Program of Thoughts for Financial Reasoning: Leveraging Dynamic In-Context Examples and Generative Retrieval

Subhendu Khatuya, Shashwat Naidu, Pawan Goyal, Niloy Ganguly
Indian Institute of Technology Kharagpur, India
subha.cse143@gmail.com, shashwatnaidu07@gmail.com,
pawang@cse.iitkgp.ac.in, niloy@cse.iitkgp.ac.in

Abstract

Despite continuous advancements in the capabilities of large language models (LLMs), numerical reasoning remains a challenging area. Techniques like chain-of-thought prompting, tree-of-thought prompting, and program-of-thought prompting guide LLMs through intermediate reasoning steps. Although in-context learning with few-shot prompting has improved performance, LLMs still lag behind state-of-the-art models on financial numerical reasoning datasets such as FinQA and ConvFinQA. In this work, we introduce FINDER, a novel two-step framework, to enhance LLM's capabilities in financial numerical reasoning. The first step utilizes a generative retriever to extract relevant facts from unstructured data, including both text and tables. This is followed by context-aware Program of Thought prompting with dynamic selection of in-context examples. Our model FINDER achieves a new state-of-the-art performance on both the FinQA and ConvFinQA datasets, surpassing previous benchmarks with execution accuracy improvements of **5.98%** and **4.05%**, respectively.

1 Introduction

Numerical reasoning, particularly within financial domains, remains a significant challenge in artificial intelligence (AI). Unlike traditional question-answering tasks (Rajpurkar et al., 2018; Yang et al., 2018), it requires not only the extraction of relevant information from diverse sources, such as tables and unstructured text, but also the construction of coherent reasoning paths to integrate and process this information. Towards this effort, datasets such as FinQA (Chen et al., 2021b) and ConvFinQA (Chen et al., 2022) have been developed to benchmark deep learning models for such numerical reasoning tasks in the financial domain.

Despite advances in task-specific models, large language models (LLMs) still struggle with numerical reasoning (Huang and Chang, 2022; Satpute et al., 2024; Zhao et al., 2024), as it requires multi-step problem-solving including fact extraction, logical inference, and mathematical computation. Even minor errors in intermediate steps can lead to incorrect solutions, making numerical tasks particularly challenging for LLMs.

A common approach to tackle numerical reasoning problem is the retriever-generator question-answering framework, initially proposed by Chen et al. (2021b). Subsequent models leveraging pre-trained language models have shown significant performance gains on such tasks (Wang et al., 2022c; Zhang et al., 2022; Wang et al., 2022a). The current state-of-the-art is the APOLLO (Sun et al., 2024b) model, a retriever-generator framework that employs a number-aware retriever and a fine-tuned BERT-based encoder-decoder generator to generate executable programs, achieving high execution accuracy.

In parallel, an emerging line of work adopts the Program-of-Thoughts (PoT) paradigm (Chen et al., 2023) for program generation, which utilizes decoder-only large language models to express the reasoning process as executable programs, disentangling computation from reasoning.These methods typically rely on in-context learning (ICL) rather than fine-tuning, offering greater flexibility and improved generalization capabilities. PoT-based approaches are more adaptable and better leverage the capabilities of modern LLMs.

In this work, we bridge these two directions by proposing a novel retriever-generator framework FINDER that replaces the retriever with an instruction-tuned generative model and the encoder-decoder based program generator with a decoder-only LLM under the PoT paradigm with some key modifications. PoT prompting (Chen et al., 2023), when applied to financial numerical reasoning tasks, typically includes the entire textual and tabular context in the prompt. This often leads to grounding errors, as the model struggles to accu-

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 31006–31018

rately identify and extract the relevant numerical information. To address this, we instruction-tune FLAN-T5 (Chung et al., 2022) using LoRA (Hu et al., 2021) to enable accurate instance-specific relevant facts extraction.

Additionally, PoT relies on static in-context examples, which may not generalize well across diverse problem instances. To address this limitation, we make use of dynamic selection of in-context examples. We enhance an existing gradient-based method, PromptPG (Lu et al., 2023) by refining both the candidate selection strategy and the reward evaluation mechanism which reduces the chance of incorrect penalties. We ensure candidate diversity (Rubin et al., 2022), by identifying a fixed-size subset of training samples from a larger pool. This subset is carefully constructed using clustering techniques to include representative questions from a range of themes (e.g., growth rates, amortization). In addition to this, we place higher-ranking examples closer to the query to optimize contextual relevance. Finally, we combine the dynamically selected diverse ordered in-context examples with the retrieved facts and format them using PoT-based prompting to get the final answer of the question.

FINDER[1] surpasses existing LLM-based approaches with the PoT paradigm, achieving a **8.56%** improvement on FinQA and a **9.60%** improvement on ConvFinQA, compared to the previous best. It sets a new SOTA with execution accuracies of **75.32%** on FinQA and **81.95%** on ConvFinQA, improving upon the current SOTA APOLLO (Sun et al., 2024b) by **5.98%** and **4.05%**, respectively.

2 Related Works

Numerical reasoning in math word problems has long been a challenging task, addressed in several works (Huang et al., 2024; Sun et al., 2024a; Liu et al., 2020). Several benchmark datasets have been developed to advance research in this area, including MathQA (Amini et al., 2019) and MaWPS (Koncel-Kedziorski et al., 2016). Additionally, Lu et al. (2023) introduced the Tabular Math Word Problems (TABMWP) dataset, which requires mathematical reasoning over both textual and tabular data. Math word problems (MWP) are often presented in a structured and consistent format, prompting some prior research to use template-based approaches (Wang et al., 2019) or tree-based methods (Jie et al., 2022; Li et al., 2023) to tackle these problems more effectively. Beyond MWPs, long-form numerical reasoning is addressed by several state-of-the-art methods.

