TAT-LLM: A Specialized Language Model for Discrete Reasoning over Tabular and Textual Data

Fengbin Zhu[1,3], **Ziyang Liu**[3], **Fuli Feng**[2], **Chao Wang**[3], **Moxin Li**[1], **Tat-Seng Chua**[1]
[1]National University of Singapore, Singapore
[2]University of Science and Technology of China, China
[3]6Estates Pte Ltd, Singapore

Abstract

In this work, we address question answering (QA) over a hybrid of tabular and textual data, involving a variety of common content in reality like SEC filings, where discrete reasoning is often required. We consider harnessing the multi-step reasoning capabilities of large language models (LLMs) to tackle this problem, which have recently achieved remarkable success in many natural language tasks. To do this, we first abstract a *Step-wise Pipeline* for tabular and textual QA to help LLMs better execute multi-step inference, containing three key steps of *Extractor*, *Reasoner* and *Executor*. We initially design an instruction to validate the pipeline on GPT-4, demonstrating promising results. However, utilizing an online LLM like GPT-4 holds various challenges in terms of cost, latency, and data security risk, which motivates us to specialize smaller LLMs in this task. We then develop a TAT-LLM model by fine-tuning LLaMA 2 with the training data generated automatically from existing datasets following the *Step-wise Pipeline*. The experimental results have verified that our TAT-LLM model can outperform all compared models, including prior best fine-tuned models and very large-scale LLMs like GPT-4 on FinQA, TAT-QA and TAT-DQA benchmarks. The models and datasets are publicly available at https://huggingface.co/next-tat

1 Introduction

The documents containing both tables and text, e.g. SEC filings, academic papers and medical reports, make a very prevalent category of content in the real world. They often feature extensive numerical data in both the tabular and textual content, necessitating discrete reasoning capabilities for machines to comprehend them. Recent research (Zhu et al., 2021; Chen et al., 2021) investigates the intelligent comprehension of such documents through question answering (QA) tasks, as exemplified in Figure 1. The model, provided with a table and

Revenue from external customers, classified by significant product and service offerings, was as follows:

(in millions) Year Ended June 30,	2019	2018	2017
Server products and cloud services	32,622	26,129	21,649
Office products and cloud services	31,769	28,316	25,573
Windows	20,395	19,518	18,593
Gaming	**11,386**	**10,353**	9,051
Search advertising	7,628	7,012	6,219
LinkedIn	6,754	5,259	2,271
Enterprise Services	**6,124**	5,846	5,542
Devices	**6,095**	5,134	5,062
Other	3,070	2,793	2,611
Total	**125,843**	**110,360**	96,571

Our commercial cloud revenue, which includes Office 365 Commercial, Azure, the commercial portion of LinkedIn, Dynamics 365, and other commercial cloud properties, was $38.1 billion, $26.6 billion and $16.2 billion in fiscal years 2019, 2018, and 2017, respectively. These amounts are primarily included in Office products and cloud services, Server products and cloud services, and LinkedIn in the table above.

Question 1: What was the percentage change in gaming between 2018 and 2019?
Discrete Reasoning Type: Arithmetic Calculation
Answer: **(11,386 - 10,353) / 10,353 = 9.98%**

Question 2: Did the total revenue in 2019 exceed that of 2018?
Discrete Reasoning Type: Comparison
Answer: **125,843 > 110,360 = Yes**

Question 3: How many revenue items are between 6,000 million and 6,500 million in 2019?
Discrete Reasoning Type: Counting
Answer: **Count(Enterprise Services#Devices) = 2**

Figure 1: Examples of QA with discrete reasoning over a hybrid of tabular and textual data.

relevant text as the context, needs to perform various types of discrete reasoning, such as arithmetic calculations, making comparisons, and counting, to answer the question.

To perform QA over hybrid tabular and textual data, a straightforward approach (Ran et al., 2019) involves taking the table, text, and question as input and generating the answer directly. This approach can be ineffective due to the complex reasoning process involved (Wei et al., 2022c). To address this issue, some works (Lei et al., 2022; Zhou et al., 2022; Zhu et al., 2023) decompose the task into multiple steps, producing intermediate results that serve as references for the final answer. These multi-step approaches typically design distinct modules at each step and often optimize these modules concurrently through multi-task learning. To date, there has been no consensus on how to decompose the answer process in existing literature.

Recently, large language models (LLMs) like GPT-4 (OpenAI, 2023) and FLAN (Wei et al., 2022a) have exhibited strong multi-step reasoning abilities (Wei et al., 2022b) with proper instructions such as chain-of-thought (CoT) (Wei et al., 2022c) and least-to-most (Zhou et al., 2023). Therefore, we consider harnessing this amazing power of LLMs for better discrete reasoning over hybrid tabular and textual data. To achieve this, we first identify three key steps in the process of tabular and textual QA from previous multi-step methods (Zhu et al., 2021, 2023; Zhou et al., 2022), and abstract a *Step-wise Pipeline*, as illustrated in Figure 2 b). Specifically, 1) *Extractor* identifies the relevant information or evidence to the question from the given context; 2) *Reasoner* generates a mathematical equation or logic rule with the obtained information; and 3) *Executor* derives the final answer by executing the mathematical equation or logic rule with the associated information. The three steps emphasize different capabilities of the tabular and textual QA model — understanding the question and context, inferring the logic for answering the question, and calculating the answer with precision. These steps produce a sequence of intermediate results, which means we can specifically model and enhance one (or more) of them given a specific application scenario.

Following the *Step-wise Pipeline*, we initially design a task instruction and validate it on GPT-4 (OpenAI, 2023), achieving promising results on multiple benchmarks. However, utilizing an online LLM presents challenges in terms of cost, latency, and data security risk. By contrast, fine-tuning a smaller language model, specifically for math word problems (Fu et al., 2023; Cobbe et al., 2021), has been proven fairly appealing. We are then motivated to explore the specialization of smaller language models for addressing this challenge following the *Step-wise Pipeline*.

