TACT: Advancing Complex Aggregative Reasoning with Information Extraction Tools

Avi Caciularu[γ] **Alon Jacovi**[γ] **Eyal Ben-David**[γ] **Sasha Goldshtein**[γ]
Tal Schuster[δ] **Jonathan Herzig**[γ] **Gal Elidan**[γ,η] **Amir Globerson**[γ,τ]

[γ]Google Research [δ]Google DeepMind [η]The Hebrew University of Jerusalem [τ]Tel Aviv University
avica@google.com

tact-benchmark.github.io

Abstract

Large Language Models (LLMs) often do not perform well on queries that require the aggregation of information across texts. To better evaluate this setting and facilitate modeling efforts, we introduce TACT—Text And Calculations through Tables, a dataset crafted to evaluate LLMs' reasoning and computational abilities using complex instructions. TACT contains challenging instructions that demand stitching information scattered across one or more texts, and performing complex integration on this information to generate the answer. We construct this dataset by leveraging an existing dataset of texts and their associated tables. For each such tables, we formulate new queries, and gather their respective answers. We demonstrate that all contemporary LLMs perform poorly on this dataset, achieving an accuracy below 38%. To pinpoint the difficulties and thoroughly dissect the problem, we analyze model performance across three components: table-generation, Pandas command-generation, and execution. Unexpectedly, we discover that each component presents substantial challenges for current LLMs. These insights lead us to propose a focused modeling framework, which we refer to as *IE as a tool*. Specifically, we propose to add "tools" for each of the above steps, and implement each such tool with few-shot prompting. This approach shows an improvement over existing prompting techniques, offering a promising direction for enhancing model capabilities in these tasks.

Figure 1: Annotated components of the TACT dataset. The answer is concise but demands advanced reasoning. Intermediate artifacts aid in analyzing LLM reasoning and designing the *IE as a tool* method. Relevant spans are underlined.

38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks.

1 Introduction

Large Language Models (LLMs) have shown exceptional capabilities across a wide range of natural language tasks. However, they still face significant challenges in solving complex problems that require reasoning over data presented in non-mathematical formats, such as word and algebraic problems [Amini et al., 2019, Dua et al., 2019]. Interestingly, research indicates that these types of problems pose difficulties not only for LLMs but also for humans [Cummins et al., 1988, Elliott, 2023]. This difficulty mirrors a broader observation about LLM reasoning capabilities: the process of transforming linguistic or graphical inputs into solvable mathematical equations is often more challenging than performing the calculations themselves [Schick et al., 2023, Das et al., 2024]. This overarching issue is particularly evident when LLMs attempt tasks that involve the aggregation of information from either single or multiple texts. These models frequently underperform in tasks that require counting, comparing, or processing similar events or entities within texts [Caciularu et al., 2022, Amouyal et al., 2023, Li et al., 2023]. This highlights a fundamental limitation: while LLMs can handle isolated data points effectively, their ability to integrate and interpret information across contexts remains a significant hurdle.

A first step towards advancing the capabilities of LLMs on complex reasoning, is to have a high-quality benchmark for evaluating and analyzing their performance in this setting. To this end, we introduce TACT—Text And Calculations through Tables. TACT instances were created by NLP and data science experts who wrote aggregative queries over texts. The experts were instructed to use tables (from the InstructIE dataset [Gui et al., 2023]) as the basis for writing the instructions, as they consolidate dispersed information from source texts into a structured format, enabling comprehensive aggregation across the text. The resulting TACT instances consist of the original text, the written instruction, and a gold answer, all requiring advanced text comprehension and reasoning (illustrated in Figure 1). Importantly, the TACT task does not include the table, so as to test the ability of the model to answer aggregative queries in an end-to-end manner, requiring advanced text comprehension and reasoning. Through TACT, the model is implicitly required to address information extraction (IE) challenges such as coreference resolution [Lee et al., 2017, Joshi et al., 2019, Kirstain et al., 2021], multi-hop reasoning [Lin et al., 2018, Dua et al., 2019, Zhao et al., 2022], summarization [Zhang et al., 2020, Goyal et al., 2022, Slobodkin et al., 2023], and multi-document processing [Caciularu et al., 2021, Hirsch et al., 2021, Caciularu et al., 2023, Zhu et al., 2024]. In Section 2, we describe our methodology for constructing this benchmark—by layering our new expert annotations over InstructIE instances—and the measures we took to ensure its difficulty and robustness.

Having evaluated LLMs on TACT and observed the challenges it presents to them (Section 4), we aim to understand the root of these difficulties and explore potential modeling improvements. We propose dissecting the problem into three tasks: table-generation, Pandas command-generation, and command-execution. Leveraging TACT's ground-truth tables and Pandas commands curated by experts, we analyze the LLM performance of each step in Section 5. Our findings reveal significant performance headroom in each task, implying that with targeted few-shot prompting, models can considerably enhance their individual task performance. Building on these results, we propose a focused modeling strategy termed the *IE as a tool* framework, which specifically addresses each phase independently (see an illustration in Figure 4 and more details in Section 3). This approach has shown to be superior to existing prompting techniques, as detailed in Section 4. The demonstrated improvements suggest a promising direction for enhancing LLM capabilities in complex reasoning tasks, aligning with our initial findings of untapped potential in each dissected component of the task.

Our contributions are summarized as follows:

- TACT: An expert-curated, diverse evaluation dataset that challenges LLMs on following aggregative queries, requiring information extraction and complex reasoning.

- A rigorous analysis of LLM performance on decomposed TACT tasks, revealing model strengths and weaknesses in table-generation, Pandas command-generation, and execution.

- Introduction of the *IE as a tool* framework, leveraging the aforementioned sub-tasks as discrete tools, demonstrating up to 12% improvement over conventional prompting techniques.

2

2   Dataset

This section introduces TACT—Text And Calculations through Tables—a novel challenge set designed to evaluate and improve the capability of LLMs on complex queries that require integration of information. The data is derived from the InstructIE [Jiao et al., 2023] test set using new expert-annotated labels, as described below. In this section, we detail the data labeling methodology employed to create TACT, highlighting the steps taken to ensure the *reliability* and *validity* of the labeled data. We first introduce the InstructIE benchmark creation methodology (Section 2.1), then we introduce our TACT dataset (Section 2.2), and finally, we explore the properties and conduct an analysis of TACT (Section 2.3).

2.1   Background: The InstructIE Dataset

InstructIE [Jiao et al., 2023] is a dataset that includes texts alongside corresponding tables, which summarize the textual content. These tables effectively organize the extracted information into sets of triples—subjects, relations, and objects—derived from the texts. To compile the tables and texts in the test set of InstructIE, which we employed for creating TACT, human annotators first defined the table topics and columns using real-world texts from the web. These texts were then utilized to craft tables that summarize them through a process combining automatic extraction and manual validation.

The primary components of InstructIE that we utilized are: **Text**– the accompanying document or collection of short documents, **Table**– a structured representation of the extracted information, where the first row serves as the table header and the subsequent rows contain the extracted data. See an illustrative example in Figure 7 in Appendix C.2, and Jiao et al. [2023] for additional details and descriptions of other components that were not included in our study. While InstructIE provides a good setup for information extraction, it does not directly test the models' abilities to aggregate the extracted information, which we target in TACT.

2.2   The TACT Dataset

Our goal is to evaluate the capabilities of LLMs in addressing aggregative, information-seeking queries, that require both text comprehension and complex reasoning. Using tables as the basis for creating such queries is highly effective, since they consolidate the essential information from their source texts into a structured format. Thus, performing an aggregation on these tables is equivalent to executing an aggregation across the entire text. We leverage and extend the use of the InstructIE dataset, which already contains structured information in table format (see above). We introduce the Text And Calculations through Tables (TACT) challenge set, aimed at verifying the capabilities of LLMs in handling complex numerical instructions (the resulting TACT dataset and its components are compared to InstructIE in Figure 7 in Appendix C.2).

