Enhancing Numerical Reasoning with the Guidance of Reliable Reasoning Processes

Anonymous ACL submission

Abstract

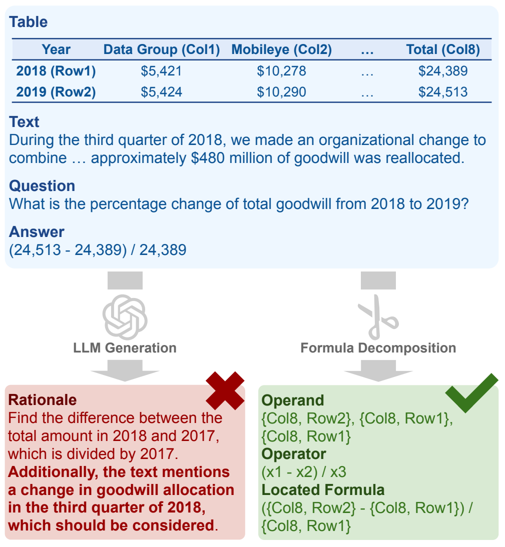
Numerical reasoning is an essential ability for NLP systems to handle numeric information. Recent research indicates that fine-tuning a small-scale model to learn generating reasoning processes alongside answers can significantly enhance performance. However, current methods have the limitation that most methods generate reasoning processes with large language models (LLMs), which are "unreliable" since such processes could contain information unrelated to the answer. To address this limitation, we introduce **E**nhancing **N**umeri**C**al reas**O**ning with **R**eliable proc**E**sses (ENCORE), which derives the reliable reasoning process by decomposing the answer formula, ensuring which fully supports the answer. Nevertheless, models could lack enough data to learn the reasoning process generation adequately, since our method generates only one single reasoning process for one formula. To overcome this difficulty, we present a series of pre-training tasks to help models learn the reasoning process generation with synthesized data. The experiments show that ENCORE yields improvement on all five experimental datasets with an average of 1.8%, proving the effectiveness of our method[1].

1 Introduction

Numerical reasoning is an essential ability for NLP systems to handle arithmetic questions in real scenarios, which refers to generating answers to numerical questions with given evidence (Geva et al., 2020). The evidence is employed to support reasoning in cases where the question does not furnish all the necessary contextual information, as seen in passages and tables in numerical data (Zhu et al., 2021b,a; Chen et al., 2021). The answer generated encompasses a range of elements, including values (Dua et al., 2019), programs (Chen et al., 2021), and formulas (Zhu et al., 2021a)[2].

[1]Our data and code will be released after review.

[2]For the sake of conciseness in this paper, we collectively refer to these elements as *formulas*.



Table

Year	Data Group (Col1)	Mobileye (Col2)	...	Total (Col8)
2018 (Row1)	$5,421	$10,278	...	$24,389
2019 (Row2)	$5,424	$10,290	...	$24,513

Text
During the third quarter of 2018, we made an organizational change to combine … approximately $480 million of goodwill was reallocated.

Question
What is the percentage change of total goodwill from 2018 to 2019?

Answer
(24,513 - 24,389) / 24,389

Rationale
Find the difference between the total amount in 2018 and 2017, which is divided by 2017. **Additionally, the text mentions a change in goodwill allocation in the third quarter of 2018, which should be considered**.

Operand
{Col8, Row2}, {Col8, Row1}, {Col8, Row1}
Operator
(x1 - x2) / x3
Located Formula
({Col8, Row2} - {Col8, Row1}) / {Col8, Row1}

Figure 1: The reasoning processes generated by `gpt-3.5-turbo` and ENCORE. The left process is described with natural language, where **bold** words are unrelated to the answer. The right process contains three parts designed in ENCORE that fully support the answer.

Currently, although LLMs have demonstrated great performance on the numerical reasoning (Chen et al., 2022a; Gao et al., 2022), we argue that it is still valuable to study and employ the small-scale model (e.g., $\text{BART}_{\text{LARGE}}$ (Lewis et al., 2020)) since their low computational efficiency and decent performance, which still have application value in real scenarios. Previous research has demonstrated that teaching small-scale models to generate reasoning processes during fine-tuning can make the prediction more accurate and explainable (Cobbe et al., 2021; Ho et al., 2023; Magister et al., 2023). For example, SCOTT (Wang et al., 2023) employs LLMs to generate reasoning processes based on questions and answers, which are used to fine-tune small-scale models. However, the reasoning processes of the current methods could be **unreliable** since most methods employ LLMs to generate the processes, where such processes could contain information that does not support the answer (e.g., bold words in the left part of Figure 1).

To address this issue, we present a novel numerical reasoning method called **E**nhancing **N**umeri**C**al reas**O**ning with **R**eliable proc**E**sses (ENCORE). Our method decomposes the operands and operators from the answers as the reasoning process, which we concatenate with the answer as the output of the model. The generated reasoning process of ENCORE is reliable since the process entirely derives from decomposing the answer, ensuring that the process does not contain answer-unrelated information and fully supports the answer, as shown in the right part of Figure 1. However, our method could lack enough data to enable the model to learn the reasoning process generation adequately, as for one answer, our method generates only one reasoning process, while methods based on LLMs can generate multiple processes (Ho et al., 2023). To overcome this difficulty, we present a series of pre-training tasks to help models learn the process generation with synthesized data.

To evaluate the effectiveness of ENCORE, we adopt it on five mainstream numerical reasoning datasets with various settings. Compared with baseline models, ENCORE brings performance improvement on all datasets and leads an average boost of 1.8%, showing the effectiveness and generalization of ENCORE. Moreover, in comparison to fine-tuning with the reasoning process generated by `gpt-3.5-turbo`, our method is superior by about 10%, which proves that the reasoning process generated by ENCORE has higher quality.

Our contributions can be summarized as follows:

- To ensure that generated reasoning processes fully support answers, we propose ENCORE, which generates reliable reasoning processes by decomposing answer formulas.
- To alleviate the difficulty of the insufficient data scale of ENCORE, we introduce a series of pre-training tasks, which enhance the generation of the reasoning process with synthesized data.
- To prove the effectiveness of ENCORE, we evaluate it with five mainstream numerical reasoning datasets, which yield performance improvement on all experimental datasets with an average boost of 1.8% compared with baselines.

2 Background

The numerical reasoning task is to generate single or multiple formulas as answers based on the given question as well as the textual and tabular evidence. About the input, textual evidence contains several paragraphs, and tabular evidence consists of one or more tables, where the first row and column are called table headers, which describe information about the cells of the corresponding rows or columns. About the output, one formula consists of operators and operands (e.g., $2 + 1 \times 3$). The operands refer to the values manipulated or processed in the formula (e.g., 2, 1, 3). The operators are arithmetic symbols or functions that are to be performed on the operands (e.g., $+$, $\times$).

However, in practical applications, answers could not be annotated with formulas. We also try to apply our method to such data without formulas. For the questions with simple calculations (e.g., DROP (Dua et al., 2019)), we can directly generate the formulas following the previous work[3]. For the more complex calculations (e.g., GSM8K (Cobbe et al., 2021)), we employ LLMs with in-context learning, which can generate answers with a few samples without fine-tuning, to prove the effectiveness of the reasoning process we designed.