We develop a TAT-LLM model by fine-tuning LLaMA 2 (Touvron et al., 2023b) with the training data generated automatically from existing expert-annotated datasets, as shown in Figure 2 c). In particular, we train the selected LLM to take the table, text, and question as input, and complete the task in three key steps following the *Step-wise Pipeline*. We construct each of the training instances with five parts: Instruction, Table, Text, Question, and Response, from available tabular and textual QA datasets. Inspired by Tab-CoT (Ziqi and Lu, 2023), the model output (or response) is formatted to a structured table, with each row corresponding to one step of the pipeline, as shown in the right part of Figure 2 c). This design is friendly to further refining the outputs at each step and automatic evaluation. Moreover, to enhance the model performance, we equip the TAT-LLM model with an *External Executor*, which strengthens the execution of logical rules and mathematical calculations to better infer the final answer. We test our TAT-LLM on three popular benchmarks: FinQA (Chen et al., 2021), TAT-QA (Zhu et al., 2021) and TAT-DQA (Zhu et al., 2022). The experimental results show that our smallest model TAT-LLM (7B) can outperform all baseline models and even beat GPT-4 on all three datasets. In summary, we make the following main contributions in this work:

- We abstract a *Step-wise Pipeline* including *Extractor*, *Reasoner* and *Executor* to assist LLMs in better performing discrete reasoning over a hybrid of tabular and textual data.
- We develop a TAT-LLM model by fine-tuning LLaMA 2 (including 7B, 13B and 70B) for tabular and textual QA following the *Step-wise Pipeline*, better supporting practical application with cost and privacy concerns.
- We conduct extensive experiments on three popular benchmarks, validating the superiority of our TAT-LLM model over both conventional methods and very large-scale LMs, e.g. GPT-4.

2 Related Work

2.1 Tabular and Textual QA

Early works on tabular and textual QA (Chen et al., 2020b; Li et al., 2021; Chen et al., 2020a) focus on answer extraction from tabular and textual data. In recent two years, a growing body of works tackle tabular and textual QA by performing discrete reasoning as massive numerical information is usually included in the table or text, such as TAT-QA (Zhu et al., 2021), TAT-HQA (Li et al., 2022), FinQA (Chen et al., 2021), MultiHiertt (Zhao et al., 2022), and TAT-DQA (Zhu et al., 2022). In this line of research, many supervised fine-tuning methods are proposed, such as DyRRen (Li et al., 2023), RegHNT (Lei et al., 2022), UniRPG (Zhou et al., 2022) and MVGE (Wei et al., 2023). Harnessing

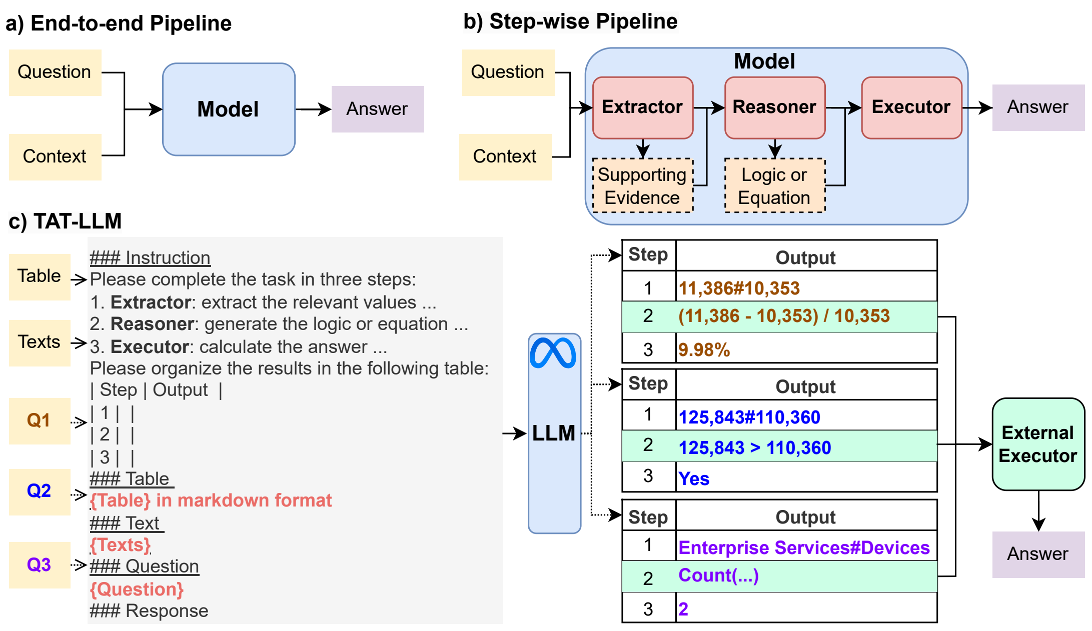

Figure 2: Comparison between a) *End-to-end Pipeline* and b) *Step-wise Pipeline*. c) Our TAT-LLM language model is developed by fine-tuning LLaMA 2 following the *Step-wise Pipeline*.

the capabilities of advanced LLMs to tackle tabular and textual QA remains a relatively underexplored area, which is the main focus of our work.

2.2 Large Language Models (LLMs)

Recently, LLMs like ChatGPT (Ouyang et al., 2022) have attracted tremendous research attention with their superb language understanding and generation abilities, bringing excellent performance to many tasks like reading comprehension (Rogers et al., 2023), discrete reasoning (Wei et al., 2022c), etc. Since these general LLMs are usually extremely large and not accessible for most researchers, open-source smaller LLMs are proposed to allow researchers to customize their own LLMs for different tasks, such as LLaMA (Touvron et al., 2023a) and Alpaca (Taori et al., 2023), with comparable performance to the general LLMs. Recently, there has been a trend in the research community that small LLMs are specialized for specific tasks through fine-tuning with instructions (Fu et al., 2023), yielding impressive performance. To the best of our knowledge, we are the first work to specialize a small LLM (e.g.,7B) in tabular and textual QA challenge.

3 Approach

In this section, we first introduce the *Step-wise Pipeline* for tabular and textual QA, and then elaborate our proposed TAT-LLM model.

3.1 *Step-wise Pipeline*

Large language models (LLMs) like ChatGPT (Ouyang et al., 2022) and FLAN (Wei et al., 2022a) have demonstrated astoundingly strong multi-step reasoning abilities (Wei et al., 2022b) following human instructions in many tasks (Zhou et al., 2023; Cobbe et al., 2021). To better leverage such amazing power of LLMs to address QA over hybrid tabular and textual data, we abstract a *Step-wise Pipeline* from the previous multi-step methods (Zhu et al., 2021; Zhou et al., 2022) on this task. As shown in Figure 2 b), this pipeline addresses the task with three key steps: *Extractor*, *Reasoner*, and *Executor*. In particular, *Extractor* serves as an information extraction module that identifies the relevant snippets or segments of information to the question from the context; 2) *Reasoner* works as a "logic thinker" to generate the right logic rule or equation that leads to the right answer to the question; 3) *Executor* is responsible for deriving the final answer by performing the logic rule or executing the equation. Compared with an *End-to-end Pipeline* that provides little insight into how the answer is derived, as shown in Figure 2 a), such a *Step-wise Pipeline* is able to produce a sequence of intermediate results, which means we can boost each component (or some of them) within the pipeline to consequently acquire improved final performance. For example, in this work, we choose to strengthen the third step by adding an extra Executor, which brings noticeable performance gains as verified experimentally in Section 4.3.

