is the most frequently solved variable in formula f_i. For example, in the simple interest formula (i.e., $\texttt{interest} = \texttt{principal} \times \texttt{interest rate} \times \texttt{time}$), we usually calculate the interest based on the principal, interest rate and time. Then, the interest is the most frequently solved variable. Note that there is exactly one element whose $\texttt{isFreqCal}$ is true in $\mathcal{V}_i$, denoted as $v_i^\star$.

$\mathcal{S}_i$ is an executable Groovy script expressing the arithmetic logic of f_i. We stipulate that the output of $\mathcal{S}_i$ must be $v_i^\star$, and $\mathcal{S}_i$ can be defined as

$$\mathcal{S}_i : \{v \in \mathcal{V}_i \mid \texttt{isFreqCal}(v) = \text{False}\} \rightarrow v_i^\star. \quad (2)$$

$\mathcal{C}_i = \{c_{i1}, c_{i2}, \ldots, c_{iM_i}\}$ is the constraint set that defines the relationships or restrictions among the variables in $\mathcal{V}_i$, where each $c \in \mathcal{C}_i$ is a logical condition on $\mathcal{V}_i$. These constraints ensure the correctness of the variable values in both mathematical and financial semantics. $\mathcal{Q}_i \subseteq \mathcal{V}_i$ is the set of solvable variables in f_i, where each $q \in \mathcal{Q}_i$ can be analytically determined given the values of other variables in $\mathcal{V}_i$. According to the definition above, there must exist $v_i^\star \in \mathcal{Q}_i$. Note that for $q \in \mathcal{Q}_i$ with $\texttt{isList}(q)$ being true, we mean solving for a random element of the list q (denoted as $q[idx]$), given the values of $\mathcal{V}_i \setminus \{q\}$ and $q \setminus \{q[idx]\}$. In this way, we ensure that the target variable to be solved must be a numerical value. For convenience of expression, in this paper, we uniformly use $\mathcal{V}_i \setminus \{q\}$ to represent the other variables in f_i except for the single numerical solution target $q \in \mathcal{Q}_i$, regardless of whether $\texttt{isList}(q)$ is true or false.

We ask experts to label the four-tuple defined in Eq. (1) for all formulas, in preparation for the subsequent generation and synthesis of formula-driven questions. The basic statistics of the constructed formula bank are listed in Table 2.

Question Generation And Synthesis

Questions in FinMathBench are divided into single-formula questions (SFQs) and multi-formula questions (MFQs) based on the number of formulas needed for their solution.

Single-Formula Question Generation By LLMs An SFQ is fundamentally defined by a triplet: 1) formula f_i, 2) solution target $q \in \mathcal{Q}_i$, and 3) specific values of $\mathcal{V}_i \setminus \{q\}$. The generation of the SFQ employs the following pipeline:

- **Variable Instantiation:** We first programmatically generate random values that fall into their $\texttt{range}$ for each variable in $\mathcal{V}_i \setminus \{v_i^\star\}$. Then, the script $\mathcal{S}_i$ is executed given the above values to obtain the specific value of $v_i^\star$. Finally, we programmatically check whether the value of $v_i^\star$ falls into its $\texttt{range}$ and whether all constraints $\mathcal{C}_i$ are satisfied. The above process is repeated until a set of variables satisfying all conditions is obtained.
- **Ground Truth Answer:** When generating the question, the value of the solution target q is masked and treated as the ground truth answer, while the remaining values are included as known conditions in the question, as shown in Figure 1. This *Mask-for-Solve* paradigm allows us to efficiently obtain answers for any target variable $q \in \mathcal{Q}_i$ using a single script $\mathcal{S}_i$.
- **Question Generation:** To enhance questions' diversity and complexity, we first leverage Qwen2.5-72B (Yang et al. 2024a) to generate 3$\sim$10 financial terms related to formula f_i, which will be merged into the question description as decoy information. Given these terms and the triplet mentioned above, we prompt (see our GitHub) DeepSeek-R1 to generate a natural language question. We require in the prompt that 1) q should be the final target to be solved, 2) values of $\mathcal{V}_i \setminus \{q\}$ should exist in the question as known conditions, and 3) to solve q, f_i must be used. Recognizing that many financial problems involve table data, we let DeepSeek-R1 include table elements in the question description with a probability of 50%. To ensure the correctness of the generated question as much as possible, we programmatically check whether

Financial Formula Bank			
# Accounting	44	# Fund	11
# Insurance	27	# Futures	20
# Banking	17	# Securities	8
# Others	21	# Total	148
Property			**Value**
# $\lvert\mathcal{V}_i\rvert$ (Median/Avg)			3.0 / 3.5
# $\lvert\mathcal{Q}_i\rvert$ (Median/Avg)			3.0 / 2.9
% Formulas with List-Type Variable			9 (6.08%)
# Types of Different Units			20
FinMathBench Dataset			
Property			**Value**
# 1-Formula Questions' Size			424
# 2-Formula Questions' Size			271
# 3-Formula Questions' Size			194
# 4-Formula Questions' Size			57
# Total Set Size			946
1-Formula Question Length (Median/Avg)			519.0 / 481.4
% 1-Formula Questions with Table			283 (66.7%)

Table 2: Statistics of the constructed financial formula bank and FinMathBench dataset.

