DocMath-Eval: Evaluating Math Reasoning Capabilities of LLMs in Understanding Financial 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

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 financial documents containing both text and tables. We evaluate a wide spectrum of 27 LLMs, including those specialized in math, coding and finance, with Chain-of-Thought and Program-of-Thought prompting methods. We found that even the current best-performing system (i.e., GPT-4) still significantly lags behind human experts in solving complex numerical reasoning problems grounded in long contexts. We believe DocMath-Eval can be used as a valuable benchmark to evaluate LLMs' capabilities to solve challenging numerical reasoning problems in expert domains.

github.com/yale-nlp/DocMath-Eval

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, 2022, 2023; Touvron et al., 2023), 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., 2021a; Cobbe et al., 2021; Lu et al., 2023b; 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 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 reasoning to complex problem-solving (Chen et al.,

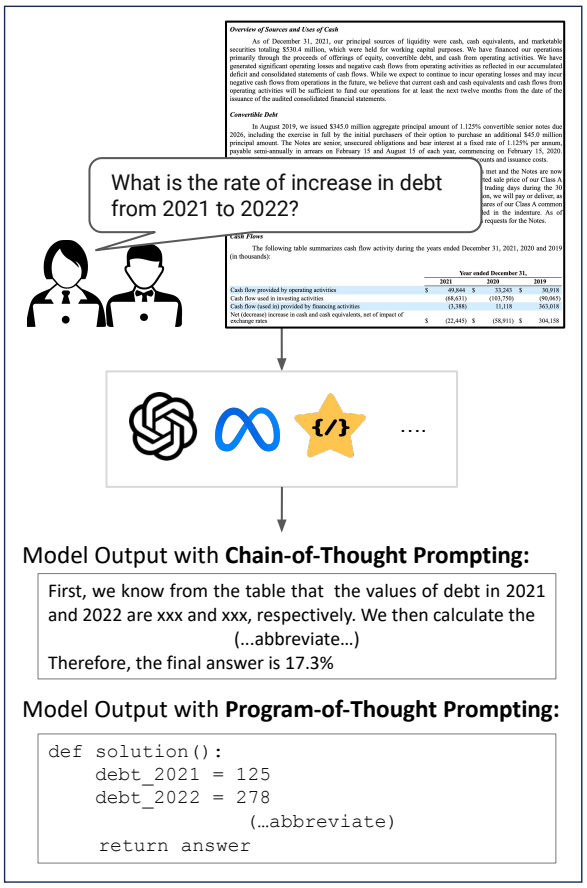

Figure 1: The overview of DocMath-Eval and the prompting methods adopted. DocMath-Eval evaluates the LLMs' performance in the context of understanding and analyzing financial documents containing both text and tables. The models are required to first locate question-relevant data points within lengthy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

[*]Equal Contributions.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16103–16120

2021; Zhu et al., 2021; Zhao et al., 2022; Li et al., 2022). Recognizing this gap, our research focuses on the finance domain (Wu et al., 2023a; Yang et al., 2023b; Callanan et al., 2023). 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 financial 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., 2022), necessitates *complex* numerical reasoning over *short* document with *one* table.

We evaluate a wide spectrum of open- and closed-source LLMs, specifically, 27 models from 17 organizations. This notably includes code-based LLMs (Xu et al., 2023; Luo et al., 2023b; Li et al., 2023a; Tunstall et al., 2023) for enhanced reasoning and programming abilities, as well as LLMs specialized in the finance domain (Xie et al., 2023). Two prompting methods, Chain-of-Thought (CoT) (Wei et al., 2022) and Program-of-Thought (PoT) (Chen et al., 2023a), are adopted for experiments. Our experimental results indicate that while the existing best-performing LLM (i.e., GPT-4) can achieve high performance in a simple setting (i.e., DM$_{\text{SimpShort}}$), it still falls short of human experts in more challenging ones. Specifically, GPT-4 significantly outperforms other open-source LLMs, achieving an accuracy of 41.2% on the most challenging evaluation set (i.e., DM$_{\text{CompLong}}$) when applying PoT 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 highlights the need for further advancements in adapting LLMs' numerical reasoning capabilities for practical application in real-world expert 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 financial 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 coding and finance. We also incorporate different prompting methods (e.g., 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 (e.g., 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

2.1 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., 2023c; Hendrycks et al., 2021b), 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 mixed-content financial documents through numerical reasoning.