To instruct LLMs to perform discrete reasoning over tabular and textual data following the *Step-wise Pipeline*, we carefully design a natural language instruction to include the three key steps in the *Step-wise Pipeline*. Please refer to Appendix D for more information. We validate it on both FinQA and TAT-QA benchmarks using ChatGPT (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023) and obtain promising results, demonstrating the rationality of applying the *Step-wise Pipeline* grounded on the LLMs. We then generate training data with publicly available tabular and textual QA datasets and specialize a smaller LLM, i.e., LLaMA 2 (Touvron et al., 2023b), to answer questions following the *Step-wise Pipeline* for better supporting practical applications with cost and privacy concerns, as elaborated at below.

3.2 TAT-LLM

Selection of Language Model. We develop our TAT-LLM by fine-tuning LLaMA 2 (Touvron et al., 2023b). LLaMA (Touvron et al., 2023a) is a series of open-source large language models trained on the large-scale publicly available corpus. It is commonly employed in the development of large language models with instruction tuning, such as Alpaca (Taori et al., 2023) and Vicuna (Chiang et al., 2023). In this study, we choose the latest LLaMA 2 (Touvron et al., 2023b) as our base language model considering it offers an extended maximum sequence length (i.e., $4,096$) compared to the previous version LLaMA (Touvron et al., 2023a), which often fails to accommodate the tabular and textual content of one instance from existing datasets. Moreover, LLaMA 2 has demonstrated remarkable performance across various benchmarks.

Construction of Training Data. Our fine-tuning is based on the training data we automatically transform from existing tabular and textual QA datasets. For implementation, we use three datasets, i.e., FinQA (Chen et al., 2021), TAT-QA (Zhu et al., 2021) and TAT-DQA (Zhu et al., 2022). We carefully design templates for each dataset to generate the training data for fine-tuning LLMs. Please refer to Appendix C for more information. As shown in Figure 2 c), each training instance is designed to be composed of five parts, including Instruction, Table, Text, Question, and Response.

Algorithm 1 : External Executor

Input O_1: the output of *Extractor*; O_2: the output of *Reasoner*; O_3: the output of *Executor*; Q_t: the predicted question type.

```
 1: $answer \leftarrow O_3$
 2: **if** $O_2$ is a valid arithmetic equation **then**
 3:     $answer \leftarrow round(eval(O_2), 4)$
 4: **else if** "#" in $O_2$ **then** # multiple values are separated with "#"
 5:     $arr \leftarrow O_2.split(\text{"\#"})$
 6:     $answer \leftarrow len(arr)$
 7: **else if** ">" in $O_2$ or "<" in $O_2$ **then**
 8:     $answer \leftarrow eval(O_2)$
 9: **else if** $O_2$ = "N.A." **then**
10:     **if** $Q_t$ = "Span" **then**
11:         $answer \leftarrow O_1$
12:     **else if** $Q_t$ = "Multiple Spans" **then**
13:         $arr \leftarrow O_1.split(\text{"\#"})$ # spans are separated with "#"
14:         $answer \leftarrow arr$
15:     **end if**
16: **end if**
```

• **Instruction:** This part provides a detailed guide for what the task is and how to complete it. It outlines the steps that should be followed to derive the answer from the provided context. We adopt the *Step-wise Pipeline* described in above sections. 1) *Extractor*: We request the model to extract the relevant information from either the table or the accompanying text. 2) *Reasoner*: Based on the extracted information, we ask the model to derive an equation or a logic rule that is used to infer the answer. 3) *Executor*: We require the model to execute the equation or logic rule to arrive at the final answer. An illustration is given in Figure 2 c), which is the same for all the training instances. The instruction part helps the model to precisely understand the task and generate accurate and meaningful outputs.

• **Table**: This part refers to the tabular data in the input context of this task. We compose this part using the tables from FinQA and TAT-QA datasets. As each table in the two datasets is stored in a Two-Dimensional (2D) array, we transform it into a markdown table. Note that we omit TAT-DQA dataset for this part, considering the context given in TAT-DQA is the document pages from PDF files, where the structure of the table is unknown.

• **Text**: This part refers to the textual data in the input context. For FinQA and TAT-QA datasets, we compose this part with the paragraphs relevant

Dataset	Train Set	Dev Set	Test Set
FinQA	6,251	883	1,147
TAT-QA	13,215	1,668	1,669
TAT-DQA	13,251	1,645	1,662

Table 1: Statistics of the train, validation and test splits for the financial tabular and textual QA datasets.

to the table; all contents from the document pages in TAT-DQA dataset are regarded as textual data.

• **Question**: This part refers to the question that is asked based on the given tabular and textual context. We directly put the question from the three datasets in this part. The goal of the task is to generate the right answer to this question.

• **Response**: This part describes the model's output based on the given table, text, and question. To better inspect the results in the intermediate steps and automatically evaluate the model output, we train the language model to format the output in a structured markdown table with two columns, i.e., step and output. Each row in the output table corresponds to one step defined in the instruction. In the end, the model draws its conclusion with a statement: "The answer is: {answer}", making its final response clear.

To train the model, the response part in the training set needs to be generated in advance. Existing tabular and textual QA datasets often provide the answer to the question and the derivation annotated by human experts. To generate the correct response for constructing our training instance, we first automatically identify the supporting evidence from the derivation of each question, such as the numbers and entities, which are used as the output of the first step. We then convert the derivation to a valid equation or logic rule as the output of the second step, making sure its execution result is consistent with the final answer. We directly use the annotated answer as the final answer, which is also the output of the last step.

Training. After constructing the training data, we train our TAT-LLM model in various sizes, including 7B, 13B, and 70B, by fine-tuning LLaMA 2 (Touvron et al., 2023b) using Low-Rank Adaptation (LoRA) (Hu et al., 2022). More details are provided in Appendix A.

External Executor. We observe that the trained model struggles in performing the execution of the mathematical equations and logic rules (e.g., counting, comparison) in the last step, according to

Type	Model	EM
Human Expert Performance		91.16
Fine-tuned	Longformer	21.90
	NeRd	48.57
	FinQANet$_{BERT}$	50.00
	DyRRen$_{BERT}$	59.37
	FinQANet$_{RoBERTa}$	61.24
	ELASTIC$_{RoBERTa}$	62.66
	DyRRen$_{RoBERTa}$	63.30
Zero-shot	Vicuna (7B)	10.11
	LLaMA 2-Chat (7B)	15.43
	LLaMA 2-Chat (70B)	32.17
	MAmmoTH (70B)	36.09
	WizardMath (70B)	47.25
	GPT3.5-Turbo	58.00
	GPT-4	<u>63.91</u>
Ours	TAT-LLM (7B)	(+0.69) **64.60**

Table 2: Performance of our TAT-LLM model and compared models on the test set of FinQA. Best results are marked in bold and numbers in red indicate the improvement over the underlined second-best results.

experimental evaluations in Section 4.3. Such incompetence of the Executor has also been found in previous works (Luo et al., 2023; Yue et al., 2023). To enhance the model's accuracy of the final output, we propose to add an *External Executor* to our model, which takes the model's intermediate outputs as input and performs the execution of the equations or logic rules to obtain the final answer. With this *External Executor*, the model can refine its output to a better one. With the output of the model formatted to a structured table, we can easily access the intermediate results. After obtaining the intermediate results, the *External Executor* is applied to refine the final answer instead of directly using the prediction of the model. The whole process is summarized in Algorithm 1.