Property	Value
Question Correctness	
# Total Generated	424
% Correctly Generated	402 (94.8%)
% Wrongly Generated	22 (5.2%)
Financial Professionalism Modification	
% Needs Modification	18 (4.5%)
% No Modification Needed	384 (95.5%)
Length of Questions w/ Mod. (Median/Avg)	493.0 / 467.0
Levenshtein Distance (Median/Avg)	6.0 / 13.5

Table 3: Statistics of SFQ quality annotation results.

the given values $\mathcal{V}_i \setminus \{q\}$ appear in the question exactly as they are. If not, we regenerate the question.

In this way, the resulting questions are novel and highly unlikely to have appeared in the training corpora of LLMs. For each f_i, we generate $|Q_i|$ questions by LLMs and ultimately obtain 424 independent SFQs, where the target to be solved in each question is different and therefore solving each question requires a different variant of f_i. It should be noted that we can generate more than $|\mathcal{Q}_i|$ SFQs for each f_i.

Multi-Formula Question Synthesis By Program An MFQ consists of multiple sub-SFQs, where the solution of one SFQ may serve as a necessary input to solve another, as shown in Figure 1. The final solution of the MFQ is obtained by nested substitution and application of multiple formulas (or their variants). Considering that it is highly challenging for LLMs to directly generate a valid MFQ that incorporates the aforementioned complex nested logic while maintaining accurate financial and mathematical semantics, we propose Algorithm 1 to synthesize complex MFQs based on the 424 generated SFQs (after human annotation discussed in the next subsection). In the evaluation, we find that this novel synthesis method brings significant challenges to LLMs. We model MFQs using tree-based directed acyclic graphs (DAGs), which inherently exhibit the hierarchical structure of the multistep reasoning nature of MFQs. Specifically, each node in the DAG corresponds to an SFQ, while the links between nodes represent variable substitution and formula nesting relationships between two SFQs. In FinMathBench, we synthesize MFQs containing 2/3/4 formulas. Note that we can include more formulas in Algorithm 1.

Algorithm 1: K-Formula Question Synthesis using DAG

Input: 424 SFQs, maximum number of questions N_{max}.
Output: N_{max} K-FQs with answers.

Set $cnt = 0$.
while $cnt < N_{\text{max}}$ **do**
  Initialize a tree-based DAG with K nodes. Select K different SFQ$_i$ (denote the triplet as $(f_i, q_i, \mathcal{V}_i \setminus \{q_i\})$), where $i = 1, 2, ..., K$. Assign each SFQ to a node.
  Concatenate "`denoted as` q_i" after the last question mark for every non-root SFQ$_i$, and "`this is the final solution target`" for the root.
  for SFQ$_i$ in topological order **do**
    Set $set = \mathcal{V}_i \setminus \{v_i^\star, q_i\}$.
    for SFQ$_i$'s child node SFQ$_j$ **do**
      Select a v_{ik} from set with the same unit as q_j. Replace v_{ik} with q_j when executing $\mathcal{S}_i$, and with the text "q_j`(`q_j `needs to be calculated through other steps)`" in SFQ$_i$.
      Set $set = set \setminus \{v_{ik}\}$.
    end for
    Run $\mathcal{S}_i$ with new variable values to obtain $v_{i,\text{new}}^\star$. If $v_i^\star$ occurs as a known condition in SFQ$_i$, replace it with $v_{i,\text{new}}^\star$.
    Check whether the new values of $\mathcal{V}_i$ still fall in their `range` and whether $\mathcal{C}_i$ is still satisfied.
    if any condition is not satisfied **then**
      Return to line 2.
    end if
  end for
  Concatenate above K SFQs with "`\n...\n`" to form a K-FQ, and the answer is the newly calculated q_i of the root.
  Set $cnt = cnt + 1$.
end while

Data Quality Validation

We conduct a rigorous quality annotation to ensure the correctness and financial professionalism of the 424 SFQs generated by LLMs. For each question, we first assign one expert to check the triplet that defines this question. Specifically, the expert should validate whether: 1) the solution target is clear and unambiguous, and is exactly the specified target $q \in \mathcal{Q}_i$, 2) all the given values of $\mathcal{V}_i \setminus \{q\}$ appear in the question and are correctly pointed to the corresponding variable names, 3) the solution of this question must and can only use the given formula f_i, 4) other statements in the question do not directly/indirectly affect the predetermined solution process. When any of the above is not satisfied, we mark this question as "wrongly generated" (otherwise, as "correctly generated"), indicating that the LLM makes some mistakes when generating this question.

For "wrongly generated" questions, the expert will correct the errors, while for "correctly generated" ones, the expert will modify the unprofessional parts of the question