TACT was created by NLP and data science experts, who employed a rigorous annotation process to transform InstructIE instances into a format suitable for aggregative instruction following. Creating the data includes the steps of assessing the text and the table, then formulating a query in natural language, and finally translating the query into a Pandas command, and executing it on the table. We chose Pandas over other languages, such as SQL, due to its simplicity. While SQL requires defining a schema, Pandas can easily operate on a single dataframe and often provide solutions with just a single line of code.

Additionally, two human passes were conducted over the dataset, where an expert human validator ensured 100% accuracy. The expert achieved this level of precision given the lack of ambiguity in the questions, further strengthening the reliability of the data. See the data creation guidelines, summarization of the annotation process, and more details about the data creation in Appendix C). The full steps are:

Initial Review and Relevance Vetting: A comprehensive review of the InstructIE dataset, focusing on texts and tables out of InstructIE's test set containing numerical data. Experts identified tables and text segments where numerical data was present and suitable for quantitative instructions. Tables were vetted for numerical integrity and alignment with the text to ensure data quality. For the remaining examples, the experts were tasked to convert the Markdown-formatted tables from InstructIE into the CSV table format (for convenient parsing into the Pandas dataframe format).

3

Numerical Aspect Identification: Experts identified *specific numerical aspects* within the text and tables—such as years, currencies, population counts, and temperatures—that enable quantitative operations like counting, calculation, and aggregation. This step identifies which aspect of the table data should be incorporated into the instruction.

Natural Language Instruction Formulation: Based on the identified numerical aspects, experts formulated clear and precise natural language instructions over the text that result in a single numerical value. These instructions targeted the numerical aspects with a focus on aggregation functions like sum, mean, and filtering.

Natural Language Query Over the Table: After formulating the natural language instructions, experts verbalized them into corresponding natural language queries over the tables (see Figure 1). These queries refined the focus on the numerical data within the table, minimized ambiguity, and helped to prepare the Pandas command.

Translation to Pandas Commands and Gold Response Extraction: Next, experts translated the previous natural language query over the table into a Pandas command. Then, they extracted the gold response by executing the formulated Pandas commands over the tables.

Command Execution and Validation: Finally, the extracted responses were manually verified against the expected outcomes derived from the formulated instructions and texts. This validation step ensured that the results were consistent with the intended instructions and the underlying data from both the text and the tables.

Each instance in the dataset consists of (see illustrative example in Figure 1):

1. Original Text and Table: Sourced from the InstructIE dataset, these elements contain the foundational data and numerical information relevant to the query. The text provides context, while the table offers structured numerical data aligned with the text content.

2. Natural Language Question: A clearly formulated query in natural language that targets specific numerical aspects identified in the text and table. These questions focus on computational tasks like sum, mean, and filtering to challenge the models' understanding and processing capabilities.

3. Natural Language Query Over the Table: After formulating the natural language question, a corresponding natural language query over the table is developed. This step refines the focus on the numerical data within the table, ensuring that the essential information for the computation is precisely delineated and consistent with the intent of the initial question.

4. Pandas Command: A precise translation of the natural language question into a Pandas command. This command is designed to replicate the expected computational process using the original column names from the table, ensuring the accuracy and consistency of the data manipulation.

5. Expected Result: The correct numerical answer derived from executing the Pandas command, serving as a benchmark to validate the models' responses against the ground truth.

The resulting TACT dataset contains 124 examples,[1] as well as additional 4 examples that serve as optional few-shot examples for in-context learning.[2] For evaluating performance on TACT, we employ exact match for the final answer, since it is a single-span (number). For intermediate steps available in TACT, such as table- and command-generation, we utilize a both similarity metrics (e.g., ROUGE [Lin, 2004]) and execution-based metrics (e.g., accuracy of the generated command's output) as described in Section 5.

2.3 Exploring the Numerical Challenges in TACT

In this section, we delve into the characteristics of the TACT dataset. TACT offers a diverse range of tasks, primarily focusing on two types of instructions–"Calculate" and "Count":

---

[1]Recent studies, through empirical validation, found that 100 examples are sufficient to conduct a high-quality evaluation of LLMs [Liang et al., 2023, Polo et al., 2024].

[2]These examples are sourced from the validation set of InstructIE, but follow the same process for constructing the TACT, as elaborated in Section 2.2.

4

"Calculate" Instructions: Out of the 124 examples, 63 are categorized under "Calculate" instructions. These tasks require the execution of basic mathematical operations to solve the instance. As depicted in Figure 2, the operations include addition, subtraction, multiplication, division, and other arithmetic functions. The distribution of these operations, such as summation (28.3%), mean calculation (10.4%), and power functions (6.1%), highlights the varied complexity and the need for precise computational understanding by the models.

"Count" Instructions: The remaining 61 examples fall under "Count" instructions, where the primary objective is to identify specific types or categories within the attached text and perform a simple counting operation. This task challenges the model's ability to accurately parse and interpret textual data, identifying relevant entities or events, and perform the proper counting.

The composition of the TACT dataset, with a balanced mix of "Calculate" and "Count" instructions, ensures a comprehensive evaluation of models across different dimensions of numerical reasoning. The operations, as detailed in the pie chart (Figure 2), further emphasize the diversity and scope of numerical challenges that TACT presents, offering a broad testbed for evaluating both fundamental and complex computational reasoning. This variety plays a pivotal role in assessing how well models can generalize their mathematical skills to real-world tasks.
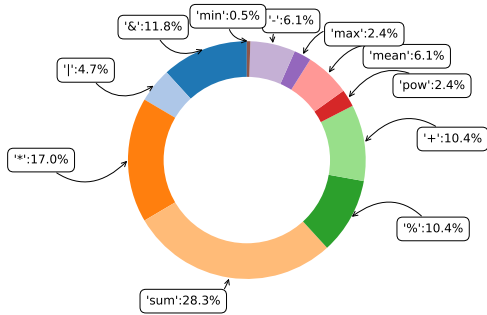
Figure 2: The TACT Dataset pandas different tokens' distribution.
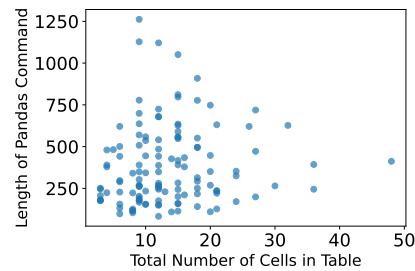
Figure 3: TACT's Pandas commands' length vs. the total number of cells in their corresponding tables.

In Figure 3, we present a comparison of the total number of cells in a table against the lengths of the corresponding Pandas commands. The figure reveals a wide distribution of data points, illustrating that the length of Pandas commands, quantified in Gemini tokens [Gemini-Team et al., 2023], does not correlate directly with the total number of cells in the tables. The varied spread of points across the graph indicates that additional factors, such as the complexity of arithmetic operations required or the specific data arrangement within the tables, might play a more significant role in determining the length of the commands than simply the volume of data.

In Table 1, we present an illustration of the diverse range of implicit tasks incorporated within TACT, which are specifically designed to test advanced text comprehension and numerical reasoning. Each task is tied to text spans that underline the specific data points or contextual clues necessary for task completion, ranging from multi-document summarization to date and time numerical reasoning. For example, one task leverages coreference resolution, requiring the model to understand and connect information spread across different parts of the text. Another task tests the model's capability for lexical matching, identifying specific words within a context. Complex arithmetic operations are also present, demanding a high level of numerical literacy to interpret numerical and financial concepts. This highlights the interplay between linguistic understanding and numerical computations, demonstrating the ability to handle a wide spectrum of real-world tasks—from simple counting to complex, multi-step mathematical operations embedded within textual data.

5

Table 1: An overview of implicit tasks in TACT, split by their types of 'Count' and 'Calc.' (Calculate) instruction types, along with textual instructions, accompanying texts and their corresponding sub-tasks. The sub-tasks' related aspects in the text are underlined.

