DocMath-Eval: Evaluating Math Reasoning Capabilities of LLMs in Understanding Long and Specialized Documents

Yilun Zhao[*1] **Yitao Long**[*2] **Hongjun Liu**[2] **Ryo Kamoi**[3] **Linyong Nan**[1]
Lyuhao Chen[4] **Yixin Liu**[1] **Xiangru Tang**[1] **Rui Zhang**[3] **Arman Cohan**[1,5]
[1]Yale University [2]New York University [3]Penn State University
[4]Carnegie Mellon University [5]Allen Institute for AI

https://github.com/yale-nlp/DocMath-Eval
https://docmath-eval.github.io/

Abstract

Recent LLMs have demonstrated remarkable performance in solving exam-like math word problems. However, the degree to which these numerical reasoning skills are effective in real-world scenarios, particularly in expert domains, is still largely unexplored. This paper introduces DocMath-Eval, a comprehensive benchmark specifically designed to evaluate the numerical reasoning capabilities of LLMs in the context of understanding and analyzing specialized documents containing both text and tables. We conduct an extensive evaluation of 48 LLMs using Chain-of-Thought and Program-of-Thought prompting techniques, aiming to comprehensively assess the capabilities and limitations of existing LLMs in DocMath-Eval. We found that even the current best-performing system (*i.e.*, GPT-4o) still significantly lags behind human experts in solving complex numerical reasoning problems grounded in long contexts. We believe that DocMath-Eval can serve as a valuable benchmark for evaluating LLMs' capabilities in solving challenging numerical reasoning problems within expert domains.

1 Introduction

Recent advancements in large language models (LLMs) have attracted significant attention due to their capabilities in solving a broad range of tasks (OpenAI, 2023; AI@Meta, 2024), including math word problems (MWPs) commonly found in academic exams (Wang et al., 2017; Miao et al., 2020; Amini et al., 2019; Cobbe et al., 2021; Hendrycks et al., 2021; Cobbe et al., 2021; Lu et al., 2023; Chen et al., 2023b). These MWPs vary from basic arithmetic to advanced algebra, showcasing LLMs' proficiency in numerical reasoning — a crucial skill for interpreting and manipulating numerical data across various contexts. Despite

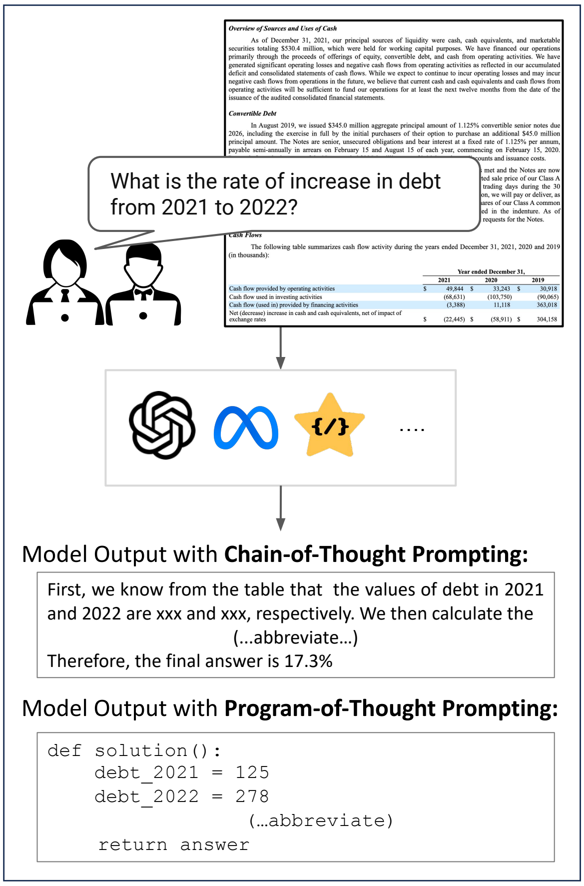

Figure 1: The overview of DocMath-Eval and the prompting methods explored. DocMath-Eval evaluates the LLMs' performance in the context of understanding and analyzing financial documents containing both text and tables. The models are required to first locate question-relevant data points within lengthy documents, and then apply numerical reasoning and specialized financial knowledge to answer the question.

this progress, there is still a significant gap in understanding the practicality of LLMs' numerical reasoning in real-world scenarios, particularly in specialized fields such as finance, medicine, and science. As illustrated in Figure 1, these expert domains necessitate LLMs to interpret complex, domain-specific documents, applying numerical

[*]Equal Contributions.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16103–16115
August 11-16, 2024

reasoning to complex problem-solving (Chen et al., 2021; Zhu et al., 2021; Zhao et al., 2022; Li et al., 2022b). Recognizing this gap, our research focuses on the finance domain (Li et al., 2022a; Wu et al., 2023a; Yang et al., 2023; Callanan et al., 2023; Xie et al., 2024). The finance industry often deals with lengthy and data-intensive documents that demand advanced numerical reasoning skills for accurate analysis and decision-making.

We introduce **DocMath-Eval**, a comprehensive and standardized benchmark that systematically evaluates the numerical reasoning capabilities of LLMs in understanding and interpreting specialized documents containing both textual and tabular data. DocMath-Eval encompasses four evaluation sets, each with varying levels of difficulty in *numerical reasoning* and *document understanding*. Specifically, We construct a new evaluation set, **DM$_{\textbf{CompLong}}$**, from scratch, to examine the LLM's capabilities in performing *complex* numerical reasoning over *extreme long* documents containing *multiple* tables. We also adapt and re-annotate four existing finance QA benchmarks to develop three additional, less challenging evaluation sets: 1) **DM$_{\textbf{SimpShort}}$** based on TAT-QA (Zhu et al., 2021) and FinQA (Chen et al., 2021), necessitates *simple* numerical reasoning over *short* document with *one* table; 2) **DM$_{\textbf{SimpLong}}$** based on MultiHiertt (Zhao et al., 2022), necessitates *simple* numerical reasoning over *long* document with *multiple* tables; and 3) **DM$_{\textbf{CompShort}}$** based on TAT-HQA (Li et al., 2022b), necessitates *complex* numerical reasoning over *short* document with *one* table.