FinQA (Chen et al., 2021b) and ConvFinQA (Chen et al., 2022) are benchmark datasets for long-form numerical reasoning over financial reports. Various approaches have been proposed, including fine-tuning and prompting-based methods. Wang et al. (2022b) pre-trained DeBERTa (He et al., 2023) on financial data. Ant Risk AI (Zhang et al., 2022) used an ensemble of specialized models, while ELASTIC (Wang et al., 2022a) introduced a cell retriever for extracting relevant gold cells. TabT5 (Andrejczuk et al., 2022) leveraged a T5 model pretrained on Wikipedia tables for reasoning tasks. LLMs enhance long-form reasoning, with Chain of Thought (CoT) (Wei et al., 2022) and Program of Thought (PoT) (Chen et al., 2023) prompting improving reasoning and prompt efficiency. (Lim et al., 2024) enhanced argument recognition in program generation using an argument aggregator. CBR-Ren (Feng et al., 2024a) combines LLMs with case-based reasoning to enhance retriever-generator models by improving critical fact retrieval.

3 Methodology

Our proposed framework for financial numerical reasoning task is divided into two phases, a generative relevant fact retriever phase, and a target answer computation phase as depicted in Figure 1.

3.1 Task Formulation

In this task, the input consists of a structured table T containing rows and columns of data, textual information, and a natural language question Q. The objective is to generate an output Y that correctly answers the question by leveraging the information encoded within the table and the textual content. Given the complexity of this task, we break it down into two primary components.

Relevant Fact Retriever: This involves identifying the relevant facts from the input data.

Target Answer Computation: This involves developing and executing a program using the retrieved facts, in-context examples and the given question to derive the final answer.

[1]Data and code are available at https://github.com/subhendukhatuya/FINDER_POT_Financial_Numeric_Reasoning.git

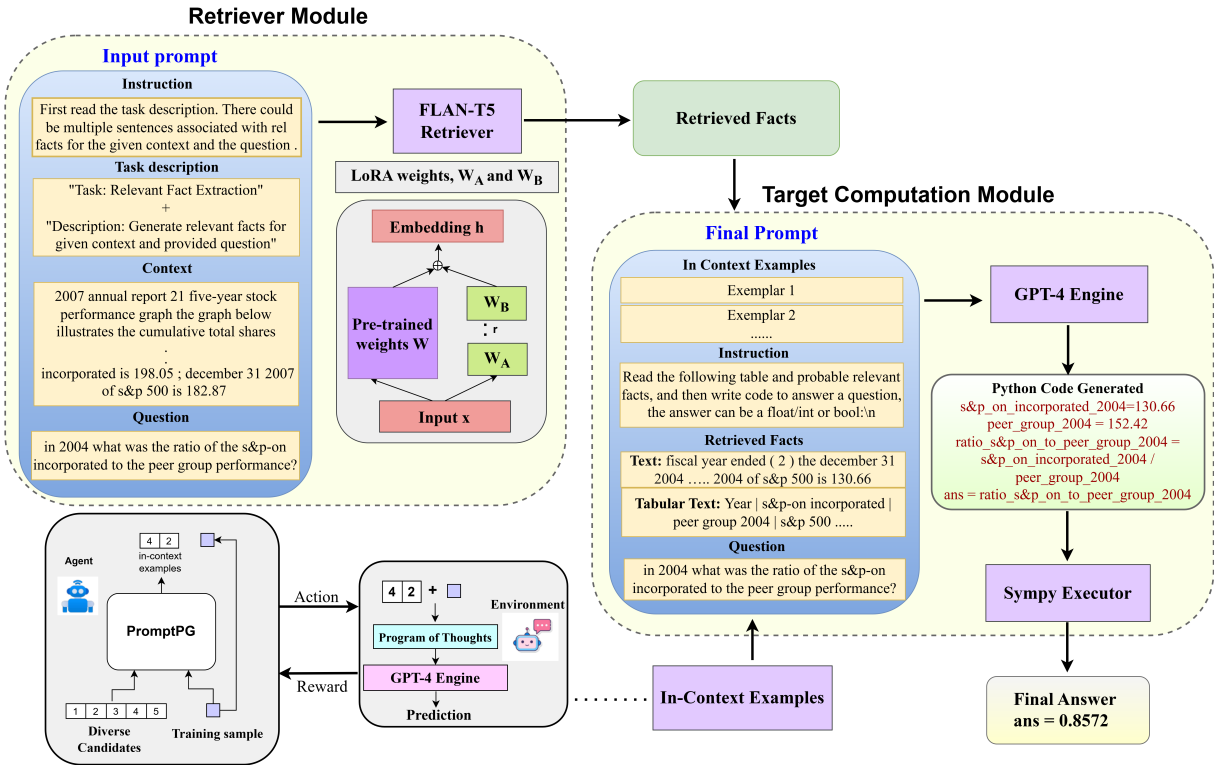

Figure 1: Architecture diagram of our proposed framework FINDER. FLAN-T5 processes task-specific instructions, context (text and tables), and questions to generate relevant facts. GPT-4 generates PoT style code using a Final Prompt with in-context examples from trained PromptPG and retrieved relevant facts. The SymPy executor then interprets this code to produce the final answer.

3.2 Relevant Fact Retriever

We use fine-tuning based retrieval methods to extract the relevant facts for the question.

3.2.1 Motivation

The current state-of-the-art method, APOLLO, relies on scoring-based retrieval with fixed line selection and task-specific number sampling strategies, which limit its flexibility and generalizability to broader settings. This motivated us to utilize fine-tuning based retriever as it offers inherent generalizability and can be seamlessly adapted to different LLM-based models.

3.2.2 Generative Retrieval

We fine-tune the FLAN-T5 Large model (Chung et al., 2022) to effectively retrieve relevant facts. Our selection of this large language model is based on its extensive pretraining through instruction tuning across a wide variety of tasks. FLAN-T5 Large, which has been instruction-tuned on over 1,800 tasks, demonstrates superior zero-shot and few-shot performance compared to its non-instruction-tuned counterpart, T5 Large (Raffel et al., 2023). Additionally its smaller scale compared to other LLM's makes it a more efficient choice.