Operation		Instruction	Relevant text spans	Implicit task
Count	Calc.			
	✓	*Calculate the sum of the years that "To Kill a Mockingbird" was published in, and the year that it won a prize according to the text.*	. . . *It was published in <u>1960</u> . . . A year after <u>its</u> release, <u>it</u> won the Pulitzer Prize* . . .	Coreference resolution
✓		*Count the number of achievements that include instructions.*	<u>1.</u> *We present FLAIR* . . . <u>2.</u> *How well can NLP models perform?* . . . <u>3.</u> *Pretrained language models have become increasingly prominent* . . .	Multi-document summarization
	✓	*Calculate the sum of squares of the stock price increases in the text.*	. . . *The S&P 500 <u>rose 1.45%</u> . . . the Nasdaq Composite <u>popped 1.07%</u> . . . The Dow Jones Industrial Average led gains, <u>rising 2.12%</u>* . . .	Complex arithmetics
✓		*Count the number of weather forecasts that include temperatures between 50 and 91 degrees.*	. . . *with temps currently ranging from the <u>upper 80s to low 90s</u> . . . Overnight, seasonal temps in the <u>upper 60s to low 70s</u> continue* . . .	Numerical range entailment
✓		*Count the number of wars in the text that have "Indian" within their names.*	. . . *The American <u>Indian Wars</u>, also known as the . . . and the <u>Indian Wars</u>* . . .	Lexical matching
✓		*Count the number of cases where the delivery date was later than June 2, 2023 and the travel time was more than 2 hours in this text.*	. . . *2. On <u>June 10, 2023</u>, XYZ Shipping's Truck 789, . . . Departing at <u>8:30 AM</u> and arriving at <u>2:00 PM</u>* . . .	Date and time numerical reasoning

3   IE as a Tool

As demonstrated by our experiments in the subsequent sections, current models face significant challenges when tackling the TACT task. To address this, we introduce a novel approach called *IE as a Tool*, which is illustrated in Figure 4. The core idea is to handle TACT instructions through the sequential use of two distinct tools: one that generates a table from the provided text and instruction, and another that formulates the corresponding Pandas command. The model then executes the command, alongside the original instruction and text, to derive the final answer. This sequence offers a natural and efficient strategy for addressing TACT's aggregative queries.

The implementation of these tools can follow multiple methods. For simplicity, we adopted a few-shot prompting approach, as detailed in the prompt templates in Appendix D.3. Our experimental results reveal that *IE as a Tool* yields up to 12% improvement in performance on TACT, outperforming conventional prompting techniques (see Section 4).
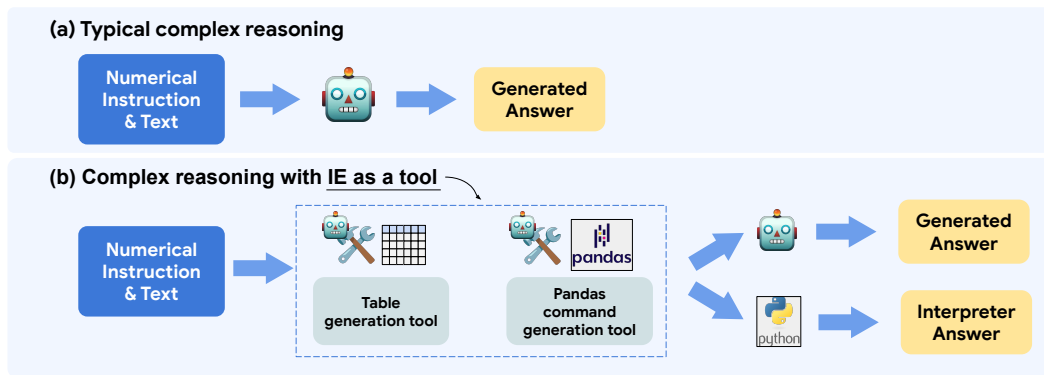
Figure 4: Possible setups for solving TACT with LLMs. (a) Typical Approach: the large language model (LLM) directly generates the answer based on the provided query and text, but without the aid of any external tools. (b) With *IE as a Tool*: This approach utilizes a three-step process. First, an information extraction tool generates a structured table from the text and query. Next, another tool formulates an appropriate Pandas command based on this table, the text and the query. Finally, all this information is fed into the LLM, which then generates the answer; or into a code interpreter that can run the Pandas command over the table.

6

4 Experimental Setup and TACT Results

We assess the performance of several models on the TACT dataset, including GPT-4o [OpenAI et al., 2024], Claude 3.5 Sonnet [Anthropic, 2024], Gemini-1.0-Ultra [Gemini-Team et al., 2023], Gemini-1.5-Pro [Reid et al., 2024], Llama-2-13b-chat, Llama-2-70b-chat [Touvron et al., 2023], Gemma-7b-it (v1.1) [Gemma-Team et al., 2024], Mistral-7b-instruct (v0.2) [Jiang et al., 2023], and Mixtral 8x7B [Jiang et al., 2024]. Their capabilities are tested using both standard prompting techniques and our *IE as a tool* method (detailed in Section 3). Subsequent analysis focuses on the specific sub-tasks of table generation (Section 5.1) and Pandas command generation (Section 5.2), providing a comprehensive evaluation of each model's performance and identifying potential areas for improvement in each sub-task.

We measure performance by averaging over four combinations of few-shot examples (by sampling the order and examples), following the procedure described in Jacovi et al. [2023b], and report the results for {0, 2, 4}-shot, using dedicated examples from the validation set of InstructIE (see more details in Appendix D).

We evaluate the LLMs' performance on the TACT task, measuring exact match, where the models are provided a numerical instruction and a text, and are tasked to produce the correct numerical answer. We report 4-shot results for Gemma-7b-it, Mistral-7b-instruct, and Mixtral 8x7B, while zero-shot results for the remaining larger models, given the results on table and Pandas query generation in the following sections, and the observation that few-shot demonstrations yield saturated performance for most of the tasks for the larger models. We propose the following experimental setups for each LLM:

- **Generic:** The baseline setting where the LLM receives a TACT instruction and a text passage, and is tasked with directly generating the answer.
- **Chain-of-Thought (CoT):** This setting is akin to the baseline, with the enhancement of adding "Let's think step-by-step" to the prefix (input). This encourages the model to generate a detailed, step-by-step CoT reasoning before producing the final answer.
- **In-context IE:** This method adopts a chain-of-thought-like approach, where the LLM first generates a table from the text, then creates a Pandas query, and finally, with all this context provided, the model generates the answer, all within the same prompt.
- **IE as a tool:** Here, the model can utilize generated tables and Pandas queries to answer the instruction, like the previous setup, however, in this variation, we employ few-shot prompted LLMs as separate tools (the number of shots applies for these tools). See Section 3 for more details. We include both *Without Pandas* and *With Pandas* variants as an ablation. We include these since as detailed in Section 5.2, even with syntactic errors in the Pandas commands, these errors may still assist the model in generating correct outputs.
- **IE as a tool (Gold):** This configuration is similar to the previous one (with Pandas command) but utilizes gold-standard (i.e., ground truth) tables (Gold Table) and/or Pandas commands (Gold Table+Pandas) from TACT, rather than relying on outputs generated by the tools. This baseline serves as an upper bound for the performance of the *IE as a tool* approach.

Table 2: Exact match accuracy evaluation results of different models on TACT, evaluated across different experimental setups, including Generic, Chain-of-Thought (CoT), In-context IE, and IE as a tool with various settings. The best-performing results are highlighted in **bold**.