Model	Size	Notes	1-Formula		2-Formula		3-Formula		4-Formula		Avg.	
			PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT
Proprietary LLMs												
Gemini-2.5-Pro			89.6	88.4	82.7	77.0	79.4	69.6	54.4	42.1	83.4	78.5
Claude-3.7-Sonnet			88.4	85.4	77.1	69.4	68.6	65.0	38.6	36.8	78.1	73.7
Claude-3.5-Sonnet			83.0	78.5	57.6	59.4	32.5	37.6	15.8	15.8	61.3	60.9
o3-mini			87.0	78.5	73.8	48.7	65.5	36.1	56.1	29.8	77.0	58.4
GPT-4o			81.8	72.9	57.6	51.3	41.2	46.9	19.3	14.0	62.8	57.8
o1-mini			85.9	73.6	62.0	48.3	51.6	18.0	26.3	14.0	68.4	51.4
Gemini-1.5-Pro			78.5	67.5	63.5	42.8	45.4	38.1	31.6	17.5	64.6	51.4
Gemini-2.5-Flash			86.8	57.8	76.4	37.3	63.9	22.2	35.1	10.5	76.0	41.8
Claude-3-Haiku			57.6	42.9	19.9	15.1	11.9	5.2	3.5	0.0	34.1	24.6
GPT-3.5-Turbo			45.3	23.8	1.85	1.8	0.5	1.0	1.8	0.0	21.0	11.4
Open-source LLMs												
Qwen3	235B	MoE	89.6	88.2	80.1	69.7	77.8	64.4	59.7	35.1	82.7	74.8
DeepSeek-V3	671B	MoE	87.7	83.5	67.2	70.1	55.2	56.2	43.9	33.3	72.5	71.0
DeepSeek-Prover-V2	671B	Math	80.7	85.6	59.8	60.2	37.1	42.8	33.3	24.6	62.9	65.9
DeepSeek-R1	671B	MoE	92.5	84.7	75.3	58.7	60.3	39.7	38.6	12.3	77.7	63.6
Qwen3	30B	MoE	89.2	79.5	67.5	54.2	57.7	42.3	36.8	17.5	73.4	60.9
Qwen3	32B		88.7	78.1	75.7	56.5	66.0	37.6	48.2	21.1	77.9	60.2
QwQ	32B		90.6	80.0	74.9	45.4	64.4	34.0	52.6	8.8	78.4	56.3
Qwen2.5	72B		78.1	57.6	47.6	37.3	36.1	22.4	14.0	12.3	56.9	41.9
Phi-4	14B		66.5	62.2	34.7	31.0	20.1	16.6	19.3	10.5	45.0	40.8
Qwen2.5	32B		79.5	58.5	39.1	33.6	22.2	10.8	12.3	1.8	52.1	38.2
Mistral-Small	24B		70.8	55.0	36.2	31.7	18.6	20.1	10.5	3.5	46.5	38.1
Llama-3.1	405B		71.2	58.3	47.2	21.0	31.4	9.3	17.5	3.5	53.0	34.3
Llama-3.1	70B		62.5	49.8	26.9	18.5	17.0	8.8	7.0	0.0	39.6	29.4
DeepSeek-Coder-V2	16B	Code	63.9	55.7	13.3	10.7	3.1	3.1	1.8	0.0	33.2	28.7
Qwen2.5-Coder	32B	Code	73.6	31.1	32.5	23.6	19.1	6.2	17.5	5.3	47.2	22.3
Llama-3	70B		51.7	34.2	20.7	7.8	8.3	5.2	1.8	0.0	30.9	18.6
XuanYuan 3	70B	Fin.	48.8	36.9	5.54	4.8	2.1	0.0	1.8	0.0	24.0	17.9
DeepSeek-Prover-V2	7B	Math	46.5	31.9	2.3	3.3	0.0	0.0	0.0	0.0	21.8	15.2
Qwen2.5	7B		53.1	26.7	15.9	6.3	4.6	3.1	0.0	0.0	29.3	14.2
Yi-1.5	34B		40.3	26.0	6.0	6.6	2.6	1.6	0.0	0.0	20.4	13.9
Mistral-Nemo	12B		45.3	18.6	3.0	3.3	3.6	1.0	1.8	0.0	21.8	9.5
Codestral	22B	Code	43.7	19.1	13.3	1.5	5.7	1.6	0.0	0.0	24.6	9.3
DeepSeek-Math	7B	Math	30.3	18.4	0.7	0.7	0.0	0.0	0.0	0.0	13.8	8.5
Mathstral	7B	Math	22.5	11.1	2.3	1.5	0.5	0.0	0.0	0.0	10.9	5.4
Qwen2.5-Math	72B	Math	11.1	13.2	0.0	0.0	0.0	0.0	0.0	0.0	5.0	5.4
Llama-3.1	8B		17.5	7.8	1.5	0.7	0.0	0.0	0.0	0.0	8.3	3.7
XuanYuan 2	70B	Fin.	18.0	5.4	0.7	0.7	0.0	0.0	0.0	0.0	8.3	2.6
DISC-FinLLM	13B	Fin.	1.4	2.1	0.0	0.0	0.0	0.0	0.0	0.0	0.6	1.0
FinGPT	8B	Fin.	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2
FinMA	7B	Fin.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 4: Results of CoT and PoT prompting on FinMathBench. We rank the models based on their accuracy under CoT prompting. Underscored numbers indicate that models using CoT prompting achieve better results compared to PoT prompting.

to make it more rigorous in finance. The Levenshtein distance (Konstantinidis 2005) is used to measure the degree of such modification. The shorter the distance, the fewer unprofessional points in the original question. Finally, out of the 424 SFQs generated by LLMs, 402 are marked as "correctly generated", among which only 18 questions need to be further modified to ensure professionalism in finance. More details of the annotation are listed in Table 3. We can see that, when provided with the structured formula information, it is highly feasible to generate high-quality SFQs using LLMs. Notably, the quality of SFQs can be further improved through iterative optimization of prompt engineering.

Data Statistics and Dataset Release

We run Algorithm 1 with $K = 2, 3, 4$. To enhance diversity, we additionally require that each SFQ appears at most three times in each K-FQs dataset. Finally, we obtain 271, 194 and 57 questions for $K = 2, 3, 4$, respectively, leading to a total of 946 questions in FinMathBench. Detailed statistics are listed in Table 2, and the dataset is available in Github.