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., 2022)	expert annotated from scratch
Question Length	19 / 20.1	23 / 24.0	30 / 30.2	35 / 38.3
# Sentences in Text	14 / 16.7	64 / 66.9	6 / 7.8	746 / 1,035.6
# Words in Text	500 / 505.1	2,247 / 2,352.6	253 / 310.8	24,736 / 35,065.6
# Table	1 / 1.0	4 / 4.0	1 / 1.0	48 / 78.2
# Rows per Table	7 / 8.0	9 / 11.6	8 / 9.3	4 / 7.9
# Columns per Table	4 / 4.0	4 / 4.4	4 / 4.0	3 / 3.7
# Text Evidence	1 / 1.3	1 / 1.0	2 / 2.3	2 / 1.9
# Table Evidence	1 / 1.0	1 / 1.0	1 / 1.0	1.3 / 1.2
% Questions w. Table Evidence	92.9%	86.4%	97.8%	76.3%
# Math Operations in Python Solution	2 / 2.1	2 / 2.4	2 / 2.2	4 / 4.9
# Code Lines in Python Solution	5 / 5.3	6 / 6.0	5 / 5.3	8 / 8.3
# Comment Lines in Python Solution	2 / 2.0	2 / 2.0	2 / 2.0	5 / 5.5
Dataset Size	1,459	793	1,621	2,101

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.

2.2 Numerical Reasoning over Documents

Numerical reasoning over documents requires models to 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., 2023b) 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., 2023a). Numerous datasets focusing on numerical reasoning within the financial domain 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., 2022) 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 and longer texts. However, as illustrated in Table 1, these four datasets focus on less challenging scenarios, where either simple numerical reasoning (e.g., calculating average or increasing rate of two metrics) is sufficient, or the input context is short (i.e., one-page document segment with only one table). 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, which is crucial in the era of LLMs.

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. Finally, we present human-level performance on each evaluation set in DOCMATH-EVAL. Table 1 describes the basic statistics of four developed evaluation sets. DOCMATH-EVAL contains a total of 5,974 questions with high-quality annotations, featuring varying difficulty levels in numerical reasoning and document understanding.

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:

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

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

In the subtasks of $\text{DM}_{\text{SimpLong}}$ and $\text{DM}_{\text{CompLong}}$, due to the length of the document exceeding the maximum input length of LLMs, we first apply retrievers to retrieve the top-n most relevant textual and tabular evidence to form a partial document, while maintaining the original relative order of the evidence within the partial document. This textual content and structured tables in the partial document are then input into the LLMs.

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., 2022) 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, solutions in text format often lack the precision and unambiguity necessary for computational problem-solving; and solutions in mathematical equations or math programs are less descriptive, with the semantic meaning of each numeric value in the equations sometimes being unclear.

To overcome the aforementioned limitations, in DocMath-Eval, we represent solutions using Python programs, as this format combines the explicitness of code execution with the descriptive power of annotated explanation (in the format of Python comments). Such a unified Python program format supports a standardized and effective evaluation framework for LLM assessment. Specifically, annotators are required to first define variables at the beginning of the Python function, starting with "`def solution():`". These variables correspond to the key elements or quantities mentioned in the question or question-relevant content in the documents. Annotators are instructed to assign meaningful names that clearly represent each element. They then write a sequence of Python statements that logically solve the problem, step by step. Additionally, annotators receive a bonus for writing detailed comments, thereby enhancing the code's readability and understandability. To ensure the accuracy and functionality of the solutions, our annotation interface automatically executes the Python function. This execution checks that the return type of the answer is either a float or an int and verifies that there are no execution errors.

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., 2022) 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, existing finance-relevant QA benchmarks (Zhu et al., 2021; Chen et al., 2021; Zhao et al., 2022; Li et al., 2022) 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, $\mathbf{DM}_{\mathbf{CompLong}}$, from scratch. This set focuses on settings that more closely align with real-world problem-solving scenarios, where models are required to perform complex numerical reasoning over long financial documents for problem solving. The annotation process for $\text{DM}_{\text{CompLong}}$ 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 tables as evidence supporting it. Following Chen et al. (2021), we use the paragraph index p_i to mark question-relevant textual evidence in the p_ith paragraph; and (t_i, r_j) to mark relevant tabular evidence in the r_jth row of the t_ith column. The same method is applied in the dataset re-annotation process, as detailed in Section 3.3. 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 complex 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. We present the human evaluation scores and inter-evaluator agreements for a subset of 200 sampled examples. Table 2 demonstrate that DOCMATH-EVAL exhibits superior annotation quality and a high degree of inter-annotator agreement.

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
Comment Comprehensiveness	87.4

Table 2: Human evaluation over 200 samples of DOCMATH-EVAL. Three internal evaluators were asked to rate the samples on a scale of 1 to 5. We report 1) percent of samples that have an average score ≥ 4 to indicate the annotation quality of DOCMATH-EVAL

3.5 Expert-level Performance Evaluation

To provide a rough but informative estimate of the performance of domain-experts on each of DOCMATH-EVAL sets, we invite two professionals with Chartered Financial Analyst licenses for 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%).