3 Methodology

In this section, we present our numerical reasoning method called ENCORE. First, we introduce the pipeline of ENCORE, which generates reliable reasoning processes fully supporting the answers (§ 3.1). Then, we propose three pre-training tasks based on ENCORE to enhance the generation performance of the reasoning process (§ 3.2).

3.1 ENCORE

In this section, we introduce ENCORE, which enhances numerical reasoning by generating reliable reasoning processes decomposed from answers. The illustration of ENCORE is shown in Figure 2.

3.1.1 Retrieve

Because evidence irrelevant to the question could mislead the model, causing performance degradation, we employ a retriever to **retrieve** the question-relevant evidence. We concatenate each text paragraph and table column with questions, then feed it into a binary classification model to generate correlation confidence. The classification model is trained with the ground evidence annotated by the dataset or the headers in the located formulas obtained in § 3.1.2. Then, we sort each text paragraph and table column with the correlation confidence

[3]`https://github.com/allenai/allennlp-reading-comprehension`

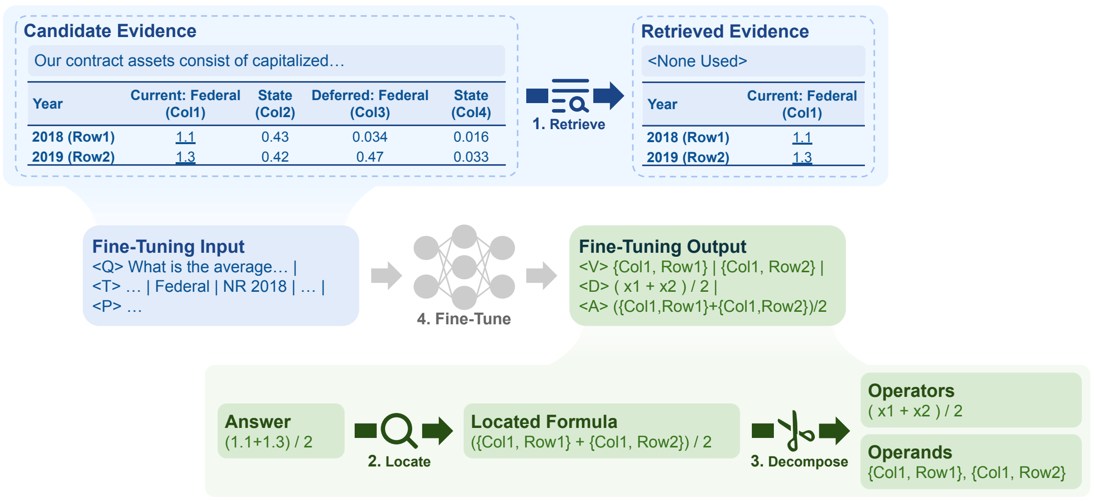


Figure 2: The illustration of ENCORE, which takes the question "What is the average current federal of 2018-2019?" as the example. ENCORE consists of four steps: **1.Retrieve** question-related evidence. **2.Locate** the table heads of each value in the formula. **3.Decompose** the located formula into operators and operands. **4.Fine-tune** the model with the input and the generated output.

of the model output. We select the top-k evidence as the retrieval result, where k is determined by the input length limit, and then we concatenate such evidence with the question as the model input.

3.1.2 Locate

This step is designed to reduce the difficulty of value memory and table understanding by changing the value format in answers. In prior work, it has been observed that current models struggle to accurately retain complex floating-point values present in the evidence (Thawani et al., 2021). Besides, the table understanding ability of most models is limited, as linearized input disrupts the structural information of the table (Jin et al., 2022). To alleviate the challenge of extracting numerical values from tables, we propose substituting values in the answer by **locating** their respective headers in the table, which we call the located formula. For instance, as illustrated in Figure 2, instead of directly using the value "1.1", we use "$\{Col1, Row1\}$" corresponding to its cell headers in the table. Consequently, the model only needs to recall the headers associated with relevant cells, lowering the difficulty of specific value memory and table understanding, thereby enhancing the reasoning performance.

We use string matching to locate the cell corresponding to each value in the formula, which could not handle the cells with the same value. An example of our labeling method is shown in Appendix B However, how to detect the question-related entities in the evidence is a long-studied problem, which has been discussed in detail by the previous works (Liu et al., 2021; Kumar et al., 2023; Wu et al., 2023). Therefore, to focus on the main topic of this paper, we will discuss how to merge the detecting methods with ENCORE in the future.

3.1.3 Decompose

The designed motivation for this step is to reduce the complexity of reasoning through multi-step generation. Current methods, which ask models to generate formulas in one step, often lead to challenges in establishing a clear correspondence between the answer and the input information. For example, most operands in the formula of the answer are typically extracted from evidence, while the majority of operators are determined by the semantics of the question (e.g., "ratio" in the question leads to the division operator). Moreover, some formulas are too complex to generate correctly in one step. To address these issues, we design a multi-step generation process for the model to achieve the numerical reasoning results. We **decompose** the formula into operators and operands, which are used to ask the model to generate before located formulas. By first generating the more straightforward operators and operands and then generating the complete formula, we can reduce the difficulty in generating formulas, thereby enhancing accuracy.

3.1.4 Fine-tune

After constructing the located formulas and corresponding operators and operands, we take them with answers as output and use the question and the retrieved evidence as input to **fine-tune** the seq2seq model. During the construction of inputs and outputs, we use tags like the form "<*A*>" to distinguish different parts. We also add other information in the output sequence to meet the requirements of different datasets, such as value scales (e.g., "billion" and "percentage" in TAT-QA) and spans (e.g., span-type answers in DROP).

3.2 Pre-Training

With the reasoning process generated by ENCORE, the model could still struggle to learn how to produce such processes because of the limit of the training data scale. To aid the model in learning to generate the reasoning process, in this section, we introduce three pre-training tasks. We synthesize questions, answers, and reasoning processes based on different templates, then pre-train the model with all these data as the multi-task training. The template and example of each pre-training task are shown in Appendix A.

We primarily design pre-training tasks for tabular evidence rather than textual evidence, for two reasons: *(1)* Most current pre-trained language models are trained on textual data, ensuring their proficiency in generating text-related reasoning processes. *(2)* Direct linearization of the tabular evidence during input disrupts the structural information, making it challenging for the model to generate table-related reasoning processes.

Table Location Prediction is designed to help the model better locate operands in the formula by learning the correspondence between the cell and the corresponding table headers. Given the row and column headers of one cell, the model should predict the value of this cell.

Table Calculation Prediction is designed to enhance operator generation. About the tabular evidence, many calculation formulas involve all values in one column as the operands, such as the average or total value of a column. We help models learn the generation of these formulas with the given column and the calculation type.

Hierarchical Table Prediction is designed for models to perform better operand extraction by comprehending the hierarchical table structure with multi-level headers (Zhao et al., 2022), where the whole table can be seen as several sub-tables. For this task, models should predict the name of the first level of each given table header.

4 Experiments

4.1 Experiment Setup

Datasets We apply ENCORE on five datasets with various settings: FinQA (Chen et al., 2021), ConvFinQA (Chen et al., 2022b), TAT-QA (Zhu et al., 2021a), MathQA (Amini et al., 2019) and DROP (Dua et al., 2019), which cover different types of evidence and answers. More detailed information can be found in Appendix C.