Model	Generic	CoT	In-context IE	IE as a Tool		
				Without/With Pandas	Gold Table	Gold Table + Pandas
Gemma-7b (4-shot)	17.1	25.4	26.2	27.1 / 28.9	33.3	45.1
Mistral-7b (4-shot)	2.4	2.6	2.6	2.7 / 3.5	4.1	10.9
Llama-2-13b (4-shot)	8.5	8.0	8.3	8.5 / 8.5	9.2	14.3
Mixtral 8x7B (4-shot)	4.4	4.2	5.2	5.9 / 6.1	6.3	12.6
Llama-2-70b (0-shot)	3.6	4.0	4.2	4.5 / 7.6	12.9	22.4
Gemini-Pro (0-shot)	28.4	34.7	12.3	40.2 / 41.9	48.5	72.1
Gemini-Ultra (0-shot)	25.4	37.3	36.6	39.7 / 41.4	49.8	72.3
GPT-4o (0-shot)	**30.1**	37.7	36.4	**41.1 / 42.2**	50.9	74.1
Claude 3.5 Sonnet (0-shot)	28.6	**37.9**	**36.8**	40.8 / 42.1	**51.3**	**74.6**

7

The results, which are depicted in Table 2, show that all models experience substantial benefits when using the *IE as a tool* approach, with a small improvement when tasked to generate a Pandas command. This particularly clear in the larger models (Gemini, GPT, and Claude), which excel on this task, where Claude outperforms the rest of the models. This is evident from the consistent performance improvement across different models when comparing the *IE as a tool* setup with the Generic baseline and other approaches. On the other hand, smaller models showed more moderate improvements when using *IE as a tool*. The gap between the performance in *IE as a tool* and the Gold variant implies a significant potential for enhancing the overall task effectiveness through the refinement of IE tools. Specifically, the larger gap for smaller models suggests that their capabilities can be dramatically increased by improving the accuracy and reliability of the generated tables and commands. This points towards the critical importance of future work on optimizing IE tools to maximize the end-task performance for complex reasoning tasks.

5 Performance Analysis via TACT Decomposition

To understand the factors contributing to the suboptimal performance of current LLMs on TACT, we decompose the problem into two constituent tasks: 1) table generation from text based on a TACT instruction and the corresponding text (Section 5.1), and 2) Pandas query generation based on the corresponding table (either gold or previously generated), instruction, and text (Section 5.2). Successful execution of these two tasks would naturally result in accurate TACT results. We assess current LLM capabilities on each task using gold outputs, revealing substantial headroom and a potential for improvement. This observation directly motivated our design of *IE as a Tool*, a method that demonstrably improved TACT performance.

5.1 Evaluating the Accuracy of Table Generation

We assess the capabilities of LLMs to generate the appropriate tables given a TACT instruction and its corresponding text, tasking the model to construct the correct table based on the specified instruction. Note that the model should infer the correct table from the TACT instruction, which only implicitly points towards the relevant information to extract. Each model is provided with TACT instructions and corresponding texts. The task requires generating tables that accurately reflect the data described in the text, and helps to seek the correct information given the instruction.

We follow the evaluation protocol from Jiao et al. [2023] and adopt a soft matching strategy [Jiao et al., 2022] by using SentenceT5-Large [Ni et al., 2022] to calculate the cosine similarity (multiplied by 100) as the semantic similarity score between the generated table and the gold table, as table contents reflect the quality of extraction. Additionally, we use the ROUGE-L F1 score [Lin, 2004] to evaluate the lexical similarity of the generated table to the gold one. We also report the Table Validity rate, where we were able to parse a syntactically correct CSV table from the generated content.

Table 3: Evaluation Results of Different Models on TACT table generation, measuring semantic similarity, ROUGE-L F1 (lexical matching), and table validity between the generated tables and the gold tables. The best-performing results are highlighted in **bold**.

	Semantic Similarity			ROUGE-L F1			Table Validity Rate (%)		
Model	0-shot	2-shot	4-shot	0-shot	2-shot	4-shot	0-shot	2-shot	4-shot
Gemma-7b	68.6	69.1	69.4	6.5	6.6	7.1	0.6	23.1	25.5
Mistral-7b	73.5	72.8	72.8	4.9	6.4	7.0	1.2	30.9	34.4
Llama-2-13b	73.4	72.7	73.1	4.3	5.2	5.5	42.7	43.5	47.4
Mixtral 8x7B	72.3	71.7	71.8	4.5	7.1	7.2	9.5	39.9	24.3
Llama-2-70b	72.3	72.4	72.7	3.9	5.1	4.9	92.5	73.4	73.8
Gemini-Pro	78.4	78.2	78.3	18.8	21.0	22.9	81.5	90.2	93.3
Gemini-Ultra	**78.6**	78.6	79.3	18.7	21.1	24.8	81.3	89.9	94.1
GPT-4o	78.2	**78.9**	79.9	19.3	**23.2**	27.3	93.4	93.6	95.7
Claude 3.5 Sonnet	78.5	78.6	**80.1**	**19.7**	23.1	**28.1**	**94.1**	**94.2**	**96.2**

8

The evaluation of various LLMs on their ability to generate accurate tables based on TACT instructions is shown in Table 3. Claude and GPT mostly outperform other models across all metrics. The Gemini models also shows strong performance but vary across different shots, indicating potential instability in its output quality. Llama-2-13b, Llama-2-70b, and Mistral-7b exhibit moderate performance, with Mistral-7b achieving a higher semantic similarity but lower table validity rates. Gemma-7b and Mixtral 8x7B show comparatively lower performance, particularly in table validity. Notably, the smaller models like Gemma-7b and Mistral-7b benefit significantly from few-shot learning, demonstrating that small models are incapable of solving this task without any aid.

While the semantic similarity between the generated tables of Gemini-Ultra and the gold standard tables is relatively high, lexical similarity remains low. However, a qualitative analysis suggests that the generated tables contain key information that addresses the instructions, despite their differences from the gold tables. This observation supports the use of semantic similarity as a more appropriate metric for evaluating table generation in this context [Jiao et al., 2023].

5.2 Evaluating the Accuracy of Pandas Command Generation

The ability to accurately generate Pandas commands is a key intermediate step in solving TACT queries, and evaluates how well LLMs can comprehend the TACT instruction and the table at once. Thus, we next evaluate the ability of LLMs to generate Pandas commands, when provided with the TACT instruction, the associated text, as well as a table extracted from the text. We consider two cases: one where the provided table is the gold one, and one where it is the one generated by the model. To assess the quality of the generated Pandas queries, we execute them using a Python interpreter, and compare the output to the gold answer.

Table 4: Evaluation Results of Different Models on TACT Pandas command generation on the generated/gold table, measured by the accuracy after executing the command with a Python interpreter. The best-performing results are highlighted in **bold**.

Model	0-shot (Generated/Gold)	2-shot (Generated/Gold)	4-shot (Generated/Gold)
Gemma-7b	0 / 0.4	1.4 / 2.3	1.9 / 2.4
Mistral-7b	0.0 / 0.1	0.4 / 0.6	0.5 / 0.9
Llama-2-13b	0.0 / 0.3	0.1 / 0.6	0.3 / 1.2
Mixtral 8x7B	1.2 / 1.8	1.3 / 2.1	1.5 / 2.9
Llama-2-70b	2.5 / 3.4	3.1 / 4.1	3.2 / 4.3
Gemini-Pro	3.4 / 3.6	3.0 / 4.8	7.3 / 8.0
Gemini-Ultra	1.1 / 1.9	4.8 / 5.0	7.7 / 8.4
GPT-4o	4.5 / 4.9	5.3 / 6.0	8.7 / 9.4
Claude 3.5 Sonnet	**5.1 / 5.9**	**5.6 / 6.4**	**9.6 / 10.1**

Table 4 presents the results, where Claude consistently outperforms the other models. GPT, Gemini, and Llama-2-70b also demonstrate relatively strong performance, though with some variability across different shot configurations. Interestingly, as in the previous experiment, the smaller models—such as Gemma-7b and Mistral-7b—showed lower performance overall but exhibited significant improvements with few-shot learning, highlighting their ability to effectively leverage additional examples. Llama-2-13b and Mixtral 8x7B delivered moderate performance but still trailed behind the larger models.

It is worth noting that the overall numbers in Table 4 are quite low, even when compared to the results in Table 2, including for gold-standard tables usage. This may seem surprising at first, but upon inspection, we found that many of the Pandas commands generated by the models contain syntax errors or other issues that lead to execution failures. In contrast, the results in Table 2 do not involve Python execution, which allows for more robust command interpretation and, as a result, better answers.