We conduct an extensive evaluation on DocMath-Eval, covering a total of 48 proprietary and open-source LLMs from 17 organizations. Two prompting methods, Chain-of-Thought (CoT) (Wei et al., 2022) and Program-of-Thought (PoT) (Chen et al., 2023a), are applied for result analysis. Our experimental results indicate that while the existing best-performing LLM on average (*i.e.,* GPT-4o) can achieve high performance in simple settings (*e.g.,* DM$_{\text{SimpShort}}$), it still falls short of human experts in more challenging ones, *i.e.,*, DM$_{\text{CompLong}}$. Moreover, Claude-3.5-Sonnet outperforms other LLMs, achieving an accuracy of 40.0% on the DM$_{\text{CompLong}}$ set when applying CoT prompting. However, it still lags far behind human expert performance, which stands at 76%. This significant gap between LLMs and human experts underscores the challenges presented by DocMath-Eval. It underscores the importance of advancing LLMs' numerical reasoning and document understanding abilities to effectively apply them in the real-world specialized domains.

We conclude our main contributions as follows:

- We introduce DocMath-Eval, a comprehensive benchmark designed to systematically evaluate LLMs' numerical reasoning ability to understand and interpret long and specialized documents. This includes a newly developed, challenging evaluation set and three adapted evaluation sets for varying difficulty levels.
- We conduct an extensive evaluation encompassing a wide range of LLMs, including those specialized in math and coding. We also incorporate different prompting methods (*i.e.,* CoT and PoT) to comprehensively assess the capabilities and limitations of existing LLMs in our task.
- Our experimental results reveal a noticeable performance gap compared to human experts in more complex scenarios (*i.e.,* problems requiring complex numerical reasoning over long documents). This highlights the limitations of current LLMs in complex real-world applications and the need for continued advancements.

2 Related Work

Math Word Problems The research community has shown significant interest in the vital role of numerical reasoning skills in LLMs. These skills are vital for models to effectively engage in complex problem-solving. To this end, a wide variety of MWP datasets have been proposed in recent years (Hosseini et al., 2014; Koncel-Kedziorski et al., 2016; Wang et al., 2017; Ling et al., 2017; Cobbe et al., 2021). More challenging datasets have recently been introduced to enhance diversity (Miao et al., 2020), difficulty (Chen et al., 2023b; Hendrycks et al., 2021), and adversarial robustness (Patel et al., 2021). However, existing MWP datasets predominantly focus on problems akin to academic exams, with a limited emphasis on real-world scenarios. Addressing this gap, our paper introduces a novel and comprehensive benchmark designed to evaluate LLMs' abilities in understanding and interpreting long and specialized documents through numerical reasoning.

Numerical Reasoning over Documents Numerical reasoning over documents requires models to

Property (Median/Avg)	**DM$_{\text{SimpShort}}$**	**DM$_{\text{SimpLong}}$**	**DM$_{\text{CompShort}}$**	**DM$_{\text{CompLong}}$ (new)**
Data Source	TAT-QA (Zhu et al., 2021) FinQA (Chen et al., 2021)	MultiHiertt (Zhao et al., 2022)	TAT-HQA (Li et al., 2022b)	expert annotated from scratch
Question Length	19 / 20.0	21 / 21.6	29 / 30.1	34 / 37.7
# Sentences in Text	14 / 16.9	64 / 66.9	6 / 7.8	535 / 752.3
# Words in Text	504 / 506.6	2,216 / 2,334.0	251 / 314.2	25,149 / 34,589.0
# Table	1 / 1.0	4 / 3.9	1 / 1.0	46 / 72.5
# Rows per Table	6 / 7.0	9 / 11.6	7 / 8.2	3 / 7.5
# Columns per Table	5 / 4.7	4 / 4.5	5 / 5.0	3 / 3.1
# Text Evidence	0 / 0.4	1 / 0.9	0 / 0.4	1 / 1.0
# Table Evidence	1 / 0.9	1 / 1.1	1 / 1.0	1 / 1.0
% Questions w. Table Evidence	92.9%	86.4%	97.8%	76.3%
# Math Operations in Python Solution	2 / 2.1	2 / 2.3	2 / 2.3	4 / 4.9
# Code Lines in Python Solution	5 / 5.3	6 / 5.9	5 / 5.3	8 / 8.2
# Comment Lines in Python Solution	2 / 2.0	2 / 2.0	2 / 2.0	2 / 3.4
Development set	200	100	200	300
Test set	800	400	800	1,200
Total Size	1,000	500	1,000	1,500

Table 1: Basic statistics of DocMath-Eval dataset. Our newly constructed evaluation set, DM$_{\text{CompLong}}$, poses unique challenges in both numerical reasoning and financial document understanding.

have a deep understanding of context and the ability to derive answers through numerical reasoning (Dua et al., 2019). Applying these models in the finance domain (Xie et al., 2023; Wu et al., 2023a; Yang et al., 2023) presents additional challenges in terms of interpreting hybrid data (Zhu et al., 2021) and utilizing domain-specific expertise (Chen et al., 2021; Zhao et al., 2024). Numerous datasets focusing on numerical reasoning over specialized documents have been proposed recently. Two notable benchmarks are TAT-QA (Zhu et al., 2021) and FinQA (Chen et al., 2021), which represent pioneering efforts in studying numerical reasoning in finance, particularly requiring the fusion of tabular and textual content. Building upon TAT-QA, a more challenging dataset named TAT-HQA (Li et al., 2022b) was developed, focusing on counterfactual questions in relation to the provided context. Additionally, MultiHiertt (Zhao et al., 2022) focuses on numerical reasoning over longer financial documents containing multiple tables. However, as illustrated in Table 1, these four datasets focus on less challenging scenarios, where either simple numerical reasoning (*e.g.,* calculating the increasing rate or average value) is sufficient, or the input context is short. Furthermore, there is a lack of a standardized benchmark for systematically evaluating models' performance across varying difficulty levels in terms of numerical reasoning and document understanding.