Metrics For MathQA, FinQA, and ConvFinQA, we employ execution accuracy as our evaluation metrics (Chen et al., 2021). About DROP and TAT-QA, we evaluate methods with the exact match (EM) (Zhu et al., 2021a). For TAT-QA, we additionally use Arithmetic EM to represent the EM on numerical reasoning questions. The definition of these metrics can be found in Appendix D.

Baselines We adopt $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019) as our baseline retrieval model. We use $\text{BART}_{\text{LARGE}}$ (Lewis et al., 2020) and T5_{3B} (Raffel et al., 2020) as our baseline seq2seq models.

Settings All experimental models are implemented with PyTorch (Paszke et al., 2019), Huggingface transformers (Wolf et al., 2020), and Fairseq (Ott et al., 2019). We adopt the pre-training tasks to TAT-QA, FinQA, and ConvFinQA since they contain tabular evidence. More detailed settings are shown in Appendix E.

4.2 Main Results

The main experiment results are summarized in Table 1, where the detailed results on each dataset are shown in Appendix F. ENCORE brings performance improvement on all experiment datasets with all used baseline models and achieves SOTA or near-SOTA results on most datasets, which proves the efficiency and generalizability of our method. Compared to $\text{BART}_{\text{LARGE}}$, our method exhibits more obvious improvements on T5_{3B}, suggesting that larger-scale models can more effectively learn the generation of reasoning processes and apply the associated capabilities to answer generation. However, our method exhibits an obvious discrepancy with the current SOTA on DROP. This is attributed to the low quality of the synthesized formulas, where the synthesized results could be incorrect, which subsequently misleads the model into erroneous reasoning processes, resulting in poor generation performance.

Method	FinQA		ConvFinQA		TAT-QA		MathQA		DROP	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
Published SOTA	69.7	68.0	<u>**76.5**</u>	76.0	N/A[†]	<u>**76.8**</u>	N/A[†]	<u>**83.0**</u>	N/A[†]	<u>**90.0**</u>
$\mathrm{BART}_{\mathrm{LARGE}}$	62.5	58.8	67.4	71.5	68.5	–	77.4	78.0	68.6	67.4
+ ENCORE	64.0	62.3	68.9	74.4	71.0	–	77.7	78.8	69.2	68.4
Δ	+1.5	+3.5	+1.5	+2.9	+2.5	–	+0.3	+0.8	+0.6	+1.0
$\mathrm{T5}_{\mathrm{3B}}$	66.9	65.0	73.3	79.6	73.8	–	78.1	78.6	77.3	77.1
+ ENCORE	<u>**71.6**</u>	<u>**69.4**</u>	**76.0**	<u>**79.8**</u>	**75.6**	**71.5**	**80.0**	**80.6**	**77.6**	**77.1**
Δ	+4.7	+4.4	+2.7	+0.2	+1.8	–	+1.9	+2.0	+0.3	+0.0

Table 1: The main experiment results of ENCORE. [†] denotes the model does not report the corresponding metric result. The experiments on TAT-QA lack results on the test set due to it is not public, where only periodic submissions are allowed for evaluation, so we only evaluate the model that performs best on the dev set. On FinQA, ConvFinQA, and MathQA, the previous SOTA results are achieved by APOLLO (Sun et al., 2022). The best results on TAT-QA and DROP are Code and MindOpt Copilot respectively, which papers have not been published. The best results of our methods are marked in **bold**. The best results of all methods are marked in <u>**underline**</u>.

Setting	EM	Arithmetic EM
ENCORE	74.1	78.6
w/o. Operand	72.7 (−1.4)	75.5 (−3.1)
w/o. Located Formula	73.9 (−0.2)	77.3 (−1.6)
w/o. Operator	73.1 (−1.0)	78.3 (−0.3)

Table 2: The performance of $\mathrm{BART}_{\mathrm{LARGE}}$ under different settings on TAT-QA using golden evidence without pre-training. The Arithmetic EM denotes the EM of the arithmetic questions.

Setting	EM	Arithmetic EM
ENCORE	75.7	81.2
w/o. Table Location	75.2 (−0.5)	79.8 (−1.4)
w/o. Table Calculation	74.6 (−1.1)	79.4 (−1.8)
w/o. Hierarchical Table	75.3 (−0.4)	80.5 (−0.7)
w/o. All	74.1 (−1.6)	78.6 (−2.6)

Table 3: The performance of ENCORE after removing different pre-training tasks of $\mathrm{BART}_{\mathrm{LARGE}}$ on TAT-QA with golden evidence.

It is noteworthy that, in comparison to datasets that solely utilize textual evidence (e.g., MathQA, DROP), our method exhibits a more significant improvement on datasets with both textual and tabular evidence (e.g., TAT-QA, FinQA). This is because our designed located formula addresses the challenge of cell location, where textual evidence does not involve this challenge. Besides, our designed pre-training tasks mainly focus on tabular evidence, so the improvement of textual evidence is less obvious when compared to tabular evidence.

4.3 Ablation Studies

In this section, we perform ablation studies to further evaluate the performance of ENCORE. We use TAT-QA as our study dataset since it covers various types of evidence and answers, which can comprehensively reflect the performance of the model.

4.3.1 Reasoning Process Studies

To verify that each designed part of the reasoning process in ENCORE is effective, we perform ablation experiments on each part separately, which is shown in Table 2. We can see that each part of the reasoning process contributes the performance improvement, which proves the effectiveness of the reasoning process designed by our method. According to the arithmetic EM, we can see that extracting operands has the most apparent impact on model performance. This is because the model regards the values in the answer as part of the formula structure, lacking the awareness of extracting values from evidence. The effect of the located formula is also apparent, which proves that it is hard for models to map the table headers to the corresponding cell value. The improvement of introducing operators is not significant since the formula operator is similar to that in the answer.

To verify the impact of generation orders of different parts of the reasoning process, we also adopt the ablations of the reasoning process format under two settings: generate operands first and operators first. Generating operands first leads to 74.1% on EM, and generating operators first is 73.6% in our experiment, showing that generating operands first is better, which is the order we used in ENCORE.

4.3.2 Pre-Training Studies

To prove that all designed pre-training tasks are effective, we conduct ablation experiments on them. Table 3 shows the experiment results of ENCORE with different pre-training settings.

From Table 3, we can observe that: *(1)* the ablation of each pre-training task leads to a drop in performance, which proves the effectiveness of all pre-training tasks; *(2)* the performance increment of Arithmetic EM is much higher than EM of all types of questions, which proves that the pre-training does improve model performance by improving numerical reasoning capabilities; *(3)* Table Calculation Prediction task leads to the most significant improvement for the model, proving that the ability to handle operator generation of the baseline is weak, while our method effectively improves the ability to handle such a reasoning process.

4.4 Analysis

Does ENCORE generate better reasoning processes than using LLMs? To compare the quality of the reasoning processes generated by ENCORE and by LLMs, we fine-tune models using reasoning processes produced by both methods. We employ `gpt-3.5-turbo` to generate reasoning processes given the question and the answer with zero-shot inference. The performance of different process sources is shown in Table 5. From the table, we can see that the model fine-tuned with the reasoning process generated by using ENCORE markedly outperforms using `gpt-3.5-turbo`. Therefore, the reasoning process synthesized by our method can better help small models generate correct results.