Experiments

LLMs And Prompting Methods

We evaluate the following 40 LLMs on FinMathBench:

- **General:** GPT-3.5&4 (OpenAI 2023), o1&o3, Gemini-1.5&2.5 (Reid et al. 2024), Claude-3&3.5&3.7, Qwen-2.5&3 (Yang et al. 2024a), QwQ, Llama-3&3.1 (AI@Meta 2024), Mistral (Jiang et al. 2023), Phi-4 (Abdin et al. 2024), Yi-1.5 (Young et al. 2024).
- **Math-specific:** DeepSeek-Math (Shao et al. 2024), DeepSeek-Prover-V2 (Xin et al. 2025), Mathstral,

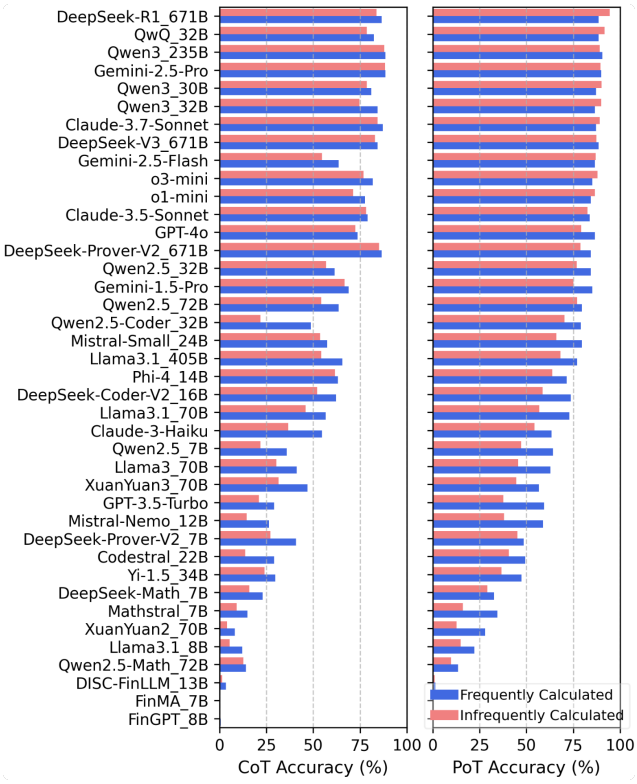

Figure 2: Accuracy comparison for solving frequently/ infrequently calculated variables on 1-FQs evaluation.

Qwen2.5-Math (Yang et al. 2024b).

- **Code-specific:** Qwen2.5-Coder (Hui et al. 2024), Codestral, DeepSeek-Coder-V2 (DeepSeek-AI et al. 2024b).
- **Finance-specific:** XuanYuan 2&3 (Zhang and Yang 2023), FinMA (Xie et al. 2023), FinGPT (Yang, Liu, and Wang 2023), DISC-FinLLM (Chen et al. 2023b).
- **Mixture of Experts (MoE):** DeepSeek-V3 (DeepSeek-AI et al. 2024a), DeepSeek-R1 (DeepSeek-AI et al. 2025), Qwen3 (Yang et al. 2025).

During the evaluation, we set temperature as 1.0, top_p as 0.95, and maximum output tokens as 16K. Evaluations of open-source LLMs are conducted on NVIDIA A100 GPUs with 80GB RAM.

Two prompts are used: (1) **Chain-of-Thought:** The CoT method (Wei et al. 2022) instructs LLMs to generate a step-by-step reasoning process, leading to the final answer. (2) **Program-of-Thought:** The PoT method (Chen et al. 2023a) instructs LLMs to generate a structured program to represent the reasoning process. The final answer is obtained through the external execution of the program rather than by direct calculation of LLMs.

Answer Extraction And Validation

For CoT prompting, we utilize regular expressions to extract potential numerical answers, including (1) numbers wrapped in tags such as `<num></num>`, (2) the last three numerical values that appear in the LLM's response. We find that this approach works very effectively across the evaluated LLMs to identify the intended answers provided by the models. For PoT prompting, we first extract Python snippets wrapped in ```` ```python``` ```` from the model's response. In cases where multiple snippets are extracted, we additionally concatenate them in sequence into one complete code and append it to the Python list. Finally, we execute all the obtained Python snippets to obtain a list of answers. For each value in the extracted answer list, we calculate its relative error against the ground truth. If the minimum relative error is below 0.1%, the model's response is considered correct.

Main Results

Table 4 shows the performance of the LLMs using CoT and PoT prompting on FinMathBench. The results demonstrate that our MFQ synthesis mechanism can effectively pose challenges to LLMs: the highest accuracy decreases from 88.4% (1-FQ, Gemini-2.5-Pro) to 42.1% (4-FQ, Gemini-2.5-Pro) under CoT prompting. The proprietary Gemini-2.5-Pro achieves the highest average accuracy of 78.5% under CoT prompting, while Qwen3-235B leads open source LLMs at 74.8%, with a small 3.7% gap. This near-parity highlights significant open-source advancements.

Across model types: (1) MoE LLMs such as Qwen3 and DeepSeek show relatively good performance and rank among the top in open source LLMs. (2) Code/Math-specific LLMs show limited performance due to their training for specific forms of tasks, leading to a sharp decline in other tasks (e.g., the general Qwen2.5-72B achieves 41.9% at CoT prompting, while Qwen2.5-Math-72B of the same size only achieves 5.4%). (3) Finance-specific LLMs underperform dramatically. Despite being trained in vast financial corpora (XuanYuan3) and financial computing (DISC-FinLLM), their accuracy is low: XuanYuan3 achieves only 17.9%, and DISC-FinLLM a mere 1.0%.