4 Experiments

4.1 Datasets, Models and Evaluation Metrics

Datasets. We use FinQA (Chen et al., 2021), TAT-QA (Zhu et al., 2021) and TAT-DQA (Zhu et al., 2022) for our experiments. See Table 1 for the statistics of splits of each dataset.

• **FinQA** (Chen et al., 2021) is an expert-annotated tabular and textual QA dataset in which the tables and text are sampled from financial reports.

Type	Model	EM	F_1
Human Expert Performance		84.1	90.8
Fine-tuned	TagOp	50.10	58.00
	TeaBReaC	55.80	63.80
	KIQA	58.20	67.40
	FinMath	58.30	68.20
	GANO	61.90	72.10
	MHST	63.60	72.70
	UniPCQA	63.90	72.20
	SoarGraph	65.40	75.30
	UniRPG	67.20	76.00
	RegHNT	70.30	77.90
	MVGE	70.90	79.10
Zero-shot	Vicuna (7B)	32.53	40.97
	LLaMA 2-Chat (7B)	37.16	45.37
	MAmmoTH (70B)	38.97	46.51
	WizardMath (70B)	39.63	45.28
	LLaMA 2-Chat (70B)	45.94	53.80
	GPT3.5-Turbo	59.47	68.11
	GPT-4	<u>71.92</u>	<u>79.71</u>
Ours	TAT-LLM (7B)	(+2.64) **74.56**	(+3.17) **82.88**

Table 3: Performance of our TAT-LLM model and compared models on the test set of TAT-QA.

It focuses on discrete reasoning capabilities like addition, subtraction, multiplication, division, and numerical comparison.

• **TAT-QA** (Zhu et al., 2021) is also built with tables and paragraphs extracted from financial reports. Most questions require discrete reasoning to generate the answers, and meanwhile, there are also cases where the answers can be extracted directly from the tables or text. Its questions are classified into four different types: *Span*, *Multiple Spans*, *Counting*, and *Arithmetic*.

• **TAT-DQA** (Zhu et al., 2022) is an extension of the TAT-QA dataset, focusing on question answering over the original long financial statements with up to three pages, and the position and structure of the tables are unknown.

Compared Models. We compare our TAT-LLM model with two kinds of models: fine-tuned models on the tabular and textual QA dataset, and LLMs in zero-shot setting. For fine-tuned models, we select the state-of-the-art supervised fine-tuned models for each dataset, which are listed separately in Table 2, Table 3 and Table 4. For LLMs, we utilize the state-of-the-art GPT3.5-Turbo (Brown et al., 2020) and GPT-4 (OpenAI, 2023). Besides, we also adopt powerful smaller LLMs that have garnered considerable research interest in the field, including Vicuna (7B) (Chiang et al., 2023) and LLaMA 2-Chat (7B and 70B) (Touvron et al., 2023b). In addition to these general LLMs, we

Type	Model	EM	F_1
Human Expert Performance		84.1	90.8
Fine-tuned	NumNet+ V2	30.60	40.10
	TagOp	33.70	42.50
	MHST	41.50	50.70
	Doc2SoarGraph	59.20	67.60
Zero-shot	Vicuna (7B)	28.44	36.72
	LLaMA 2-Chat (7B)	34.52	42.32
	MAmmoTH (70B)	35.42	42.82
	WizardMath (70B)	36.44	41.55
	LLaMA 2-Chat (70B)	41.91	49.74
	GPT3.5-Turbo	52.74	61.40
	GPT-4	<u>64.46</u>	<u>72.20</u>
Ours	TAT-LLM (7B)	(+4.99) **69.45**	(+5.55) **77.75**

Table 4: Performance of our TAT-LLM model and compared models on the test set of TAT-DQA.

also compare our models with LLMs specialized in math word problems, including MAmmoTH (70B) (Yue et al., 2023) and WizardMath (70B) (Luo et al., 2023), since our task involves numerical reasoning. All LLMs are tested in zero-shot setting, because financial tabular and textual data inputs tend to be lengthy, making it impractical to include additional in-context examples due to input length limits.

Evaluation Metrics. For TAT-QA and TAT-DQA datasets, we adopt the Exact Match (EM) and the numeracy-focused (macro-averaged) F_1 score (Zhu et al., 2021, 2022). Both two metrics measure the overlap between a bag-of-words representation of the gold and predicted answers. The numeracy-focused F_1 score is set to 0 unless the predicted number is exactly equal to the ground truth. We use EM for FinQA, which is the same as the metric of Execution Accuracy originally used in FinQA (Chen et al., 2021).

For zero-shot prediction using LLMs, we omit the scale prediction and compare the predicted value with the ground truth value only, assuming the scale prediction is always correct. For our TAT-LLM model, since we instruct the model to output a structured table that is friendly to automatic evaluation, we take the scale into account.

4.2 Main Results

We first compare the performance of our TAT-LLM with previous methods on each dataset respectively. The experimental results are summarized in Table 2 for FinQA, Table 3 for TAT-QA and Table 4 for TAT-DQA. From the tables, we make the following observations. 1) Our TAT-

Model	FinQA	TAT-QA		TAT-DQA	
	EM	EM	F_1	EM	F_1
GPT-3.5-Turbo	58.00	59.47	68.11	52.74	61.40
GPT-4	63.91	71.92	79.71	64.46	72.20
Fine-tuned with respective training set:					
TAT-LLM (7B)	(+0.69) 64.60	(+2.64) 74.56	(+3.17) 82.88	(+4.99) 69.45	(+5.55) 77.75
Fine-tuned with a combination of training sets:					
TAT-LLM$_{All}$ (7B)	(+1.22) 65.13	(+4.57) 76.49	(+5.42) 85.13	(+6.92) 71.38	(+8.04) 80.24
TAT-LLM$_{All}$ (13B)	(+8.02) 71.93	(+5.59) 77.51	(+6.24) 85.95	(+7.76) 72.22	(+8.36) 80.56
TAT-LLM$_{All}$ (70B)	(+12.90) **76.81**	(+9.50) **81.42**	(+8.78) **88.49**	(+12.09) **76.55**	(+11.70) **83.90**

Table 5: Performance of the TAT-LLM model trained with combination of all three training sets.

