BizBench: A Quantitative Reasoning Benchmark for Business and Finance

Rik Koncel-Kedziorski[†]**, Michael Krumdick**[†]
Viet Lai, Varshini Reddy, Charles Lovering, Chris Tanner
Kensho Technologies
michael.krumdick@kensho.com

Abstract

Answering questions within business and finance requires reasoning, precision, and a wide-breadth of technical knowledge. Together, these requirements make this domain difficult for large language models (LLMs). We introduce BizBench, a benchmark for evaluating models' ability to reason about realistic financial problems. BizBench comprises eight quantitative reasoning tasks, focusing on question-answering (QA) over financial data via program synthesis. We include three financially-themed code-generation tasks from newly collected and augmented QA data. Additionally, we isolate the reasoning capabilities required for financial QA: reading comprehension of financial text and tables for extracting intermediate values, and understanding financial concepts and formulas needed to calculate complex solutions. Collectively, these tasks evaluate a model's financial background knowledge, ability to parse financial documents, and capacity to solve problems with code. We conduct an in-depth evaluation of open-source and commercial LLMs, comparing and contrasting the behavior of code-focused and language-focused models. We demonstrate that the current bottleneck in performance is due to LLMs' limited business and financial understanding, highlighting the value of a challenging benchmark for quantitative reasoning within this domain.

1 Introduction

Large language models (LLMs) show strong performance on question-answering (QA) and code generation tasks (Austin et al., 2021; OpenAI, 2023). Nonetheless, it remains particularly difficult for models to reason about quantities and numbers (Hendrycks et al., 2021c). This poses issues for using LLMs for real-world problems in business
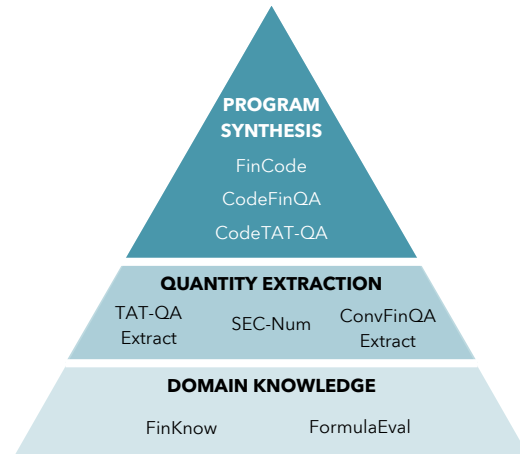

Figure 1: Overview of BizBench's eight tasks. Each level of the pyramid corresponds to a task category, with higher levels requiring increasingly complex capabilities.

and finance, as these fields can require transparent and precise reasoning capabilities.

To facilitate the development of better models for business and finance, we introduce a new benchmark for evaluating financial quantitative reasoning, with a focus on question-answering over structured and unstructured financial data.

Financial questions often require multi-step reasoning (Chen et al., 2021b). Professional workflows necessitate a transparent reasoning process to promote user trust, but how LLMs reason remains opaque. While chain-of-thought (CoT) prompting, where reasoning steps are generated as part of model output, improves performance on reasoning tasks (Suzgun et al., 2023; Kojima et al., 2022), the answers generated from CoT are often not a direct product of the generated reasoning (Wei et al., 2022; Wang et al., 2023; Lanham et al., 2023).Unlike generating CoT, generating executable code (program synthesis) ties model outputs to specific opereations, functions or instructions (Madaan et al., 2022; Chen et al., 2023).

[†]Equal contribution.

Task	Test	Train	Novel Task	CG	FK	DC	DS
Program Synthesis							
FinCode	121	16	✓	✓	✓		
CodeFinQA	844	4,669		✓	✓	✓	
CodeTAT-QA	392	2,864		✓	✓		✓
Quantity Extraction							
ConvFinQA (E)	916	-			✓	✓	
TAT-QA (E)	248	-			✓	✓	
SEC-Num	2,000	6,845	✓		✓	✓	
Domain Knowledge							
FinKnow	877	-			✓		
FormulaEval	50	-	✓	✓	✓		

Table 1: Characteristics of BizBench tasks including test data size, novel supervised-finetuning data size, the inclusion of novel task and novel data, and if the task involves code generation (CG), financial knowledge (FK), document context (DC), and data structures (DS).

In BizBench, we frame multi-step QA as a program synthesis task. This formulation provides increased transparency because the exact rationale for a model's answer can be audited.

BizBench consists of three interrelated *types* of tasks for assessing transparent and accurate financial reasoning: program synthesis, quantity extraction, and domain knowledge. Figure 1 shows an overview of BizBench.

Program Synthesis. We introduce a novel QA task, **FinCode**, built from professional exams. We also create two code-generation tasks **CodeTAT-QA** and **CodeFinQA** by reformulating the existing financial QA datasets TAT-QA (Zhu et al., 2021) and FinQA (Chen et al., 2021b). Both of these tasks require leveraging raw text and tables.

Quantity Extraction. We introduce **SEC-Num**, a numerical span-identification task over corporate earnings reports filed with the U.S. Securities and Exchange Commission (SEC). Additionally, we study span-based QA subsets of ConvFinQA (Chen et al., 2022) and TAT-QA (Zhu et al., 2021).

Financial Domain Knowledge. We introduce two tasks: **FormulaEval**, a novel code-completion task which tests a model's knowledge of financial formulas, and **FinKnow**, which tests non-quantitative subsets of business and finance exams.

BizBench complements existing benchmarks for business and financial NLP (Xie et al., 2023; Shah et al., 2022). Existing benchmark tasks include sentiment analysis, named entity prediction, and stock price prediction. BizBench focuses on a model's capacity to leverage numeric information in structured and unstructured financial data to answer quantitative financial questions.

Our evaluations of open-source and commercial pre-trained language models – in both few-shot and fine-tuned setups – provide a detailed view of the state-of-the-art NLP for quantitative financial reasoning. Model size, instruction-tuning, and code-specific pretraining all meaningfully impact performance, but significant improvement is still needed for even the best models to be useful in high-stakes, real-world workflows.

The major contributions of this work include:

- A new benchmark of eight tasks for evaluating quantitative reasoning in finance and business.

- Novel data for studying the problems of program synthesis, quantity extraction, and knowledge of the financial domain.

- Detailed evaluation of a variety of LLMs, showing how model size, data, code-tuning, and alignment impact task performance.

- Error analysis showing that top-performing models fall short mainly due to gaps in their financial knowledge.

Along with releasing the dataset at `https://huggingface.co/datasets/kensho/bizbench`, we've established a leaderboard on a held-out, curated subset of the evaluation data at `https://benchmarks.kensho.com/`, open to public submission. Through the leaderboard, we hope to encourage progress towards developing solutions to these challenging problems while mitigating the potential for data contamination.

2 Reasoning in Business and Finance

Business and finance professionals are frequently required to reason about quantities. They often search for specific quantities within large sets of distractors (e.g., searching for the revenue of a particular division for a specific quarter, and doing so from a report that contains a comprehensive detailing of financial metrics across many divisions). They also rely on financial domain knowledge across banking, accounting, and finance to manipulate these quantities in useful ways. For example, computing an EBITDA margin requires first identifying a series of relevant quantities (net income, interest expense, taxes, depreciation, amortization, and revenue) and then applying a specific formula to these quantities (taking the sum of the first five quantities and computing their percentage of revenue). Since large sums of money are often at stake, financial professionals need to communicate transparent rationales for their decisions.

Researchers have developed tasks to measure some aspects of quantitative financial reasoning capabilities of AI systems. FinQA consists of 8,281 questions written by financial professionals, paired with context and tables from earnings reports of the largest companies listed on American stock exchanges (Chen et al., 2021b). Answering FinQA questions often requires multiple reasoning steps, including extracting quantities, performing mathematical operations with them, and reasoning about time periods such as fiscal quarters and years. Additionally, these questions test the financial knowledge of currency and scale (e.g. percentage, millions, billions). Similar to FinQA, TAT-QA consists of questions, answers, and relevant context of text and tables (Zhu et al., 2021). The 16,552 questions in TAT-QA cover a range of skills including span-extraction, comparison, and arithmetic.

Program synthesis improves transparency of model outputs, allowing for auditing of reasoning steps — which in turn increases trust and usability. Since large sums of money are often at stake, financial professionals need to communicate transparent rationales for their decisions. Generating programs allows QA models to avoid arithmetic calculations, which is challenging for smaller models.