Observation 1: Most LLMs struggle with direct calculation The numerical values in FinMathBench are randomly generated rather than simple integers, which to some extent increases the difficulty of calculation. We identify two primary calculation error types: (1) rounding errors which can be potentially improved by requiring more decimal places in prompts, (2) fundamental arithmetic errors unrelated to rounding. The evaluation results show that only 4 out of the 40 evaluated LLMs achieve higher accuracy under CoT prompting compared to PoT, indicating a widespread weakness in direct calculation by LLMs. The average accuracy of 40 LLMs on CoT drops by 20.2% compared to PoT for 1-FQs and 47.2% for 4-FQs, as 4-FQs require more calculation. Additionally, for reasoning models like QwQ, splitting and repeatedly verifying calculation steps during thinking does not notably improve calculation quality. Sometimes, calculations remain incomplete before hitting token limits.

Observation 2: LLMs exhibit bias towards frequently solved variables, with lower accuracy on infrequent solution targets When dividing the 424 1-FQs in FinMathBench according to $\texttt{isFreqCal}(q)$ = true (148/424) or false (276/424), we find that all 40 LLMs show lower accuracy for infrequently solved variables under CoT prompting, while 32 models maintain this accuracy disparity un-

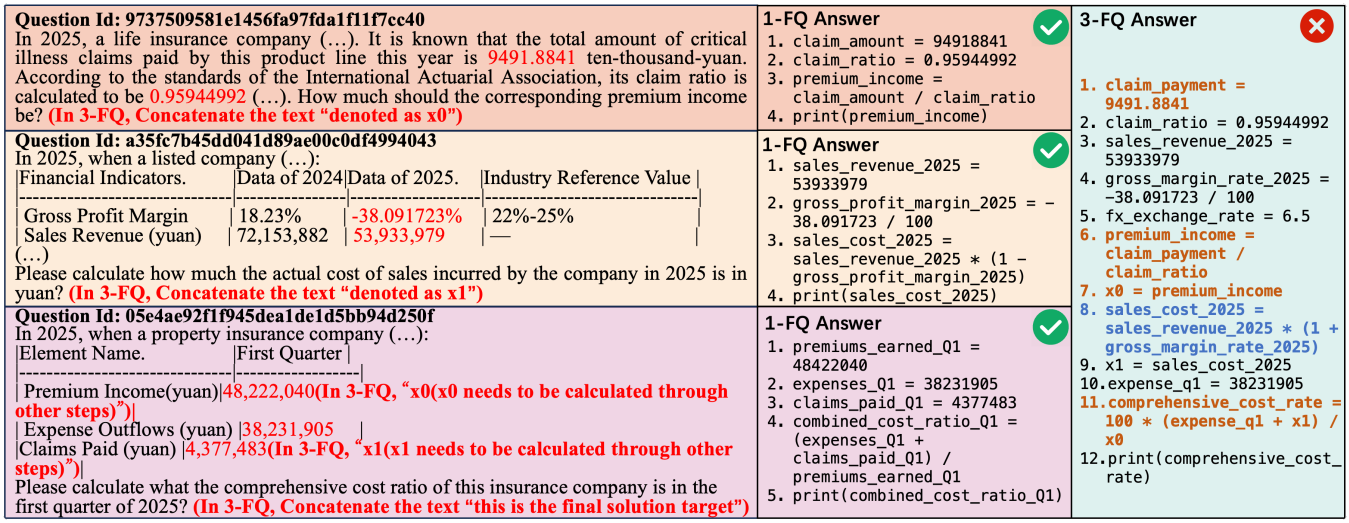

Figure 3: Example of errors made by GPT-4o under PoT prompting, where GPT-4o correctly answers three 1-FQs individually but fails when they are synthesized into a 3-FQ. The blue part (line 16) indicates the Formula Comprehension Error, and the orange parts (lines 3, 13-14, 21) indicate Unit Conversion Errors. For clarity and brevity, the questions shown are summarized in English. The full, original Chinese versions are available in our Github.

der PoT prompting, as shown in Figure 2. A general trend emerges: stronger LLMs exhibit smaller accuracy gaps mentioned above, while weaker models show larger gaps. We guess that this phenomenon may stem from imbalances in training data composition, where real-world financial corpora heavily emphasize routine calculations (e.g., computing loan payments) while largely excluding inverse operations (e.g., deriving loan terms from given payments).

Observation 3: LLMs mistakenly "correct" valid but extreme values to fit typical ranges When provided with financial values that are theoretically correct but hardly occur in reality, LLMs tend to "correct" these values to conventional ranges, leading to critical calculation errors. For instance, in one 1-FQ in FinMathBench featuring a fund subscription fee rate of 0.1089 (10.89%)—theoretically possible but highly atypical since the fee rate typically ranges between 0–5%, 11 of 34 (the remaining 6 models fail to even mention this necessary fee rate) models incorrectly modify it to 0.1089% (0.001089). Similarly, in another 1-FQ, a given stock turnover rate of 1.41 (141%) is incorrectly converted to 1.41% by 14 out of 36 models, as normal turnover rates rarely exceed 15%. Details of the above examples are in our Github. This behavior is observed in both state-of-the-art and weaker LLMs, which reveals a fundamental conflict: LLMs prioritize the statistical priors learned from training data over the fidelity to contextual data. This is particularly unacceptable in finance where extreme values *do occur* during extreme market crises.