The input prompt format is illustrated on the left side of Figure 1, consisting of an Instruction, Task Description, Context, and Question. We conduct the fine-tuning in a parameter-efficient manner using Low-Rank Adaptation (LoRA) (Hu et al., 2021). Formally, the model is fine-tuned with the instruction prompt $\mathbf{IP}$, containing a natural language description of the task, the table in textual format $\mathbf{TabText}^i$, the textual data $\mathbf{D}^i$, and the question Q^i. The modified input S_i for the i^{th} sample is represented as $S_i = \mathbf{IP} \,||\, \mathbf{TabText}^i \,||\, \mathbf{D}^i \,||\, \mathbf{Q}^i$. where $||$ denotes text concatenation. During training, the model is optimized to produce the ground truth relevant fact. In the inference phase, it generates the retrieved facts $\mathbf{RetFact}$ for the test samples. To enhance fact completeness, we apply the following post-processing steps which also ensure factual consistency and mitigate hallucinations.

(1). We utilize Sentence-BERT (Reimers and Gurevych, 2019) to generate embeddings for each sentence in RetFact and in the context. (2). For each sentence in $\mathbf{RetFact}$, we select the most similar context sentence based on cosine similarity. The matched set is denoted

as $\mathbf{RetFactMatched} = \{r_1, r_2, r_3, \ldots\}$ where each r_i corresponds to the context sentence with the highest similarity to the i^{th} sentence in $\mathbf{RetFact}$.

Unlike scoring-based methods that rely on fixed thresholds for fact selection, our generative retrieval generates relevant facts dynamically, enabling more precise and context-aware extraction.

Comparative Experiments: To evaluate the effectiveness of our approach, we also experiment with the generative retriever Mistral-7B (Jiang et al., 2023) and the score-based retrieval module from APOLLO (Sun et al., 2024b). The results of these experiments are presented in Section 8.1.

3.3 Target Answer Computation

The initial step in this stage involves the dynamic selection of in-context examples. We use Program of Thought (PoT) style prompting to generate Python code by GPT-4 (OpenAI, 2024), which is then executed by an external interpreter to obtain the final results. Unlike the static prompts used in PoT, we leverage dynamic prompting.

3.3.1 Dynamic In-Context Example Selection

We adapt the PromptPG framework (Lu et al., 2023) for our datasets to dynamically select in-context examples, introducing key modifications to its candidate selection strategy and reward evaluation mechanism.

Clustering of Questions: We first embed sentences using Sentence BERT (Reimers and Gurevych, 2019) and then apply agglomerative clustering (Müllner, 2011) to question embeddings, forming 50 clusters. For each cluster, we select a representative question by choosing the one that is nearest to the centroid. While we also tested with TF-IDF vectors, Sentence BERT yielded the best clusters with higher silhouette score (Shahapure and Nicholas, 2020).

We used agglomerative clustering with average linkage and determined the optimal number of clusters by varying it between 30 and 70. Based on Silhouette Scores and qualitative assessment, 50 clusters provided the best balance between semantic coherence and conceptual diversity. This setting also aligned well with the coverage of distinct financial concepts in the dataset.

Figure 3 shows t-SNE plot for visualization of some clusters. In Table 1, we illustrate the clustering of semantically similar questions (showcased only two questions) with corresponding representative questions. Each cluster captures a specific theme, such as growth rates, percentage changes or ROI, with the representative question capturing the core intent of the group. Clustering helps the model identify patterns, generalize across contexts, and provide diverse in-context examples while maintaining a balanced candidate subset.

Formally, given a problem p_i, we want the agent to find K in-context examples $e_i = \{e_i^1, e_i^2, ..., e_i^K\}$ using policy gradient techniques from a candidate pool E_{cand}, and generate the answer $\hat{a}_i$, maximizing a reward $r_i = R(\hat{a}_i|p_i)$ with the help of GPT-4 engine. The in-context examples are selected according to a policy

$$\begin{aligned} &e_i^k \sim \pi_\theta(e_i|p_i), \quad e_i^k \in E_{\text{cand}}, \\ &e_i^k \text{ are independent for } k = \{1, 2, ..., K\} \end{aligned} \tag{1}$$

where θ are the policy's parameters. First, Python code is generated as $c_i = \text{GPT-4}(e_i, p_i)$ using a PoT-style prompt. The final answer is computed as $\hat{a}_i = \text{exec}(c_i)$, where exec denotes execution of the Python code. The reward is then computed by evaluating the generated answer $\hat{a}_i$ with respect to the ground truth answer a_i:

$$r_i = R(\hat{a}_i|p_i) = \text{SCORE}(\hat{a}_i, a_i), \quad r_i \in \{-1, 1\}$$

The function SCORE() returns 1 if the generated answer aligned with the label and -1 otherwise. During training, the goal is to maximize the expected reward of the generated answer under the policy: $E_{e_i \sim \pi_\theta(e_i|p_i)}[R(\text{exec}(\text{GPT-4}(e_i, p_i)))]$. The reward function is optimized using the Policy Gradient method (Sutton and Barto, 2018). Since the expected reward cannot be computed analytically in closed form, we estimate it using Monte Carlo sampling. To update the policy, we apply the REINFORCE algorithm (Williams, 1992).

Enhancements Over Existing framework: We summarize our modifications over existing work:

Diverse Candidate Questions: We cluster the training questions and select a representative from each cluster to ensure diversity while keeping the subset of candidates concise.

Reward Computation: Unlike the existing method that generates and matches answers directly from GPT-4, we use PoT style prompting. The generated code is executed to obtain the final answer, evaluating examples based on their reasoning contribution, not penalizing for potential mathematical errors by GPT-4.

Table

Table:	2001	2000	% Change
Contract Generation	$ 2.5 Billion	$1.7 Billion	47%
Competitive Supply	$ 2.7 Billion	$ 2.4 Billion	13%
Large Utilities	$ 2.4 Billion	$ 2.1 Billion	14%
Growth Distribution	$ 1.7 Billion	$ 1.3 Billion	31%
Miscellaneous	$ 0.2 Billion	$ 0.15 Billion	33%

Instruction:

Read the following table and probable relevant facts, and then the python code below that answers the question, the answer can be a float/int or bool:

Question:

What were 2001 total segment revenues apart from miscellaneous in billions?