Model	FinQA	TAT-QA		TAT-DQA	
	EM	EM	F_1	EM	F_1
GPT-4	63.91	71.92	79.71	64.46	72.20
TAT-LLM$_{All}$ (7B)					
w/o *External Executor*	48.47	58.69	67.21	54.84	63.68
w *External Executor*	65.13	76.49	85.13	71.38	80.24
gains (+)	16.66	17.80	17.92	16.54	16.56
TAT-LLM$_{All}$ (13B)					
w/o *External Executor*	60.05	62.60	70.73	59.95	68.61
w *External Executor*	71.75	76.79	85.05	71.86	80.50
gains (+)	11.70	14.19	14.32)	11.91	11.89
TAT-LLM$_{All}$ (70B)					
w/o *External Executor*	70.10	76.61	83.55	71.74	78.99
w *External Executor*	**76.81**	**81.42**	**88.49**	**76.55**	**83.90**
gains (+)	6.71	4.81	4.94	4.81	4.91

Table 6: Effectiveness of the *External Executor* for TAT-LLM$_{ALL}$ with different sizes.

LLM (7B) significantly outperforms all the previous models on each of the three datasets, including the previous best fine-tuned models and the state-of-the-art GPT-4 (OpenAI, 2023). In particular, our TAT-LLM reaches 64.60%, 74.56% and 69.45% in terms of EM on the test set of FinQA, TAT-QA and TAT-DQA respectively, i.e. an increase of 0.69, 2.64 and 4.99 points compared to GPT-4. These results well demonstrate the noticeable effectiveness of our TAT-LLM model. The results also confirm the rousing potential of specializing smaller language models, like our TAT-LLM (7B) model, for specific tasks to yield better performance than very large-scale models like GPT-4. 2) However, the performance of GPT-4 and our TAT-LLM (7B) obviously lags behind that of human experts, showing that this task is still challenging. 3) The strong general LLM GPT3.5-Turbo (Ouyang et al., 2022) and LLaMA 2-Chat (70B) (Touvron et al., 2023b) underperform the best fine-tuned models, evidencing that supervised fine-tuning is still essential for achieving advanced performance for this challenge. 4) The LLMs specialized in mathematical reasoning, i.e., WizardMath (Luo et al., 2023) and MAmmoTH (Yue et al., 2023), largely underperform GPT3.5-Turbo, GPT-4 and our TAT-LLM model, showing that current numerically-enhanced LLMs still struggle in discrete reasoning over tabular and textual QA. This again speaks for the considerable value of specializing models for specific tasks.

4.3 In-Depth Analysis

Fine-tuning with Combined Data. In addition to training our TAT-LLM model on a single tabular and textual QA dataset, we also train the TAT-LLM$_{All}$ (7B) with a combination of the train sets from FinQA, TAT-QA and TAT-DQA datasets. As shown in Table 5, training on a combined dataset brings clear performance improvements over training on a single dataset. This suggests that incorporating larger-sized and more diverse data into the training process can potentially enhance the model's overall capabilities. Furthermore, we check the performance with various sizes of the base LLaMA 2 model, and find that larger models consistently achieve significant performance improvements across all three datasets, aligning with the prevailing consensus in the field. Compared to GPT-4, TAT-LLM$_{ALL}$ (70B) increases 12.90%, 9.50% and 12.09% on the EM over FinQA, TAT-QA and TAT-DQA, respectively, which strongly validates the rationality of our solution.

Different Fine-tuning Strategies. To verify the effect of our fine-tuning strategy, i.e. *Step-wise Pipeline* plus *External Executor*, we compare it with separately applying *End-to-end Pipeline* or *Step-wise Pipeline* on LLaMA 2 (7B) models. The results are summarized in Figure 3. Firstly, we can see our TAT-LLM (7B) clearly outperforms the two variants following the *End-to-end Pipeline* and the *Step-wise Pipeline* on all tabular and textual QA datasets for both evaluation metrics, demonstrating the effectiveness of our fine-tuning strategy. Besides, we find that the *Step-wise Pipeline* also outperforms the *End-to-end Pipeline* in all situations, showing the rationality of incorporating interme-

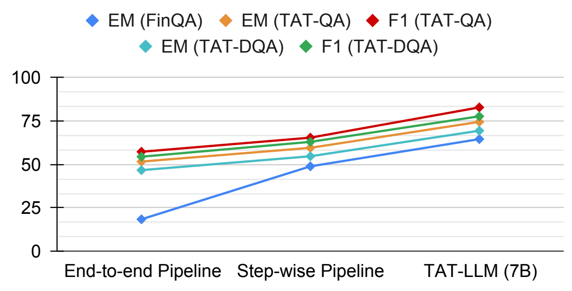

Figure 3: Comparison of different training strategies.

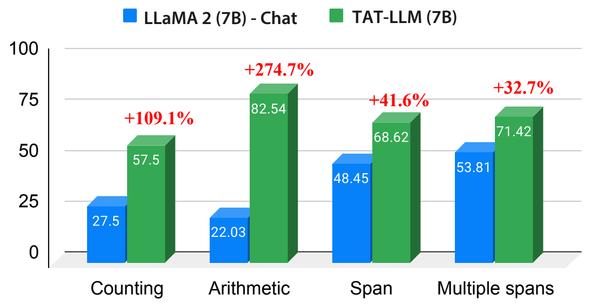

Figure 4: Performance comparison in terms of EM between TAT-LLM (7B) and LLaMA 2-Chat (7B) for different question types on TAT-QA.

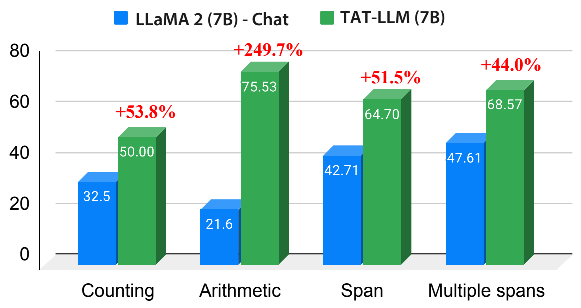

Figure 5: Performance comparison in terms of EM between TAT-LLM (7B) and LLaMA 2-Chat (7B) for different question types on TAT-DQA.

diate reasoning steps, which is in line with previous research on chain-of-thoughts reasoning (Wei et al., 2022c). Moreover, it is observed that the *End-to-end Pipeline* achieves notably poor performance on FinQA compared to that on TAT-QA and TAT-DQA. This is probably because FinQA requires more complex discrete reasoning steps than TAT-QA and TAT-DQA, and omitting such steps in fining-tuning would largely degrade model performance. Finally, we find a significant performance increase from the *Step-wise Pipeline* to TAT-LLM. This is due to the incorporation of the reliable *External Executor* which largely guarantees the reasoning accuracy of the answer. In the next section, we perform detailed analysis on the effectiveness of the *External Executor*.