3 DocMath-Eval

In this section, we first offer a formal definition of the DocMath-Eval task. We then explain the rationale and methodology for adopting Python program as the standardized solution format for DocMath-Eval. Subsequently, we detail the data annotation process used to construct the challenging DM$_{\text{CompLong}}$ evaluation set, as well as the data re-annotation process for compiling the other three evaluation sets. Table 7 in the Appendix presents the profiles of the seven annotators involved. Finally, we present human-level performance on each evaluation set in DocMath-Eval.

3.1 Task Formulation

We formally define the task of DocMath-Eval in the context of LLMs as follows: Presented with a numerical reasoning question q and a financial document consisting of textual contents E and structured tables T, the task is to generate the numeric-value answer a:

$$\hat{a} = \arg\max_{a} P_{\mathbf{LM}}(a \mid q, E, T) \qquad (1)$$

To obtain the best candidate answer $\hat{a}$, we use greedy decoding in all our LLM evaluations.

3.2 Solution Format Standardization

We observe that existing finance QA datasets feature solutions in various formats. Specifically, TAT-QA (Zhu et al., 2021) and TAT-HQA (Li et al., 2022b) utilize text, while MultiHiertt (Zhao et al., 2022) employs mathematical expressions, such as `100/3`, and FinQA (Chen et al., 2021) uses math programs, such as `divide(100,3)`, for solution annotations. This diversity in annotation formats hinders the development of a unified evaluation framework to assess LLM performance across different benchmarks. Additionally, text-based solutions often fall short in precision and clarity, making them less suitable for computational problem-solving; and the solutions presented as mathematical equations or programs can be less descriptive, with the intended semantic meaning of the equations sometimes being unclear.

To overcome the aforementioned limitations, in DocMath-Eval, we represent solutions using Python programs (Zhao et al., 2024). Such a unified Python program format supports a standardized and effective evaluation framework for LLM assessment. Specifically, annotators are instructed to initially define variables at the start of the Python function, beginning with "`def solution():`". These variables should align with the primary elements or quantities referenced in the question or relevant content in the documents. They then write a Python program that methodically address the problem, solving it step by step. Additionally, annotators receive a bonus for writing detailed comments, thereby enhancing the code's readability and understandability. To verify the correctness and performance of the solutions, our annotation interface automatically runs the Python function. This process checks that the output is either a float or int and ensures that the execution finishes without any errors.

3.3 Data Re-Annotation From Public Datasets

We re-annotate four existing datasets and incorporate them into DocMath-Eval. Specifically, we re-annotate TAT-QA (Zhu et al., 2021) and FinQA (Chen et al., 2021) for $\text{DM}_{\text{SimpShort}}$, MultiHiertt (Zhao et al., 2022) for $\text{DM}_{\text{SimpLong}}$, and TAT-HQA (Li et al., 2022b) for $\text{DM}_{\text{CompShort}}$.

Question Validation and Re-annotation We instruct the annotators to identify and remove questions with incorrect annotations or those whose answers are not numerical. Annotators are then asked to enhance each question by adding a scale descriptor to ensure clarity and specificity. For example, *"Question: What is the average payment volume per transaction for American Express? **(in billions)**"*. They were also asked to correct any identified errors in the original questions.

Solution Validation and Re-annotation As outlined in Section 3.2, we require annotators to rewrite the original solutions into a unified Python format, standardizing variable names and adding comments to enhance the readability of the solutions. Regarding the supporting evidence annotation, we initially convert the original evidence annotations to our format. We then highlight these evidences in the annotation interface, and direct annotators to verify their correctness.

3.4 Data Annotation From Scratch

In real-world scenarios, financial professionals typically need to handle documents spanning tens of pages, along with problems that require more complex numerical reasoning combined with financial knowledge. However, as previously discussed, existing benchmarks (Zhu et al., 2021; Chen et al., 2021; Zhao et al., 2022; Li et al., 2022b) focus on less challenging scenarios, where either simple numerical reasoning is sufficient, or the input context is short. To bridge this gap, we have developed a new, challenging evaluation set, $\mathbf{DM}_{\textbf{CompLong}}$, from scratch. This set focuses on settings that more closely align with real-world scenarios, where models are required to perform complex numerical reasoning over long financial documents for problem solving. The annotation process is as follows:

Source Document Collection Following previous work (Zhu et al., 2021; Chen et al., 2021; Zhao et al., 2022), we use the quarterly (i.e., Form 10-Q) and annual reports (i.e., Form 10-K) of companies as our source documents, which are publicly available at the open-source database[1] of U.S. Securities and Exchange Commission. After collecting all the source documents, we utilize a commercial API[2] to extract their textual and tabular content. Subsequently, we apply a heuristic-based method to preprocess these two formats of content. The preprocessed documents are then passed to expert annotators for question annotation.

Data Annotation Given a financial document, annotators are first required to briefly read its content and determine the data points to be used in the question. They must then compose the question and highlight the selected paragraphs or ta-

[1] https://www.sec.gov/edgar/search/
[2] https://sec-api.io/

bles as evidence supporting it. Finally, the annotators are required to write down the solution to the question in Python program format, as discussed in Section 3.2. We set up a *bonus payment system* for complex annotations that involve difficult document comprehension and numerical reasoning. Specifically, to increase the difficulty of document understanding, we award bonuses to annotators for questions that necessitate information from: 1) multiple tables, 2) multiple sections, or 3) a combination of tables and textual content. To enhance the challenge in numerical reasoning, we provide bonuses for questions requiring financial expertise or involving complex mathematical operations. If such annotations are validated during the quality validation stage, a bonus payment will be added.