6 Related Work

Information Extraction (IE) and Text-to-Table IE is the process of automatically extracting structured information from unstructured text, involving sub-tasks like named entity recognition,

9

relation extraction, and event extraction. Many works have leveraged large language models (LLMs) to provide effective solutions for IE [Ma et al., 2023, Lu et al., 2023, Zhou et al., 2024]. Recently, Wu et al. [2022] presented the concept of text-to-table, and Jiao et al. [2023] introduced InstructIE, a benchmark that includes triplets of an IE instruction, their associated text, and the relevant content in a tabular format (see Section 2). We employ a labeling methodology on top of InstructIE to distill an aggregative instruction following challenge set. Yuan et al. [2024] presented an effort with a similar focus on numerical tasks but with a narrower scope, limited to financial data. Another related research realm is open information extraction, which aims to extract information without predefined schemas, typically focusing on simple structures from short texts [Banko et al., 2007, Mausam et al., 2012, Stanovsky et al., 2018, Zhan and Zhao, 2020].

Complex and Numerical Reasoning A persistent challenge for LLMs lies in their ability to solve numerical problems, particularly those involving mathematical calculations [Geva et al., 2020, Imani et al., 2023, Chang et al., 2024, Ahn et al., 2024] or the need to stitch and aggregate information across the text [Li et al., 2023, Sprague et al., 2024, Jacovi et al., 2024]. Some works propose to evaluate models on such tasks, by presenting challenge datasets based on financial data [Chen et al., 2021, Yuan et al., 2024]. DROP [Dua et al., 2019] and IIRC [Ferguson et al., 2020], two reading comprehension benchmarks involving reasoning, showcase the complexity of comprehensive numerical reasoning. DROP focuses on discrete reasoning over paragraphs, requiring models to perform operations like addition, counting, and sorting, while IIRC evaluates the ability to handle incomplete contexts and locate additional sources of information. TACT focuses on a more practical and common use-case: reasoning over texts given natural language instructions, necessitating the integration of information scattered throughout the text.

Semantic Parsing is the process of converting natural language into a machine-interpretable representation, such as a formal query or command [Pasupat and Liang, 2015, Yoran et al., 2022, Mekala et al., 2023, Bogin et al., 2023]. *IE as a tool* also aligns with previous research on semantic parsing, as we utilize executable Pandas command generation over texts and tables, demonstrating how LLMs can interpret and convert complex instructions into executable code operations.

Multi-step Reasoning and LLM Tools Our research is closely related to various methodologies that utilize LLM tools for task resolution, as explored in recent studies [Parisi et al., 2022, Mialon et al., 2023, Schick et al., 2023, Hao et al., 2023, Patil et al., 2023]. These methods train LLMs to utilize APIs independently during inference, contrasting with our *IE as a tool* approach, which employs a static strategy for addressing complex reasoning tasks. Furthermore, prior research has emphasized enhancing task resolution through multi-step processes [Berant et al., 2014, Drozdov et al., 2023, Zhou et al., 2023, Fu et al., 2023]. Unlike these approaches, which apply general multi-step reasoning or tool triggering across various domains, *IE as a tool* specifically concentrates on constructing tables and executing commands for numerical reasoning, thereby targeting a more focused application of multi-step numerical reasoning.

7   Conclusion

In this paper, we introduced TACT—Text And Calculations through Tables, a dataset designed to assess the reasoning capabilities of Large Language Models (LLMs) through complex, aggregative instructions. TACT features numerical instructions that require processing and integrating information dispersed across one or more texts for producing the correct answer. By leveraging the InstructIE dataset [Jiao et al., 2023], experts annotated and transformed instances into a format suitable for aggregative instruction following, ensuring high precision and relevance. To better understand the performance of LLMs on TACT, we provide further analysis that evaluates performance on two distinct sub-tasks that are likely to be relevant for solving TACT (table-generation and Pandas command-generation). We also provide a modeling scheme, *IE as a tool*, that is based on this decomposition, and show that it improves performance on TACT. Future work could focus on further enhancing the performance of LLMs on TACT by developing and integrating new, more sophisticated tools that are specifically designed for handling complex, aggregative instructions over text.

10

Acknowledgements

We thank Jonathan Berant, Yonatan Bitton, Mor Geva, and Eran Ofek for their valuable feedback and constructive suggestions, and Ayelet Shasha Evron for her assistance in designing the figures for this paper.

References

J. Ahn, R. Verma, R. Lou, D. Liu, R. Zhang, and W. Yin. Large language models for mathematical reasoning: Progresses and challenges. In N. Falk, S. Papi, and M. Zhang, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024: Student Research Workshop, St. Julian's, Malta, March 21-22, 2024*, pages 225–237. Association for Computational Linguistics, 2024. URL `https://aclanthology.org/2024.eacl-srw.17`.

A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1245. URL `https://aclanthology.org/N19-1245`.

S. Amouyal, T. Wolfson, O. Rubin, O. Yoran, J. Herzig, and J. Berant. QAMPARI: A benchmark for open-domain questions with many answers. In S. Gehrmann, A. Wang, J. Sedoc, E. Clark, K. Dhole, K. R. Chandu, E. Santus, and H. Sedghamiz, editors, *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 97–110, Singapore, Dec. 2023. Association for Computational Linguistics. URL `https://aclanthology.org/2023.gem-1.9`.

Anthropic. The Claude 3 Model Family: Opus, Sonnet, Haiku, 2024. URL `https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf`.

M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In M. M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2670–2676, 2007. URL `http://ijcai.org/Proceedings/07/Papers/429.pdf`.

J. Berant, V. Srikumar, P.-C. Chen, A. Vander Linden, B. Harding, B. Huang, P. Clark, and C. D. Manning. Modeling biological processes for reading comprehension. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1159. URL `https://aclanthology.org/D14-1159`.

B. Bogin, S. Gupta, P. Clark, and A. Sabharwal. Leveraging Code to Improve In-context Learning for Semantic Parsing. *CoRR*, abs/2311.09519, 2023. doi: 10.48550/ARXIV.2311.09519. URL `https://doi.org/10.48550/arXiv.2311.09519`.

A. Caciularu, A. Cohan, I. Beltagy, M. Peters, A. Cattan, and I. Dagan. CDLM: Cross-document language modeling. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2648–2662, Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.225. URL `https://aclanthology.org/2021.findings-emnlp.225`.

A. Caciularu, I. Dagan, J. Goldberger, and A. Cohan. Long context question answering via supervised contrastive learning. In M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2872–2879, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.207. URL `https://aclanthology.org/2022.naacl-main.207`.

11

A. Caciularu, M. Peters, J. Goldberger, I. Dagan, and A. Cohan. Peek across: Improving multi-document modeling via cross-document question-answering. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1989, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.110. URL `https://aclanthology.org/2023.acl-long.110`.

Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie. A Survey on Evaluation of Large Language Models. *ACM Trans. Intell. Syst. Technol.*, 15(3), mar 2024. ISSN 2157-6904. doi: 10.1145/3641289. URL `https://doi.org/10.1145/3641289`.

Z. Chen, W. Chen, C. Smiley, S. Shah, I. Borova, D. Langdon, R. Moussa, M. Beane, T. Huang, B. R. Routledge, and W. Y. Wang. Finqa: A dataset of numerical reasoning over financial data. In M. Moens, X. Huang, L. Specia, and S. W. Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3697–3711. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.300. URL `https://doi.org/10.18653/v1/2021.emnlp-main.300`.

D. D. Cummins, W. Kintsch, K. Reusser, and R. Weimer. The Role of Understanding in Solving Word Problems. *Cognitive Psychology*, 20(4):405–438, 1988. ISSN 0010-0285. doi: https://doi.org/10.1016/0010-0285(88)90011-4. URL `https://www.sciencedirect.com/science/article/pii/0010028588900114`.

D. Das, D. Banerjee, S. Aditya, and A. Kulkarni. MATHSENSEI: A Tool-Augmented Large Language Model for Mathematical Reasoning. *arXiv preprint arXiv:2402.17231*, 2024.

A. Drozdov, N. Schärli, E. Akyürek, N. Scales, X. Song, X. Chen, O. Bousquet, and D. Zhou. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=gJW8hSGBys8`.

D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2368–2378. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1246. URL `https://doi.org/10.18653/v1/n19-1246`.

E. C. Elliott. Why Word Problems are Hard for High School Math Students: Problem Formulation and Disciplinary Literacy. 2023.