Does ENCORE work well on various answer formats? To prove our method can handle data from multiple scenes at the same time, we adapt ENCORE to the unified setting by merging multiple datasets. We transfer the answer formats of MathQA, FinQA, and numerical reasoning questions of TAT-QA into the domain-specific language format (Amini et al., 2019) to unify the answer format (e.g., $2 + 1 \times 3 \rightarrow add(2, multiple(1, 3))$). Then, we train the models in this unified setting and evaluate them on the unified and divided datasets respectively to evaluate the performance.

The experimental results are shown in Table 6, from which we can observe that: *(1)* compared with single training, the unified setting achieves much better performance since more training examples make the model learn more numerical reasoning knowledge; *(2)* ENCORE can further improve the performance under the unified setting by 1.5% compared with the baselines, demonstrating the generalizability under different answer types.

Is the reasoning process format of ENCORE still effective for in-context learning? Although ENCORE brings great performance improvement, it cannot handle the questions annotating answers without formulas (e.g., GSM8K). Considering the brilliant in-context learning ability of the current LLMs, we conduct experiments to verify whether the reasoning process format of ENCORE can still improve the performance without fine-tuning.

We compare our method with two prompt methods: generate directly and with Chain-of-Thought (CoT) (Wei et al., 2022), where CoT asks LLMs to generate answers with the prompt "think it step by step" using 8-shot prompt following Fu et al. (2023). The detailed prompts we used are shown in Appendix G. We evaluate ENCORE on the arithmetic subset of TAT-QA, FinQA under the 3-shot setting and GSM8K with 8-shot since the questions of GSM8K are harder than the above two datasets. As shown in Table 7, compared with CoT, ENCORE brings an average performance improvement of 8.9% on all datasets and LLMs, which shows that our method is still effective under the in-context learning setting.

What is the performance of ENCORE on different answer types? We categorize the predictions based on answer types and sources, which are shown in Table 4. About the performance of different question types, compared with the baseline model, ENCORE improves the performance of arithmetic questions with 4.9%, showing that our method does improve the numerical reasoning ability. Furthermore, our method also shows enhancements for other types of answers, indicating that the reasoning process generation can elevate the reasoning for various answer types to some extent. About the results of different evidence sources, ENCORE increases the performance of table-source and hybrid-source questions, showing that generating located formulas indeed lowers the difficulty of the table understanding.

However, ENCORE suffers from performance degradation on the text-source and span-type, which are mainly span extraction questions. There are two reasons for this result. Firstly, there are no fixed rules for annotating span-type answers, which leads to performance fluctuations during the prediction. Besides, our method focuses on improving the numerical reasoning ability, and the additionally generated information (e.g., operands, operators) could reduce the span extraction ability.

Method	Type				Source			Total
	Span	Spans	Arithmetic	Count	Text	Table	Hybrid	
$\text{BART}_{\text{LARGE}}$	73.0	77.0	73.7	37.5	58.9	73.1	82.4	72.6
+ ENCORE	71.6	77.9	78.6	46.9	56.3	76.2	84.6	74.1
Δ	-1.4	$+0.9$	$+4.9$	$+9.4$	-2.6	$+3.1$	$+1.8$	$+1.5$

Table 4: The exact match of $\text{BART}_{\text{LARGE}}$ with and without ENCORE on TAT-QA, which uses the golden evidence and is without pre-training. Type denotes the types of dataset questions. Source denotes the evidence type that contains the answer-related information, whereas hybrid includes both text and table.

Method	Arithmetic EM
$\text{BART}_{\text{LARGE}}$	73.7
+ `gpt-3.5-turbo`†	74.7
+ ENCORE	**78.6**

Table 5: The performance on TAT-QA with different reasoning process sources. † denotes fine-tuning with the rationale generated by `gpt-3.5-turbo`. The best performance is marked in **bold**.

Dataset	$\text{BART}_{\text{LARGE}}$		+ ENCORE	
	Single	Unified	Single	Unified
MathQA	79.3	**82.7**	79.5	<u>**84.4**</u>
TAT-QA*	73.7	**79.5**	78.6	<u>**79.8**</u>
FinQA	63.1	**66.1**	65.0	<u>**68.0**</u>
Mixture†	-	**79.5**	-	<u>**81.0**</u>

Table 6: The execution accuracy on the single and the unified dataset with golden evidence. * denotes the numerical reasoning questions. The best of each method is highlighted in **bold**. The best of each dataset is highlighted in <u>**underline**</u>. † denotes the result on the dev set mixture of all three datasets.

Method	TAT-QA*	FinQA	GSM8K
`code-davinci-002`	36.2	12.8	19.3
+ CoT	45.4	19.8	60.3
+ ENCORE	**46.0**	**35.1**	**66.3**
`gpt-3.5-turbo`	25.2	9.9	7.9
+ CoT	38.2	28.2	63.1
+ ENCORE	**55.2**	**39.8**	**71.3**
Llama2-70b	18.5	13.7	16.0
+ CoT	41.9	21.6	54.4
+ ENCORE	**49.2**	**35.1**	**55.3**

Table 7: The execution accuracy of in-context learning with different prompt methods and LLMs. * denotes the numerical reasoning questions. The best performances of different datasets and LLMs are marked in **bold**.

What are the main error types of ENCORE? To better understand how our method improves the numerical reasoning performance of models and to better observe the direction of future improvement, we study the current error distribution on numerical reasoning questions. We categorize the error cases into three types: *(1) operand* denotes that the extracted operators is incorrect; *(2) operator* means that the model makes mistakes in the operator generation; *(3) other* is the error other than the above two categories. We randomly select 256 numerical questions and then analyze them manually, which is shown in Figure 3.

From Figure 3, we can observe that: *(1)* with ENCORE, the model makes fewer mistakes on all error types, showing that our method can significantly improve the model performance on both operator generation and operand extraction; *(2)* the most significant error drop is in the *operand* error type since the operand extraction and the located formula make the model not need to memorize specific values, lowering the difficulty of reasoning; *(3)* the *operand* error types still account for the main part of the bad cases with ENCORE, which requires follow-up work to continue to improve the operand extraction ability of models.

4.5 Case Study

To better understand how ENCORE improves the numerical reasoning ability, we show an example case of TAT-QA in Figure 4, which requires locating the cell value based on the column name and the row name. The baseline model generates a wrong number 754, which does not exist in the table, showing that the baseline method makes it hard to detect the question-related value in the table.

With ENCORE, the model correctly corresponds the header names in the question to $\{col10, row2\}$, and then extracts the corresponding value 774 by locating the header without model reasoning. That is because our method obviates the need for the model to memorize specific values and reduces the complexity of table understanding, thereby decreasing the difficulty of the operand extraction.