4 Experiment Setup

4.1 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

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

this end, we evaluate a wide range of models:

- **General**: GPT-3.5&4 (OpenAI, 2022, 2023), Gemini (Google, 2023) Llama-2 (Touvron et al., 2023), Mistral (Jiang et al., 2023), MPT (Team, 2023), WizardLM (Luo et al., 2023b), Yi (01.AI, 2023), Baichuan (Yang et al., 2023a), Aquila (BAAI, 2023), Qwen (Bai et al., 2023), Vicuna (Zheng et al., 2023) Phi-1.5&2 (Li et al., 2023b), and DeepSeek (DeepSeek, 2023).
- **Math-specific**: WizardMath (Luo et al., 2023a).
- **Finance-specific**: FinMA (Xie et al., 2023).
- **Code-based**: StarCoder (Li et al., 2023a), StarChat (Tunstall et al., 2023), CodeLlama (Rozière et al., 2023), WizardCoder (Luo et al., 2023b), and Lemur (Xu et al., 2023).
- **Mixture of Experts (MoE)**: Mixtral of experts (Mistral.AI, 2023).

By default, we use `chat` or `instruct` versions for each model, when available, otherwise, we used their `base` version. Additionally, we select the most recent, largest, and best-performing checkpoint available as of paper submission (i.e, December 13th, 2023). All the model weights of evaluated open-sourced LLMs can be found at HuggingFace Model Hub[3].

4.2 Retrieval

We experiment with both dense and sparse retrieval models. For dense retrievers, we specifically experiment with OpenAI Ada Embeddings[4] and Contriever (Izacard et al., 2022), while for sparse retrievers, we use BM25 (Robertson et al., 1995). These retrievers are employed in the subtasks of $\text{DM}_{\text{SimpLong}}$ and $\text{DM}_{\text{CompLong}}$ to extract the top-n most related textual and tabular evidence from the source document. The extracted evidence is then provided within the LLM input context to answer the given question.

4.3 Prompting Methods

Following recent LLM reasoning benchmark works (Lu et al., 2023a; Chen et al., 2023c), we evaluate two established prompting methods, with examples of prompt illustrated in Figure 2.

[3] https://huggingface.co/models

[4] platform.openai.com/docs/guides/embeddings

Chain-of-Thoughts Prompting Method:

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

[user input]
Document:
{document}

Question: {question}

Using the information from the document, let's think step by step to answer the question.
```

Program-of-Thought Prompting Method:

```
[system prompt]
You are a financial expert, you are supposed to generate a Python program to answer the given question. The returned value of the program is supposed to be the answer.

[user input]
Document:
{document}

Question: {question}

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

Figure 2: Examples of CoT and PoT prompts we used.

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

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

4.4 Implementation Details

The implementation details, including 1) LLM parameter setting, 2) tabular data serialization, and 3) final answer extraction and evaluation are discussed in Appendix A.1.

5 Experimental Results

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.