J. Ferguson, M. Gardner, H. Hajishirzi, T. Khot, and P. Dasigi. IIRC: A dataset of incomplete information reading comprehension questions. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1137–1147. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.86. URL `https://doi.org/10.18653/v1/2020.emnlp-main.86`.

Y. Fu, H. Peng, A. Sabharwal, P. Clark, and T. Khot. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=yf1icZHC-l9`.

Gemini-Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Gemma-Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

12

M. Geva, A. Gupta, and J. Berant. Injecting numerical reasoning skills into language models. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.89. URL `https://aclanthology.org/2020.acl-main.89`.

T. Goyal, J. J. Li, and G. Durrett. News Summarization and Evaluation in the Era of GPT-3. *arXiv preprint arXiv:2209.12356*, 2022.

H. Gui, J. Zhang, H. Ye, and N. Zhang. InstructIE: A Bilingual Instruction-based Information Extraction Dataset. *ArXiv*, abs/2305.11527, 2023. URL `https://api.semanticscholar.org/CorpusID:258823375`.

S. Hao, T. Liu, Z. Wang, and Z. Hu. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023. URL `https://openreview.net/forum?id=BHXsb69bSx`.

E. Hirsch, A. Eirew, O. Shapira, A. Caciularu, A. Cattan, O. Ernst, R. Pasunuru, H. Ronen, M. Bansal, and I. Dagan. iFacetSum: Coreference-based interactive faceted summarization for multi-document exploration. In H. Adel and S. Shi, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 283–297, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-demo.33. URL `https://aclanthology.org/2021.emnlp-demo.33`.

S. Imani, L. Du, and H. Shrivastava. Mathprompter: Mathematical reasoning using large language models. In S. Sitaram, B. B. Klebanov, and J. D. Williams, editors, *Proceedings of the The 61st Annual Meeting of the Association for Computational Linguistics: Industry Track, ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 37–42. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-INDUSTRY.4. URL `https://doi.org/10.18653/v1/2023.acl-industry.4`.

A. Jacovi, A. Caciularu, O. Goldman, and Y. Goldberg. Stop Uploading Test Data in Plain Text: Practical Strategies for Mitigating Data Contamination by Evaluation Benchmarks. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5075–5084, Singapore, Dec. 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.308. URL `https://aclanthology.org/2023.emnlp-main.308`.

A. Jacovi, A. Caciularu, J. Herzig, R. Aharoni, B. Bohnet, and M. Geva. A Comprehensive Evaluation of Tool-Assisted Generation Strategies. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13856–13878, Singapore, Dec. 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.926. URL `https://aclanthology.org/2023.findings-emnlp.926`.

A. Jacovi, M. Ambar, E. Ben-David, U. Shaham, A. Feder, M. Geva, D. Marcus, and A. Caciularu. Coverbench: A challenging benchmark for complex claim verification. *arXiv preprint arXiv:2408.03325*, 2024.

A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Y. Jiao, S. Li, Y. Xie, M. Zhong, H. Ji, and J. Han. Open-Vocabulary Argument Role Prediction For Event Extraction. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5404–5418, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.395. URL `https://aclanthology.org/2022.findings-emnlp.395`.

Y. Jiao, M. Zhong, S. Li, R. Zhao, S. Ouyang, H. Ji, and J. Han. Instruct and extract: Instruction tuning for on-demand information extraction. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages

10030–10051, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.620. URL `https://aclanthology.org/2023.emnlp-main.620`.

M. Joshi, O. Levy, L. Zettlemoyer, and D. Weld. BERT for coreference resolution: Baselines and analysis. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5803–5808, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1588. URL `https://aclanthology.org/D19-1588`.

Y. Kirstain, O. Ram, and O. Levy. Coreference resolution without span representations. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 14–19, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.3. URL `https://aclanthology.org/2021.acl-short.3`.

K. Lee, L. He, M. Lewis, and L. Zettlemoyer. End-to-end neural coreference resolution. In M. Palmer, R. Hwa, and S. Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1018. URL `https://aclanthology.org/D17-1018`.

J. Li, M. Wang, Z. Zheng, and M. Zhang. LooGLE: Can Long-Context Language Models Understand Long Contexts? *arXiv preprint arXiv:2311.04939*, 2023.

P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, B. Yuan, B. Yan, C. Zhang, C. A. Cosgrove, C. D. Manning, C. Re, D. Acosta-Navas, D. A. Hudson, E. Zelikman, E. Durmus, F. Ladhak, F. Rong, H. Ren, H. Yao, J. WANG, K. Santhanam, L. Orr, L. Zheng, M. Yuksekgonul, M. Suzgun, N. Kim, N. Guha, N. S. Chatterji, O. Khattab, P. Henderson, Q. Huang, R. A. Chi, S. M. Xie, S. Santurkar, S. Ganguli, T. Hashimoto, T. Icard, T. Zhang, V. Chaudhary, W. Wang, X. Li, Y. Mai, Y. Zhang, and Y. Koreeda. Holistic evaluation of language models. *Transactions on Machine Learning Research (TMLR)*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=iO4LZibEqW`. Featured Certification, Expert Certification.

C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-1013`.

X. V. Lin, R. Socher, and C. Xiong. Multi-hop knowledge graph reasoning with reward shaping. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1362. URL `https://aclanthology.org/D18-1362`.

D. Lu, S. Ran, J. Tetreault, and A. Jaimes. Event extraction as question generation and answering. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1666–1688, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.143. URL `https://aclanthology.org/2023.acl-short.143`.

Y. Ma, Y. Cao, Y. Hong, and A. Sun. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10572–10601, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.710. URL `https://aclanthology.org/2023.findings-emnlp.710`.

Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. Open language learning for information extraction. In J. Tsujii, J. Henderson, and M. Paşca, editors, *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL `https://aclanthology.org/D12-1048`.

14

D. Mekala, J. Wolfe, and S. Roy. ZEROTOP: Zero-shot task-oriented semantic parsing using large language models. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5792–5799, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.354. URL `https://aclanthology.org/2023.emnlp-main.354`.

G. Mialon, R. Dessi, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Roziere, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, E. Grave, Y. LeCun, and T. Scialom. Augmented language models: a survey. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=jh7wH2AzKK`. Survey Certification.

J. Ni, G. Hernandez Abrego, N. Constant, J. Ma, K. Hall, D. Cer, and Y. Yang. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.146. URL `https://aclanthology.org/2022.findings-acl.146`.

OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Łukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Łukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph. Gpt-4 technical report, 2024.

A. Parisi, Y. Zhao, and N. Fiedel. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022.

P. Pasupat and P. Liang. Compositional semantic parsing on semi-structured tables. In C. Zong and M. Strube, editors, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1142. URL `https://aclanthology.org/P15-1142`.

15

S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

F. M. Polo, L. Weber, L. Choshen, Y. Sun, G. Xu, and M. Yurochkin. tinyBenchmarks: evaluating LLMs with fewer examples. *arXiv preprint arXiv:2402.14992*, 2024.

M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-b. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023. URL `https://openreview.net/forum?id=Yacmpz84TH`.

A. Slobodkin, A. Caciularu, E. Hirsch, and I. Dagan. Don't add, don't miss: Effective content preserving generation from pre-selected text spans. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12784–12800, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.852. URL `https://aclanthology.org/2023.findings-emnlp.852`.

Z. R. Sprague, X. Ye, K. Bostrom, S. Chaudhuri, and G. Durrett. MuSR: Testing the limits of chain-of-thought with multistep soft reasoning. In *The International Conference on Learning Representations (ICLR)*, 2024.

G. Stanovsky, J. Michael, L. Zettlemoyer, and I. Dagan. Supervised open information extraction. In M. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 885–895, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1081. URL `https://aclanthology.org/N18-1081`.

H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

X. Wu, J. Zhang, and H. Li. Text-to-table: A new way of information extraction. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2518–2533. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-LONG.180. URL `https://doi.org/10.18653/v1/2022.acl-long.180`.

O. Yoran, A. Talmor, and J. Berant. Turning tables: Generating examples from semi-structured tables for endowing language models with reasoning skills. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6016–6031, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.416. URL `https://aclanthology.org/2022.acl-long.416`.