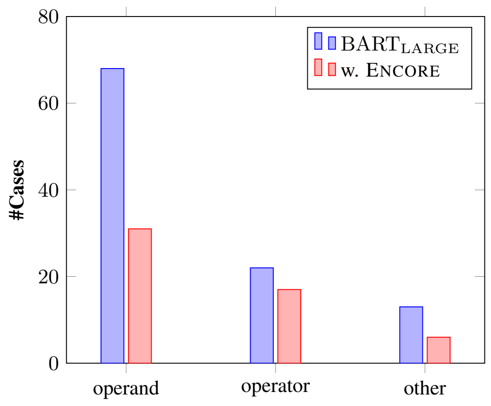
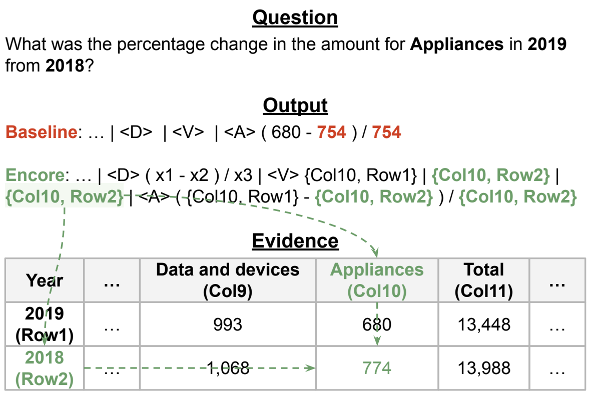


Figure 3: The number of bad cases of numerical reasoning questions on TAT-QA using BART$_{\text{LARGE}}$ with and without ENCORE under different error types. #Cases denotes the number under different error types.



Question

What was the percentage change in the amount for **Appliances** in **2019** from **2018**?

Output

Baseline: ... | <D> | <V> | <A> (680 - **754**) / **754**

Encore: ... | <D> (x1 - x2) / x3 | <V> {Col10, Row1} | **{Col10, Row2}** | **{Col10, Row2}** | <A> ({Col10, Row1} - **{Col10, Row2}**) / **{Col10, Row2}**

Evidence

Year	...	Data and devices (Col9)	Appliances (Col10)	Total (Col11)	...
2019 (Row1)	...	993	680	13,448	...
2018 (Row2)	...	1,068	774	13,988	...

Figure 4: An example of TAT-QA dev set of BART$_{\text{LARGE}}$ with and without ENCORE. The correct entities are highlighted in **green**. The incorrect entities are highlighted in **red**.

5 Related Work

Numerical Reasoning Numerical reasoning is a widely researched topic to handle questions about documents with rich numerical information. Previous works have found that it is hard for the model to generate the numerical answers directly and explore generating reasoning processes for enhancing interpretability and extra supervision for answer generation (Ling et al., 2017). Based on this finding, many researchers have designed model structures to generate reasoning processes implicitly (Ling et al., 2017; Zhu et al., 2021a; Lei et al., 2022; Shao et al., 2022; Zhang and Moshfeghi, 2022). Considering the powerful few-shot inference ability of LLMs without fine-tuning, Wei et al. (2022) presents the chain-of-thought to generate the reasoning process by LLMs themselves, attracting much attention (Wang et al., 2022; Ye and Durrett, 2022; Kojima et al., 2022; Chen et al., 2022a).

Although LLMs have demonstrated impressive performance in the numerical reasoning task, their substantial computational overhead hinders their deployment in practical applications. To address this issue, we explore the use of small-scale models with low computation for numerical reasoning, enhancing their reasoning capabilities by training them to generate reasoning processes.

Answering with Reasoning Processes Previous research has indicated that concurrently generating reasoning processes while producing answers can significantly enhance the accuracy of the responses (Wei et al., 2022; Chu et al., 2023). Subsequent studies have revealed that LLMs exhibit varying performance when generating different types of reasoning processes (Chen et al., 2022a; Gao et al., 2022; Ziqi and Lu, 2023). Apart from LLMs, researchers also find that fine-tuning small-scale models using reasoning processes generated by LLMs can also improve performance (Cobbe et al., 2021; Ho et al., 2023; Wang et al., 2023; Magister et al., 2023). Considering the lower computational overhead and acceptable performance of small-scale models, such works remain worthy of research.

However, the aforementioned methods confront the challenge wherein the reasoning process generated by LLMs could not fully support the answers. To address this limitation, our method directly obtains the reliable reasoning process by decomposing operators and operands from the answers. Our method enhances the quality of the generated reasoning process, thereby improving the accuracy of the generated answers.

6 Conclusion

We propose a novel numerical reasoning method called ENCORE to address the limitations of the reasoning process generation. Compared with previous methods, ENCORE can guarantee to generate reliable reasoning processes that fully support the answer and aim models to learn to generate the process with pre-training. According to experiments, ENCORE brings significant performance improvement over all five experimental datasets, leading to an average improvement of 1.8% compared with baselines, which shows the effectiveness and generalization of our method. Meanwhile, our method is superior by about 10% in comparison to the reasoning process generated by `gpt-3.5-turbo`, which proves the higher quality of the reasoning processes generated by ENCORE.

Limitations

ENCORE has two limitations, including: *1*. About the operand extraction, we directly check whether each operand appears in the evidence, which could lead mistake. For future work, we will adapt better grounding methods, enhancing the extraction accuracy. *2*. It is required that the training data be labeled with formulas, demanding high label overhead. In future work, we will employ LLMs to synthesize formulas, thereby reducing label cost.

Ethics Statement

All datasets and models used in this paper are publicly available, and our usage follows their licenses and terms.

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 *Proc. of AACL*.

Ewa Andrejczuk, Julian Eisenschlos, Francesco Piccinno, Syrine Krichene, and Yasemin Altun. 2022. Table-to-text generation and pre-training with TabT5. In *Proc. of EMNLP Findings*.

Kunlong Chen, Weidi Xu, Xingyi Cheng, Zou Xiaochuan, Yuyu Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. 2020. Question directed graph attention network for numerical reasoning over text. In *Proc. of EMNLP*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022a. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *ArXiv preprint*.

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 *Proc. of EMNLP*.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022b. ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proc. of EMNLP*.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2023. A survey of chain of thought reasoning: Advances, frontiers and future.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of AACL*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of AACL*.

Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. 2023. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proc. of ACL*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proc. of ACL*.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: Recent advances.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners.

Vishwajeet Kumar, Yash Gupta, Saneem Chemmengath, Jaydeep Sen, Soumen Chakrabarti, Samarth Bharadwaj, and Feifei Pan. 2023. Multi-row, multi-span distant supervision for Table+Text question answering. In *Proc. of ACL*.

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 *Proc. of COLING*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proc. of ACL*.

Xiao Li, Yin Zhu, Sichen Liu, Jiangzhou Ju, Yuzhong Qu, and Gong Cheng. 2022. Dyrren: A dynamic retriever-reranker-generator model for numerical reasoning over tabular and textual data. *ArXiv*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *ACL*.

Qian Liu, Dejian Yang, Jiahui Zhang, Jiaqi Guo, Bin Zhou, and Jian-Guang Lou. 2021. Awakening latent grounding from pretrained language models for semantic parsing. In *Proc. of ACL Findings*.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proc. of ACL*.

Rungsiman Nararatwong, Natthawut Kertkeidkachorn, and Ryutaro Ichise. 2022. KIQA: Knowledge-infused question answering model for financial table-text data. In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*.

OpenAI. 2023. Gpt-4 technical report.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. of AACL*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Proc. of NeurIPS*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine reading comprehension with numerical reasoning. In *Proc. of EMNLP*.