5.1 Main Results

We draw the following findings and conclusions based on the results illustrated in Table 3.
```

Model	Size	Backbone	Notes	$DM_{SimpShort}$		$DM_{SimpLong}$		$DM_{CompShort}$		$DM_{CompLong}$		Avg. Acc	
				CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	**CoT**	PoT
Human Expert				91.0		87.0		84.0		76.0			
GPT-4-1106	–	–	–	**89.3**	**87.4**	**62.6**	**63.2**	**80.0**	74.4	**38.8**	**41.2**	**67.7**	**66.5**
GPT-4-0613	–	–	–	87.9	84.9	56.5	59.3	78.0	**74.5**	38.5	38.8	65.2	64.4
GPT-3.5-1106	–	–	–	80.0	80.8	45.8	45.9	50.3	57.6	23.7	27.0	49.9	52.8
deepseek	67B	–	–	**76.5**	**68.5**	**40.7**	**37.0**	**52.6**	**50.5**	20.0	**20.5**	**47.5**	**44.1**
Mixtral	8x7B	–	MoE	76.2	45.6	39.5	25.5	48.2	27.0	**21.6**	12.0	46.4	27.5
GPT-3.5-0613	–	–	–	70.1	77.6	44.1	44.5	45.0	48.1	21.7	22.7	45.2	48.2
Gemini-Pro	–	–	–	76.7	77.7	34.7	46.3	39.4	52.1	12.7	27.1	40.9	50.8
WizardLM	70B	Llama-2	–	66.8	6.1	34.8	1.8	36.6	4.1	16.5	1.3	38.7	3.3
Llama-2	70B	–	–	53.4	55.7	32.9	33.1	33.6	44.7	13.3	10.8	33.3	36.1
Lemur	70B	Llama-2	code-based	50.9	55.4	27.1	27.6	33.9	45.9	12.2	13.3	31.1	35.5
Llama-2	13B	–	–	51.7	44.9	24.6	20.9	30.4	38.8	10.6	7.8	29.3	28.1
Llama-2	7B	–	–	35.0	29.8	19.4	10.5	22.0	20.2	7.9	4.1	21.1	16.2
CodeLlama	34B	Llama-2	code-based	41.6	49.4	5.3	1.2	26.0	32.3	2.8	1.0	18.9	21.0
Baichuan2	13B	Llama-2	–	30.5	18.3	15.8	7.6	15.0	7.8	7.2	2.8	17.1	9.1
Yi	34B	–	–	34.3	2.3	13.1	1.1	13.3	1.1	7.7	0.6	17.1	1.3
Qwen	14B	–	–	28.7	1.0	17.2	0.3	12.0	0.5	4.8	0.0	15.6	0.4
WizardMath	70B	Llama-2	math	6.0	0.2	33.8	1.1	4.9	0.1	16.3	0.0	15.2	0.3
Mistral	7B	Llama-2	–	24.6	15.7	9.5	5.7	14.4	10.7	4.0	1.7	13.1	8.5
WizardCoder	34B	Llama-2	code-based	25.6	24.8	9.2	12.7	11.7	12.1	3.8	3.7	12.6	13.4
MPT	30B	–	–	22.2	0.0	9.8	0.0	11.5	0.0	5.3	0.0	12.2	0.0
AquilaChat2	34B	–	–	10.4	1.9	6.3	0.6	5.4	1.3	3.0	0.3	6.3	1.1
phi-2	2.7B	–	–	7.3	1.6	3.2	0.9	4.5	0.7	1.8	0.5	4.2	0.9
Vicuna	33B	Llama-1	–	1.9	0.1	9.1	0.1	1.5	0.0	3.7	0.0	4.0	0.1
Pixiu (FinMA)	30B	Llama-1	finance	2.1	0.0	6.4	0.0	1.7	0.0	2.7	0.0	3.2	0.0
StarChat-beta	15.5B	StarCoder	code-based	2.2	9.3	2.0	1.5	2.3	5.2	0.8	1.8	1.8	4.5
phi-1.5	1.3B	–	–	2.1	0.3	1.1	0.0	2.2	0.3	1.8	0.2	1.8	0.2
StarCoder	15.5B	–	code-based	2.5	1.0	0.9	0.2	1.7	0.5	0.8	0.1	1.5	0.5

Table 3: Results for CoT and PoT prompting on DOCMATH-EVAL. For $DM_{SimpLong}$ and $DM_{CompLong}$, we use the Ada Embedding-based retriever to retrieve top-10 evidence as input document.

GPT-* Significantly Outperforms Other Open-source LLMs Proprietary models demonstrate the best performance on each evaluation set of DOCMATH-EVAL. Notably, GPT-4 significantly outperforms other LLMs, achieving accuracies of 89.3% on $DM_{SimpShort}$ and 80.0 % on $DM_{CompShort}$, respectively, when utilizing CoT prompting. In contrast, open-source LLMs lag considerably behind, indicating a substantial need for future efforts in model development to bridge the performance gap.

Significant Performance Gap to Human Expert in the Complex Settings While the current best-performing LLM (i.e., GPT-4) achieves performance comparable to human experts in simple problem settings, we find significant performance gaps in more challenging settings. Specifically, GPT-4 achieves an accuracy of **41.2%** on $DM_{CompLong}$ with PoT, which is far behind the human expert performance of **76.0%**. This underscores the need for ongoing development in the field of LLMs, particularly in *complex problem-solving* within expert domains.