Z. Yuan, K. Wang, S. Zhu, Y. Yuan, J. Zhou, Y. Zhu, and W. Wei. FinLLMs: A Framework for Financial Reasoning Dataset Generation with Large Language Models. *CoRR*, abs/2401.10744, 2024. doi: 10.48550/ARXIV.2401.10744. URL `https://doi.org/10.48550/arXiv.2401.10744`.

J. Zhan and H. Zhao. Span model for open information extraction on accurate corpus. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9523–9530. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6497. URL `https://doi.org/10.1609/aaai.v34i05.6497`.

J. Zhang, Y. Zhao, M. Saleh, and P. Liu. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*,

16

pages 11328–11339. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/zhang20ae.html`.

Y. Zhao, Y. Li, C. Li, and R. Zhang. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.454. URL `https://aclanthology.org/2022.acl-long.454`.

D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. V. Le, and E. H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=WZH7099tgfM`.

W. Zhou, S. Zhang, Y. Gu, M. Chen, and H. Poon. UniversalNER: Targeted distillation from large language models for open named entity recognition. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=r65xfUb76p`.

A. Zhu, A. Hwang, L. Dugan, and C. Callison-Burch. FanOutQA: Multi-Hop, Multi-Document Question Answering for Large Language Models. *arXiv preprint arXiv:2402.14116*, 2024.

17

A   Limitations

While our work and the introduced tools specifically target numerical complex reasoning tasks, they are not designed to address the full spectrum of natural language processing challenges. Consequently, their application to non-numerical tasks may not yield optimal results. Another significant limitation is the sequential use of tools, which can introduce and propagate errors through the processing stages, potentially compromising the accuracy and reliability of the final response. This propagation of errors underscores the need for careful handling and validation at each step to mitigate compounding inaccuracies. Future work should aim to develop more versatile tools and methodologies that can handle a broader range of tasks while minimizing the risk of error propagation.

B   License and Intended Use

The TACT benchmark, including templates and instructions, is licensed under the Creative Commons Attribution NoDerivs 4.0 International License (CC BY ND 4.0). Contributions derived from InstructIE [Jiao et al., 2023], such as texts and tables, are provided under the terms of this license. Users must assume responsibility for their use in accordance with the obligations to the creators of InstructIE. The intended use of these materials is for the improvement and evaluation of large language models (LLMs), and we assume full responsibility for any potential violations of rights and confirm adherence to the licensing agreements associated with the data used in this study.[3]

We emphasize the strict use of the TACT dataset exclusively for evaluation purposes, prohibiting its inclusion in NLP model training datasets to mitigate potential biases and contamination. We implement measures to prevent data contamination as outlined by Jacovi et al. [2023a], and we require that any future redistribution or use of the data adheres to these same guidelines. Additionally, redistribution of any part of the dataset is advised against without robust measures to block web-crawler access. To facilitate the tracing and management of potential data contamination within web-crawled corpora, a distinct 64-character identifier string is appended to each dataset instance.

C   TACT Data Creation and Guidelines

In this section, we include additional details and material regarding the data creation process. We provide the official guidelines that were given to the experts for creating the data (Appendix C.1) along with the concluded data creation process (Appendix C.2).

C.1   Guidelines

The data creation process was guided by a comprehensive set of guidelines, prepared to equip the participants with the necessary skills and knowledge for the task. These prerequisites included familiarity with basic Python, proficiency in the Pandas library, and an understanding of data aggregation concepts such as sum, mean, and filtering. The original instructions are depicted in Figure 5.

C.2   Creation Process

In the creation of TACT, we engaged NLP and data science experts (with a PhD degree specializing in NLP), each with a minimum of four years of experience, to ensure high-quality data curation. These experts reported that labeling each example typically required between 16 to 20 minutes. Approximately 70% of this time was dedicated to carefully reading the provided tables and accompanying texts, and formulating challenging numerical questions that draw on the data. The remaining time was allocated to writing the queries, including one in Pandas and two in natural language, and executing the Pandas query on the table to verify that the intended results were achieved. This meticulous process guarantees that our dataset can both challenge LLMs and accurately reflects realistic scenarios where mathematical reasoning is essential. The summarized data creation process

---

[3]The data, the full licence details are all available in `https://huggingface.co/datasets/google/TACT`, and the TACT metadata is available in `https://huggingface.co/api/datasets/google/TACT/croissant`.

18

Data Creation Guidelines

Prerequisites
- Familiarity with basic Python and the Pandas library.
- Understanding of data aggregation concepts (e.g., sum, mean, filtering).

Task Objective
To extract and compute numerical aspects from provided text and table data in a replicable and unambiguous manner using the Pandas library.

Instructions

1. **Careful Review and Assessment:**
   - Read the text thoroughly.
   - Examine the attached instruction and provided table.
   - **Relevance Assessment:** Determine if the table contains numerical data that directly relates to the text and aligns with the given instructions.
   - Disregard tables if they:
     - i. Were overlooked during the annotation phase, especially those that consist of only a single row without any complex or challenging information leading to numerical computation.
     - ii. Contain inaccurate information or do not accurately reflect the content of the text.
     - iii. Are poorly structured, failing to clearly communicate key aspects.
     - iv. Focus solely on analyzing the structure rather than the content of the text, such as instructions aimed at extracting linguistic features, semantic relationships, and discourse structures.

2. **Numerical Aspect Identification:**
   - Identify a numerical aspect within both the text and the table that allows for counting, calculation, or aggregation (combining multiple values).
   - Ensure this aspect can be uniquely determined from the data.

3. **Natural Language Instruction Creation:**
   - Formulate a clear, well-defined question in natural language that addresses the chosen numerical aspect. Log the expected (numerical) answer for the question.
   - **Focus on Aggregation:** Consider adding to the question a requirement for summation, average, count, or other calculations on the data. You can add a complex function or combination as well, such as the sum of squares.
   - **Mitigate Ambiguity:** Ensure the instruction has a single, straightforward interpretation.

4. **Natural Language Query Over the Table:**
   - After formulating the natural language instruction, develop a corresponding natural language query over the table. This query refines the focus on the numerical data within the table, ensuring that the essential information for computation is accurately delineated and consistent with the intent of the initial instruction.

5. **Pandas Command Formulation:**
   - Translate your natural language instruction into a precise Pandas command.
   - **Column Name Adherence:** Use the original column names from the provided table.
   - **Clarity:** Strive for a command that is easy to understand and replicate.
   - **Expected Output:** The final command should yield a numerical answer.

6. **Command Execution and Validation:**
   - Execute your Pandas command.
   - **Verification:** Meticulously verify that the result aligns with the original question and the data within both the text and table.

Example

Text: " The warehouse floor was a Tetris puzzle of wooden crates from the recent shipment. Stacked against the far wall, which was about 10 feet high, twelve large crates towered over the rest, each a hefty 25 kilograms. It took about 4 hours to drop those. Closer to the loading dock, a cluster of six medium-sized 2 crates, 15 kilograms apiece, awaited their turn, which will roughly take 2 hours to unload. Tucked into a corner, five small crates, light at 8 kilograms each, seemed almost lost in the vast space, with about 30 minutes unloading time. This shipment was a mix of sizes and weights, ready to be distributed to various destinations."

Table:

Crate Size	Quantity	Weight (kg)
Large	12	25
Medium	6	15
Small	5	8

Natural Language Instruction: Calculate the total weight of the medium crates in the shipment, as if their quantity was equal to the number of small crates.

Natural Language Query Over the Table : Given the following table, write a single-line pandas command in python, to calculate the product of 'Weight (kg)' when its 'Crate Size' is equal to 'Medium', with the 'Quantity' when its 'Crate Size' is equal to 'Small'.

Pandas Command Formulation:
```
df[df['Crate Size'] == 'Medium']['Weight (kg)'].astype(int).item()
* df[df['Crate Size'] == 'Small']['Quantity'].astype(int).item()
```
Result: 75

Figure 5: The data creation guidelines for the TACT Dataset. This figure presents a comprehensive set of guidelines designed to assist annotators in extracting and computing numerical aspects from provided text and table data using the Pandas library. The guidelines include steps for reviewing and assessing the relevance of data, identifying numerical aspects, formulating natural language instructions and queries, translating these into precise Pandas commands, and validating the results. An example is provided to demonstrate the process, from text and table review to the execution and verification of the computed result.