Effect of *External Executor* w.r.t. Different Model Sizes. Here we analyze the effectiveness of the *External Executor* on models with different sizes. According to the experimental results shown in Table 6, we can see that ablating the *External Executor* significantly degrades the performance across all model sizes on all datasets, causing decreases of larger than 15% on the EM of the three datasets for TAT-LLM$_{ALL}$ (7B). This shows that the *External Executor* is essential in ensuring the answer correctness. Also, we find that larger models w/o *External Executor* perform better than smaller models w/o *External Executor*, and the gains by the *External Executor* get smaller as the model size increases. The largest TAT-LLM$_{ALL}$ (70B) w/o *External Executor* only decreases less than 7% on all three datasets, which is far less than TAT-LLM$_{ALL}$ (7B). This is because larger models have stronger discrete reasoning abilities than smaller models and make less mistakes in answer calculation. Hence, the advantage of ensuring accurate results in discrete reasoning through the *External Executor* appears to be less conspicuous.

Improvements w.r.t. Different Question Types. Here we analyze the improvements achieved by TAT-LLM model regarding different question types of TAT-QA and TAT-DQA datasets. We compare the performance of TAT-LLM (7B) and LLaMA 2-Chat (7B) for each question type. The results are summarized in Figure 4 and Figure 5 for TAT-QA and TAT-DQA, respectively. We can observe substantial performance improvements for the *Arithmetic* questions, which are over two-fold, for both datasets. The larger performance gains are attributed to the effective fine-tuning strategy of TAT-LLM as well as the reliable *External Executor* that ensures accurate execution of the equations and logic rules. Large performance improvements are also found for the *Counting* questions, namely 109.1% for TAT-QA and 53.8% for TAT-DQA. This is probably due to executing more correct counting operations. For the *Span* and *Multiple Spans* types, relatively smaller improvements are observed, which are still over 30%. This is potentially because these questions involve simpler or no discrete reasoning, mainly numerical comparison.

5 Conclusion

In this work, we first abstract a *Step-wise Pipeline* for tabular and textual QA based on the previous multi-step methods. Following this pipeline, we develop TAT-LLM model by specializing smaller language models (i.e., 7B) in discrete reasoning over tabular and textual data. We validate its effectiveness with extensive experiments, showing that our TAT-LLM model can outperform both conventional methods and very large-scale LMs like GPT-4 on this task. Our work well demonstrates that specialized models might be a promising direction towards more advanced models on specific tasks that can compete with human experts.

Limitations

Despite the impressive performance on all three datasets i.e., FinQA (Chen et al., 2021), TAT-QA (Zhu et al., 2021), and TAT-DQA (Zhu et al., 2022), our TAT-LLM model still has much room for further improvement, as shown in error analysis in Appendix B. In addition, our TAT-LLM model is designed for the documents that contain both tabular and textual data and feature rich numerical values. This means it may have limited advantages over other kinds of documents like pure textual documents. Also, our model may not be directly applied to understand the documents with a large number of pages (e.g., >100 pages) due to the constraint of the maximum input sequence length.

Ethics Statement

In this work, we first abstract a *Step-wise Pipeline* for tabular and textual QA. Then, we propose the TAT-LLM model by specializing a smaller language model (i.e., LLaMA 2 7B, 13B and 70B) in discrete reasoning over a hybrid of tabular and textual data. Our TAT-LLM model is developed on open-source tools and datasets to assist human beings in processing and understanding such kind of data. Thus, we do not anticipate any potential risks or negative ethical issues.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W Cohen. 2020a. Open question answering over tables and text. In *International Conference on Learning Representations*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020b. 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.

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

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

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10421–10430. PMLR.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Fangyu Lei, Shizhu He, Xiang Li, Jun Zhao, and Kang Liu. 2022. Answering numerical reasoning questions in table-text hybrid contents with graph-based encoder and tree-based decoder. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1379–1390. International Committee on Computational Linguistics.

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

Xiao Li, Yawei Sun, and Gong Cheng. 2021. Tsqa: tabular scenario based question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13297–13305.

Xiao Li, Yin Zhu, Sichen Liu, Jiangzhou Ju, Yuzhong Qu, and Gong Cheng. 2023. Dyrren: A dynamic retriever-reranker-generator model for numerical reasoning over tabular and textual data. AAAI Press.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei

Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct.

OpenAI. 2023. 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.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine reading comprehension with numerical reasoning. In *EMNLP-IJCNLP*, pages 2474–2484.

Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2023. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *ACM Comput. Surv.*, 55(10).

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models.

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

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022c. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Yifan Wei, Fangyu Lei, Yuanzhe Zhang, Jun Zhao, and Kang Liu. 2023. Multi-view graph representation learning for answering hybrid numerical reasoning question.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning.

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. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Yongwei Zhou, Junwei Bao, Chaoqun Duan, Youzheng Wu, Xiaodong He, and Tiejun Zhao. 2022. Unirpg: Unified discrete reasoning over table and text as program generation. *arXiv preprint arXiv:2210.08249*.

Fengbin Zhu, Wenqiang Lei, Fuli Feng, Chao Wang, Haozhou Zhang, and Tat-Seng Chua. 2022. Towards complex document understanding by discrete reasoning. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4857–4866.

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. Association for Computational Linguistics.

Fengbin Zhu, Moxin Li, Junbin Xiao, Fuli Feng, Chao Wang, and Tat Seng Chua. 2023. Soargraph: Numerical reasoning over financial table-text data via semantic-oriented hierarchical graphs. In *Companion Proceedings of the ACM Web Conference 2023*, page 1236–1244. Association for Computing Machinery.

Jin Ziqi and Wei Lu. 2023. Tab-CoT: Zero-shot tabular chain of thought. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10259–10277. Association for Computational Linguistics.

A Implementation Details

Since the question type and the scale of the answer have been annotated in both TAT-QA and TAT-DQA datasets, we add two more steps in the instruction part of the input instance, in addition to the three steps in the proposed *Step-wise Pipeline* shown in Figure 2 c). One step is Question Type Predictor, with which we limit the model to select one value from the following question types: *Span*, *Multiple Spans*, *Counting*, and *Arithmetic*. For *Span* and *Multiple Spans* questions, we set the output of the *Reasoner* to "N.A.". The other step is Scale Predictor, with which we restrict the model to choose one from the following values: *Thousand*, *Million*, *Billion*, *Percent* and *None*. For all the templates used to construct training instances of our TAT-LLM model, please refer to Appendix C.