Error Analysis

To gain a deeper insight into the challenges posed by our MFQ synthesis mechanism to LLMs, we conduct a focused analysis on cases where GPT-4o (under the PoT prompt) produces incorrect answers on the 3-FQs evaluation while the corresponding SFQs are all answered correctly on the 1-FQs evaluation. Out of the 194 3-FQs in FinMathBench, 58 cases exhibit above phenomenon. We classify the errors into four categories: **(1) Python Syntax Errors** (3/64): the generated Python snippets have some syntax errors and cannot be executed. **(2) Formula Comprehension Errors** (34/64): the model uses the wrong formulas (or variants) for solution, indicating that the model either fails to understand the financial semantics behind the question or does not master the corresponding mathematical knowledge. **(3) Data Retrieval Errors** (18/64): the model selects incorrect values in a complex context and substitutes them into the formula, leading to an incorrect final result. **(4) Unit Conversion Errors** (9/64): the model incorrectly adds values with different units directly or misinterprets 0.2% as 0.2, etc. It should be noted that a single incorrectly answered question may involve multiple types of error simultaneously. Therefore, the denominator in the above error distribution is 64 (greater than 58). Figure 3 shows an example of such errors, which indicates that our MFQ synthesis mechanism indeed poses significant challenges to LLMs.

Conclusion

This paper introduces FinMathBench, the first formula-driven and fully LLM-generated benchmark aimed at assessing LLMs in financial math reasoning. The proposed question generation and synthesis methods avoid data contamination and enable fine-grained difficulty control. Comprehensive evaluation of 40 LLMs reveals previously obscured flaws, such as systematic bias toward the frequently solved targets in formulas, and dangerous "correction" of valid but extreme numerical values, etc. We believe that this research provides valuable insights into advancing LLMs in finance.

References

Abdin, M. I.; Aneja, J.; Behl, H. S.; Bubeck, S.; Eldan, R.; Gunasekar, S.; Harrison, M.; Hewett, R. J.; Javaheripi, M.; Kauffmann, P.; Lee, J. R.; Lee, Y. T.; Li, Y.; Liu, W.; Mendes, C. C. T.; Nguyen, A.; Price, E.; de Rosa, G.; Saarikivi, O.; Salim, A.; Shah, S.; Wang, X.; Ward, R.; Wu, Y.; Yu, D.; Zhang, C.; and Zhang, Y. 2024. Phi-4 Technical Report. *CoRR*, abs/2412.08905.

AI@Meta. 2024. The Llama 3 Herd of Models. *CoRR*, abs/2407.21783.

Amini, A.; Gabriel, S.; Lin, S.; Koncel-Kedziorski, R.; Choi, Y.; and Hajishirzi, H. 2019. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 2357–2367. Association for Computational Linguistics.

Austin, J.; Odena, A.; Nye, M. I.; Bosma, M.; Michalewski, H.; Dohan, D.; Jiang, E.; Cai, C. J.; Terry, M.; Le, Q. V.; and Sutton, C. 2021. Program Synthesis with Large Language Models. *CoRR*, abs/2108.07732.

Biderman, S.; Prashanth, U. S.; Sutawika, L.; Schoelkopf, H.; Anthony, Q.; Purohit, S.; and Raff, E. 2023. Emergent and Predictable Memorization in Large Language Models. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Chen, B.; Yu, H.; Wang, X.; Miao, Y.; Yang, B.; and Zhou, J. 2021. FinQA: Answering Financial Questions through Program Synthesis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2711–2724.

Chen, W.; Ma, X.; Wang, X.; and Cohen, W. W. 2023a. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Trans. Mach. Learn. Res.*, 2023.

Chen, W.; Wang, Q.; Long, Z.; Zhang, X.; Lu, Z.; Li, B.; Wang, S.; Xu, J.; Bai, X.; Huang, X.; and Wei, Z. 2023b. DISC-FinLLM: A Chinese Financial Large Language Model based on Multiple Experts Fine-tuning. *CoRR*, abs/2310.15205.

Chen, W.; Yin, M.; Ku, M.; Lu, P.; Wan, Y.; Ma, X.; Xu, J.; Wang, X.; and Xia, T. 2023c. TheoremQA: A Theorem-driven Question Answering Dataset. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, 7889–7901. Association for Computational Linguistics.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *CoRR*, abs/2110.14168.

DeepSeek-AI; Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; Zhang, X.; Yu, X.; Wu, Y.; Wu, Z. F.; Gou, Z.; Shao, Z.; Li, Z.; Gao, Z.; Liu, A.; Xue, B.; Wang, B.; Wu, B.; Feng, B.; Lu, C.; et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *CoRR*, abs/2501.12948.

DeepSeek-AI; Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; Dai, D.; Guo, D.; Yang, D.; Chen, D.; Ji, D.; Li, E.; Lin, F.; et al. 2024a. DeepSeek-V3 Technical Report. *CoRR*, abs/2412.19437.