Quality Validation We implement a comprehensive quality validation protocol to ensure that each annotated example meets the required standards. For every question annotation, we assign it to another annotator, recognized for their high performance in annotation, to verify its accuracy. This process involves manually locating the question-relevant evidence in the documents using our retrieval-based search toolkits. They then compare this evidence with the original annotations and correct any errors found. Additionally, validators are tasked with confirming the accuracy of the annotated solutions. We offer bonus payments to annotators for identifying erroneous annotations. Ultimately, 232 of the annotated questions are flagged as erroneous and are subsequently revised. Table 6 in the Appendix presents the human evaluation scores and inter-evaluator agreements for a subset of 200 sampled examples. DocMath-Eval exhibits superior annotation quality and a high degree of inter-annotator agreement.

3.5 Expert-level Performance Evaluation

To give a general yet insightful estimate of the performance on each of the DocMath-Eval sets, we enlisted two professionals who hold Chartered Financial Analyst licenses to conduct the evaluation. Regarding human expert performance on $\text{DM}_{\text{SimpShort}}$ and $\text{DM}_{\text{SimpLong}}$, we report the same results as those in the original papers, with accuracy of 91% and 87%, respectively. For $\text{DM}_{\text{CompShort}}$ and $\text{DM}_{\text{CompLong}}$, We randomly sample 25 examples from each set, asking the expert evaluators to answer the questions individually within a four-hour period. They achieve accuracy of 88% and 80% on $\text{DM}_{\text{CompShort}}$ (average 84%); and accuracy of 72% and 80% on $\text{DM}_{\text{CompLong}}$ (average 76%).

Chain-of-Thought Prompt

```
[System Input]:
```

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

```
[User Input]:
```

{Document context}

Question: {question}

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

Figure 2: Example of *zero*-shot CoT prompt used.

3.6 Dataset Release

Table 1 presents the data statistics of four developed evaluation sets. DocMath-Eval contains a total of 4,000 questions with high-quality annotations, featuring varying difficulty levels in numerical reasoning and document understanding. We randomly partitioned the dataset into two subsets: *testmini* and *test*. The *testmini* subset includes 800 examples and is intended for model development and validation. The *test* subset consists of the remaining 3,200 examples, which are reserved for standard evaluation. To avoid data contamination (Deng et al., 2024), the features directly related to the ground truth for the *test* set are kept private. Instead, we have developed and manage an online evaluation platform, where researchers can assess models and participate in a leaderboard.

4 Experiment Setup

This section discusses the experiment setup, including the evaluated LLMs, prompting methods, and our implementation details.

4.1 Evaluated Large Language Models

Our goal is to investigate the capabilities of current state-of-the-art LLMs on DocMath-Eval to better understand their strengths and limitations. To this end, we evaluate a wide range of models, including 32 general-purpose LLMs, 4 math-specific LLMs, 6 code-based LLMs, and 7 mixture of experts (MoE) models. The specific details of each evaluated LLM, including the exact version used, can be found in Table 8 in the Appendix.

4.2 Prompting Methods

Following recent works on LLM reasoning benchmarks (Lu et al., 2024; Chen et al., 2023b), we evaluate two commonly used prompting methods for math reasoning:

Chain-of-Thought The CoT method (Wei et al., 2022) instructs the LLMs to explicitly outline their reasoning process step by step before arriving at the final answer. Figure 2 presents the CoT prompt used in our experiment.

Program-of-Thought The PoT method (Chen et al., 2023a) separates computation from the reasoning process by instructing the LLMs to produce a structured program that encapsulates the reasoning steps. The final answer is obtained by executing the generated program. Figure 3 in Appendix presents the PoT prompt we used.

4.3 Implementation Details

LLM Experiment The experiments involving open-sourced LLMs were conducted using the `vLLM` framework (Kwon et al., 2023). In all the experiments, we used a temperature setting of 1.0 and maximum output length of 512. Given the extensive context length of input document, the main evaluation of DocMath-Eval is conducted under a *zero-shot* setting, aiming to assess LLMs' capabilities to generate accurate answers without few-shot demonstrations or additional training.

Input Tabular Data Serialization Building on previous work that evaluated LLMs on table-relevant tasks (Chen, 2023; Zhao et al., 2023a,b), we present our method for processing tabular data in documents. Specifically, we separate headers or cells in different columns using a vertical bar (|), and rows using a newline. This approach allows for the direct feeding of flattened table input into LLMs. In our preliminary study, we found that most LLMs can comprehend these table formats well. Nevertheless, we believe that future research could explore more effective methods for encoding tabular data (Fang et al., 2024).

RAG-based Setting for $\mathrm{DM}_{\mathrm{CompLong}}$ For the $\mathrm{DM}_{\mathrm{CompLong}}$ subset, the input document length is extremely long and exceeds the context length limit of evaluated LLMs. Therefore, in our main experiments with $\mathrm{DM}_{\mathrm{CompLong}}$, we evaluate models using the retrieval-augmented generation (RAG) setting. In this setting, external retrievers are employed to extract the top-n most relevant textual and tabular evidence from the source document. We maintain the original relative order of the evidence and input it into the LLMs to answer the given question. We experiment with commonly-used sparse retriever, *i.e.,* BM25 (Robertson et al., 1995), and three dense retrievers, including OpenAI Embedding 3 small & large versions (Neelakantan et al., 2022) and Contriever (Izacard et al., 2022).

Final Answer Extraction For LLMs using CoT prompting, we adopt the answer extraction process from Chen et al. (2023b) and Lu et al. (2024) to extract the final answer from the model's output. For LLMs employing PoT prompting, we first develop a heuristic method to extract the generated python solution from the model response. We then execute it to obtain the final answer.

5 Results and Analysis

We next discuss our main findings from the experiments and our analysis of the $\mathrm{DM}_{\mathrm{CompLong}}$ subset.

5.1 Main Results

Table 2 and Table 9 in the Appendix present the LLM performance on the DocMath-Eval testmini and test sets, respectively.