Llama-2 Achieves Robust Performance We observe that the Llama-2 models generally outperform their variants in DOCMATH-EVAL task. For instance, Llama-2-7B shows superior performance compared to Mistral-7B across all evaluation sets. Furthermore, despite its specialization in the finance domain, PiXiu does not demonstrate competitive performance in DOCMATH-EVAL. These findings suggest that Llama-2 is more versatile and robust for our specific task, indicating the needs for future research on exploring the development of LLMs in specialized domains.

5.2 Program-of-Thought Analysis

We observe that the PoT prompting method consistently improves performance over the CoT method in GPT-* models and code-based LLMs. In contrast, the performance of several general LLMs, such as Mistral and WizardLM degrades with PoT prompting. To better analyze the reasons for these differing performance outcomes, we examine the execution rate of each LLM under PoT prompting, measuring how many of the generated Python programs are executable. Figure 3 in Appendix illustrates the relationship between execution rate and accuracy across different models. It demonstrates that the degraded performance when applying PoT

prompting is attributable to the low execution rate. For instance, although WizardLM achieves competitive performance with CoT, it struggles to consistently generate executable Python solutions, leading to lower accuracy with the PoT prompting.

5.3 LLM Document Understanding Analysis

We develop a metric, **DU**-F1, designed to gauge the nuanced **D**ocument **U**nderstanding capabilities of LLMs. Specifically, we apply rule-based methods to extract all explicit values (i.e., those not derived from the computation of other values) present within the CoT or PoT output. Subsequently, a comparative analysis is conducted by juxtaposing these extracted direct values with those obtained from the ground truth Python-format solution. The evaluation criterion employs the F1-score, quantifying the LLMs' efficacy in evidence extraction. The relationships between DU-F1 and accuracy across different LLMs in CoT and PoT prompting are illustrated in Figure 4 and Figure 5 in Appendix respectively. The final accuracy of LLMs correlates with DU-F1, indicating that enhancing the document understanding abilities of LLMs can improve their overall performance on DocMath-Eval.

5.4 Error Analysis

Table 3 reveals the notable superiority of GPT-* models over other LLMs. Despite this, the accuracy falls short of that achieved by human experts. To further understand the strengths and weaknesses of GPT-*, we undertook an extensive analysis of errors. This analysis centered on 100 randomly selected examples from the dataset where GPT-3.5-0613 exhibited failure. We pinpoint four common errors prone to occurring in current LLMs. A detailed explanation for each error type is provided in Table 6 in the Appendix. Our error analysis reveals that LLMs are likely to make mistakes in calculations. To disentangle the computational abilities from the final accuracy, we applied an external calculator (Inaba et al., 2023) for CoT output to do computation. Figure 6 illustrates the calibrated results of LLMs with an external calculator.

5.5 Analysis of Evidence Extraction