3 Task Details

The main focus of BizBench is evaluating a model's financial understanding through program synthesis. We additionally provide tasks in two domains we view to be the building blocks in this regard: quantity extraction and financial domain knowledge. In order to answer a complex program synthesis question, the model must be able to understand and extract relevant values within a question or context. It must also have some knowledge of the necessary formulas and other information required to then compute the answer. Additional skills and features of each task are shown in Table 1.

3.1 Program Synthesis

Program synthesis requires a model to generate logically sound code that can be executed to answer some question. Each example contains a natural language question, optionally text or structured data source, and a Python program that produces a numeric answer to the question. We measure accuracy by comparing this numeric output with a ground truth reference value. Answers within 1% of the reference are considered to be correct.

Question: *An investment project costing $500 today will generate profiles of $500 in year five and year ten. If the discount rate is 10%, what is the project's NPV? Answer to the nearest dollar.*

```python
Python Program Solution

1 initial_investment = 500
2 discount_rate = 0.1
3 pv = lambda year, cash_flow:
      cash_flow / ((1 + discount_rate) ** year)
4 npv = (pv(5, 500) + pv(10, 500)) -
      initial_investment
5 round(npv)
```

Figure 2: Example from **FinCode**. In this dataset, answering questions requires financial background knowledge and the ability to synthesize code.

Questions for these tasks are written by financial professionals. The associated reference code is written either by human annotators or converted from human-generated equations (see Figure 3) for a sample semantically-rich Python program.

Indicator	FinCode	CodeFinQA	CodeTAT-QA
Context	-	131.3	-
Table	-	64.8	147.0
Question	14.4	5.4	6.7
Avg. Total Numerals	14.4	201.6	153.7
Addition	0.81	0.25	0.13
Subtraction	1.04	0.49	2.43
Multiplication	1.40	0.62	0.41
Division	0.73	0.73	0.43
Percentage	0.21	0.62	0.39
Exponent	0.13	0.00	0.00
Avg. Total Operators	4.32	2.71	3.79
Lines of code	7.0	3.8	3.0
Parenthesis	4.6	0.3	1.6

Table 2: Complexity of program synthesis in BizBench including the average count of numerals within each part of the input; the average count of mathematical operators used in the solution; and the solution complexity in terms of average lines of code and parentheses;.

We highlight some important features of these tasks: First, the questions are written by financial professionals using real-world data and financial knowledge. As such, they are closer to the kinds of questions that business and financial professionals answer as part of their workflows. They present different challenges from questions found in existing numerical reasoning datasets because they involve resolving complex quantity references, avoiding abundant distractor quantities, and using implicit financial background knowledge. These tasks are different from existing code generation tasks in that they require grounding generated code in real-world text or data structure context that also require financial background knowledge.

Secondly, the code we provide with these tasks is *semantically-rich*, by which we mean that the relationship between the code, the question it answers, and the context it uses is intuitive. This is accomplished through the use of a simple procedural style and descriptive variable names. Semantically-rich code explains the model's final answer, allowing model answers to be audited. Although we do not enforce the generation of semantically rich code, we find that in practice providing the model with semantically rich examples is sufficient to elicit this behavior.

Further details of the data collection process for the following tasks can be found in Appendix A.

FinCode consists of 137 questions, programs, and answers taken from Certified Financial Analyst (CFA) and Certified Public Accountant (CPA) practice exams. Each question is annotated with Python code that references quantities from the question text and applies mathematical operations to compute the requested numeric answer. The financial background knowledge required to answer these questions is requisite for professionals earning certification. In total, 46 of the 137 examples were written from scratch by financial professionals, and the remaining 91 were generated by an LLM and then verified by financial professionals.

Figure 2 shows a typical example from this dataset. The question statement discusses a potential investment project. Answering this question requires understanding domain-specific terms – such as "discount rate" and "NPV" (net present value) – and how the concepts fit together into a formula for determining the answer. The code to answer this question is complex, requiring multiple steps of financial arithmetic. See Table 2 for an analysis of the complexity of FinCode and the other BizBench datasets.

CodeFinQA is a subset of FinQA for which we provide code solutions to the questions. This dataset of 5,513 question/context/code triples can be used for finetuning or dynamic prompting. For the CodeFinQA task, models are given the same input as FinQA: text, table, and the question. The output is Python code which is executed to produce an answer. A sample is shown in Figure 3.

CodeTAT-QA is a subset of QA pairs from the TAT-QA dataset that can be answered using information in the provided table. We adapt this dataset for QA over structured financial data, a common task in business and finance workflows. We believe QA models can benefit from learning how to access

Context: *"... Q4 2022 revenue totaled 28.9B, compare to the same period last year of 27.8B. We saw stronger sales in our leasing division with a 14% increase ..."*
Question: *What was the percent change in revenue from 2021 to 2022?"*

Original FinQA derivation
```
1 divide(subtract(28.9, 27.8), 27.8)
```

Our Python Program
```
1 revenue_2022 = 28.9
2 revenue_2021 = 27.8
3 change = revenue_2022 - revenue_2021
4 percent_change = change / revenue_2021
5 answer = percent_change * 100
```

Figure 3: **CodeFinQA** example. Comparison of our added code solution and the original FinQA equation. Our Python program is executable with named variables for easier verification (lines 1-2), quantity composition (lines 3-4), and with the expected scale or unit (line 5).

Question: *What was the change in Foreign revenue in 2019 from 2018?*

Balance Sheet

Revenue, in millions	2019	2018	2017
Domestic	204.2	140.3	56.0
Foreign	11.8	19.9	14.2
Income before income taxes	216.0	160.2	70.2

Dataframe Access
```
1 df = DataFrame(data=balance_sheet_table)
2 foreign_2019 = df["Foreign"]["2019"]
3 foreign_2018 = df["Foreign"]["2018"]
4 answer = foreign_2019 - foreign_2018
```

Figure 4: Example from **CodeTAT-QA**. The task requires accessing and manipulating data via a dataframe, simulating QA scenarios where data comes from structured sources.

these data sources programmatically.

Each question is augmented not only with a program, but also with a structured table representation that the model can utilize to compute its solution. Information from the table can be accessed through the dataframe by specifying the row and column labels, as shown in Figure 4.

3.2 Quantity Extraction

Quantity extraction tasks require models to identify numbers in text and tables from natural language descriptions. Although this task can sometimes require minimal financial background knowledge (e.g. reading a number directly from a table), it is a necessary sub-task for many complex tasks that do. It is also a valuable task in its own right, as many business and finance workflows can benefit

from high precision automated quantity extraction. BizBench includes three quantity extraction tasks: a new dataset of SEC filings and labeled quantities (SEC-Num), and extraction-only subsets of TAT-QA and ConvFinQA.

SEC-Num is a novel dataset for quantity extraction from SEC filings. Recently, the SEC implemented a machine-readable labeling scheme for structuring data within human-readable documents.[1] Under these rules, filers are required to annotate quantities within reports with natural language descriptions of each quantity reported. We treat these descriptions as labels and define the SEC-Num task as follows: given a document snippet and a target label as input, the expected output is the quantity span from the snippet corresponding to the label. This open-vocabulary task generalizes Loukas et al. (2022), who focus on the most frequent labels and develop a classification task. A snippet of the original SEC filing is shown in Table 14 in the Appendix.

The data processing pipeline for SEC-Num begins with 202 10-K and 10-Q filings from the SEC EDGAR data portal. From these, we split each document into pages, each of which may contain multiple paragraphs and tables with a large number of quantities. For each unambiguous quantity label, we create a datapoint (x, y) where x is a snippet/label pair and y is the corresponding number from the snippet for the given label. The resulting dataset has 8,845 datapoints, which we split into 6,845 train and 2,000 test datapoints. Full statistics of this data are available in Table 1.

TAT-QA Extract (E) and **ConvFinQA Extract (E)** are subsets of questions from TAT-QA and ConvFinQA respectively, which can be answered using a numeric span from the context text or tables. See Table 1 for our dataset statistics.

3.3 Domain Knowledge

These tasks test the financial domain knowledge of an AI system. Here, models must demonstrate internal understanding of business and financial terms, practices, and formulae.

FinKnow contains 877 multiple choice questions and answers collected from CFA practice exams and the business ethics, microeconomics, and professional accounting exams from the MMLU dataset (Hendrycks et al., 2021b). The CFA exam questions have three potential choices, while the

Figure 5: Truncated example from **FormulaEval**. The model is prompted with the function signature and docstring and generates the code highlighted in cyan.

questions from the MMLU dataset have four. We exclude incomplete questions and questions that require numeric extraction or numerical reasoning. In total, this dataset contains 418 CFA, 86 business ethics, 224 microeconomics and 149 professional accounting question-answer pairs. We evaluate the models in a zero-shot setup. For each question, we compute the log probability of each potential answer and select the highest as the model's choice.