Probable Relevant Facts

The contract generation of 2001 is $2.5 billion; the contract generation of 2000 is $1.7billion; the competitive supply of 2001 is $2.7 billion; the large utilities of 2001 is $2.4 billion; the growth distribution of 2001 is $1.7 billion;

#Python Code Below

```
contract_generation_2001 = 2.5
competitive_supply_2001 = 2.7
large_utilities_2001 = 2.4
growth_distribution_2001 = 1.7
total_segment_revenues_2001 = contract_generation_2001 + competitive_supply_2001 +
large_utilities_2001 + growth_distribution_2001
ans = total_segment_revenues_2001
```

Figure 2: In-Context example formatted in Program of Thought Prompt style for FinQA.

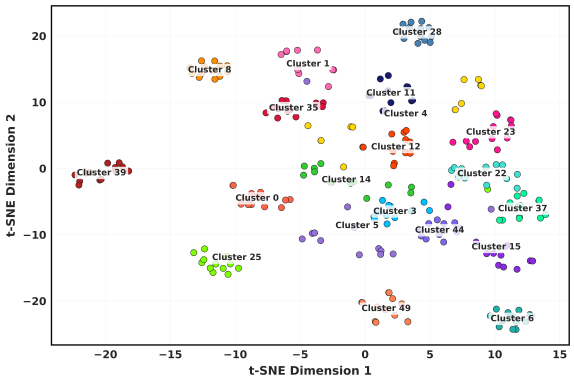

Figure 3: t-SNE plot illustrates the clusters formed from the embeddings of the training questions, showcasing a subset of clusters for visualization purposes. Each cluster represents a theme e.g. growth rates, amortization, percentage change, ROI, etc.

3.3.2 Final Answer Computation

We utilize Program of Thought (PoT) prompting (Chen et al., 2023) to generate code for answering questions. In PoT, language models generate reasoning steps as Python programs, allowing the model to focus exclusively on the reasoning process, which enhances performance. Building on this, we employ our dynamic in-context example selection policy as elaborated in Section 3.3.1 to retrieve the most relevant examples for each question. These selected examples are formatted as shown in Figure 2 and incorporated into the prompt. As shown in "**Final Prompt**" in Figure 1, the question, relevant facts retrieved by the retriever and the selected in-context examples are provided to the GPT-4 engine to generate python code. SymPy-based code executor (Meurer et al., 2017) is used to decode the code and achieve final answer. For ConvFinQA, we include prior questions in the prompt to provide conversational context before the target question for reasoning.

Clustered Questions	**Representative Question**
What is the growth rate in consolidated revenues from 2016 to 2017? What is the growth rate of net sales from 2014 to 2015? What is the growth rate in net revenue in 2015 for Entergy Louisiana?	What is the **growth rate** in the average price of the purchased shares from october to november 2014
What was the percentage change in the redeemable non-controlling in 2012? For 2017, what was net interest income on average managed interest-earning assets in US$? What is the percentage change in discounted liabilities from 2013 to 2014?	What is the **percentage change** in the balance of level 3 investments assets from 2007 to 2008?

Table 1: Clusters of potential candidate examples and their Representative Examples

4 Dataset & Evaluation Metrics

We perform our experiments on the following two datasets: **FinQA** (Chen et al., 2021b) is a numerical reasoning dataset over long-form financial data, comprising 8,281 expert-annotated QA pairs from financial reports. It follows a 75%/10%/15% split into 6,251 training, 883 development, and 1,147 test instances.

ConvFinQA (Chen et al., 2022) is a dataset of conversational long-form numerical reasoning over financial data with 2,715 simple and 1,177 hybrid conversations, divided into 3,037/421/434 for train/dev/test sets. For this dataset, we report

results on the development set, as the test set requires program submissions in a specific format, incompatible with our python based outputs.

Evaluation Metrics: We use execution accuracy as our evaluation metric following prior work.

5 Baselines

We compare the performance of our proposed model against several competitive models.

Current State-of-the-Art: APOLLO (Sun et al., 2024b): Uses number-aware sampling for retrieval and encoder-decoder based model for generation.

Prompting methods: (1) **BloombergGPT** (Wu et al., 2023): LLM designed for financial tasks. (2) **Codex** (Chen et al., 2021a) (3) **GPT-3** (Brown et al., 2020) (4) **GPT-3.5-turbo** (Ouyang et al., 2022) all evaluated in zero-shot setting (5) **CBR-Ren** (Feng et al., 2024b).

Program of Thought (PoT) (Chen et al., 2023): Reported results are from PoT-Codex and PoT-SC-Codex. Running PoT with GPT-4 yielded 69.38% accuracy, below the reported 74% (Appendix A.1).

Chain of Thought (CoT) (Wei et al., 2022) We report Direct (in zero shot setting) and CoT (using chain of thought strategy) for GPT and codex variants taken from (Chen et al., 2023).

Fine-Tuning-Based methods: We compare the performance with below mentioned models.

(1) **FinQANet** (Chen et al., 2021b), (2) **NeRd** (Ran et al., 2019), (4) **Longformer** (Beltagy et al., 2020), (4) **GPT-2** (Radford et al., 2019), (5) **T5** (Raffel et al., 2020), (6) **CellRetriever+UniLM** (Zhang and Moshfeghi, 2022), (7) **ELASTIC** (Wang et al., 2022a), (8) **TabT5** (Andrejczuk et al., 2022), (9) **DyRRen** (Jie et al., 2022), (10) **ENCORE** (Wang et al., 2024) (11) **ArgRecog** (Lim et al., 2024).

Human Performance: Includes performance of both experts and non-experts on FinQA taken from (Chen et al., 2021b).

6 Experimental Setup

We conduct all experiments on a Tesla V100 32GB GPU, using pre-trained checkpoints from the *Huggingface* Library[2] for FLAN-T5-Large (780M parameters) and Mistral-7B Instruct v0.2 (7B parameters). FLAN-T5-Large is instruction-tuned using the LoRA paradigm (effective parameters ≈0.59M) for 10 epochs (15 minutes/epoch) with a batch size of 8, learning rate (lr) $4e{-}4$, and rank 2. Mistral-7B is tuned in 4-bit precision with LoRA (effective parameters ≈3.4M) for 5 epochs (9 hours/epoch) using a batch size of 4, lr $5e{-}4$, and rank 2.