We analyze the impact of retrieval performance on the final accuracy of LLMs in long document settings. Initially, we evaluate the performance of the retriever model used. As illustrated in Table 4, the Ada embedding achieves the best performance. Specifically, it attains a R@10 of 83% and 69.2% on $\text{DM}_{\text{CompShort}}$ and $\text{DM}_{\text{CompLong}}$, respectively. As demonstrated in Table 5, improved performance of the retriever module consistently enhances the final accuracy of LLMs in our task. This finding underscores the necessity for future work in developing more advanced information retrieval techniques.

Evaluation Set	Retriever	R@5	R@10
$\text{DM}_{\text{CompShort}}$	BM25	22.7	36.8
	Contriever	57.5	66.6
	Ada Embedding	74.0	83.0
$\text{DM}_{\text{CompLong}}$	Contriever	31.7	45.8
	BM25	41.7	52.3
	Ada Embedding	55.6	69.2

Table 4: Results of retrieving top-n question-relevant evidence from the source documents.

Model	top-n	Retriever	Acc
Llama-2-70B	5	BM25	6.6
	5	Contriever	6.8
	5	Ada Embedding	10.7
	10	Contriever	6.5
	10	BM25	8.9
	10	Ada Embedding	13.3
	–	Oracle	17.0
GPT-3.5	5	Contriever	14.6
	5	BM25	15.0
	5	Ada Embedding	19.0
	10	Contriever	14.9
	10	BM25	17.1
	10	Ada Embedding	21.7
	–	Oracle	29.4

Table 5: Results of the CoT prompting approach under various retrieval settings on $\text{DM}_{\text{CompLong}}$. A correlation is observed between LLM performance and question-relevance of the retrieved evidence.

6 Conclusion

This paper presents DocMath-Eval, a comprehensive benchmark tailored to assess LLMs' capabilities in numerical reasoning and problem-solving, particularly in the realm of financial document analysis. Our comprehensive experiments over 27 LLMs with CoT and PoT prompting methods demonstrate that although the top-performing current model excels in simple problem settings, it falls short of human expert performance in problems requiring numerical reasoning over long contexts. We contend that DocMath-Eval serves as a valuable benchmark for future work on evaluating LLMs' proficiency in tackling complex numerical reasoning tasks within expert domains.

Ethical Consideration

For the DOCMATH-EVAL annotation, we hired 7 graduate students (5 females and 2 males) majoring in finance-related disciplines, all of whom passed our quality exams in Python and finance. Before beginning the official annotation process, each annotator received a two-hour training session to familiarize themselves with the annotation requirements and learn how to use the annotation interface. For $\text{DM}_{\text{CompLong}}$ annotation from scratch, we consider the following as a unit task: (1) create one math reasoning question and annotate corresponding supporting evidence, (2) compose a Python-format solution for a given question, and (3) validate two annotated examples. We pay approximately \$2.5 for each unit task. On average, an annotator can complete 5 unit tasks per hour after training and practice. For dataset re-annotation from existing datasets (i.e., $\text{DM}_{\text{SimpShort}}$, $\text{DM}_{\text{SimpLong}}$, $\text{DM}_{\text{CompLong}}$), we consider the following as a unit task: (1) validate two questions and their original solutions and (2) convert one original solution to Python format and annotate corresponding supporting evidence. We pay around \$1 for each unit task. On average, an annotator can complete 12 unit tasks per hour after training and practice. The hourly rates are in the range of \$10 to \$15, depending on the different working speeds, which is above the local average wage for similar jobs. We recommend that annotators spend a maximum of 4 hours per day to reduce pressure and maintain a comfortable pace. In total, the approximate working hours to construct DOCMATH-EVAL is 700 hours. The whole annotation work lasted three weeks.