FormulaEval is a novel code-completion task designed to determine whether formulae for different business, economic, and financial measures are memorized and accessible without external knowledge sources. Using these formulae is required for the program synthesis tasks and is an important part of many business and financial workflow.

There are two main types of functions within this task: standalone functions and class functions. The standalone functions represent common financial formulas, such as computing the simple interest rate accrual on a loan. Many formulae involve reasoning about the structured relationships between a common set of items, such as computing EBITDA or Net Income from a balance sheet. To evaluate these types of formulae, we implement shared classes that represent financial documents (Balance Sheet, Income Statement, Statement of Cash Flows) with attributes representing items that you might find within these documents.

The model is given a function stub including a docstring and type hints. For the functions that are part of a class, the model is also given the class definition. The model is then tasked with completing the implementation of the function. An example of task input and expected output is shown in Figure

Model	Size	Domain Knowledge		Quantity Extraction			Program Synthesis			Avg.
		FinKnow	FormulaEval	ConvFinQA (E)	TAT-QA (E)	SEC-Num	FinCode	CodeFinQA	CodeTAT-QA	
		0-shot	0-shot	3-shot	3-shot	3-shot	8-shot	3-shot	3-shot	
Falcon	7B	40.9	14.0	66.5	62.9	27.3	3.3	2.0	7.4	28.0
Falcon	40B	43.9	6.0	82.8	80.6	52.3	8.3	18.4	38.5	41.3
MPT	7B	39.7	48.0	71.7	62.1	31.7	6.6	6.6	30.4	37.1
MPT	30B	42.4	60.0	85.5	81.5	56.4	6.6	31.0	64.8	53.5
StarCoder	16B	37.9	18.0	79.7	75.0	57.8	9.9	31.2	70.2	47.5
Llama 2	7B	41.7	52.0	86.1	83.1	56.2	7.4	21.9	37.0	48.2
Llama 2	13B	42.1	52.0	88.2	88.7	61.4	6.6	33.4	65.1	54.7
Llama 2	70B	44.9	80.0	92.8	94.4	74.7	24.0	57.3	79.1	68.4
CodeLlama	7B	36.5	52.0	82.4	77.4	53.1	10.7	34.0	70.9	52.1
CodeLlama	13B	37.3	56.0	87.0	81.5	61.3	9.9	39.1	82.1	56.8
CodeLlama	34B	40.0	70.0	88.1	83.9	67.0	17.4	52.4	81.4	62.5
Mistral	7B	44.4	66.0	91.5	87.9	65.1	15.7	48.8	75.0	61.8
Mixtral	8x7B	47.1	94.0	92.8	91.1	73.1	19.0	58.5	83.9	69.9
GPT 3*	175B	47.9	82.0	91.6	91.9	77.4	26.5	61.5	84.7	72.3
GPT 3.5*	-	60.3	76.0	92.4	84.2	76.0	36.1	67.5	87.6	72.5
GPT 4*	-	80.1	100.0	94.0	90.3	79.3	63.6	78.8	90.6	84.6

Table 3: Performance of state-of-the-art models on BizBench in zero-shot and few-shot in-context learning settings. The best performance for each task is in **bold.** The second-best performance is underlined. Closed-source models are represented by asterisks (*), whereas all other models are open-source.

5. In total, we collected 50 functions: 14 of which are standalone functions and 36 functions coming from four class implementations.

Evaluation is done by synthesized unit testing. Greedy model-generated and gold function implementations are compared by checking their outputs on 100 randomly sampled inputs. If their outputs match on all the inputs, we consider the model-generated function implementation to be correct. We report the overall accuracy on this task.

4 Few-Shot Experiments

We benchmark state-of-the-art open-source and proprietary models on the BizBench dataset using the EleutherAI LM Evaluation harness (Gao et al., 2022), establishing how performance ranges across model sizes and pretraining strategies. For large models, we provide zero- and few-shot results. We evaluate **Falcon** (Penedo et al., 2023), **MPT** (MosaicML, 2023), **StarCoder** (Li et al., 2023), **Llama-2** and **CodeLlama** (Touvron et al., 2023), **Mistral/Mixtral** (Jiang et al., 2023, 2024), and OpenAI's **GPT**(s) (Brown et al., 2020; OpenAI, 2023). Table 3 shows the models and their sizes.

We were unable to benchmark pre-existing financial LLMs due to three factors: The models we hoped to benchmark were either proprietary (e.g., BloombergGPT (Wu et al., 2023)), designed for languages other than English (e.g. BBT-Fin-T5 (Lu et al., 2023b) and XuanYuan 2.0 (Zhang and Yang, 2023)) or did not include code in their financial tuning and thus led to very poor results (e.g. FinMA (Xie et al., 2023), FinGPT (Yang et al., 2023), and Fin-LLaMA (William Todt, 2023)). BizBench requires models to be adept in both finance and coding. We discuss this limitation further in Section 7.

Model size and alignment impact performance: generally, larger models perform better, and models using RLHF and instruction tuning, such as GPT variants, perform best of all. There are notable outliers: Mistral 7B outperforms larger models–MPT, Falcon, and Llama 2–across a range of settings.

4.1 Few-Shot Error Analysis

We analyze how models err on program synthesis to better understand where they can improve. We categorize model errors into four categories: **extracting** the wrong number from context, using the wrong **formula**, or code **syntax** errors. Also, some questions are **ambiguous** about the desired answer format (e.g. decimal vs percentage). Our use of strict matching metrics causes some trivial errors.[2]

Upper Limits of Performance. GPT-4 is the best performing model on seven of the eight tasks benchmarked here. For the most difficult task – FinCode – GPT-4 achieves 27.5% absolute improvement over the next best model. Despite this, GPT-4 still fails to answer over 36% of the questions. We further analyzed these errors to better characterize the current limits in performance.

Of the 44 errors GPT-4 made, there were zero extraction or syntax errors. We deemed 4 errors to be due to potentially ambiguous questions. Of the formulaic errors, one error was the result of answering in the wrong units (trillions of dollars rather than

---

[2]Appendix B shows examples of these error types.

billions of dollars) and two were more basic math mistakes (Error counting number of months, and an error with order of operations). The remaining 37 errors all came from limits in GPT-4's business and financial knowledge. Examples of these errors can be found in Appendix B. This suggests that improving LLMs business and financial background knowledge is key to improving performance.

Comparing Code and Language Models. Llama-2 and CodeLlama models allow us to measure the benefits of further code training. This additional training results in improvements on our financial program synthesis benchmarks (by an absolute 16.4% and 8.6% for 7B and 13B respectively) albeit at the trade-off of slightly worse quantity extraction and domain knowledge performance.

To further understand these differences in performance, we categorize CodeFinQA errors found in 50 incorrect outputs from CodeLlama-34B, as well as 25 from each of CodeLlama-7B and Llama-2-7B.[3] The largest category of errors are numerical extraction errors, followed by incorrect formula errors. We find no clear difference in the error type distribution arising from parameters (CodeLlama-34B vs. CodeLlama-7B) or code pretraining (CodeLlama-7B vs. Llama-2-7B). Qualitatively, there are problems with scaling, e.g., confusing millions with billions or ratios with percentages. The models struggle with the semantics of gains and losses, particularly the sign of financial values. For example, the following all indicate a loss: *"loss of $100"*, *"-$100"*, *"($100)"*.

Models pretrained on language-only versus code solve different sets of questions. Llama-2-70B correctly answers 33% of the questions that CodeLlama-34B fails on; and CodeLlama-34B correctly answers 25% of those Llama-2-70B fails on. An oracle mixture of these models would achieve an accuracy of 68% on the CodeFinQA task.