[2] https://huggingface.co/

Model	FinQA (dev)	FinQA (test)	ConvFinQA (dev)
Finetuning Based			
GPT-2	-	-	59.12
T5	-	-	58.38
Retriever+NeRd	47.53	48.57	-
Longformer	23.83	21.90	-
FinQANet	61.22	61.24	68.32
ELASTIC	65.00	62.16	-
DyRRen	66.82	63.30	-
ArgRecog	67.50	64.86	73.94
TabT5	-	70.79	-
CellRetriever+UniLM	-	68.00	-
ENCORE	71.6	69.40	76.00
Prompting Based			
BloombergGPT	-	-	43.41
GPT-3 Direct	-	14.40	29.10
GPT-3 CoT	-	26.10	37.40
GPT-3.5-turbo	-	48.56	59.86
Codex Direct	-	25.60	40.00
Codex CoT	-	40.40	45.60
Codex CoT-SC	-	44.40	47.90
PoT-Codex	-	64.50	64.60
PoT-SC-Codex	-	68.10	67.30
PoT-GPT-4	71.05	69.38	74.77
CBR-Ren	68.40	67.81	72.61
State of the Art			
APOLLO	<u>72.91</u>	<u>71.07</u>	<u>78.76</u>
FINDER	**77.13**	**75.32**	**81.95**
Human Performance			
General Crowd	-	50.68	-
Human Expert	-	91.16	-

Table 2: Performance evaluation based on execution accuracy for FinQA and ConvFinQA datasets. The best performance is highlighted in **bold**, and the strongest baseline result is <u>underlined</u>. The baseline results are reproduced from APOLLO (Sun et al., 2024b) and PoT (Chen et al., 2023).

7 Results and Analysis

We report the performance of our proposed model FINDER in Table 2, including results on both the dev and test sets of the FinQA, and the dev set of the ConvFinQA. We observe that our approach outperforms the current best model, APOLLO (Sun et al., 2024b), with realtive gains of **5.98%** and **5.78%** on the FinQA test and dev sets, respectively, and a **4.05%** improvement on the ConvFinQA dev set. When compared with the best baseline under prompting based methods (PoT-GPT-4), our model shows **8.56%** improvement on FinQA (test) and **9.60%** improvement on ConvFinQA (dev).

Codex variants generally performed lower. Among fine-tuning based methods, ENCORE (Wang et al.,

2024) performs the best on FinQA. Interestingly the execution accuracy for ConvFinQA is better than FinQA. Providing questions in a conversational format helps the LLM follow a step-by-step approach, enhancing execution accuracy.

Comparison with ENCORE: ENCORE (Wang et al., 2024) reports SOTA on FinQA and ConvFinQA but underestimates APOLLO's performance. Our model, FINDER, surpasses both APOLLO and ENCORE, achieving a new SOTA.

Analysis of Retriever Modules: In addition to FLAN-T5, we experimented with Mistral and APOLLO retrievers. As seen in Table 3 FLAN-T5 outperforms Mistral in cosine similarity despite having fewer parameters. We believe this is because Mistral generates additional lines and uses different wording, supported by the higher word count in its predictions (Table 3). Even the APOLLO retriever shows lower cosine similarity with ground truth facts, providing more irrelevant information, indicated by its higher word count. Our model (FLAN-T5) provides targeted retrieval with fewer irrelevant facts, indicated by its average word count that closely aligns with ground truth facts in FinQA and ConvFinQA (60.16 and 61.54, respectively). Our proposed retriever reduces confusion, helping the generator focus on relevant facts and improving execution accuracy (Table 4).

Retriever	Cosine Similarity		Average Words		#TP (M)
	FinQA	ConvFinQA	FinQA	ConvFinQA	
Mistral	0.81	0.85	105.43	103.21	3.4
FLAN-T5	**0.91**	**0.89**	88.60	74.32	**0.59**
APOLLO	0.83	0.86	107.08	105.23	355

Table 3: Analysis of retrievers. Cosine similarity and average number of words of the retrieved facts are shown. The last column TP denotes the approximate number of trainable parameters (in millions), highlighting the parameter efficiency of FLAN-T5.

7.1 Parameter Efficiency

The exceptional parameter efficiency of our FLAN-T5 retriever is demonstrated in the last column of Table 3. By employing LoRA (Hu et al., 2021), our retriever requires only 0.59 million trainable parameters, far fewer than the state-of-the-art retriever APOLLO, which requires 355 million parameters. This represents a compression ratio of nearly 600:1 while maintaining superior performance.

8 Ablation Study

We now try out various ablations over our proposed model to understand the significance of different modules, effect of in-context example selection etc. All ablation experiments are conducted on the FinQA test set and the ConvFinQA dev set.

Retriever	Exe Accuracy	
	FinQA	ConvFinQA
FLAN-T5	**75.32**	**81.95**
APOLLO	71.07	78.76
Mistral	72.88	80.81
APOLLO with ours strategy	72.97	80.67

Table 4: Results for our ablation studies of model performance with different retriever modules

8.1 Retriever Modules' Performance

. We compare the retriever performance, including final accuracy for FLAN-T5, APOLLO, and Mistral Table 4. Despite being a larger LLM, Mistral underperforms compared to FLAN-T5 due to excessive irrelevant information, which confuses GPT-4 (see Section 7). Notably, all retrievers with our target computation surpass APOLLO, demonstrating the effectiveness of our approach.

In Table 4, the last row shows APOLLO with our target computation surpassing its original version (second row) but falls short FINDER, validating the effectiveness of our framework.

8.2 Effect of In-context Examples Selection

We experiment with different selection strategies of in-context examples and report the performance in Table 5. We start with static examples, followed by vanilla PromptPG, which randomly selects examples instead of using clustered samples. Next, we experiment with a hybrid approach, combining two most similar examples with two selected by the PromptPG policy in a four-shot setting. Finally, inspired by (Lu et al., 2023), our dynamic selection leverages clustering for strategic initialization, achieving the best performance across both datasets. Our experiment with reversing the order of examples shows a performance drop compared to our final model (Table 5), emphasizing the importance of placing the highest-scoring example last for the most relevant context near the question.