We train our TAT-LLM model on one NVIDIA DGX-A100 with eight A100 GPUs. The quantization is 8 bit. We use Adam optimizer with a learning rate of $3e-4$ and warmup over the first 3% steps to train. The maximum sequence length is $4,096$ and the maximum number of epochs is set to 3. The batch size is set to 4 and the gradient accumulation step is 10.

For comparing with fine-tuned models, we take the results from their original papers respectively. For each LLM, we utilize three different templates to perform zero-shot inference to obtain prediction results. We select the best result as the reported result in Section 4.2. Please refer to Appendix D for details of the templates we use for zero-shot inference. We utilize the latest version[1] GPT3.5-Turbo (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023) via OpenAI APIs. We set the temperature as 0, top p as 1.0, max token as $1,000$, and other parameters as default. We obtain the official trained checkpoint of Vicuna (Chiang et al., 2023), LLaMA 2-Chat (Touvron et al., 2023b), MAmmoTH (Yue et al., 2023) and WizardMath (Luo et al., 2023) from Hugginface. The inference is done on one NVIDIA DGX-A100 with eight A100 GPUs. The parameters `num_beam` and `do_sample` are 1 and false respectively.

B Error Analysis

To further diagnose our TAT-LLM model, we randomly sample 100 error instances of TAT-LLM (7B) from the test set of FinQA and analyze

[1]Date: Sep 2023

Step	Error(%)	Example
Extractor	Wrong Evidence (48%)	**Q**: What is the percentage change in cash flow hedges in 2011 compared to 2010? **G**: 153.7, 139.9 **P**: 153.7, 375.0
	Missed Evidence (15%)	**Q**: What was the total impairment costs recorded from 2003 to 2005 in millions? **G**: 0.6, 0.5, 4.7 **P**: 0.6, 0.5
	Redundant Evidence (8%)	**Q**: What was the average number of shares issued to employees from 2013 to 2015? **G**: 439000, 411636 **P**: 439000, 411636, 556000
Reasoner	Wrong Operators (19%)	**Q**: what was the percent of the change in the stock price performance for hum from 2010 to 2011? **G**: (201 - 125) / 125 **P**: 201 - 125
	Wrong Values (10%)	**Q**: What was the average cash flow from 2004 to 2006? **G**: (950.4 + 957.4 + 769.1) / 3 **P**: (957.4 + 957.4 + 769.1) / 3

Table 1: Examples of errors on the test set of FinQA and corresponding percentages in each step. Q, G and P denote question, ground truth and prediction.

the reasons. Since we adopt the *External Executor* for reliable execution, the errors only occur to the *Extractor* and *Reasoner*. As shown in Table A1, we list all kinds of errors with examples and their corresponding percentages. We can observe that most of the errors come from the *Extractor*, where the *Wrong Evidence* takes 48% of the total errors. This shows that our TAT-LLM (7B) model still faces challenges in precisely interpreting the meaning of the values, sometimes due to the unique terminology. Training the model with more tabular and textual data in this domain might be a possible approach to enhancing its evidence extraction ability. Additionally, the *Wrong Operator* and the *Wrong Values* may be partially caused by the randomness of the LLM as the generation length increases, which fails to generate the full equation or wrongly copies the extracted evidence.

C Templates for Fine-tuning

To prepare data for training our TAT-LLM model, the below templates are applied to construct the training instances following the *Step-wise Pipeline* on the three tabular and textual QA datasets, i.e., Table A2 for FinQA, Table A3 for TAT-QA and Table A4 for TAT-DQA, respectively.

For comparison, we also present the templates we used to prepare the training instances following the *End-to-end Pipeline* in Table A5 for FinQA, Table A6 for TAT-QA, and Table A7 for TAT-DQA.

D Templates for Zero-shot Inference

For zero-shot inference with the baseline LLMs such as GPT-3.5-Turbo (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023), we utilize three different templates to build the instances and feed them to the LLMs to get the predictions. We report the best results in Section 4.2. The following are details of the templates we used. 1) *End-to-end Pipeline* template, i.e., Table A8 for FinQA and TAT-QA, and Table A9 for TAT-DQA; 2) *Chain-of-Thought template*, which adds "Let's complete this task step by step" to the *End-to-end Pipeline* template; 3) *Step-wise Pipeline* template that is similar to the templates for fine-tuning, with the part of “Response” removed.

```
Below is an instruction that describes a question answering task in the finance domain, paired with an input table and its
relevant text that provide further context. The given question is relevant to the table and text. Generate an appropriate
answer to the given question.

### Instruction:
Given a table and a list of texts in the following, what is the answer to the question? Please complete the task in three
steps:
1. In the first step, extract the relevant numerical values from the provided table or texts. Store these in the variable
‘{evidence}‘. If there are multiple values, separate them using the ’#’ symbol.
2. In the second step, generate an equation using the extracted numerical values. Store this equation in the variable
‘{equation}‘.
3. In the third step, calculate the answer based on the equation and store it in the variable ‘{answer}‘.
Please organize the results in the following table:
| step | output |
| 1 | {evidence} |
| 2 | {equation} |
| 3 | {answer} |
Finally, present the calculated answer in the format: "The answer is: {answer}"

### Table
{table}

### Text
{text}

### Question
{question}

### Response
|step | output|
|1 | {gold_evidence} |
|2 | {gold_equation} |
|3 | {gold_answer} |
The answer is: {gold_answer}
```

Table 2: The template for constructing training instances following *Step-wise Pipeline* with FinQA dataset.

```
Below is an instruction that describes a question answering task in the finance domain, paired with an input table and its
relevant text that provide further context. The given question is relevant to the table and text. Generate an appropriate
answer to the given question.

### Instruction
Given a table and a list of texts in the following, answer the question posed using the following five-step process:
1. Step 1: Predict the type of question being asked. Store this prediction in the variable ‘{question_type}‘. The value of
‘{question_type}‘ can be one of the following:‘Single span‘, ‘Multiple spans‘, ‘Count‘, or ‘Arithmetic‘.
2. Step 2: Extract the relevant strings or numerical values from the provided table or texts. Store these pieces of evidence
in the variable ‘{evidence}‘. If there are multiple pieces of evidence, separate them using the ’#’ symbol.
3. Step 3: if the ‘{question_type}‘ is ‘Arithmetic‘, formulate an equation using values stored in ‘{evidence}‘. Store this
equation in the variable ‘{equation}‘. For all other question types, set the value of {equation} to ’N.A.’.
4. Step 4: Predict or calculate the answer based on the question type, evidence and equation. Store it in the variable
‘{answer}‘. If there are multiple values, separate them using the ’#’ symbol.
5. Step 5: If the value of the ‘{answer}‘ is numerical, predict its scale and store it in a variable named ‘{scale}‘. The
value of ‘{scale}‘ can be one of the following: ‘none‘, ‘percent‘, ‘thousand‘, ‘million‘, or ‘billion‘. For non-numerical
values, set the value of ‘{scale}‘ to ’none’.
Please organize the results in the following table:
| step | output |
| 1 | {question_type} |
| 2 | {evidence} |
| 3 | {equation} |
| 4 | {answer} |
| 5 | {scale} |
Finally, present the final answer in the format: "The answer is: {answer} #### and its corresponding scale is: {scale}"

### Table
{table}

### Text
{text}

### Question
{question}

### Response
| step | output |
| 1 | {gold_question_type} |
| 2 | {gold_evidence} |
| 3 | {gold_equation} |
| 4 | {gold_answer} |
| 5 | {gold_scale} |
The answer is: {gold_answer} #### and its corresponding scale is: {gold_scale}
```