Zhihong Shao, Fei Huang, and Minlie Huang. 2022. Chaining simultaneous thoughts for numerical reasoning. In *ArXiv preprint*.

Jia Sun, Hang Zhang, Chen Lin, Yeyun Gong, Jian Guo, and Nan Duan. 2022. Apollo: An optimized training approach for long-form numerical reasoning. *ArXiv*.

Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro Szekely. 2021. Representing numbers in NLP: a survey and a vision. In *Proc. of AACL*.

Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. 2023. SCOTT: Self-consistent chain-of-thought distillation. In *Proc. of ACL*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *ArXiv preprint*.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*.

Jian Wu, Yicheng Xu, Yan Gao, Jian-Guang Lou, Börje Karlsson, and Manabu Okumura. 2023. TACR: A table alignment-based cell selection method for HybridQA. In *Proc. of ACL Findings*.

Ramil Yarullin and Sergei Isaev. 2023. Numerical embeddings for reasoning over text and tables.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. In *Proc. of NeurIPS*.

Jiaxin Zhang and Yashar Moshfeghi. 2022. ELASTIC: Numerical reasoning with adaptive symbolic compiler. In *Proc. of NeurIPS*.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proc. of ACL*.

Fan Zhou, Mengkang Hu, Haoyu Dong, Zhoujun Cheng, Shi Han, and Dongmei Zhang. 2022a. Tacube: Pre-computing data cubes for answering numerical-reasoning questions over tabular data. *ArXiv preprint*.

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

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021a. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *ACL*.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021b. Retrieving and reading: A comprehensive survey on open-domain question answering. *ArXiv preprint*.

Jin Ziqi and Wei Lu. 2023. Tab-CoT: Zero-shot tabular chain of thought. In *Proc. of ACL Findings*.

A Template and Example of Pre-Training

The templates and examples of our pre-training method are shown in Table 8.

B Example of Automatically Labeling

Take Table 9, the question "*I want to know the balance sum from 2018 to 2020*" and the labeling answer "$(113,246 - 18,967) + (120,523 - 19,786) + (125,843 - 21,355)$" as an example.

The grammar of the answer formula format is shown in Table 10, where we omit the specific descriptions of *Operator* and *Operand*. Following this grammar, the given answer can be parsed as "*(Operand Operator Operand) Operator (Operand Operator Operand) Operator (Operand Operator Operand)*". We replace all *Operand* with $x_1, x_2, ...$, and keep all *Operator* as the original symbols. Then, we can get the formula structure "$(x_1 - x_2) + (x_3 - x_4) + (x_5 - x_6)$".

About the operands, we can directly extract the values corresponding to the operands. In this example, they are "$113,246$, $18,967$, $120,523$, $19,786$, $125,843$, and $21,355$", which are also the extracted values. After that, we employ the string match method and locate their positions in the table and get $\{Col_2, Row_1\}$, $\{Col_1, Row_1\}$, $\{Col_2, Row_2\}$, $\{Col_1, Row_2\}$, $\{Col_2, Row_3\}$, $\{Col_1, Row_3\}$, which are the extracted entities. Although the method above is sample yet effective, how to detect the question-related entities in the evidence completely correct should be carefully studied (e.g., MITQA (Kumar et al., 2023), TACR (Wu et al., 2023)). Considering the complexity of this issue, we leave how to solve it as future work.

Correspondingly, the ENCORE output of this example is "<V> $\{Col_2, Row_1\}$ | $\{Col_1, Row_1\}$ | $\{Col_2, Row_2\}$ | $\{Col_1, Row_2\}$ | $\{Col_2, Row_3\}$ | $\{Col_1, Row_3\}$ | <D> $(x_1 - x_2) + (x_3 - x_4) + (x_5 - x_6)$ | <A> $(\{Col_2, Row_1\} - \{Col_1, Row_1\}) + (\{Col_2, Row_2\} - \{Col_1, Row_2\}) + (\{Col_2, Row_3\} - \{Col_1, Row_3\})$ ".

C Experiment Datasets

In this section, we discuss the detailed information on the datasets of the experiments. The settings of these datasets are shown in Table 11

FinQA A financial HybridQA dataset while it only contains numerical reasoning questions, just like MathQA. The operations of FinQA are fewer and more straightforward than MathQA, which only includes less than ten elementary operations. Following the settings of TAT-QA, we also locate the program values' positions as extracted entities. FinQA contains 8,281 examples and is released as training (6,251), validation (883), and test (1,147) following a 75%/10%/15% split.

ConvFinQA A context-dependent version of FinQA, which decomposes the complex numerical reasoning questions into multiple steps. During the validation, it evaluates the model predictions of all steps.

TAT-QA A financial HybridQA dataset containing textual as well as tabular evidence and four types of answers including *span*, *multi-span*, *count*, and *arithmetic*. We follow the origin derivation format of the dataset and extract all values in the derivation, then locate the positions of the values in the table as extracted entities. TAT-QA consists of 16,552 question-answer pairs and is split into the training set (80%), development set (10%), and test set (10%).

MathQA A numerical reasoning dataset containing DSL-format questions involves more than one hundred scientific operations. Because the maximum step number of one answer is 66, which is too long for the model to generate, we only train the model on the example with the number of answer program steps less than ten that have covered the 93% dataset questions. MathQA consists of 37k problems, split into (80/12/8)% training/dev/test sets. We do not add the values to the inputs of MathQA since we do not design the alias for the text.

DROP A reading comprehension dataset contains three types of answers, including *spans*, *date* and *number*, where *number* questions only require + and - operations. Considering that it does not annotate the calculation process, we first extract all numbers in the evidence, calculate the similarity between the question and the words around every number, then select the legal calculation formula with maximum similarity. DROP includes a total of 96,567 question-answer pairs and is partitioned into training (80%), development (10%), and test (10%) sets.

GSM8K The GSM8K is a collection of 8,500 meticulously created and linguistically varied math word problems suitable for grade school levels,

Pre-Training Task	**Question**	**Answer**
Table Location Prediction	What is **{ Col_i , Row_j }** ? *What is **{ Col3 , Row2 }** ?*	The cell value of the headers. ***0.47***
Table Calculation Prediction	What is the max/min/sum/average of **Col_i** ? *What is the sum of **Current : Federal** ?*	The formula of the column. ***{ Col1 , Row1 } + { Col1, Row2 }***
Hierarchical Table Prediction	What is the **{ Col_i , Row_j }** belong to ? *What is the **{ Col2 , Row2 }** belong to ?*	The first-level header of the cell. ***Current***

Table 8: Templates and examples of pre-training data. In each block, the first line is the template, and the second line is the example marked with the *italics*. All answers to the examples are extracted from Figure 2. The replaceable parts are highlighted in **bold**.

Year	**Outcome** (Col_1)	**Income** (Col_2)
2018 (Row_1)	18,967	113,246
2019 (Row_2)	19,766	120.523
2020 (Row_3)	21,355	125,843
2021 (Row_4)	22,312	130,725

Table 9: An example of tabular evidence.

Rules
Formula $\rightarrow$ Formula Operator Formula
Formula $\rightarrow$ (Formula)
Formula $\rightarrow$ Operand

Table 10: An example of answer formula grammar.