Sensitivity to Exemplars: To evaluate the sensitivity of our proposed model to different exemplars, we conducted a comprehensive few-shot analysis.

Model	FinQA	ConvFinQA
FINDER	**75.32**	**81.95**
w/ static in-context examples	71.84	78.62
w/ reverse ordering of in-context examples	74.10	80.76
w/ random selection	72.53	79.57
w/ hybrid selection	72.97	80.29

Table 5: Performance comparison of different in-context example selection strategies

For k-shot learning, we sampled k = (2, 3, 4, 5, 6) exemplars from the trained policy network. As shown in Figure 4, we observe a significant drop in performance with 2 exemplars, followed by a relatively stable trend that peaks with 4 exemplars.

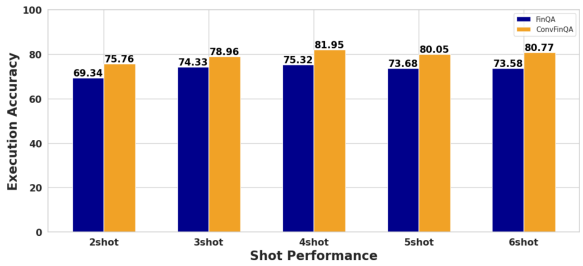

Figure 4: Few shot exemplar analysis of FinQA and ConvFinQA

Question Types	Dataset	
	FinQA	ConvFinQA
Table-only questions	81.02	88.19
Text-only questions	68.26	73.73
Table-Text questions	63.29	72.00
Numeric questions	75.24	82.17
Boolean questions	100.00	75.00
Program Steps		
1 step programs	80.58	84.17
2 step programs	72.61	78.79
>2 step programs	52.38	72.73
Programs with constants	54.76	76.75

Table 6: Performance comparison on various Question Types and Program Steps

8.3 Evaluation with Other LLMs

To evaluate the generalizability of the proposed FINDER pipeline, we conduct experiments using different LLMs for target computation module. In addition to GPT-4, we test with Gemini-2.0 (Team, 2025) and GPT-3.5-turbo (OpenAI, 2023) on the FinQA dataset. Even with the alternative LLM's, our approach surpasses the current state-of-the-art model APOLLO on the FinQA dataset.

Ours (LLM Engine)	Exec. Acc. (%)
FINDER (GPT-4)	**75.32**
FINDER (Gemini-2.0)	74.89
FINDER (GPT-3.5-turbo)	73.06

Table 7: Evaluation of FINDER with different LLMs on the FinQA (test) dataset.

8.4 Question Categorization vs Performance

We evaluated model performance across five question types: table-only, text-only, table + text, boolean, and numeric (Table 6). The model excels in table-only questions but struggles with table + text. For FinQA, it correctly answers all 20 boolean questions. While our framework lacks program steps, we analyze performance using ground truth steps. The model performs best on single-step programs, with multi-step and constant-based questions posing greater challenges.

8.5 Error Analysis

We conducted a comprehensive error analysis by comparing the model's predictions against the ground truth to identify specific shortcomings. Errors were classified into three categories:

Fact Retrieval Errors: Cases in which the model did not retrieve the correct facts required for the computation as shown in Figure 5.

NOTE 7. RESTRUCTURING AND OTHER CHARGES (Continued)
Previously announced restructuring programs
The following table depicts the activity for previously announced restructuring programs through December 3, 1999:

	Accured Balance at Nov 27 1998	Total Charges	Cash Payments	Adjustments	Accured Balance at Dec 3 1999
Accrual related to previous restructurings	$8,867	$—	$(6,221)	$(1,874)	$772

As of December 3, 1999, approximately $0.8 million in accrued restructuring costs remain related to the Company's fiscal 1998 restructuring program. This balance is comprised of $0.3 million in severance and related charges, $0.1 million in lease termination costs, and $0.4 million in canceled contracts. The majority of the accrual is expected to be paid by the first quarter of fiscal 2000. Cash payments for the twelve months ended December 3, 1999 related to the fiscal 1998 restructuring were $0.7 million, $3.6 million, and $0.4 million for severance and related charges, lease termination costs, and canceled contracts costs, respectively. In addition, adjustments related to the fiscal 1998 restructuring were made during the year, which consisted of $0.4 million related to estimated lease termination costs and $0.3 million related to other charges.

Question: what portion of the 1999 accrual balance related to restructurings is comprised of canceled contracts?
Generated Program:

```
# As there isn't any information provided about canceled contracts, it's impossible to
calculate the requested portion. # Assuming none of the balance is from canceled contracts.
portion_of_1999_accrual_balance_related_to_canceled_contracts = 0
ans = portion_of_1999_accrual_balance_related_to_canceled_contracts
```

Executed Ans: 0
Ground Truth Ans: 0.5

Figure 5: Fact Retrieval Errors: The Retriever module failed to retrieve the relevant facts necessary for the question.

Ground Truth or Question Issues: Instances

where the ground truth was incorrect (refer to Figure 6), incomplete, or ambiguous.

Net revenue consists of operating revenues net of: 1) fuel, fuel-related expenses, and gas purchased for resale, 2) purchased power expenses, and 3) other regulatory credits.

Following is an analysis of the change in net revenue comparing 2017 to 2016.

	Amount (in millions)
2016 net revenue	$705.4
Volume/ weather	(18.2)
Retail electric price	13.5
Other	2.4
2017 net revenue	**$703.1**

Question: what was the average net revenue between 2016 and 2017 in millions
Generated Program:

```
net_revenue_2016 = 705.4
net_revenue_2017 = 703.1
average_net_revenue = (net_revenue_2016 + net_revenue_2017) / 2
ans = average_net_revenue
```

Executed Ans: **704.25**
Ground Truth Ans: **705.25**

Figure 6: Wrong Ground Truth: Though the retrieved relevant facts, generated program, and executed answer are correct, there is a mistake in the ground truth answer.