DeepSeek-AI; Zhu, Q.; Guo, D.; Shao, Z.; Yang, D.; Wang, P.; Xu, R.; Wu, Y.; Li, Y.; Gao, H.; Ma, S.; Zeng, W.; Bi, X.; Gu, Z.; Xu, H.; Dai, D.; Dong, K.; Zhang, L.; Piao, Y.; Gou, Z.; Xie, Z.; Hao, Z.; Wang, B.; Song, J.; Chen, D.; Xie, X.; Guan, K.; You, Y.; Liu, A.; Du, Q.; Gao, W.; Lu, X.; Chen, Q.; Wang, Y.; Deng, C.; Li, J.; Zhao, C.; Ruan, C.; Luo, F.; and Liang, W. 2024b. DeepSeek-Coder-V2: Breaking the Barrier of Closed-Source Models in Code Intelligence. arXiv:2406.11931.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. In Vanschoren, J.; and Yeung, S., eds., *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Hui, B.; Yang, J.; Cui, Z.; Yang, J.; Liu, D.; Zhang, L.; Liu, T.; Zhang, J.; Yu, B.; Lu, K.; Dang, K.; Fan, Y.; Zhang, Y.; Yang, A.; Men, R.; Huang, F.; Zheng, B.; Miao, Y.; Quan, S.; Feng, Y.; Ren, X.; Ren, X.; Zhou, J.; and Lin, J. 2024. Qwen2.5-Coder Technical Report. arXiv:2409.12186.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de Las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. *CoRR*, abs/2310.06825.

Kiela, D.; Bartolo, M.; Nie, Y.; Kaushik, D.; Geiger, A.; Wu, Z.; Vidgen, B.; Prasad, G.; Singh, A.; Ringshia, P.; Ma, Z.; Thrush, T.; Riedel, S.; Waseem, Z.; Stenetorp, P.; Jia, R.; Bansal, M.; Potts, C.; and Williams, A. 2021. Dynabench: Rethinking Benchmarking in NLP. In Toutanova, K.; Rumshisky, A.; Zettlemoyer, L.; Hakkani-Tür, D.; Beltagy, I.; Bethard, S.; Cotterell, R.; Chakraborty, T.; and Zhou, Y., eds., *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, 4110–4124. Association for Computational Linguistics.

Konstantinidis, S. 2005. Computing the Levenshtein distance of a regular language. In Dinneen, M. J.; Speidel, U.; and Taylor, D. P., eds., *Proceedings of the IEEE ITSOC Information Theory Workshop 2005 on Coding and Complexity, ITW 2005, Rotorua, New Zealand, August 29 - September 1, 2005*, 113–116. IEEE.

Ling, W.; Yogatama, D.; Dyer, C.; and Blunsom, P. 2017. Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems. In Barzilay, R.; and Kan, M., eds., *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, 158–167. Association for Computational Linguistics.

Lu, P.; Bansal, H.; Xia, T.; Liu, J.; Li, C.; Hajishirzi, H.; Cheng, H.; Chang, K.; Galley, M.; and Gao, J. 2024. MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Lu, P.; Qiu, L.; Chang, K.; Wu, Y. N.; Zhu, S.; Rajpurohit, T.; Clark, P.; and Kalyan, A. 2023a. Dynamic Prompt Learning via Policy Gradient for Semi-structured Mathematical Reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Lu, P.; Qiu, L.; Chang, K.; Wu, Y. N.; Zhu, S.; Rajpurohit, T.; Clark, P.; and Kalyan, A. 2023b. Dynamic Prompt Learning via Policy Gradient for Semi-structured Mathematical Reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Luo, H.; Sun, Q.; Xu, C.; Zhao, P.; Lou, J.; Tao, C.; Geng, X.; Lin, Q.; Chen, S.; and Zhang, D. 2023. WizardMath: Empowering Mathematical Reasoning for Large Language Models via Reinforced Evol-Instruct. *CoRR*, abs/2308.09583.

Ma, Z.; Ethayarajh, K.; Thrush, T.; Jain, S.; Wu, L.; Jia, R.; Potts, C.; Williams, A.; and Kiela, D. 2021. Dynaboard: An Evaluation-As-A-Service Platform for Holistic Next-Generation Benchmarking. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 10351–10367.

Magar, I.; and Schwartz, R. 2022. Data Contamination: From Memorization to Exploitation. In Muresan, S.; Nakov, P.; and Villavicencio, A., eds., *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, 157–165. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Reid, M.; Savinov, N.; Teplyashin, D.; Lepikhin, D.; Lillicrap, T. P.; Alayrac, J.; Soricut, R.; Lazaridou, A.; Firat, O.; Schrittwieser, J.; Antonoglou, I.; Anil, R.; Borgeaud, S.; Dai, A. M.; Millican, K.; Dyer, E.; Glaese, M.; Sottiaux, T.; Lee, B.; Viola, F.; Reynolds, M.; Xu, Y.; Molloy, J.; Chen, J.; Isard, M.; Barham, P.; Hennigan, T.; McIlroy, R.; Johnson, M.; Schalkwyk, J.; Collins, E.; Rutherford, E.; Moreira, E.; Ayoub, K.; Goel, M.; Meyer, C.; Thornton, G.; Yang, Z.; Michalewski, H.; Abbas, Z.; Schucher, N.; Anand, A.; Ives, R.; Keeling, J.; Lenc, K.; Haykal, S.; Shakeri, S.; Shyam, P.; Chowdhery, A.; Ring, R.; Spencer, S.; Sezener, E.; and et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*, abs/2403.05530.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *CoRR*, abs/2402.03300.