To analyze the generated code complexity, we measure output length in lines and the number of parentheses for Llama-2-70B on FinCode compared to the values for the gold data shown in Table 2. While the average gold answer contains 7 lines of code, Llama-2-70B outputs contain 6.2, with incorrect responses at 6.4 and correct responses at 5.8 lines. Llama-2-70B may only answer questions where simpler code suffices. However, looking at the number of parentheses we find that Llama-2-70B outputs contain on average 4.0 com-
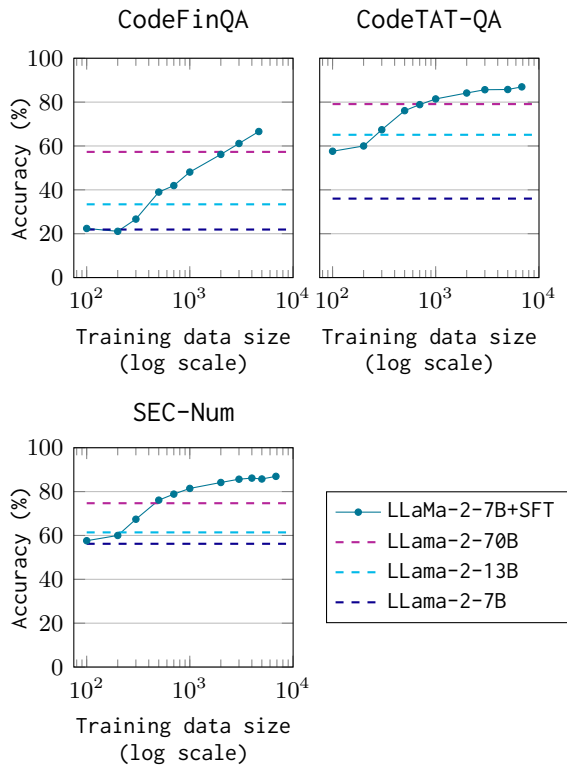

Figure 6: Performance of different finetuned Llama-2-7B models using increasing amounts of training data. Few-shot performance of various Llama 2 models is shown for comparison.

pared to 4.6 in the gold. Incorrect outputs have an average of 3.9 and correct outputs 4.2. Llama-2-70B may write more structured code when it understands the question better. Further analysis is needed to confirm this hypothesis.

5 Supervised Finetuning

We finetune Llama-2-7B in a number of settings to evaluate how the availability of both in-domain and multi-task training data impacts performance. Specifically, we investigate how different amounts of training data examples impact performance across CodeFinQA, CodeTAT-QA, and SEC-Num. To test this, we downsample the training data from 100 to 5,000 samples. Figure 6 shows the relationship between accuracy and training data size for finetuned Llama-2-7B models across these sizes compared to the few-shot results of the pretrained models. Next, we evaluate Llama-2-7B with multi-task training across these same datasets, CodeFinQA, CodeTAT-QA, and SEC-Num.

Analysis. Llama-2-7B requires fewer than 500 samples to outperform the Llama-2-13B on all three datasets. To outperform Llama-2-70B, Llama-

Source tasks			Target task		
			→ ★	→ ▲	→ ■
No SFT			21.9	37.0	56.2
Single			62.4	**86.5**	85.7
	▲	■	21.2	77.3	84.7
★		■	65.4	49.0	**87.3**
★	▲		65.1	81.9	69.9
★	▲	■	**66.6**	78.3	86.1

Table 4: Performance of LLaMa 2 7B model fine-tuned on different combinations of CodeFinQA(★), CodeTAT-QA(▲), and SEC-Num(■) in comparison with no supervised-finetuning (row 1) and supervised finetuning on the same task (row 2).

2-7B needs to be finetuned on only 3,000 samples for CodeFinQA, 1,000 samples for CodeTAT-QA, and only 500 samples for SEC-Num. Notably, when Llama-2-7B is finetuned with the full training datasets, it demonstrates a substantial improvement – achieving a 9%, 7%, and 13% higher accuracy on CodeFinQA, CodeTAT-QA, and SEC-Num compared to Llama-2-70B – while incurring only a fraction of the inference cost.

We study how knowledge is transferred across these three tasks. Table 4 shows a comparison between pretraining, supervised finetuning and multitask learning. In zero-shot transfer learning setting (e.g. training on SEC-Num and CodeFinQA and evaluating on CodeTAT-QA), the performance on CodeTAT-QA and SEC-Num improved with additional training data from other tasks, while performance on CodeFinQA slightly decreased. When the training data of the test task is included, data from other tasks has varying impact. Compared to SFT, the performance on CodeFinQA and SEC-Num increased consistently across all training mixes, while the performance on CodeTAT-QA decreased consistently.

6 Related Work

BizBench focuses on code generation and financial information extraction, complementing other financial NLP benchmark datasets which tend to focus on non-quantitative tasks (Malo et al., 2014; Sinha and Khandait, 2021), or quantitative trading tasks (Soun et al., 2022; Han et al., 2023).

Financial NLP. Non-quantitative tasks like sentiment detection (Malo et al., 2014), named entity recognition (Salinas Alvarado et al., 2015), classification (Sinha and Khandait, 2021), question answering (Maia et al., 2018), and boundary detection tasks (Au et al., 2021) are included in the financial benchmarks FLARE, introduced within PIXIU (Xie et al., 2023), and FLUE (Shah et al., 2022). Lu et al. (2023a) additionally include entity extraction, event extraction, and natural language to SQL task for Chinese. Recently, stock movement prediction (Soun et al., 2022) and pair trading (Han et al., 2023) were introduced as prediction tasks. BizBench complements these benchmarks by specifically focusing on quantitative reasoning via program synthesis, providing the first tasks of this kind within financial NLP.

Code generation. The APPS and HumanEval benchmarks test a model's ability to write code from arbitrary natural language specifications (Hendrycks et al., 2021a; Chen et al., 2021a). Austin et al. (2021) introduce MBPP, a collection of simple programming questions, as well as a Python version of the MathQA dataset introduced in Amini et al. (2019). We take inspiration from this approach to augmenting existing data when creating CodeTAT-QA and CodeFinQA. Like in our work, Lai et al. (2022) constructs a dataset of code solutions to data science problems which involve using libraries like numpy and pandas. Our coding tasks also involve reading comprehension, quantity manipulation, and financial knowledge.

Quantity extraction. Previous work explores extracting numbers, metric scales and semantic descriptions in financial (Loukas et al., 2022) and scientific documents (Harper et al., 2021; Elazar et al., 2019). Many financial QA datasets require precise quantity extraction such as HybridQA (Chen et al., 2020), TAT-QA (Zhu et al., 2021), Multi-HierTT (Zhao et al., 2022), FinQA (Chen et al., 2021b), and ConvFinQA (Chen et al., 2022). Our BizBench benchmark includes three datasets dedicated to testing numerical extraction capabilities, with our novel SEC-Num dataset encompassing values, scales, currency, dates, times, and periods.

7 Conclusion

Numerical reasoning is critically important for business and finance, as well as other domains, because small errors can incur large costs. To address this, we introduce BizBench, a benchmark for business and finance that measures models' abilities across three categories of relevant tasks: domain knowledge, quantity extraction, and program synthesis. The benchmark includes eight tasks, five of which provide novel data or meaningful extensions of existing datasets. We focus on program synthesis, as

success on these challenging tasks requires strong capabilities in both domain knowledge and quantity extraction. Additionally, we evaluate many state-of-the-art models, illustrating that models need improvement to meet the demands required for reasoning in real-world, high-stakes domains.

Acknowledgement

Limitations

This work presents new and reformulated data for evaluating the financial reasoning capabilities of LLMs. As mentioned in Section 3.1, portions of FinCode, CodeFinQA, and CodeTAT-QA were generated by LLMs, namely WizardCoder (Luo et al., 2023) and a GPT-3 variant, `text-davinci-003`. We assumed that the code is correct if it produced an answer that is identical or approximately identical to our gold truth answer. This approach could provide false positives (Python code that generated correct answers via an incorrect solution) and false negatives (Python code that had correct solutions by generated output that did not match our criterion). We have not manually reviewed all the data used to train these models, nor have we manually inspected all the model outputs included in BizBench. Therefore, we cannot make claims about the presence or absence of toxic, biased, or personal information in this data.

For tasks that involve providing context data to models, such data primarily consists of public records filed in accordance with strict government regulations. However, even under these regulations, biases may exist within the data. FinKnow and FinCode questions come from Internet sources and may contain bias. FormulaEval was created with assistance of financial professionals, and despite these professionals' expertise, the data could contain errors.

While BizBench aims to be comprehensive, it does not span the entire domains of business and finance. Moreover, BizBench only contains English text, limiting its applicability to finance problems in other languages. Various financial topics, documents, and problems – especially those concerning private markets – are not addressed by this bench-

mark. By evaluating the model's financial knowledge via it's ability to generate code, we constrain the set of models that can be fairly evaluated with our benchmark. Additionally, many financial professionals do not possess coding skills, implying that this may only be an artificial constraint.