Limitations

In this work, we propose DOCMATH-EVAL and conduct comprehensive analysis of different LLMs' capabilities in solving knowledge-intensive math reasoning problems in finance domains. However, there are still some limitations: First, our method for extracting final answer from model output (Appendix A.1) is still not perfect. In some cases, this methods fails to locate the answer, leading to the reported accuracy being an approximate lower bound. Moreover, among recently released finance-specific LLMs (Wu et al., 2023b; Yang et al., 2023b; Xie et al., 2023), we only evaluate FinMA, as it is the only work with a checkpoint available at HuggingFace and compatible with the vllm framework. Due to computational resource constraints, we do not tune LLMs on a large-scale finance-domain data ourselves. However, we believe that training on finance data can help improve LLMs' capabilities in solving problems in DOCMATH-EVAL.

Acknowledgement

We are grateful for the compute support provided by Microsoft Research's Accelerate Foundation Models Research (AFMR) program. We would also like to thank the anonymous reviewers and area chairs for constructive discussions and feedback.

References

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

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

BAAI. 2023. Wudao: Open-source llm release.

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

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.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023c. Theoremqa: A theorem-driven question answering dataset.

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

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

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

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.

Google. 2023. Gemini.

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

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

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.

Tatsuro Inaba, Hirokazu Kiyomaru, Fei Cheng, and Sadao Kurohashi. 2023. Multitool-cot: Gpt-3 can use multiple external tools with chain of thought prompting.

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*.

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

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

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*.

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

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro

von Werra, and Harm de Vries. 2023a. Starcoder: may the source be with you!

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

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

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023a. Mathvista: Evaluating math reasoning in visual contexts with gpt-4v, bard, and other large multimodal models.

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

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

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

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

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

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

OpenAI. 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.

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

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

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

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