Dataset	**Domain**	**Evidence**	**Answer**
FinQA	Finance	Hybrid	Formula
ConvFinQA	Finance	Hybrid	Formula
TAT-QA	Finance	Hybrid	Span
MathQA	MWP	Text	Choice
DROP	Wikipedia	Text	Span
GSM8K	MWP	Text	Formula

Table 11: The settings of the experimental datasets. ConvFinQA is the only context-dependent dataset.

Method	**Dev**		**Test**	
	EM	**F_1**	**EM**	**F_1**
Discriminative Methods				
NumNet+ (Ran et al., 2019)	81.1	84.4	81.5	84.8
QDGAT (Chen et al., 2020)	**84.1**	**87.1**	**84.5**	**87.6**
Generative Methods				
GPT-3.5* (OpenAI, 2023)	-	-	-	64.1
GPT-4* (OpenAI, 2023)	-	-	-	80.9
BART (Lewis et al., 2020)	68.6	71.7	67.4	70.7
BART w. ENCORE	69.2	72.2	68.4	71.4

Table 12: The performance of different methods of DROP. * denotes the few-shot setting. The best results are marked in **bold**.

crafted by professional problem writers. This dataset is divided into two sections: 7,500 problems for training and 1,000 for testing. Each problem is designed to be solved in 2 to 8 steps, predominantly involving a series of basic arithmetic operations (addition, subtraction, multiplication, division) to attain the ultimate solution. These problems are intended to be solvable by middle school students and are ideal for enhancing multi-step mathematical reasoning skills.

D Evaluation Metrics

Exact Match measures the percentage of predictions that match the ground truth answers. Usually, two arithmetic answers are considered equal if their four decimal places are equal, following the rule of rounding function.

Numeracy-Focused F_1 Score measures the average token-level overlap between the predictions and the ground truth answers, which can reduce false negative labeling. When an answer has multiple spans, the numeracy-focused F_1 performs a one-to-one alignment greedily based on the bag-of-word overlap on the set spans to ensure every current span can get the highest F_1 value, then compute micro-average F_1 over each span.

Program Accuracy measures the accuracy of operands and operators between the predicted programs and the golden programs.

Execution Accuracy measures the accuracy of the execution result of the predicted programs.

E Hyper-Parameter Settings

Our model is implemented with PyTorch (Paszke et al., 2019), transformers (Wolf et al., 2020) and Fairseq (Ott et al., 2019). About the retrieval model, we set the learning rate as 2e-5 and the dropout as

Method	Exe	Prog
FinQANet (Chen et al., 2021)	-	79.2
ELASTIC (Zhang and Moshfeghi, 2022)	-	**83.0**
BART (Lewis et al., 2020)	79.6	78.0
BART w. ENCORE	80.5	78.8
T5 (Raffel et al., 2020)	81.8	78.6
T5 w. ENCORE	82.9	80.6

Table 13: The performance of different methods of MathQA test set. Exe denotes the execute accuracy, and Prog denotes the program accuracy. The best results are marked in **bold**.

Method	Dev		Test	
	EM	F_1	EM	F_1
TagOp (Zhu et al., 2021a)	55.2	62.7	50.1	58.0
TaCube (Zhou et al., 2022a)	57.1	65.6	-	-
KIQA (Nararatwong et al., 2022)	-	-	58.2	67.4
UniRPG (Zhou et al., 2022b)	70.2	77.9	67.4	75.5
RegHNT (Lei et al., 2022)	73.6	81.3	70.3	78.0
AeNER (Yarullin and Isaev, 2023)	**78.5**	**86.0**	**75.0**	**83.2**
BART (Lewis et al., 2020)	65.7	73.0	-	-
BART w. ENCORE	71.0	78.9	-	-
T5 (Raffel et al., 2020)	73.8	80.9	-	-
T5 w. ENCORE	75.8	82.8	71.5	79.5

Table 14: The performance of different publicly available methods of TAT-QA. The best results are marked in **bold**.

0.1. We select three negative examples for every positive instance, set batch size as 16, and max epoch as 20, which takes 10 hours for training. For every question, we retrieve the top 5 as candidate evidence. About the generation models of PLMs, we consider it a Seq2Seq task with label-smoothed cross-entropy loss. We set the learning rate as 1e-5, dropout as 0.1, and weight decay as 0.05. We use max tokens as 8192 during every step, update the model parameter every four updates, and warm up with 5000 updates. We set the max epoch as 1 for pre-training, 100 for $\text{BART}_{\text{LARGE}}$, and 20 for T5_{3B}, save the model every ten epochs during fine-tuning and use early-stop checkpoints. Any other hyper-parameters following the default settings of the package. To lower the difficulty of table understanding, we mark the numerical order of each column in the table. About the LLMs generation models, we set top_p as 0.95 and temperature as 0.

We employ one NVIDIA A100 40G GPU as our experiment device. The retrieval model takes around 6 hours for training. The PLMs generation model takes around 1 hour for pre-training, 12 hours for $\text{BART}_{\text{LARGE}}$ fine-tuning, and 48 hours for T5_{3B} fine-tuning. The LLMs generation model takes about 1.5 hours to infer for 1k examples.

Model	FinQA		ConvFinQA	
	Exe	Prog	Exe	Prog
FinQANet (Chen et al., 2021)	61.2	58.9	68.9	68.2
DyRRen (Li et al., 2022)	63.3	61.3	-	-
APOLLO (Sun et al., 2022)	68.0	**65.6**	76.0	74.6
TabT5† (Andrejczuk et al., 2022)	**70.8**	**68.0**	-	-
BART (Lewis et al., 2020)	58.8	54.4	71.5	69.5
BART w. ENCORE	62.3	57.2	74.4	72.2
T5 (Raffel et al., 2020)	65.0	58.3	79.6	77.3
T5 w. ENCORE	**69.4**	63.7	**79.8**	**77.9**

Table 15: The performance of different methods of FinQA and ConvFinQA private test sets. Exe denotes the execute accuracy, and Prog denotes the program accuracy. † denotes results with ensemble. The best results are marked in **bold**. The best results of methods without ensemble are marked in **underline**.

F Detailed Experiment Results

In this section, we show the detailed results of each experiment dataset in Table 12, Table 13, Table 14 and Table 15.

G Prompts of ENCORE with LLMs

In this section, we present the prompts we used with LLMs in § 4.4 in Table 16, Table 17 and Table 18.

Answer the given question based on the given evidence.
You should generate an formula to answer the arithmetic question.
When answering the question, you should firstly generate the used entities.
Then you generate the formula structure.
Finally you generate the answer formula based on the entities and the formula structure.