Annotations included in BizBench were done by individuals with financial knowledge known to the authors. They volunteered to annotate and received no payment for their annotations. All annotators were made aware of the intended use of their data and consented to this use.

We assume that framing QA as program synthesis allows for easier auditing of model reasoning processes. This assumption is supported by the authors' experiences but may not hold in all settings.

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.

Willy Au, Abderrahim Ait-Azzi, and Juyeon Kang. 2021. Finsbd-2021: the 3rd shared task on structure boundary detection in unstructured text in the financial domain. In Companion Proceedings of the Web Conference 2021, pages 276–279.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. arXiv preprint arXiv:2108.07732.

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. arXiv.

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

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1026–1036, Online. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 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.

Yanai Elazar, Abhijit Mahabal, Deepak Ramachandran, Tania Bedrax-Weiss, and Dan Roth. 2019. How large are lions? inducing distributions over quantitative attributes. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3973–3983, Florence, Italy. Association for Computational Linguistics.

Leo Gao, Jonathan Tow, Stella Biderman, Charles Lovering, Jason Phang, Anish Thite, Fazz, Niklas Muennighoff, Thomas Wang, sdtblck, tttyuntian, researcher2, Zdeněk Kasner, Khalid Almubarak, Jeffrey Hsu, Pawan Sasanka Ammanamanchi, Dirk Groeneveld, Eric Tang, Charles Foster, kkawamu1, xagi-dev, uyhcire, Andy Zou, Ben Wang, Jordan Clive, igor0, Kevin Wang, Nicholas Kross, Fabrizio Milo, and silentv0x. 2022. EleutherAI/lm-evaluation-harness: v0.3.0.

Weiguang Han, Boyi Zhang, Qianqian Xie, Min Peng, Yanzhao Lai, and Jimin Huang. 2023. Select and trade: Towards unified pair trading with hierarchical reinforcement learning. In Proceedings of

the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '23, page 4123–4134, New York, NY, USA. Association for Computing Machinery.

Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval-2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 306–316, Online. Association for Computational Linguistics.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021a. Measuring coding challenge competence with apps. Advances in Neural Information Processing Systems.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. Proceedings of the International Conference on Learning Representations (ICLR).

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

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.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

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.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2022. Ds-1000: A natural and reliable benchmark for data science code generation. ArXiv, abs/2211.11501.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. 2023. Measuring faithfulness in chain-of-thought reasoning. arXiv preprint arXiv:2307.13702.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. StarCoder: may the source be with you! Transactions on Machine Learning Research.

Lefteris Loukas, Manos Fergadiotis, Ilias Chalkidis, Eirini Spyropoulou, Prodromos Malakasiotis, Ion Androutsopoulos, and Georgios Paliouras. 2022. FiNER: Financial numeric entity recognition for XBRL tagging. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4419–4431, Dublin, Ireland. Association for Computational Linguistics.

Dakuan Lu, Jiaqing Liang, Yipei Xu, Qi He, Yipeng Geng, Mengkun Han, Ying Xin, Hengkui Wu, and Yanghua Xiao. 2023a. Bbt-fin: Comprehensive construction of chinese financial domain pre-trained language model, corpus and benchmark. ArXiv, abs/2302.09432.

Dakuan Lu, Jiaqing Liang, Yipei Xu, Qianyu He, Yipeng Geng, Mengkun Han, Yingsi Xin, Hengkui Wu, and Yanghua Xiao. 2023b. Bbt-fin: Comprehensive construction of chinese financial domain pre-trained language model, corpus and benchmark.

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

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 1384–1403, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: financial opinion mining and question answering. In Companion proceedings of the the web conference 2018, pages 1941–1942.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. Journal of the Association for Information Science and Technology, 65(4):782–796.

MosaicML. 2023. MPT-30B: Raising the bar for open-source foundation models.

OpenAI. 2023. GPT-4 technical report.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for ccon llm: outperforming curated corpora with web data, and web data only. arXiv preprint arXiv:2306.01116.

Julio Cesar Salinas Alvarado, Karin Verspoor, and Timothy Baldwin. 2015. Domain adaption of named entity recognition to support credit risk assessment. In Proceedings of the Australasian Language Technology Association Workshop 2015, pages 84–90, Parramatta, Australia.

Raj Shah, Kunal Chawla, Dheeraj Eidnani, Agam Shah, Wendi Du, Sudheer Chava, Natraj Raman, Charese Smiley, Jiaao Chen, and Diyi Yang. 2022. When FLUE meets FLANG: Benchmarks and large pre-trained language model for financial domain. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 2322–2335, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ankur Sinha and Tanmay Khandait. 2021. Impact of news on the commodity market: Dataset and results. In Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 2, pages 589–601. Springer.

Yejun Soun, Jaemin Yoo, Minyong Cho, Jihyeong Jeon, and U Kang. 2022. Accurate stock movement prediction with self-supervised learning from sparse noisy tweets. In 2022 IEEE International Conference on Big Data (Big Data), pages 1691–1700. IEEE.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Findings of the Association for Computational Linguistics: ACL 2023, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. ArXiv.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023. Towards understanding chain-of-thought prompting: An empirical study of what matters. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.

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.

Pedram Babaei William Todt, Ramtin Babaei. 2023. Fin-llama: Efficient finetuning of quantized llms for finance. https://github.com/Bavest/fin-llama.

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.

Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2023. PIXIU: A large language model, instruction data and evaluation benchmark for finance. arXiv preprint arXiv:2306.05443.

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

Xuanyu Zhang and Qing Yang. 2023. Xuanyuan 2.0: A large chinese financial chat model with hundreds of billions parameters. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM '23, page 4435–4439, New York, NY, USA. Association for Computing Machinery.

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.

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   Data Collection

A.1   FinCode

This code was annotated by a combination of human and LLMs. For the first phase of data collection, we manually annotated a small set of examples. These examples were concatenated together to create a few-shot prompt that was provided to a GPT-3 variant, `text-davinci-003`, which generated candidate programs for the remaining questions. Programs were manually verified before being added to the dataset. This process was additionally used to identify the most challenging problems for further investigation. These problems were sent to financial professionals to solve by hand. These financial professionals were all full-time members of the finance or accounting staff at a fortune 500 company. After we manually converted these solutions into Python code, we repeated our initial bootstrapping process with the new set of examples. In total, we were able to collect 137 question and program pairs. Of these, 46 were written from scratch by financial professionals, and 91 were generated by an LLM and then verified by financial professionals.

A.2   CodeFinQA

To produce code solutions for these questions, we convert the FinQA equations into Python programs using a deterministic process. This process defines variables with dummy names (i.e., a, b, c) for numbers in the provided equation, and reformulates the operations from the equation into valid Python code over these variables. We then use a code-generation model – WizardCoder 15B (Luo et al., 2023) – to rewrite the programs to enhance readability. The prompt consists of input and output examples. Each input questions is paired with deterministic code, and the outputs are logical, human-written code with appropriate variable naming conventions.

This prompt is used to seed a bootstrapping process for data annotation. First, we find the deterministic programs which are most similar to our seed datapoints and rewrite them using the seeds as prompts. We verify that the rewritten program executes to produce the correct answer for its question. Subsequently, we find the most similar seed or previously rewritten datapoints for the remaining program until convergence. This processed produced a dataset of 5,513 question, context, code triples.

A.3   CodeTAT-QA

To create the code for CodeTAT-QA, we first convert the tables from text list format to Pandas DataFrames with named columns and rows. During this process, we flatten table hierarchy by attaching sub-table headers to corresponding row labels and normalize number representations, mirroring the action of contemporary document processing models. Information from the table can be accessed through the dataframe by specifying the row and column labels, as shown in Figure 4.

To produce programs for each datapoint, we start with a deterministic process to convert the arithmetic derivation provided in TAT-QA into valid Python code which accesses cells in the table using their row and column names. If this process identifies the values from the derivation in the table and verifies the provided answer, we pass the deterministically-generated program to a bootstrapping process similar to what was used to create CodeFinQA. Some derivations cannot be converted to programs which execute to the gold answer, and many questions from TAT-QA can not be treated as functions over DataFrames alone. This technique was able to produce code for 3,256 of the 16,000 questions of the TAT-QA dataset.

B   Error analysis

The full counts from our error categorization are shown in Table 5. We note that many outputs contain multiple error types.

We show two common error types i.e. incorrect data extraction and question misinterpretation in Figure 9 and Figure 10.

We additionally show two errors of GPT-4 on FinCode in Figure 7 and Figure 8.