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 Kam-

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

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

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

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023b. Fingpt: Open-source financial large language models. *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. 2023a. Knowledgemath: Knowledge-intensive math word problem solving 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. 2023b. QTSumm: Query-focused summarization over tabular data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1157–1172, Singapore. Association for Computational Linguistics.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

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 Experiment

A.1 Implementation Details

LLM Experiment The experiments for open-sourced LLMs were conducted using the `vLLM` framework (Kwon et al., 2023). For all the experiments, we set temperature as 1.0, Top P as 1.0, and maximum output length as 512.

Input Tabular Data Serialization Building on previous work that evaluated LLMs on table-relevant tasks (Chen, 2023; Zhao et al., 2023b,c), 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 discovered that GPT-* and llama-2 can effectively understand such table representations. Nevertheless, we believe future research could explore more effective methods for encoding tabular data.

Final Answer Extraction For LLM with CoT prompting, we adopt the answer extraction pipeline from Chen et al. (2023b) to identify the final answer from the model's output. For LLM with PoT prompting, we first develop a heuristic method to extract the generated python solution from the model's output. If this python solution is executable, we execute it to obtain the final answer.

Once we obtain the final answer from model's output, we compare it with the ground-truth answer for accuracy measurement.

Error Type	Representative Question	Explanation
Calculation Error (31/100)		The evidence retrieval is accurate, but there are errors in the calculation formulas and/or the final results during the generation.
Table Misunderstanding (26/100)		Model faces challenges in comprehending and parsing cell values, particularly in complex tables that lack perfect alignment in the input. This difficulty arises as we serialize tabular data.
Incomplete or Incorrect Evidence Retrieval (23/100)	What is the total value of common stock issued by the Registrant in USD?	The challenge lies in locating accurate evidence, stemming from the ambiguity of the questions. The values required for intermediate reasoning steps are not explicitly stated, leading to difficulties for the retriever in identifying the correct evidence.
Exceeding Context Length (11/100)		The evidence paragraphs surpass the context length limit.
Other errors (9/100)		

Table 6: Case study on DOCMATH-EVAL's failure cases.

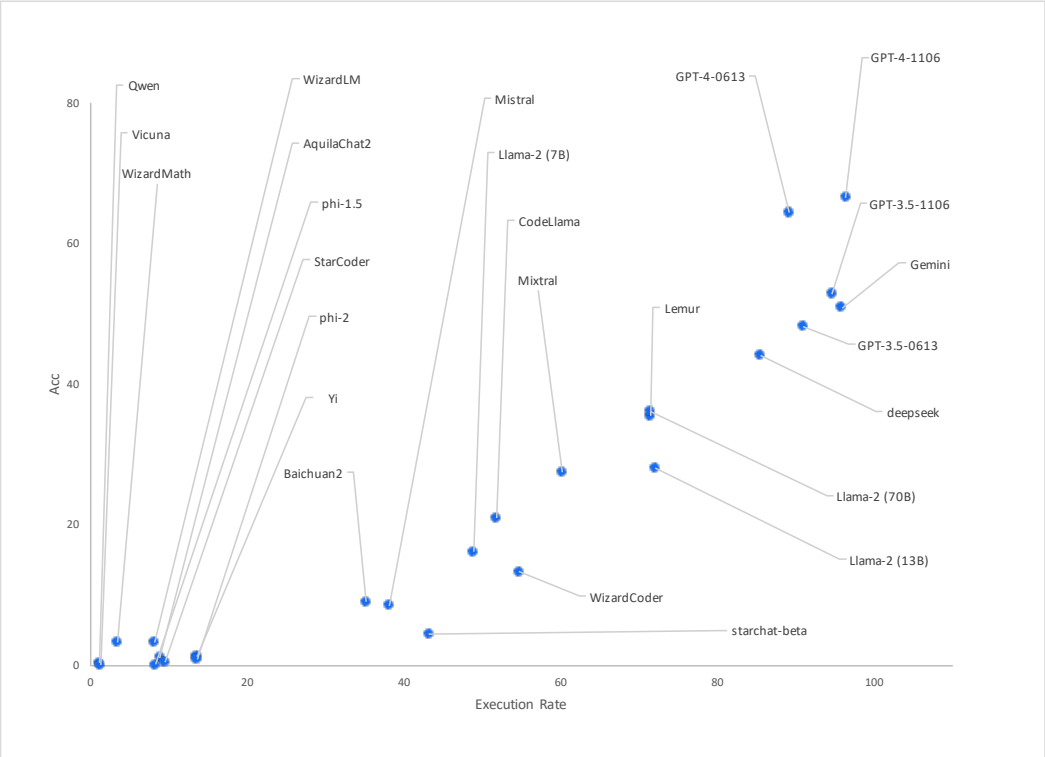

Figure 3: Execution rate and accuracy results for various LLMs using PoT prompting on DocMath-Eval.

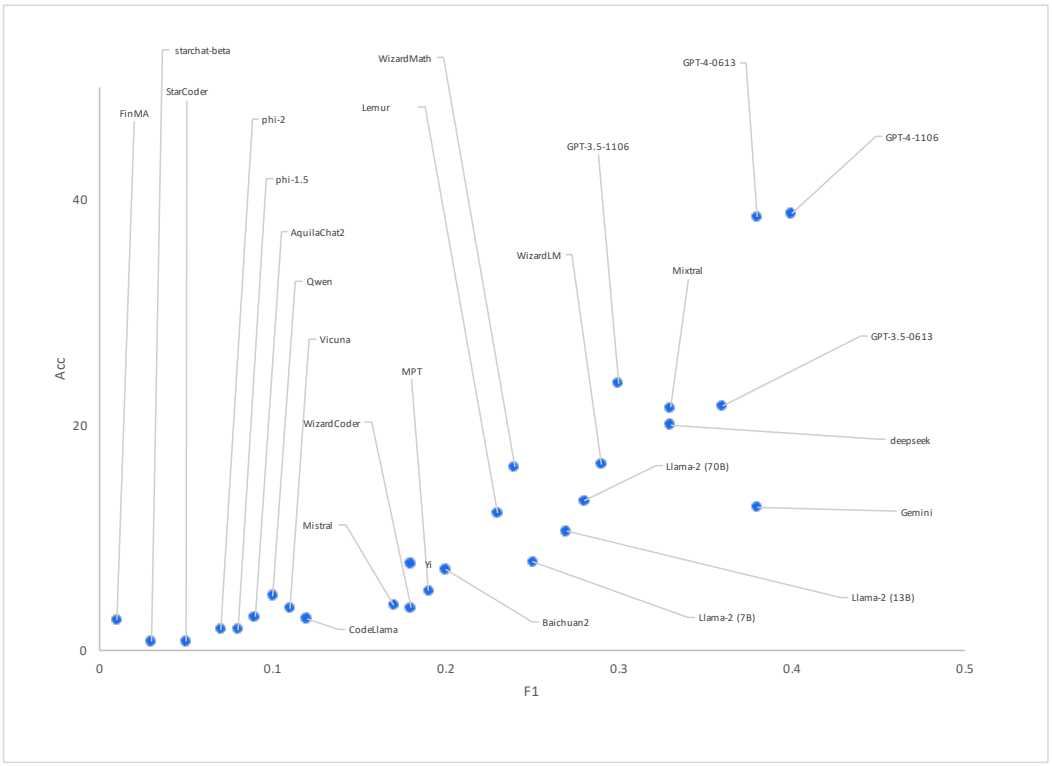

Figure 4: F1-score in retrieval evaluation and accuracy results for various LLMs using CoT prompting on $\text{DM}_{\text{CompLong}}$

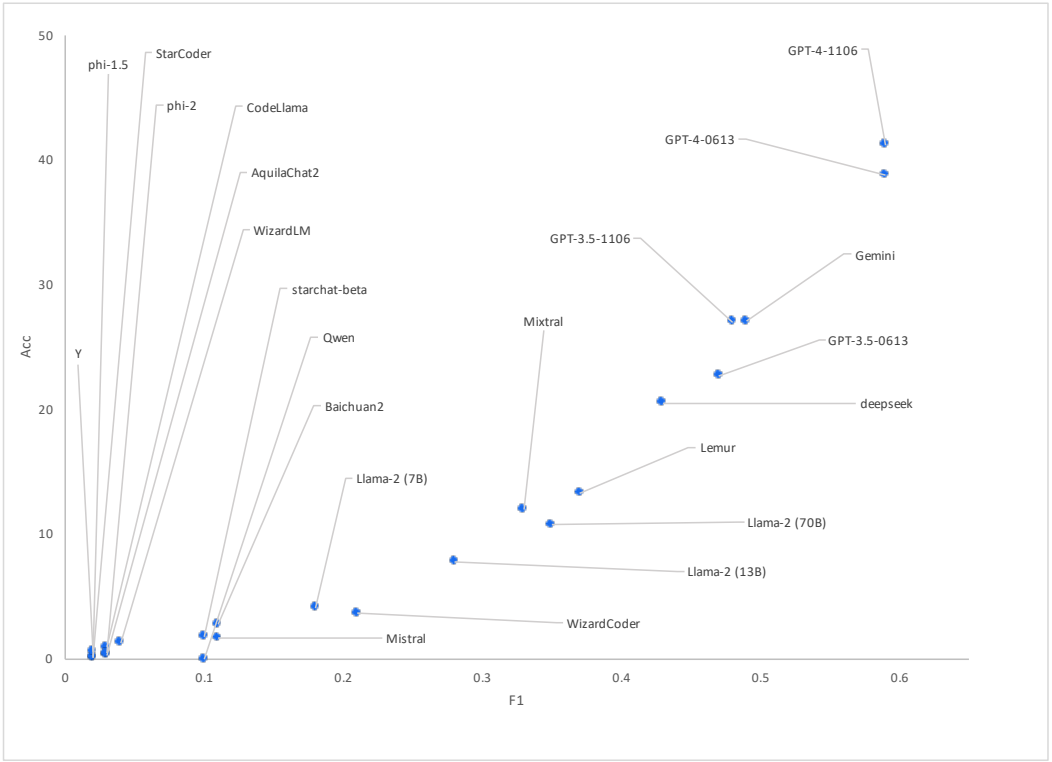

Figure 5: F1-score in retrieval evaluation and accuracy results for various LLMs using PoT prompting on $DM_{CompLong}$

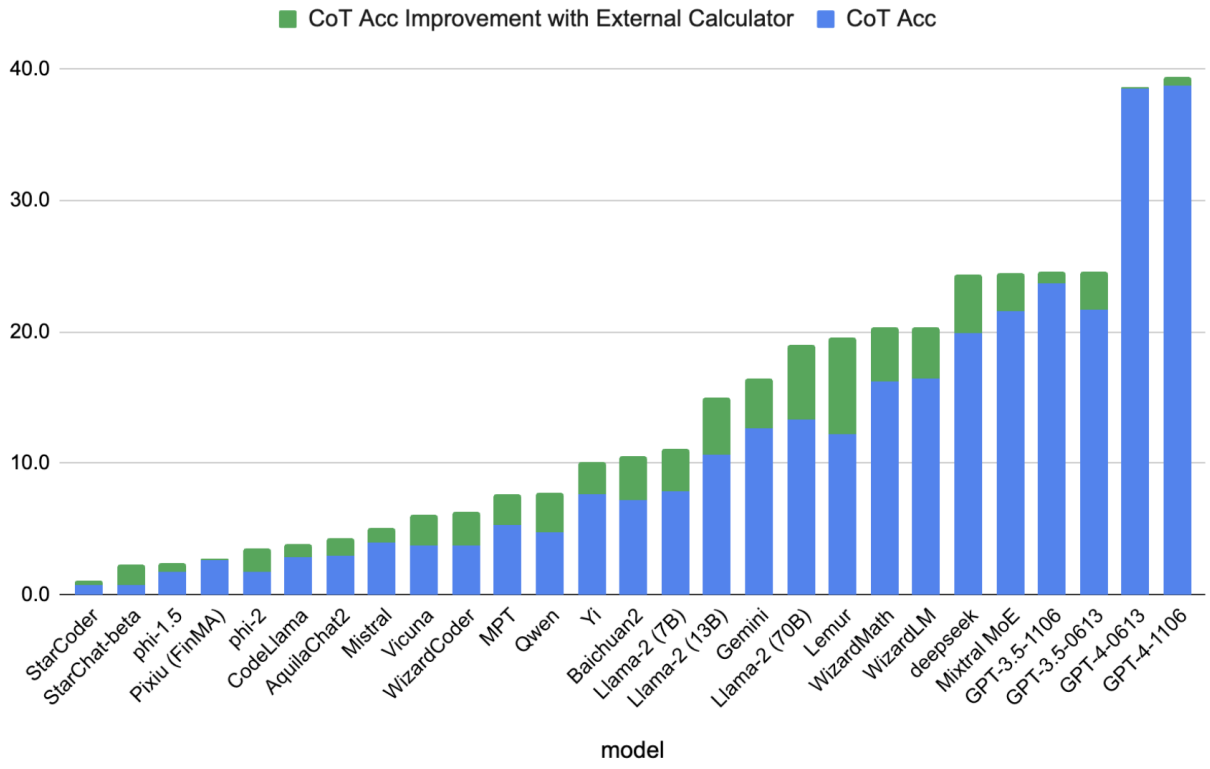

Figure 6: Accuracy Improvement with External Calculator in CoT