Read the following text and table, and then answer a question:
17. Income Taxes
Income before income taxes for the Company's domestic and foreign operations was as follows:
— | — | Years Ended June 30, | —
($ 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
Quesetion: What was the change in Foreign in 2019 from 2018?
Entities: 11.8 | 19.9
Formula: x0 - x1
Answer: 11.8 - 19.9

Read the following text and table, and then answer a question:
Effective Income Tax Rate
A reconciliation of the United States federal statutory income tax rate to our effective income tax rate is as follows: In 2019 and 2018 we had pre-tax losses of $19,573 and $25,403, respectively, which are available for carry forward to offset future taxable income. We made determinations to provide full valuation allowances for our net deferred tax assets at the end of 2019 and 2018, including NOL carryforwards generated during the years, based on our evaluation of positive and negative evidence, including our history of operating losses and the uncertainty of generating future taxable income that would enable us to realize our deferred tax.
— | Year Ended | Year Ended
— | December 31, 2018 | December 31, 2019
United States federal statutory rate | 21.00% | 21.00%
State taxes, net of federal benefit | 1.99% | -0.01%
Valuation allowance | -21.96% | -24.33%
Cumulative effect of accounting change | — | 2.07%
R&D Credit | 1.34% | 1.53%
Other | -0.38% | -0.27%
Effective income tax rate | 1.99% | -0.01%
Question: What was the 2019 percentage change in pre-tax losses?
Entities: 19,573 | 25,403 | 25,403
Formula: (x0 + x1) / x2
Answer: (19573 + 25403) / 25403

Read the following text and table, and then answer a question:
Year Ended May 31 | Expected life (in years) | risk-free interest rate | Volatility | Dividend yield | Weighted-average fair value per share
2019 | 4.6 | 2.7% | 24% | 1.7% | $10.77
2018 | 4.7 | 2.0% | 22% | 1.5% | $9.34
2017 | 4.8 | 1.0% | 23% | 1.5% | $8.18
Question: What was the average dividend yield for the 3 years from 2017 to 2019?
Entities: 1.7% | 1.5% | 1.5%
Formula: (x0 + x1 + x2) / 3
Answer: (1.7 + 1.5 + 1.5) / 3

Table 16: The prompt of TAT-QA.

Solve the following questions with the programs.
The program consists of a sequence of operations.
Each operation takes a list of arguments.
There are 6 mathematical operations: add, $subtract$, $multiply$, $divide$, $greater$, exp.
And 4 table aggregation operations: $table-max$, $table-min$, $table-sum$, $table-average$.
The mathematical operations take arguments of either numbers from the given text and table, or a numerical result from a previous step.
The table operations take arguments of table row names.
We use the special token #n to denote the result from the nth step.
The given information is enough to solve the question.

Read the following text and table, and then answer a question:
$ in millions | year ended december 2014 | year ended december 2013 | year ended december 2012
fixed income currency and commodities client execution | $ 8461 | $ 8651 | $ 9914
equities client execution1 | 2079 | 2594 | 3171
commissions and fees | 3153 | 3103 | 3053
securities services | 1504 | 1373 | 1986
total equities | 6736 | 7070 | 8210
total net revenues | 15197 | 15721 | 18124
operating expenses | 10880 | 11792 | 12490
pre-tax earnings | $ 4317 | $ 3929 | $ 5634
Question: what was the percentage change in pre-tax earnings for the institutional client services segment between 2012 and 2013?
Entities: 3929, 5634, 5634
Formula: divide(subtract(x0, x1), x2)
Answer: divide(subtract(3929, 5634), 5634)

Read the following text and table, and then answer a question:
during the year ended march 31 , 2012 , the company has recorded $ 3.3 million in stock-based compensation expense for equity awards in which the prescribed performance milestones have been achieved or are probable of being achieved .
- | number of shares (in thousands) | weighted average grant date fair value (per share)
restricted stock and restricted stock units at beginning of year | 407 | $ 9.84
granted | 607 | 18.13
vested | -134 (134) | 10.88
forfeited | -9 (9) | 13.72
restricted stock and restricted stock units at end of year | 871 | $ 15.76
Question: during the 2012 year , did the equity awards in which the prescribed performance milestones were achieved exceed the equity award compensation expense for equity granted during the year?
Entities: 607, 18.13, 1000, 3.3, 1000000
Formula: greater(multiply(multiply(x0, x1), x2), multiply(x3, x4))
Answer: greater(multiply(multiply(607, 18.13), const_1000), multiply(3.3, const_1000000))

Read the following text and table, and then answer a question:
- | september 24 2005 | september 25 2004 | september 27 2003
beginning allowance balance | $ 47 | $ 49 | $ 51
charged to costs and expenses | 8 | 3 | 4
deductions (a) | -9 (9) | -5 (5) | -6 (6)
ending allowance balance | $ 46 | $ 47 | $ 49
Question: what was the highest ending allowance balance, in millions?
Entities: ending allowance balance
Formula: table_max(x0, none)
Answer: table_max(ending allowance balance, none)

Table 17: The prompt of FinQA.

Answer the given question.
You firstly generate the used values, which must be mentioned in the question.
Then you generate the formula structure.
Finally you generate the answer formula based on the values and the formula structure.
You only need to generate the formula without any other words, not to calculate the answer.

Question: An aquarium holds an equal number of clownfish and blowfish. 26 of the blowfish stay in their own tank, and the remaining blowfish swim into a display tank. An equal number of clownfish join the blowfish in the display tank, but then a third of these clownfish swim back into their own tank. If the aquarium holds a combined total of 100 fish, how many clownfish are now in the display tank?
Entities: total_fish = 100 | blowfish_in_own_tank = 26
Formula: total_blowfish_fish = total_fish / 2 | blowfish_in_display_tank = total_blowfish_fish - blowfish_in_own_tank | clownfish_in_display_tank = blowfish_in_display_tank | ans = clownfish_in_display_tank * 2 / 3
Answer: (100 / 2 - 26) * 2 / 3

Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?
Entities: total_eggs = 16 | eaten_eggs = 3 | baked_eggs = 4 | dollars_per_egg = 2
Formula: sold_eggs = total_eggs - eaten_eggs - baked_eggs | ans = sold_eggs * dollars_per_egg
Answer: (16 - 3 - 4) * 2

Question: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
Entities: bolts_of_blue_fiber = 2
Formula: bolts_of_white_fiber = num_of_blue_fiber / 2 | ans = bolts_of_blue_fiber + bolts_of_white_fiber
Answer: 2 + (2 / 2)

Question: Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?
Entities: cost_of_original_house = 80000 | cost_of_repair = 50000 | increase_rate = 1.5
Formula: value_of_house = (1 + increase_rate) * cost_of_original_house | ans = value_of_house - cost_of_repair - cost_of_original_house
Answer: ((1 + 1.5) * 80000) - 50000 - 80000

Question: Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed. How many cups of feed does she need to give her chickens in the final meal of the day if the size of Wendi's flock is 20 chickens?
Entities: numb_of_chickens = 20 | cups_for_each_chicken = 3 | cups_in_the_morning = 15 | cups_in_the_afternoon = 25
Formula: cups_for_all_chicken = num_of_chickens * cups_for_each_chicken | ans = cups_for_all_chicken - cups_in_the_morning - cups_in_the_afternoon
Answer: (20 * 3) - 15 - 25

Question: Marissa is hiking a 12-mile trail. She took 1 hour to walk the first 4 miles, then another hour to walk the next two miles. If she wants her average speed to be 4 miles per hour, what speed (in miles per hour) does she need to walk the remaining distance?
Entities: average_mile_per_hour = 4 | total_trail_miles = 12
Formula: remaining_miles = total_trail_miles - 4 - 2 | total_hours = total_trail_miles / average_mile_per_hour | remaining_hours = total_hours - 2 | ans = remaining_miles / remaining_hours
Answer: (12 - 4 - 2) / ((12 / 4) - 2)

(Ignore two examples because the whole prompt exceeds the length of one single page.)

Table 18: The prompt of GSM8K.