While the current best-performing LLM, GPT-4o, achieves performance comparable to human experts in simple problem settings (*i.e.,* $\mathrm{DM}_{\mathrm{SimpShort}}$ and $\mathrm{DM}_{\mathrm{CompShort}}$), we find significant performance gaps in more challenging settings. Specifically, GPT-4o achieves an accuracy of 41.0% on $\mathrm{DM}_{\mathrm{CompLong}}$ with PoT, which is far behind the human expert performance of 76.0%. This underscores the need for ongoing LLM development, particularly in complex problem-solving over long and specialized documents. Most open-source LLMs still lag behind the proprietary LLMs. However, the two DeepSeek-V2-* models come close to matching the performance of the leading proprietary models. The DeepSeek-V2 even outperforms GPT-4o on the $\mathrm{DM}_{\mathrm{CompLong}}$ subset. This suggests that open-source LLMs have the potential to bridge the performance gap with the leading proprietary models in the near future.

The code-specific and proprietary LLMs generally perform as well as or better with PoT prompt-

Model	Size	Notes	$DM_{SimpShort}$		$DM_{CompShort}$		$DM_{SimpLong}$		$DM_{CompLong}$		Avg. Acc	
			PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT
Human Expert			91.0		87.0		84.0		76.0			
Proprietary LLMs												
GPT-4o			84.0	86.0	69.5	76.5	56.0	64.0	41.0	36.7	60.8	62.4
GPT-4-Turbo			85.5	82.5	80.0	81.0	56.0	53.0	38.7	38.3	62.9	61.9
Claude-3-Opus			80.5	79.5	73.5	77.5	51.0	61.0	42.0	39.7	60.6	61.8
Claude-3.5-Sonnet			78.0	77.0	76.0	69.5	54.0	61.0	44.0	40.0	61.8	59.2
Claude-3-Sonnet			82.5	80.0	80.5	73.0	55.0	56.0	40.3	35.3	62.7	58.5
Gemini-1.5-Flash			85.0	78.0	78.5	69.5	55.0	46.0	40.0	31.7	62.8	54.5
Gemini-1.5-Pro			85.5	80.5	80.0	58.0	58.0	55.0	40.3	30.0	63.7	52.8
Claude-3-Haiku			74.5	79.0	71.5	58.5	55.0	50.0	36.7	31.7	57.1	52.5
GPT-4o-Mini			88.5	69.5	77.0	69.5	53.0	56.0	38.7	28.0	62.5	52.2
GPT-3.5-Turbo			71.0	60.5	52.5	39.0	41.0	28.0	28.7	15.0	46.8	34.0
Open-source LLMs												
DeepSeek-V2	236B	MoE	87.0	82.0	75.5	69.5	61.0	56.0	43.0	39.7	64.4	59.8
Mistral-Large	123B		85.0	83.5	76.5	81.0	56.0	55.0	41.0	31.3	62.8	59.7
DeepSeek-Coder-V2	236B	Code, MoE	85.0	79.0	78.0	66.5	56.0	54.0	41.0	37.7	63.1	57.3
Llama-3.1	70B		74.5	76.5	68.0	71.0	53.0	50.0	34.7	29.3	55.3	54.1
Qwen2	72B		26.5	74.0	24.5	72.5	8.0	45.0	7.0	27.0	16.4	52.4
Llama-3	70B		84.5	73.5	64.0	63.5	52.0	42.0	41.0	28.3	59.0	50.1
Mixtral-8x22B	141B	MoE	30.0	74.0	21.5	57.0	25.0	47.0	14.7	24.0	21.5	47.6
Gemma-2	9B		79.0	66.5	65.0	54.5	50.0	39.0	24.3	17.7	51.4	41.8
DeepSeek-Coder-V2-Lite	16B	Code	66.0	67.5	51.0	53.5	27.0	30.0	22.0	20.3	40.9	41.6
WizardLM-2	141B	MoE	62.5	60.5	56.5	55.5	25.0	34.0	17.7	18.0	39.5	40.0
C4AI Command R+	104B		35.5	65.5	39.0	51.0	19.0	31.0	8.7	18.3	24.3	39.9
Yi-1.5	9B		18.0	68.5	24.5	56.0	2.0	14.0	4.0	14.0	12.4	38.1
Yi-1.5	34B		0.5	64.5	1.0	53.0	0.0	14.0	0.0	15.3	0.4	36.9
Mistral-Nemo	12B		52.5	59.5	37.5	44.0	28.0	37.0	15.3	16.7	31.7	36.8
Llama-3.1	8B		62.0	60.0	44.0	42.5	32.0	33.0	19.0	14.3	37.6	35.1
DBRX	132B	MoE	41.0	57.0	29.5	43.0	32.0	30.0	12.0	16.3	26.1	34.9
Codestral	22B	Code	39.0	51.5	38.5	41.5	18.0	23.0	17.3	13.0	28.1	31.0
Llama-3	8B		49.5	56.5	21.5	31.0	24.0	29.0	10.0	12.3	24.5	30.1
Qwen2	7B		13.0	56.0	9.5	33.0	4.0	31.0	2.3	10.0	7.0	29.9
Mathstral	7B	Math	43.5	55.0	32.5	35.0	10.0	23.0	11.3	11.7	24.5	29.8
GLM-4	9B		69.5	44.0	53.5	34.0	33.0	20.0	17.7	8.7	41.5	25.3
Aya-23	35B		1.5	44.0	1.0	25.5	0.0	20.0	0.0	11.7	0.6	24.3
DeepSeek-V2-Lite	16B	MoE	7.0	45.5	3.5	18.0	1.0	17.0	1.0	10.3	3.1	21.9
Mixtral-8x7B-v0.1	46B	MoE	0.5	39.0	2.0	17.0	0.0	25.0	0.0	12.7	0.6	21.9
DeepSeek-Math	7B	Math	2.0	46.0	1.0	27.0	1.0	4.0	0.3	8.0	1.0	21.8
Llama-2	70B		32.5	43.5	16.5	25.0	1.0	8.0	2.0	7.0	13.1	20.8
WizardLM-2	7B		47.0	42.0	30.5	28.5	5.0	6.0	7.3	5.7	22.7	20.5
Mistral-v0.3	7B		49.5	40.0	40.5	28.0	25.0	9.0	11.3	5.7	29.9	20.3
WizardMath	7B	Math	22.5	32.0	12.0	22.5	6.0	7.0	3.7	3.3	10.8	15.7
InternLM2-Math-Plus	7B	Math	28.5	27.5	15.0	14.0	7.0	9.0	4.7	4.0	13.5	13.0
StarCoder2	15B	Code	47.5	21.0	34.0	15.5	11.0	6.0	8.3	4.3	24.9	11.5
InternLM2	7B		18.0	20.0	4.5	11.0	9.0	10.0	2.7	2.3	7.8	9.9
Gemma-1	7B		1.0	20.0	0.0	7.5	0.0	7.0	0.0	3.3	0.2	9.0
Llama-2	7B		4.0	17.0	4.0	11.5	0.0	2.0	1.3	2.7	2.5	8.4
DeepSeek-Coder-V1	33B	Code	19.0	18.5	8.5	8.5	2.0	2.0	3.7	1.7	8.5	7.6
WizardCoder	33B	Code	32.5	16.0	17.5	8.0	5.0	2.0	5.0	1.0	15.0	6.6
Aya-23	8B		1.0	13.0	0.0	9.0	0.0	2.0	0.3	2.3	0.4	6.6
Gemma-1	2B		4.0	8.0	1.5	7.5	0.0	2.0	0.0	0.0	1.4	4.1