C   Hyperparameters

To finetune experiments for CodeFinQA, CodeTAT-QA, and SEC-Num, we use Adam optimizer with an initial learning rate of 2e-6. The training process takes 3 epochs with a batch size of 32. The maximum token length is set to 2048. The model is finetuned on 8 x Nvidia A100-80GB GPUs. For both finetuned and pretrained models we use greedy decoding whenever applicable.

D   Task Examples

Examples of input and expected output for each Bizbench task is shown in the following tables and figures:

Model	*Samples*	**Extraction**	**Formula**	**Generation**	**Ambiguous Question**
CodeLlama-34B	50	29	23	2	4
Llama-2-7B	25	16	13	2	1
CodeLlama-7B	25	17	13	2	2

Table 5: Errors by category for different models.

Question: *Michael Poole is an equity analyst at Dave Associates, a financial services firm. Poole is estimating the firm's cost of equity using the dividend discount model approach. He has learnt that the Gordon's growth model is particularly useful in deriving the required rate of return when this approach is used. The company has paid a dividend of $2.5 per share in the previous year. The current market price per share is $25. The company's retention rate and return on equity is 40% and 10%, respectively. What is the cost of equity using the dividend discount model? Answer as a percentage to two decimal places.*

```
GPT-4 Error Case
1 dividend = 2.5
2 price_per_share = 25
3 retention_rate = 0.4
4 return_on_equity = 0.1
5 growth_rate = retention_rate * return_on_equity

  cost_of_equity = (dividend / price_per_share) *
      (1 + growth_rate)
  round(cost_of_equity * 100, 2)

  next_dividend_per_share =
    (dividend * (1 + growth_rate))
  cost_of_equity =
    (next_dividend_per_share / price_per_share) +
        growth_rate
  round(100.0 * cost_of_equity, 2)
```

Figure 7: GPT-4 Error Sample from FinCode. The model generated code is shown in pink with the ground truth shown in cyan. GPT-4 improperly uses the current dividend value rather than the following one. It additionally incorrectly combines the two ratios with multiplication rather than addition.

- FinKnow: See Table 6

- FormulaEval: See Table 7

- ConvFinQA Extract: See Table 8 and Chen et al. (2022)

- TAT-QA Extract: See Table 9 and Zhu et al. (2021)

- SEC-Num: See Table 10

- FinCode: See Table 11

- CodeFinQA: See Table 12

- CodeTAT-QA: See Table 13

Question: *Pine Co. purchased land for $450000 as a factory site. An existing building on the site was razed before construction began. Additional information is as follows: Cost of razing old building $60000 Title insurance and legal fees to purchase land $30000 Architect's fees $9 5000 New building construction cost $1850000 What amount should Pine capitalize as the cost of the completed factory building? Answer to the nearest dollar.*

```
GPT-4 Error Case
1 land_cost = 450000
2 razing_cost = 60000
3 title_insurance_and_legal_fees = 30000
4 architect_fees = 95000
5 construction_cost = 1850000

  total_cost = razing_cost + architect_fees +
      construction_cost
  round(total_cost)

  total_cost = architect_fees + construction_cost
  round(total_cost)
```

Figure 8: GPT-4 Error Sample from FinCode. The model generated code is shown in pink with the ground truth shown in cyan. The generated code incorrectly includes the razing costs into the capitalized cost for the building.

Question: *What portion of total company used area is company owned?*

Table input:
Sq. ft. in thousands	United States	Other countries	Total
Owned	4530	2417	6947
Leased	1037	1341	2378
Total	5567	3758	9325

```
1 # Incorrect program
2 owned_area = 4530
3 total_area = 9325
4 percent_owned = owned_area / total_area
```

```
1 # Golden program
2 company_owned_area = 6947
3 total_company_used_area = 9325
4 answer = company_owned_area /
      total_company_used_area
```

Figure 9: An example of extraction error in **CodeFinQA** task. Model-generated incorrect code is highlighted in pink and the golden code is highlighted in cyan. In this incorrect solution, the model extracted the owned area in the US (*owned_area = 4530*) while it should have extracted the total owned area (*company_owned_area=6947*).

CFA Economics	**Question**	*As a firm increases the quantity of its product produced, the distance between its ATC and AVC curve:*
	Choices	A. starts increasing. B. starts decreasing. C. remains constant.
	Answer	B
Microeconomics	**Question**	*Which of the following is a characteristic of monopolistic competition?*
	Choices	A. P > MC. B. Efficiency. C. Mostly price competition. D. P =MR.
	Answer	A
Ethics	**Question**	*The Theory of _____ posits that 3 three levels of moral reasoning exist which an individual can engage in to assess ethical issues, dependant on their cognitive capacity.*
	Choices	A. Egoism B. Cognitive moral development C. Power distance D. Uncertainty avoidance
	Answer	B
Finance	**Question**	*If a company engages in share repurchases, leverage will increase:*
	Choices	A. only if the repurchase is financed with debt. B. only if the repurchase is financed with excess cash. C. whether the repurchase is financed with debt or with excess cash.
	Answer	C

Table 6: Examples from **FinKnow** across different topics.

Context:

```python
1  @dataclass
2  class IncomeStatement:
3      """
4      Representation of a Income Statement. All attributes are listed in
5      dollars.
6      """
7
8      # Sales
9      revenue: float
10     cost_of_goods_sold: float
11     administrative_expenses: float
12     depreciation: float
13     amortization: float
14     # Income from non-core operations
15     other_investment_income: float
16     # Total tax burden in dollars
17     taxes: float
18     # Total interest expense in dollars
19     interest: float
20     shares_outstanding: float
21     current_share_price: float
22
23     def gross_profit(self):
24         return self.revenue - self.cost_of_goods_sold
25
26     def operating_expenses(self):
27         return self.administrative_expenses + self.depreciation + self.amortization
28
29     def operating_income(self):
30         return self.gross_profit() - self.operating_expenses()
31
32     def ebit(self):
33         return self.operating_income() + self.other_investment_income
34
35     def ebitda(self):
36         return self.ebit() + self.amortization + self.depreciation
37
38     def pretax_income(self):
39         return self.ebit() - self.interest
40
41     def net_income(self):
42         return self.pretax_income() - self.taxes
43
44     def market_capitalization(self):
45         return self.shares_outstanding * self.current_share_price
46
47     def earnings_per_share(self):
```

Output:

```python
48         return self.net_income() / self.shares_outstanding
```

Table 7: Example from **FormulaEval**. The model is given the class definition, docstrings, and some functions (L1-L47) to generate the missing return statement (L48).

Context:

DEVON ENERGY CORPORATION AND SUBSIDIARIES
NOTES TO CONSOLIDATED FINANCIAL STATEMENTS – (Continued)

Proved Undeveloped Reserves
The following table presents the changes in Devon's total proved undeveloped reserves during 2013 (in MMBoe).

		U.S.	Canada	Total
:—	:—	:—	:—	:—
	proved undeveloped reserves as of december 31 2012	407	433	840
	extensions and discoveries	57	38	95
	revisions due to prices	1	-10 (10)	-9 (9)
	revisions other than price	-91 (91)	13	-78 (78)
	conversion to proved developed reserves	-116 (116)	-31 (31)	-147 (147)
	proved undeveloped reserves as of december 31 2013	258	443	701

At December 31, 2013, Devon had 701 MMBoe of proved undeveloped reserves. This represents a 17 percent decrease as compared to 2012 and represents 24 percent of total proved reserves. Drilling and development activities increased Devon's proved undeveloped reserves 95 MMBoe and resulted in the conversion of 147 MMBoe, or 18 percent, of the 2012 proved undeveloped reserves to proved developed reserves. Costs incurred related to the development and conversion of Devon's proved undeveloped reserves were $1.9 billion for 2013. Additionally, revisions other than price decreased Devon's proved undeveloped reserves 78 MMBoe primarily due to evaluations of certain U.S. onshore dry-gas areas, which Devon does not expect to develop in the next five years. The largest revisions relate to the dry-gas areas in the Cana-Woodford Shale in western Oklahoma, Carthage in east Texas and the Barnett Shale in north Texas.

A significant amount of Devon's proved undeveloped reserves at the end of 2013 related to its Jackfish operations. At December 31, 2013 and 2012, Devon's Jackfish proved undeveloped reserves were 441 MMBoe and 429 MMBoe, respectively. Development schedules for the Jackfish reserves are primarily controlled by the need to keep the processing plants at their 35,000 barrel daily facility capacity. Processing plant capacity is controlled by factors such as total steam processing capacity, steam-oil ratios and air quality discharge permits. As a result, these reserves are classified as proved undeveloped for more than five years. Currently, the development schedule for these reserves extends though the year 2031.