Table 3: The template for constructing training instances following *Step-wise Pipeline* with TAT-QA dataset.

```
Below is an instruction that describes a question answering task in the finance domain, paired with an input document
that has one or multiple pages that provide further context. The given question is relevant to the document. Generate an
appropriate answer to the given question.

### Instruction
Given a document that has one or multiple pages in the following, answer the question posed using the following five-step
process:
1. Step 1: Predict the type of question being asked. Store this prediction in the variable ‘{question_type}‘. The value of
‘{question_type}‘ can be one of the following:‘Single span‘, ‘Multiple spans‘, ‘Count‘, or ‘Arithmetic‘.
2. Step 2: Extract the relevant strings or numerical values from the provided document. Store these pieces of evidence in
the variable ‘{evidence}‘. If there are multiple pieces of evidence, separate them using the ’#’ symbol.
3. Step 3: if the ‘{question_type}‘ is ‘Arithmetic‘, formulate an equation using values stored in ‘{evidence}‘. Store this
equation in the variable ‘{equation}‘. For all other question types, set the value of {equation} to ’N.A.’.
4. Step 4: Predict or calculate the answer based on the question type, evidence and equation. Store it in the variable
‘{answer}‘. If there are multiple values, separate them using the ’#’ symbol.
5. Step 5: If the value of the ‘{answer}‘ is numerical, predict its scale and store it in a variable named ‘{scale}‘. The
value of ‘{scale}‘ can be one of the following: ‘none‘, ‘percent‘, ‘thousand‘, ‘million‘, or ‘billion‘. For non-numerical
values, set the value of ‘{scale}‘ to ’none’.
Please organize the results in the following table:
| step | output |
| 1 | {question_type} |
| 2 | {evidence} |
| 3 | {equation} |
| 4 | {answer} |
| 5 | {scale} |
Finally, present the final answer in the format: "The answer is: {answer} #### and its corresponding scale is: {scale}"

### Text
{pages}

### Question
{question}

### Response
| step | output |
| 1 | {gold_question_type} |
| 2 | {gold_evidence} |
| 3 | {gold_equation} |
| 4 | {gold_answer} |
| 5 | {gold_scale} |
The answer is: {gold_answer} #### and its corresponding scale is: {gold_scale}
```

Table 4: The template for constructing training instances following *Step-wise Pipeline* with TAT-DQA dataset.

```
Below is an instruction that describes a question answering task in the finance domain, paired with an input table and its
relevant text that provide further context. The given question is relevant to the table and text. Generate an appropriate
answer to the given question.

### Instruction
Given a table and a list of texts in the following, what is the answer to the question? Please output the answer in the
format of "The answer is:".

### Table
{table}

### Text
{text}

### Question
{question}

### Response
The answer is: {answer}
```

Table 5: The template for constructing training instances following *End-to-end Pipeline* with FinQA dataset

```
Below is an instruction that describes a question answering task in the finance domain, paired with an input table and its
relevant text that provide further context. The given question is relevant to the table and text. Generate an appropriate
answer to the given question.

### Instruction
Given a table and a list of texts in the following, what is the answer to the question? Please predict the answer and store
it in a variable named ‘{answer}‘. If there are multiple values, separate them using the ’#’ symbol. If the value of the
‘{answer}‘ is numerical, predict its scale and store it in a variable named ‘{scale}‘. The value of ‘{scale}‘ can be one of
the following: ‘none‘, ‘percent‘, ‘thousand‘, ‘million‘, or ‘billion‘. For non-numerical values, set the value of ‘{scale}‘
to ’none’. Finally, present the final answer in the format of "The answer is: {answer} #### and its corresponding scale is:
{scale}"

### Table
{table}

### Text
{text}

### Question
{question}

### Response
The answer is: {gold_answer} #### and its corresponding scale is: {gold_scale}
```

Table 6: The template for constructing training instances following *End-to-end Pipeline* with TAT-QA dataset

```
Below is an instruction that describes a question answering task in the finance domain, paired with an input document
that has one or multiple pages that provide further context. The given question is relevant to the document. Generate an
appropriate answer to the given question.

### Instruction
Given a document that has one or multiple pages in the following, what is the answer to the question? Please predict the
answer and store it in a variable named ‘{answer}‘. If there are multiple values, separate them using the ’#’ symbol.
If the value of the ‘{answer}‘ is numerical, predict its scale and store it in a variable named ‘{scale}‘. The value of
‘{scale}‘ can be one of the following: ‘none‘, ‘percent‘, ‘thousand‘, ‘million‘, or ‘billion‘. For non-numerical values, set
the value of ‘{scale}‘ to ’none’. Finally, present the final answer in the format of "The answer is: {answer} #### and its
corresponding scale is: {scale}"

### Document
{pages}

### Question
{question}

### Response
The answer is: {gold_answer} #### and its corresponding scale is: {gold_scale}
```

Table 7: The template for constructing training instances following *End-to-end Pipeline* with TAT-DQA dataset

```
Below is an instruction that describes a question answering task in the finance domain, paired with an input table and its relevant text that provide further context. The given question is relevant to the table and text. Generate an appropriate answer to the given question.

### Instruction
Given a table and a list of texts in the following, what is the answer to the question? (Let's complete this task step by step.) Please output the answer in the format of "The answer is:".

### Table
{table}

### Text
{text}

### Question
{question}

### Response
```

Table 8: The template for zero-shot inference using the baseline LLMs on FinQA and TAT-QA datasets.

```
Below is an instruction that describes a question answering task in the finance domain, paired with an input document that has one or multiple pages that provide further context. The given question is relevant to the document. Generate an appropriate answer to the given question.

### Instruction
Given a document with one or multiple pages in the following, what is the answer to the question? (Let's complete this task step by step.) Please output the answer in the format of "The answer is:".

### Text
{pages}

### Question
{question}

### Response
```

Table 9: The template for zero-shot inference using the baseline LLMs on TAT-DQA datasets.