Table 2: LLM performance on the *testmini* set of DocMath-Eval. We utilize the average accuracy achieved through CoT prompting as the metric for ranking model performance. For $DM_{CompLong}$, we use the OpenAI Embedding 3 Large retriever to retrieve top-10 evidence as input document. Numbers underlined indicate that models using PoT prompting outperform those using CoT prompting.

ing compared to CoT prompting. This is likely because LLMs are prone to making errors during complex mathematical computations, as revealed in concurrent work (Zhao et al., 2024). Additionally, for math-specific LLMs, InternLM2-Math-Plus outperforms its base model in CoT performance, with average accuracy rising from 9.9% to 13.0%. This highlights the impact of instruction-tuning in improving math reasoning abilities.

5.2 Analysis on $DM_{CompLong}$ Set

We next conduct a detailed analysis of the RAG setting, long-context LLMs, and model failure cases.

RAG Analysis We analyze the impact of retriever performance on the final accuracy of RAG-based LLM systems by selecting the Llama-3-70B and GPT-4o models for our study. As demonstrated in Table 3, the OpenAI Embedding-3 significantly

top-n	Retriever	R@n	Llama-3	GPT-4o
3	Contriever	22.3	13.7	16.0
	BM25	29.5	13.7	15.7
	Embedding-3-Small	44.7	19.0	24.0
	Embedding-3-Large	48.2	22.0	27.0
5	Contriever	32.0	15.3	22.0
	BM25	38.0	15.3	20.7
	Embedding-3-Small	57.1	21.0	29.0
	Embedding-3-Large	62.0	24.0	32.7
10	Contriever	45.3	18.3	25.7
	BM25	47.9	20.3	23.0
	Embedding-3-Small	71.2	25.3	31.7
	Embedding-3-Large	75.8	26.3	36.7
–	Oracle		35.3	42.0

Table 3: Results of the Llama-3-70B and GPT-4o with CoT prompting approaches under various retrieval settings on the $\text{DM}_{\text{CompLong}}$ testmini set. A correlation is observed between LLM performance and the question-relevance of the retrieved evidence.

Model	RAG	Long Context
GPT-4o	36.7	**40.3**
Gemini-1.5-Pro	30.0	**37.3**
Claude-3-Sonnet	**35.3**	34.7
Gemini-1.5-Flash	31.7	**34.3**
Claude-3-Haiku	**31.7**	31.0
DeepSeek-V2	**39.7**	38.7
DeepSeek-Coder-V2	**37.7**	36.0
Llama-3.1-70B	**29.3**	26.3
Llama-3.1-8B	**14.3**	9.0
Mistral-Nemo	**16.7**	4.7
Phi-3-Medium	12.7	**13.0**
GLM-4-9B	8.7	**9.7**

Table 4: Results of the CoT prompting approach under various retrieval settings on $\text{DM}_{\text{CompLong}}$ testmini set.

outperforms Contriever and BM25. Additionally, improved retriever performance consistently boosts the final accuracy of the models in our task. These results highlight the need for future work to develop more advanced information retrieval techniques for enhancing complex problem-solving over long and specialized documents.

Long-Context LLM Analysis In addition to using RAG for analyzing long specialized documents, recent advancements have extended the input length of LLMs to handle lengthy documents (Su et al., 2023). We compare models with a context length limit of over 100K under both the RAG (as used in the main results) and Long-Context settings, where the entire document is input. As illustrated in Table 4, the evaluated models generally achieve close performance under RAG and long-context settings. This indicates that models with extended context lengths can effectively process lengthy inputs without a significant drop in performance compared to the RAG setting.

Error Type	Explanation
Inaccurate Evidence Retrieval (39 / 100)	The challenge lies in finding accurate evidence, especially in situations where the values needed for intermediate reasoning steps are not explicitly stated. This makes it difficult for the retriever to identify the correct evidence.
Calculation Error (28 / 100)	The reasoning process is accurate, but there are errors in the intermediate or final computations.
Table Misunderstanding (16 / 100)	The model faces challenges in comprehending and parsing cell values, particularly in complex tables.
Exceeding Context Length (8 / 100)	The input document exceeds the context length limit.
Others	

Table 5: Error types and explanations of GPT-3.5-turbo failure cases on the $\text{DM}_{\text{CompLong}}$ testmini set.