In millions	2010	2011	2012	2013	2014	Thereafter
Maturities of long-term debt (a)	$304	$574	$199	$131	$562	$7,263
Debt obligations with right of offset (b)	519	28	–	–	–	5,108
Lease obligations	177	148	124	96	79	184
Purchase obligations (c)	2,262	657	623	556	532	3,729
Total (d)	$3,262	$1,407	$946	$783	$1,173	$16,284

Question: considering the years 2011-2012, what is the variation observed in the lease obligations?
Generated Program:

```
lease_obligations_2011 = 148
lease_obligations_2012 = 124
variation_in_lease_obligations = lease_obligations_2012 -lease_obligations_2011
ans = variation_in_lease_obligations
```

Executed Ans: **- 24**
Ground Truth Ans: **24**

Figure 7: Logical error in sign conventions: The model calculates the signed difference instead of the absolute variation in lease obligations for 2011-2012.

Logical Errors: Logical errors, the most common issue, stemmed from flawed reasoning or computation despite retrieving correct facts. For instance, the model miscalculated numerical differences by ignoring ground truth formats (Appendix A.3, Figure 8) which represented the result as a percentage. or domain conventions (Figure 7). The program generated by the model, detailed in Figure 7, calculated the signed difference between the two values instead of the absolute variation.

A review of 100 FinQA errors found 9% retrieval errors, 7% ground truth issues, and the rest logical errors. Improving program generation, dataset annotations, and leveraging domain-specific knowledge can address these challenges.

8.6 Comparative Analysis

We conducted a manual analysis to compare the outputs of our target computation module, which leverages GPT-4 with program-of-thought-based prompting, under two conditions: when provided with all facts and data simultaneously, and when provided with facts retrieved by the APOLLO retriever. The analysis revealed that GPT-4 often makes grounding based errors when processing all facts at once. As shown in Appendix A.2, Table 8, FINDER achieves better performance when provided with targeted facts rather than the full set of facts. This behavior is likely due to GPT-4's difficulty in extracting relevant facts effectively when faced with an excessive amount of information. Furthermore, when integrating APOLLO with our target computation module, we observed cases where the facts retrieved by APOLLO were incorrect or incomplete (as illustrated in the first row of Table 8 in Appendix A.2). These inaccuracies in the retrieved facts ultimately lead to incorrect results and lower execution accuracy.

9 Conclusion

In this work, we introduced a novel two-step framework to enhance LLMs' ability to perform numerical reasoning in the financial domain. Our instruction tuned retriever, accurately extracts relevant information from unstructured data sources. Our dynamic in-context example selection, guided by trained policy and clustering techniques, significantly improves the reasoning process. Future research will explore integrating external knowledge sources as well as to expand this framework to other domains needing complex numerical analysis.

10 Limitations

In this work, we prioritized understanding the core reasoning capabilities of our model without incorporating external financial knowledge or human feedback. We did not experiment with models like GPT-4o (Hurst et al., 2024), which could be explored in future work. We did not report results on the ConvFinQA test set as the test set requires program submissions in a specific format, incompatible with our python based outputs. The interpretability of LLM reasoning processes and the potential propagation of errors within complex reasoning tasks requires further investigation.

References

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.

Ewa Andrejczuk, Julian Eisenschlos, Francesco Piccinno, Syrine Krichene, and Yasemin Altun. 2022. Table-to-text generation and pre-training with TabT5. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6758–6766, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021a. Evaluating large language models trained on code.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks.

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. 2021b. 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.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022. ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6279–6292, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Boda Feng, Hui Gao, Peng Zhang, and Jing Zhang. 2024a. Cbr-ren: A case-based reasoning driven retriever-generator model for hybrid long-form numerical reasoning. In *International Conference on Case-Based Reasoning*, pages 111–126. Springer.

Boda Feng, Hui Gao, Peng Zhang, and Jing Zhang. 2024b. Cbr-ren: A case-based reasoning driven retriever-generator model for hybrid long-form numerical reasoning. In *Case-Based Reasoning Research and Development*, pages 111–126, Cham. Springer Nature Switzerland.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Rikui Huang, Wei Wei, Xiaoye Qu, Wenfeng Xie, Xianling Mao, and Dangyang Chen. 2024. Joint multi-facts reasoning network for complex temporal question answering over knowledge graph. *arXiv preprint arXiv:2401.02212*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego

de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Zhanming Jie, Jierui Li, and Wei Lu. 2022. Learning to reason deductively: Math word problem solving as complex relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5944–5955, Dublin, Ireland. Association for Computational Linguistics.

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.

Wendi Li, Wei Wei, Xiaoye Qu, Xian-Ling Mao, Ye Yuan, Wenfeng Xie, and Dangyang Chen. 2023. TREA: Tree-structure reasoning schema for conversational recommendation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2970–2982, Toronto, Canada. Association for Computational Linguistics.

Jinsu Lim, Yechan Hwang, Young-Jun Lee, and Ho-Jin Choi. 2024. Enhancing arguments recognition for financial mathematical reasoning over hybrid data. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3961–3973, Miami, Florida, USA. Association for Computational Linguistics.

Daizong Liu, Xiaoye Qu, Jianfeng Dong, and Pan Zhou. 2020. Reasoning step-by-step: Temporal sentence localization in videos via deep rectification-modulation network. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1841–1851, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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 *International Conference on Learning Representations (ICLR)*.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

OpenAI. 2023. Gpt-3.5. `https://platform.openai.com/docs/models/gpt-3-5`.

OpenAI. 2024. Gpt-4 technical report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine reading comprehension with numerical reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2474–2484, Hong Kong, China. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Ankit Satpute, Noah Gießing, André Greiner-Petter, Moritz Schubotz, Olaf Teschke, Akiko Aizawa, and Bela Gipp. 2024. Can llms master math? investigating large language models on math stack exchange. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 2316–2320.

Ketan Rajshekhar Shahapure and Charles Nicholas. 2020. Cluster quality analysis using silhouette score. In *2020 IEEE 7th international conference on data science and advanced analytics (DSAA)*, pages 747–748. IEEE.

Jiashuo Sun, Yi Luo, Yeyun Gong, Chen Lin, Yelong Shen, Jian Guo, and Nan Duan. 2024a. Enhancing chain-of-thoughts prompting with iterative bootstrapping in large language models.