Price Revisions

2013 – Reserves increased 94 MMBoe primarily due to higher gas prices. Of this increase, 43 MMBoe related to the Barnett Shale and 19 MMBoe related to the Rocky Mountain area.

2012 – Reserves decreased 171 MMBoe primarily due to lower gas prices. Of this decrease, 100 MMBoe related to the Barnett Shale and 25 MMBoe related to the Rocky Mountain area.

2011 – Reserves decreased 21 MMBoe due to lower gas prices and higher oil prices. The higher oil prices increased Devon's Canadian royalty burden, which reduced Devon's oil reserves.

Revisions Other Than Price

Total revisions other than price for 2013, 2012 and 2011 primarily related to Devon's evaluation of certain dry gas regions, with the largest revisions being made in the Cana-Woodford Shale, Barnett Shale and Carthage area.

Question:

What is the balance of proved undeveloped reserves in 2012 in US?

Answer: 407

Table 8: Example from **ConvFinQA**. The titles were bolded for readability of the text. We do not use any additional tokens to mark the title during prompting the models.

Table:			
-	2019%	2018%	2017%
Rate of inflation	2.9	2.9	3.0
Rate of increase in salaries	2.7	2.7	2.6
Discount rate	2.3	2.5	2.6

Question:

How much is the 2019 rate of inflation?

Answer: 2.9

Question:

How much is the 2018 rate of inflation?

Answer: 2.9

Question:

What is the 2019 average rate of inflation?

Answer: 2.9

Question:

What is the 2019 average rate of increase in salaries?

Answer: 2.7

Question:

What is the difference between 2019 average rate of inflation and 2019 average rate of increase in salaries

Answer: 0.2

Table 9: Example from **TAT-QA Extract**. Each table can have multiple associated question-answer pairs.

Context:

Contractual Obligations
The following table summarizes scheduled maturities of the Company's contractual obligations for which cash flows are fixed and determinable as of June 30, 2022:

—	—	—	—	—	—	—	—
				Payments Due in Fiscal			
(In millions)	Total	2023	2024	2025	2026	2027	Thereafter
Debt service (1)	$8,151	$429	$170	$665	$161	$661	$6,065
Unconditional purchase obligations (2)	4,742	2,852	705	637	132	133	283
Gross unrecognized tax benefits and interest – current (3)	2	2	—	—	—	—	—
Transition Tax payable(4)	215	27	42	65	81	—	—
Total contractual obligations(5)	$13,110	$3,310	$917	$1,367	$374	$794	$6,348

(1) Includes long-term and current debt and the related projected interest costs. Refer to Note 7 – Leases for information regarding future minimum lease payments relating to the Company's finance leases. **Interest costs on long-term and current debt** in fiscal 2023, 2024, 2025, **2026**, 2027 and thereafter are projected to be $174 million, $170 million, $165 million, $161 million, **$161 million** and $1,765 million, respectively. Projected interest costs on variable rate instruments were calculated using market rates at June 30, 2022.

(2) Unconditional purchase obligations primarily include: royalty payments pursuant to license agreements, inventory commitments, information technology contract commitments, capital expenditure commitments, advertising commitments and third-party distribution commitments. Future royalty and advertising commitments were estimated based on planned future sales for the term that was in effect at June 30, 2022, without consideration for potential renewal periods.

(3) Refer to Note 9 – Income Taxes for information regarding unrecognized tax benefits. As of June 30, 2022, the noncurrent portion of the Company's unrecognized tax benefits, including related accrued interest and penalties, was $73 million. At this time, the settlement period for the noncurrent portion of the unrecognized tax benefits, including related accrued interest and penalties, cannot be determined and therefore was not included.

(4) The Transition Tax may be paid over an eight-year period and this amount represents the remaining liability as of June 30, 2022.

(5) Refer to Note 7 – Leases for information regarding future minimum lease payments relating to the Company's operating leases.

Label:

Projected interest costs on long-term and current debt Due in fiscal 2026

Answer: 161

Table 10: Example from **SEC-Num**. The titles were bolded to enhance the readability of the text. We do not use any additional tokens to mark the title during prompting the models.

Question:

Mill Co. reported pretax income of $152500 for the year ended December 31. During the year-end audit the external auditors discovered the following errors: Ending inventory $30000 Overstated Depreciation expense $64000 What amount should Mill report as the correct pretax income for the year ended December 31? Answer to the closest dollar.

Program:

```
1 reported_income = 152500
2 ending_inventory = 30000
3 depreciation_expense = 64000
4 pretax_income = reported_income - ending_inventory - depreciation_expense
5 round(pretax_income)
```

Answer: 58500

Question:

Suppose that the market price of Company X is $45 per share and that of Company Y is $30. If X offers three-fourths a share of common stock for each share of Y, the ratio of exchange of market prices would be: Answer to three decimal places.

Program:

```
1 price_x = 45
2 price_y = 30
3 exchange_ratio = 3.0 / 4.0
4 ratio_of_exchange_market_prices = price_x / price_y * exchange_ratio
5 round(ratio_of_exchange_market_prices, 3)
```

Answer: 1.125

Question:

A bond is currently priced at 89.187 per 100 par value. If yields increase by 10bp, the value of bond falls to 88.215. However, if yields decrease by the same amount the value of the bond rises to 90.237. What will the approximate modified duration for the bond be? Answer to two decimal places.

Program:

```
1 price_1 = 89.187
2 price_2_rise = 90.237
3 price_2_fall = 88.215
4 yield_rise = 0.001
5 yield_fall = -0.001
6 modified_duration = (price_2_rise - price_2_fall) / (2 * price_1 * yield_rise)
7 round(modified_duration, 2)
```

Answer: 11.34

Table 11: Examples from **FinCode**. The model needs to generate the code highlighted in cyan.

Context:

NOTES TO CONSOLIDATED FINANCIAL STATEMENTS (continued)
ACE Limited and Subsidiaries

The following table shows changes in the Company's restricted stock for the years ended December 31, 2007, 2006, and 2005:

TABLE

	Number of Restricted Stock	Weighted Average Grant Date Fair Value
Unvested restricted stock December 31 2005	3488668	$41.26
Granted	1632504	$56.05
Vested and issued	(1181249)	$40.20
Forfeited	(360734)	$44.04
Unvested restricted stock December 31 2006	3579189	$48.07
Granted	1818716	$56.45
Vested and issued	(1345412)	$44.48
Forfeited	(230786)	$51.57
Unvested restricted stock December 31 2007	3821707	$53.12
Granted	1836532	$59.84
Vested and issued	(1403826)	$50.96
Forfeited	(371183)	$53.75
Unvested restricted stock December 31 2008	3883230	$57.01

END TABLE

Under the provisions of FAS 123R, the recognition of deferred compensation, a contra-equity account representing the amount of unrecognized restricted stock expense that is reduced as expense is recognized, at the date restricted stock is granted is no longer permitted. Therefore, upon adoption of FAS 123R, the amount of deferred compensation that had been reflected in unearned stock grant compensation was reclassified to additional paid-in capital in the company 2019s consolidated balance sheet.

Restricted stock units

The Company's 2004 LTIP also provides for grants of other awards, including restricted stock units. The Company generally grants restricted stock units with a 4-year vesting period, based on a graded vesting schedule. Each restricted stock unit represents the Company's obligation to deliver to the holder one share of Ordinary Shares upon vesting. During 2007, the Company awarded 108,870 restricted stock units to officers of the Company and its subsidiaries with a weighted-average grant date fair value of $56.29. During 2006, 83,370 restricted stock units, with a weighted-average grant date fair value of $56.36, were awarded to officers of the Company and its subsidiaries. During 2005, 80,550 restricted stock units, with a weighted-average grant date fair value of $44.59, were awarded to officers of the Company and its subsidiaries.

The Company also grants restricted stock units with a 1-year vesting period to non-management directors. Delivery of Ordinary Shares on account of these restricted stock units to non-management directors is deferred until six months after the date of the non-management directors' termination from the Board. During 2007, 29,676 restricted stock units were awarded to non-management directors. These units will vest in May 2008. During 2006, 23,092 restricted stock units were awarded to non-management directors. These units vested in May 2007. During 2005, 26,186 restricted stock units were awarded to non-management directors. These units vested in May 2006.