Error Analysis To better understand the strengths and weaknesses of LLMs, we conduct an extensive error analysis. This analysis focuses on 100 randomly selected examples from the $\text{DM}_{\text{CompLong}}$ testmini set where GPT-3.5-turbo failed. We identify four common types of errors in current LLMs: inaccurate evidence retrieval, calculation errors, table misunderstandings, and exceeding context length. A detailed explanation for each type is provided in Table 5.

6 Conclusion

This paper introduces DocMath-Eval, a comprehensive benchmark designed to evaluate the capabilities of LLMs in numerical reasoning over long and specialized documents. Our experiments show that even the best-performing current models still fall short of human expert performance on problems requiring complex reasoning over extended contexts. This highlights the need for future research to improve LLMs' proficiency in complex numerical reasoning tasks within expert domains.

Limitations

There are some limitations in our study that we believe can be addressed in future work. First, our approach to extracting the final answer from the model's output is not yet flawless. In certain instances, this method fails to accurately identify the answer, causing the reported accuracy to be an approximate lower limit. Additionally, we suggest that future research could investigate training large language models (LLMs) on finance-specific data to improve their performance on the DocMath-Eval benchmark (Wu et al., 2023b; Luukkonen et al., 2023; Xie et al., 2023).

Acknowledgement

We are grateful for the compute support provided by Microsoft Research's Accelerate Foundation Models Research (AFMR) program. We extend our gratitude to the anonymous reviewers and area chairs for their valuable discussions and feedback.

References

AI@Meta. 2024. Llama 3 model card.

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

Ethan Callanan, Amarachi Mbakwe, Antony Papadimitriou, Yulong Pei, Mathieu Sibue, Xiaodan Zhu, Zhiqiang Ma, Xiaomo Liu, and Sameena Shah. 2023. Can gpt models be financial analysts? an evaluation of chatgpt and gpt-4 on mock cfa exams.

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. 2023a. 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. 2023b. 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*.

Chunyuan Deng, Yilun Zhao, Yuzhao Heng, Yitong Li, Jiannan Cao, Xiangru Tang, and Arman Cohan. 2024. Unveiling the spectrum of data contamination in language models: A survey from detection to remediation.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Xi Fang, Weijie Xu, Fiona Anting Tan, Ziqing Hu, Jiani Zhang, Yanjun Qi, Srinivasan H. Sengamedu, and Christos Faloutsos. 2024. Large language models (LLMs) on tabular data: Prediction, generation, and understanding - a survey. *Transactions on Machine Learning Research*.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

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*.

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

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

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

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 mathematical reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations*.

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

Risto Luukkonen, Ville Komulainen, Jouni Luoma, Anni Eskelinen, Jenna Kanerva, Hanna-Mari Kupari, Filip Ginter, Veronika Laippala, Niklas Muennighoff, Aleksandra Piktus, Thomas Wang, Nouamane Tazi, Teven Scao, Thomas Wolf, Osma Suominen, Samuli Sairanen, Mikko Merioksa, Jyrki Heinonen, Aija Vahtola, Samuel Antao, and Sampo Pyysalo. 2023. FinGPT: Large generative models for a small language. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2710–2726, Singapore. Association for Computational Linguistics.

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.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and code embeddings by contrastive pre-training.

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

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.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding.

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. 2023a. Bloomberggpt: A large language model for finance.

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

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

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

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. Fingpt: Open-source financial large language models. *FinLLM Symposium at IJCAI 2023*.

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, Hongjun Liu, Yitao Long, Rui Zhang, Chen Zhao, and Arman Cohan. 2024. Financemath: Knowledge-intensive math reasoning in finance domains.

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. 2023a. 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. 2023b. 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 Appendix

Annotation Quality	**%S $\geq$ 4**
Question Fluency	97.4
Question Correctness	96.0
Evidence Relevance	88.5
Evidence Completeness	91.3
Final Answer Correctness	97.9
Python Solution Correctness	97.6
Variable Value Correctness	98.5
Python Solution Conciseness	89.1
Variable Name Meaningfulness	95.4

Table 6: Human evaluation was conducted on 200 samples from DOCMATH-EVAL, with three internal reviewers asked to rate each sample on a scale from 1 to 5. We present the percentage of samples that received an average score of 4 or higher, as an indicator of the annotation quality of DOCMATH-EVAL.

Chain-of-Thought Prompt

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

```python
def solution():
    # Define variables name and value
    based on the given context
    ...
    # Do math calculation to get
    the answer
    ...
    # return answer
    return answer
```

[User Input]:
{Document context}

Question: {question}

Continue the program to answer the question. The returned value of the program is supposed to be the answer:

```python
def solution():
    # Define variables name and value
    based on the given context
```

Figure 3: Example of *zero*-shot PoT prompt used.

Annotator ID	Finance Industry Experience	Annotation Sets
1	1 working and 1 internship at US	New subset, Annotation validation
2	>= 2 internship at US	New subset, Annotation validation
3	1 working at Singapore and 2 internship at US	New subset
4	2 working and >= 1 internship at US	New subset
5	1 internship at US, 2 internship at China	Re-annotation on three subsets, Annotation validation
6	Graduate student majored in computer science	Re-annotation on three subsets, Annotation validation
7	Graduate student majored in statistics	Re-annotation on three subsets

Table 7: Details of annotators involved in dataset construction.

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

Table 8: Details of the LLMs evaluated in this study.