Wang, Y.; Liu, X.; and Shi, S. 2017. Deep Neural Solver for Math Word Problems. In Palmer, M.; Hwa, R.; and Riedel, S., eds., *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, 845–854. Association for Computational Linguistics.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E. H.; Le, Q. V.; and Zhou, D. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Xie, Q.; Han, W.; Zhang, X.; Lai, Y.; Peng, M.; Lopez-Lira, A.; and Huang, J. 2023. PIXIU: A Large Language Model, Instruction Data and Evaluation Benchmark for Finance. *CoRR*, abs/2306.05443.

Xin, H.; Ren, Z. Z.; Song, J.; Shao, Z.; Zhao, W.; Wang, H.; Liu, B.; Zhang, L.; Lu, X.; Du, Q.; Gao, W.; Zhang, H.; Zhu, Q.; Yang, D.; Gou, Z.; Wu, Z. F.; Luo, F.; and Ruan, C. 2025. DeepSeek-Prover-V1.5: Harnessing Proof Assistant Feedback for Reinforcement Learning and Monte-Carlo Tree Search. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; Zheng, C.; Liu, D.; Zhou, F.; Huang, F.; Hu, F.; Ge, H.; Wei, H.; Lin, H.; Tang, J.; Yang, J.; Tu, J.; Zhang, J.; Yang, J.; Yang, J.; Zhou, J.; Zhou, J.; Lin, J.; Dang, K.; Bao, K.; Yang, K.; Yu, L.; Deng, L.; Li, M.; Xue, M.; Li, M.; Zhang, P.; Wang, P.; Zhu, Q.; Men, R.; Gao, R.; Liu, S.; Luo, S.; Li, T.; Tang, T.; Yin, W.; Ren, X.; Wang, X.; Zhang, X.; Ren, X.; Fan, Y.; Su, Y.; Zhang, Y.; Zhang, Y.; Wan, Y.; Liu, Y.; Wang, Z.; Cui, Z.; Zhang, Z.; Zhou, Z.; and Qiu, Z. 2025. Qwen3 Technical Report. arXiv:2505.09388.

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; Lin, H.; Yang, J.; Tu, J.; Zhang, J.; Yang, J.; Yang, J.; Zhou, J.; Lin, J.; Dang, K.; Lu, K.; Bao, K.; Yang, K.; Yu, L.; Li, M.; Xue, M.; Zhang, P.; Zhu, Q.; Men, R.; Lin, R.; Li, T.; Xia, T.; Ren, X.; Ren, X.; Fan, Y.; Su, Y.; Zhang, Y.; Wan, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; and Qiu, Z. 2024a. Qwen2.5 Technical Report. *CoRR*, abs/2412.15115.

Yang, A.; Zhang, B.; Hui, B.; Gao, B.; Yu, B.; Li, C.; Liu, D.; Tu, J.; Zhou, J.; Lin, J.; Lu, K.; Xue, M.; Lin, R.; Liu, T.; Ren, X.; and Zhang, Z. 2024b. Qwen2.5-Math Technical Report: Toward Mathematical Expert Model via Self-Improvement. *CoRR*, abs/2409.12122.

Yang, H.; Liu, X.; and Wang, C. D. 2023. FinGPT: Open-Source Financial Large Language Models. *CoRR*, abs/2306.06031.

Young, A.; Chen, B.; Li, C.; Huang, C.; Zhang, G.; Zhang, G.; Li, H.; Zhu, J.; Chen, J.; Chang, J.; Yu, K.; Liu, P.; Liu, Q.; Yue, S.; Yang, S.; Yang, S.; Yu, T.; Xie, W.; Huang, W.; Hu, X.; Ren, X.; Niu, X.; Nie, P.; Xu, Y.; Liu, Y.; Wang, Y.; Cai, Y.; Gu, Z.; Liu, Z.; and Dai, Z. 2024. Yi: Open Foundation Models by 01.AI. *CoRR*, abs/2403.04652.

Zhang, X.; and Yang, Q. 2023. XuanYuan 2.0: A Large Chinese Financial Chat Model with Hundreds of Billions Parameters. In Frommholz, I.; Hopfgartner, F.; Lee, M.; Oakes, M.; Lalmas, M.; Zhang, M.; and Santos, R. L. T., eds., *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM 2023, Birmingham, United Kingdom, October 21-25, 2023*, 4435–4439. ACM.

Zhao, Y.; Liu, H.; Long, Y.; and Zhang, R. 2022. MultiHiertt: A Benchmark for Multi-Hop Reasoning over Financial Tables. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 1234–1245.

Zhao, Y.; Liu, H.; Long, Y.; and Zhang, R. 2024a. FinanceMATH: Knowledge-Intensive Math Reasoning in Finance Domains. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 12841–12858.

Zhao, Y.; Long, Y.; Liu, H.; Kamoi, R.; Nan, L.; Chen, L.; Liu, Y.; Tang, X.; Zhang, R.; and Cohan, A. 2024b. DOCMATH-EVAL: Evaluating Math Reasoning Capabilities of LLMs in Understanding Long and Specialized Documents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 16103–16115.

Zhu, K.; Chen, J.; Wang, J.; Gong, N. Z.; Yang, D.; and Xie, X. 2024. DyVal: Dynamic Evaluation of Large Language Models for Reasoning Tasks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zhu, S.; Chen, B.; Liu, B.; and Wang, H. 2021. TAT-QA: Fast Transfer Learning for Table-based Question Answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 5853–5859.