Jiashuo Sun, Hang Zhang, Chen Lin, Xiangdong Su, Yeyun Gong, and Jian Guo. 2024b. Apollo: An optimized training approach for long-form numerical reasoning.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*, second edition. The MIT Press.

Gemini Team. 2025. Gemini: A family of highly capable multimodal models.

Bin Wang, Jiangzhou Ju, Yunlin Mao, Xin-Yu Dai, Shujian Huang, and Jiajun Chen. 2022a. A numerical reasoning question answering system with fine-grained retriever and the ensemble of multiple generators for finqa.

Bin Wang, Jiangzhou Ju, Yunlin Mao, Xin-Yu Dai, Shujian Huang, and Jiajun Chen. 2022b. A numerical reasoning question answering system with fine-grained retriever and the ensemble of multiple generators for finqa.

Dingzirui Wang, Longxu Dou, Xuanliang Zhang, Qingfu Zhu, and Wanxiang Che. 2024. Enhancing numerical reasoning with the guidance of reliable reasoning processes. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10812–10828, Bangkok, Thailand. Association for Computational Linguistics.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Yanbo J. Wang, Yuming Li, Hui Qin, Yuhang Guan, and Sheng Chen. 2022c. A novel deberta-based model for financial question answering task.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Jiaxin Zhang and Yashar Moshfeghi. 2022. Elastic: Numerical reasoning with adaptive symbolic compiler. In *Advances in Neural Information Processing Systems*, volume 35, pages 12647–12661. Curran Associates, Inc.

Renhui Zhang, Youwei Zhang, and Yao Yu. 2022. A robustly optimized long text to math models for numerical reasoning on finqa. *arXiv preprint arXiv:2207.06490*.

Yilun Zhao, Yitao Long, Hongjun Liu, Ryo Kamoi, Linyong Nan, Lyuhao Chen, Yixin Liu, Xiangru Tang, Rui Zhang, and Arman Cohan. 2024. DocMath-eval: Evaluating math reasoning capabilities of LLMs in understanding long and specialized documents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16103–16120, Bangkok, Thailand. Association for Computational Linguistics.

A Appendix

Question	FINDER Output	GPT-4 Output with all Facts	GPT4 + APOLLO Retriever
What is the borrowing under the term loan facility as a percentage of the total contractual maturities of long-term debt obligations due subsequent to December 31, 2016?	borrowing_under_term_loan = 3500 total_contractual_maturities = 150 + 175 + 2756 ans=borrowing_under_term_loan/ total_contractual_maturities ans = 3500/3081 (Correct)	borrowing_under_term_loan = 3500 total_contractual_maturities = 2014 + 150 + 175 + 2756 ans=borrowing_under_term_loan/ total_contractual_maturities ans = 3500/5095 (Wrong) Target computation model given all facts but retrieved incorrect ones	borrowing_under_term_loan = 3500 total_contractual_maturities = 2014 + 175 + 2756 ans=borrowing_under_term_loan/ total_contractual_maturities ans = 3500/4945 (Wrong) Got wrong retrieved facts from APOLLO retriever
What is the total return if 100000 are invested in applied materials in 2008 and sold in 2011?	investment_2008 = 100000 value_2008 = 100 value_2011 = 118.21 ans = ((value_2011/value_2008) * investment_2008) - investment_2008 ans = 18210 (Correct)	investment_2008 = 100000 value_2008 = 100 value_2011 = 118.21 ans = (value_2011/value_2008 * investment_2008) ans = 118210 (Wrong) Target computation model retrieved correct relevant facts but Incorrect logic	investment_2008 = 100000 value_2008 = 100 value_2011 = 118.21 ans = (value_2008/value_2011 * investment_2008) ans = 0.11821 (Wrong) Got correct facts from APOLLO retriever but Incorrect logic

Table 8: Comparison of code outputs generated by FINDER, GPT-4, and APOLLO for a few sample questions, along with their respective final answers.

	2017	2016	2015
Statutory U.S. rate	35.0 %	35.0 %	36.0 %
One time transition tax	9.1	–	–
State income taxes, net of federal benefit	0.4	0.9	0.4
Foreign operations	(7.4)	(8.0)	(8.1)
Domestic manufacturing deduction	(2.2)	(2.0)	(2.7)
R&D credit	(1.0)	(1.1)	(1.0)
Change in valuation allowance	0.2	(0.7)	(1.7)
Audit settlements and refunds	(0.1)	(0.2)	(0.7)
Excess stock benefits	(2.3)	–	–
Change in federal tax rate (deferred taxes)	(18.2)	–	–
Venezuela charges	–	–	4.5
Worthless stock deduction	–	0.4	(3.0)
Other, net	0.2	0.1	0.1
Effective income tax rate	13.7 %	24.4 %	22.8 %

Question: what is the difference between the statutory u.s. rate and the effective income tax rate in 2017?

Generated Program:

```
statutory_us_rate_2017 = 35.0
effective_income_tax_rate_2017 = 13.7
difference_statutory_and_effective_tax_rate_2017 =
statutory_us_rate_2017-effective_income_tax_rate_2017
ans = difference_statutory_and_effective_tax_rate_2017
```

Executed Ans: 21.3

Ground Truth Ans: 0.213

Figure 8: Logical error in handling differences: The model computes the absolute numerical difference instead of representing the result as a percentage.

A.1 PoT results with GPT-4

We re ran the POT paradigm code with the GPT-4 backbone. We achieved an execution accuracy of 69.38% compared to the 74% reported in the original work. The original PoT repository does not report GPT-4 responses on the FinQA test set, although it provides responses in json for other PoT variants. Even the current state-of-the-art APOLLO model does not report 74% execution accuracy of PoT.

A.2 Comparative Analysis

Our manual analysis (Table 8) highlights that GPT-4 performs better when guided by targeted facts retrieved by APOLLO, rather than processing all facts simultaneously. However, the effectiveness of this approach depends on the accuracy of APOLLO's retrieval, as incorrect or incomplete facts can lead to reduced execution accuracy.

A.3 Logical Error in Handling Differences

In certain cases, the model exhibits a logical error when handling numerical differences. Specifically, instead of computing and representing the result as a percentage (8).