Model	Size	Notes	$DM_{SimpShort}$		$DM_{CompShort}$		$DM_{SimpLong}$		$DM_{CompLong}$		Avg. Acc	
			PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT
Human Expert			91.0		87.0		84.0		76.0			
Proprietary LLMs												
GPT-4-Turbo			88.9	86.2	78.6	77.8	31.5	61.2	20.2	35.6	53.4	62.0
GPT-4o			87.0	86.4	69.6	75.9	62.2	66.2	41.5	31.7	62.5	60.7
Claude-3.5-Sonnet			87.2	81.8	71.0	70.0	65.0	61.5	40.8	31.5	63.0	57.4
GPT-4o-Mini			88.8	76.0	77.2	72.4	56.8	50.2	36.9	29.2	62.4	54.3
Gemini-1.5-Pro			87.9	82.5	80.4	58.4	63.0	58.0	41.8	31.2	65.6	54.2
Claude-3-Haiku			77.8	82.2	72.4	61.5	55.0	53.2	34.6	30.8	57.4	54.2
Gemini-1.5-Flash			87.9	80.5	79.9	67.0	60.5	53.8	38.9	28.1	64.1	54.1
GPT-3.5-Turbo			75.6	64.0	47.6	34.2	46.2	38.8	23.2	13.2	45.3	34.3
Open-source LLMs												
Mistral-Large	123B		87.9	85.0	74.4	79.8	58.8	58.8	37.2	30.6	61.8	60.0
DeepSeek-V2	236B	MoE	88.9	86.1	76.9	66.6	58.2	57.2	40.2	32.0	63.8	57.3
DeepSeek-Coder-V2	236B	Code, MoE	88.2	80.9	74.9	67.2	56.8	54.8	37.5	31.5	61.9	55.7
Llama-3.1	70B		81.0	80.1	65.8	70.8	49.2	53.8	32.1	26.6	54.9	54.4
Qwen2	72B		27.8	77.5	25.0	70.1	16.8	49.0	5.7	25.1	17.4	52.4
Llama-3	70B		86.2	80.9	64.8	62.0	51.5	45.0	35.0	26.3	57.3	51.2
Mixtral-8x22B	141B	MoE	27.4	74.9	23.1	59.4	21.5	46.8	14.8	22.3	20.9	47.8
DeepSeek-Coder-V2-Lite	16B	Code	68.8	71.4	52.2	44.9	28.8	35.0	18.0	19.8	40.6	40.9
Gemma-2	9B		82.0	70.1	62.4	49.2	45.5	33.2	24.4	17.7	50.9	40.6
Yi-1.5	34B		1.0	71.2	0.9	58.2	0.2	17.8	0.2	13.7	0.6	39.7
C4AI Command R+	104B		37.6	67.5	36.5	50.6	19.2	35.0	7.6	15.1	23.8	39.6
WizardLM-2	141B	MoE	62.1	59.4	49.5	47.6	31.2	36.0	19.0	16.1	38.9	37.3
DBRX	132B	MoE	46.8	64.0	33.1	41.5	26.5	29.8	10.7	18.4	27.3	37.0
Mistral-Nemo	12B		51.1	66.0	32.1	40.5	29.0	37.0	15.7	15.0	30.3	36.9
Yi-1.5	9B		23.2	69.5	17.4	45.0	1.2	15.0	2.2	11.9	11.1	35.0
Llama-3.1	8B		66.5	63.0	41.9	34.1	35.5	35.5	14.7	13.8	37.0	33.9
Codestral	22B	Code	43.1	58.0	36.8	37.2	25.0	29.5	14.2	14.0	28.4	32.8
Llama-3	8B		51.6	57.8	22.8	31.2	22.5	25.0	9.3	11.5	24.9	29.7
Mathstral	7B	Math	45.8	54.4	31.1	34.0	11.0	24.0	9.1	11.7	24.0	29.5
Qwen2	7B		15.4	52.6	6.6	33.2	4.2	29.0	2.6	11.4	7.0	29.4
GLM-4	9B		68.0	48.8	46.5	32.1	32.5	22.8	18.2	11.1	39.5	27.2
Aya-23	35B		0.9	46.9	0.5	26.5	0.0	19.5	0.5	10.1	0.5	24.6
Mixtral-8x7B-v0.1	46B	MoE	1.1	42.5	0.4	19.6	0.2	24.2	0.2	13.0	0.5	23.4
DeepSeek-Math	7B	Math	1.4	47.1	0.4	28.0	1.2	12.5	0.5	7.9	0.8	23.3
Mistral-v0.3	7B		48.6	40.8	28.5	24.8	19.5	18.0	12.8	7.6	26.5	21.5
DeepSeek-V2-Lite	16B	MoE	7.6	49.1	4.6	16.9	2.5	15.2	1.0	8.3	3.7	21.5
Llama-2	70B		27.1	45.1	16.8	26.0	2.0	8.0	1.1	6.9	11.6	21.4
WizardLM-2	7B		47.0	42.6	31.2	31.2	9.2	8.5	7.2	4.8	23.4	21.3
WizardMath	7B	Math	22.1	34.2	14.4	24.8	5.8	6.2	3.4	4.6	11.1	17.2
InternLM2-Math-Plus	7B	Math	30.0	30.2	15.0	16.2	10.8	11.2	4.2	3.3	14.2	14.3
StarCoder2	15B	Code	51.0	29.8	32.0	16.8	9.5	5.2	8.4	3.0	25.1	13.4
InternLM2	7B		17.4	24.5	9.4	11.8	9.0	6.8	2.9	4.3	8.9	11.5
Gemma-1	7B		0.5	23.2	0.2	6.0	0.0	7.8	0.2	3.2	0.3	9.5
Llama-2	7B		5.6	20.4	2.4	11.8	1.0	3.2	0.4	2.2	2.3	9.3
WizardCoder	33B	Code	38.4	19.9	17.5	8.4	7.2	3.2	6.1	1.8	17.2	8.2
Aya-23	8B		0.5	14.2	0.1	8.5	0.0	2.8	0.0	2.2	0.2	6.8
DeepSeek-Coder-V1	33B	Code	19.4	16.2	8.5	6.6	3.2	2.2	3.6	1.2	8.7	6.4
Gemma-1	2B		5.9	9.1	2.5	6.0	0.8	2.8	0.2	1.0	2.3	4.5

Table 9: Results of Chain-of-Thought and Program-of-Thought prompting on the *test* set of DocMath-Eval. We use average Accuracy using CoT prompting as the ranking indicator of model performance. For $DM_{CompLong}$, we use the OpenAI Embedding 3 Large retriever to retrieve top-10 evidence as input document. Numbers underscored indicate that models with PoT prompting achieves better results than with CoT prompting.