ESPP

The ESPP gives participating employees the right to purchase Ordinary Shares through payroll deductions during consecutive "Subscription Periods." Annual purchases by participants are limited to the number of whole shares that can be purchased by an amount equal to ten percent of the participant's compensation or $25,000, whichever is less. The ESPP has two six-month Subscription Periods, the first of which runs between January 1 and June 30 and the second of which runs between July 1 and December 31 of each year.

Question:

What is the net change in the number of unvested restricted stocks in 2007?

Program:

```
1   unvested_restricted_stocks_2006 = 3579189
2   unvested_restricted_stocks_2007 = 3821707
3   unvested_restricted_stocks_2008 = 3883230
4   change_in_unvested_restricted_stocks = unvested_restricted_stocks_2007 -
        unvested_restricted_stocks_2006
5   answer = change_in_unvested_restricted_stocks
```

Answer: 242518

Table 12: Examples from **CodeFinQA**. The titles were made bold to enhance the readability of the text. We do not use any additional tokens to mark the title during prompting the models. The model only needs to generate the code highlighted in cyan.

Table:

(in millions)	March 29, 2019	March 30, 2018
Net investment hedges – Foreign exchange forward contracts sold	116	-
Balance sheet contracts – Foreign exchange forward contracts purchased	963	697
Balance sheet contracts – Foreign exchange forward contracts sold	122	151

Question:

What is the change between Foreign exchange forward contracts purchased for March 29, 2019 and March 30, 2018?

Program:

```
1  table = {
2      "Net investment hedges -- Foreign exchange forward contracts sold": {
3          "March 29, 2019": 116,
4          "March 30, 2018": "-",
5      },
6      "Balance sheet contracts -- Foreign exchange forward contracts purchased": {
7          "March 29, 2019": 963,
8          "March 30, 2018": 697,
9      },
10     "Balance sheet contracts -- Foreign exchange forward contracts sold": {
11         "March 29, 2019": 122,
12         "March 30, 2018": 151,
13     },
14 }
15 df = pandas.DataFrame(data=table)
16 march_29_2019_foreign_exchange_balance_sheet_contract = \
17     df["Balance sheet contracts -- Foreign exchange forward contracts purchased"]["March 29, 2019"]
18 march_30_2018_foreign_exchange_balance_sheet_contract = \
19     df["Balance sheet contracts -- Foreign exchange forward contracts purchased"]["March 30, 2018"]
20 answer = march_29_2019_foreign_exchange_balance_sheet_contract - \
21     march_30_2018_foreign_exchange_balance_sheet_contract
```

Answer: 266

Table:

	2019	2018	2017
Domestic	204.2	140.3	56.0
Foreign	11.8	19.9	14.2
Income before income taxes	216.0	160.2	70.2

Question:

What was the change in Foreign in 2019 from 2018?

Program:

```
1  table = {
2      "Domestic": {
3          "2019": 204.2,
4          "2018": 140.3,
5          "2017": 56.0
6      },
7      "Foreign": {
8          "2019": 11.8,
9          "2018": 19.9,
10         "2017": 14.2
11     },
12     "Income before income taxes": {
13         "2019": 216.0,
14         "2018": 160.2,
15         "2017": 70.2
16     }
17 }
18 df = pandas.DataFrame(data=table)
19 foreign_2019 = df["Foreign"]["2019"]
20 foreign_2018 = df["Foreign"]["2018"]
21 answer = foreign_2019 - foreign_2018
```

Answer: -8.1

Table 13: Examples from **CodeTAT-QA** dataset. The model only needs to generate the code highlighted in cyan.

<div style="border:1px solid black; padding:10px">

Text

Contractual Obligations
The following table summarizes scheduled maturities of the Company's contractual obligations for which cash flows are fixed and determinable as of June 30, 2022:

—	—	—	—	—	—	—	—
			Payments Due in Fiscal			
(In millions)	Total	2023	2024	2025	2026	2027	Thereafter
Debt service (1)	$8,151	$429	$170	$665	$161	$661	$6,065
Unconditional purchase obligations (2)	4,742	2,852	705	637	132	133	283
Gross unrecognized tax benefits and interest – current (3)	2	2	—	—	—	—	—
Transition Tax payable(4)	215	27	42	65	81	—	—
Total contractual obligations(5)	$13,110	$3,310	$917	$1,367	$374	$794	$6,348

(1) Includes long-term and current debt and the related projected interest costs. Refer to Note 7 – Leases for information regarding future minimum lease payments relating to the Company's finance leases. Interest costs on long-term and current debt in fiscal 2023, 2024, 2025, 2026, 2027 and thereafter are projected to be $174 million, $170 million, $165 million, $161 million, $161 million and $1,765 million, respectively. Projected interest costs on variable rate instruments were calculated using market rates at June 30, 2022.

(2) Unconditional purchase obligations primarily include: royalty payments pursuant to license agreements, inventory commitments, information technology contract commitments, capital expenditure commitments, advertising commitments and third-party distribution commitments. Future royalty and advertising commitments were estimated based on planned future sales for the term that was in effect at June 30, 2022, without consideration for potential renewal periods.

(3) Refer to Note 9 – Income Taxes for information regarding unrecognized tax benefits. As of June 30, 2022, the noncurrent portion of the Company's unrecognized tax benefits, including related accrued interest and penalties, was $73 million. At this time, the settlement period for the noncurrent portion of the unrecognized tax benefits, including related accrued interest and penalties, cannot be determined and therefore was not included.
...

Table Preview

				Payments Due in Fiscal			
(In millions)	Total	2023	2024	2025	2026	2027	Thereafter
Debt service (1)	$8,151	$429	$170	$665	$161	$661	$6,065
Unconditional purchase obligations (2)	4,742	2,852	705	637	132	133	283
Gross unrecognized tax benefits and interest – current (3)	2	2	—	—	—	—	—
Transition Tax payable(4)	215	27	42	65	81	—	—
Total contractual obligations(5)	$13,110	$3,310	$917	$1,367	$374	$794	$6,348

SEC-annotated value-label pairs
{"value": "161", "label": "Projected interest costs on long-term and current debt Due in fiscal 2026"}
{"value": "65", "label": "Transition Tax payable, Payments Due in Fiscal 2025"}
{"value": "283", "label": "Unconditional purchase obligations, Payments Due in Fiscal Thereafter"}
{"value": "170", "label": "Projected interest costs on long-term and current debt Due in fiscal 2024"}
{"value": "215", "label": "Transition Tax payable"}
{"value": "637", "label": "Unconditional purchase obligations, Payments Due in Fiscal 2025"}
{"value": "42", "label": "Transition Tax payable, Payments Due in Fiscal 2024"}
{"value": "429", "label": "Debt service, Payments Due in Fiscal 2023"}
{"value": "81", "label": "Transition Tax payable, Payments Due in Fiscal 2026"}
{"value": "4,742", "label": "Unconditional purchase obligations"}
{"value": "132", "label": "Unconditional purchase obligations, Payments Due in Fiscal 2026"}
{"value": "27", "label": "Transition Tax payable, Payments Due in Fiscal 2023"}
{"value": "665", "label": "Debt service, Payments Due in Fiscal 2025"}
{"value": "161", "label": "Debt service, Payments Due in Fiscal 2026"}
{"value": "133", "label": "Unconditional purchase obligations, Payments Due in Fiscal 2027"}

</div>

Table 14: Original annotation from SEC filings for the ticker symbol **EL** (2022/06/30). Pairs with identical label are discarded. We present a preview of the table in markdown format for better readability.

Question: *What are the total proceeds from the issuance of employee options during February 2004, in millions?*

Context input:
In February 2004 , the company issued to eligible employees 1032717 options with an exercise price of $11.19 per share, the fair market value of the class a common stock on the date of grant ...

... the fair value of ATC Mexico plan options granted during 2002 were $3611 per share as determined by using the Black-Scholes option pricing model. As described in note 11, all outstanding options were exercised in march 2004.

```
1  # Incorrect program
2  proceeds_from_issuance_of_employee_options = 3611
3  answer = proceeds_from_issuance_of_employee_options
       / 1000000
```

```
1  # Golden program
2  options_issued = 1032717
3  exercise_price = 11.19
4  proceeds = options_issued * exercise_price
5  answer = proceeds / 1000000
```

Figure 10: An example where the model misunderstands the question and generates an incorrect solution with incorrect values. Model-generated incorrect code is highlighted in pink and the golden code is highlighted in cyan. The model, however, was able to detect the answer must be in millions.