19

with an accompanying example, following the guidelines above and the description in Section 2.2, is illustrated in Figure 6. The resulting dataset and its components are compared to InstructIE in Figure 7. [Jiao et al., 2023]

(a) Review and assess the text and the table, identify numerical aspects.

📖 **Text:** *The warehouse floor was a Tetris puzzle of wooden crates from the recent shipment. Stacked against the far wall, which was about 10 feet high, <u>twelve large crates</u> towered over the rest, each a hefty <u>25 kilograms</u>. It took about 4 hours to drop those. Closer to the loading dock, a cluster of <u>six medium-sized crates</u>, <u>15 kilograms</u> apiece, awaited their turn, which will roughly take 2 hours to unload. Tucked into a corner, five small crates, <u>light at 8 kilograms</u> each...*

📄 **Table:**

Crate size	Quantity	Weight (kg)
Large	12	25
Medium	6	15
Small	5	8

(b) Formulate a query in natural language, then write it as a query over the table.

📝 **TACT Instruction:** *Calculate the total weight of the medium crates in the shipment, as if their quantity was equal to the number of small crates.*

📝 **Query over the table:** *Given the following table, write a single-line pandas command in python, to <u>calculate</u> the product of 'Weight (kg)' when its 'Crate Size' is equal to 'Medium', with the 'Quantity' when its 'Crate Size' is equal to 'Small'.*

(c) Translate the queries into a pandas command, and execute it on the table.

🐼 **Pandas Command:**
```
df[df['Crate Size'] == 'Medium']['Weight (kg)'].astype(int).item() *
df[df['Crate Size'] == 'Small']['Quantity'].astype(int).item()
```

👉 **Answer:**
75

Figure 6: The summarized data creation process for the TACT benchmark. This illustration outlines the systematic guidelines employed for annotating numerical data derived from textual and tabular content within the TACT framework. It elaborates on the sequential steps necessary for annotators to effectively review text and tables, identify numerical data, formulate and translate these into natural language instructions and corresponding Pandas queries, and finally, execute and validate these commands. An example accompanies the instructions to showcase the entire process from initial review to the successful execution and verification of the result.

20
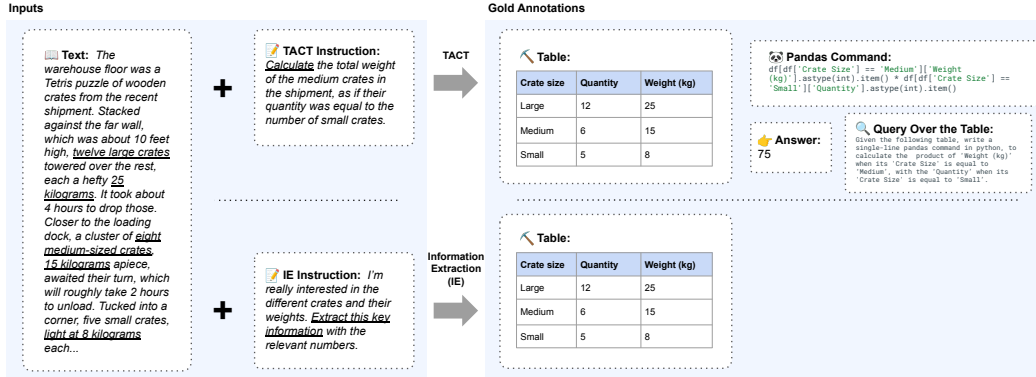
Figure 7: TACT (above) and typical Information Extraction (IE) (below). TACT includes an instruction, and allows various intermediate outputs before the answer generation, while IE focuses on table generation. The TACT instruction, the Pandas command, the query over the table, and the answer, were all created by NLP experts.

D Additional Experimental Details

In this section, we include additional experimental details, including the Large Language Models' (LLMs') settings we used in our experiments (Appendix D.1), the few-shot prompting setup (Appendix D.2), as well as the templates we used (Appendix D.3).

D.1 Models and Inference Details

In our experiments, all large language models (LLMs) were configured to operate with a temperature setting of 0.8. This parameter choice was aimed at striking an optimal balance between diversity and coherence in the model's responses, facilitating more varied yet still plausible outputs for the generation of the final answer as well as the tools. For tasks involving table generation and Pandas command generation, we parse the model's response to extract the desired output from the generated text. Conversely, for tasks requiring a numerical answer, we identify and consider the final number in the generated text as the definitive response.

For the inference of open-source models in our study – Llama-2-13b-chat[4], Llama-2-70b-chat[5] [Touvron et al., 2023], Gemma-7b-it[6] [Gemma-Team et al., 2024], Mistral-7b-instruct[7] [Jiang et al., 2023], and Mixtral 8x7B[8] [Jiang et al., 2024], we utilized 8 H100 GPUs. For the rest of the evaluated models, Gemini-1.0-Ultra, Gemini-1.0-Pro [Gemini-Team et al., 2023], we used the developers API[9].

D.2 Few-shot Prompting

In the few-shot evaluation settings, prompts were constructed by randomly choosing few-shot examples out of a pool of 4 demonstrations (that were manually curated from the validation set of InstructIE). These prompts were adjusted to include the maximum number of demonstrations that, along with the query, stayed within the context length limit for all the evaluated models. We ended up with 4 different prompts for each example, to maintain uniformity in the number of shots. For zero-shot examples we ran the same prompt, but the temperature sampling maintained randomness. This approach resulted in effectively 496 examples for each performance result report in Sections 4 and 5. The robustness of this methodology allowed us to achieve statistically significant results, demonstrating the reliability and efficacy of our few-shot evaluation framework in mimicking real-world analytical tasks.

---

[4] huggingface.co/meta-llama/Llama-2-13b-chat-hf

[5] huggingface.co/meta-llama/Llama-2-70b-chat-hf

[6] huggingface.co/google/gemma-1.1-7b-it

[7] huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

[8] huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

[9] ai.google.dev

21

```
 1   In this task, you should calculate the numerical answer that is the
          solution for the given instruction and text.  The answer should be
          numerical, using digits only.
 2
 3   Example1:
 4   Instruction: Calculate the sum of the year that "To Kill a Mockingbird"
          was published in and the year that it won a prize according to the
          text.
 5   Text: ... It was published in 1960 ... A year after its release, it won
          the Pulitzer Prize ...
 6   Answer: 4021
 7
 8   Example2:
 9   Instruction: Count the number of achievements that include instructions.
10   Text: 1. We present FLAIR ... 2. How well can NLP models perform? ... 3.
          Pretrained language models have become increasingly prominent ...
11   Answer: 3
12
13   Example3:
14   Instruction: Calculate the sum of squares of the stock price increases in
          the text.
15   Text: ... The S&P 500 rose 1.45% ... the Nasdaq Composite popped 1.07%
          ... The Dow Jones Industrial Average led gains, rising 2.12% ...
16   Answer: 0.0558
17
18   Now your turn:
19   Instruction: [Your custom instruction here]
20   Text: [Your relevant text spans here]
21   Answer:
22   ...
```

Figure 8: Example of the Generic TACT task prompt provided to our evaluated models. The prompt consists of basic instructions, several few-shot in-context examples, and the instance input. The examples demonstrate how to generate the answer to the instruction, based on given instruction and text. Note that the examples illustrated are slightly modified from the original TACT, for simplicity.

D.3 Prompt Templates

Next, we detail the template structures employed for our few-shot prompts. It's important to note that we tested multiple templates and settings to ensure that substantial effort was devoted to reporting the strongest baseline prompt implementations. Our approach included the creation of prompts designed for the specific tasks in this paper, each complemented by examples that demonstrate their goal. The Generic TACT task prompt is depicted in Figure 8, and the Table Generation task prompt, Table Generation task prompt, and the overall task prompt are illustrated in Figures 9, 10, and 11, respectively. Figure 11 matches both the *In-context IE* and *IE as a tool* setups, as for the latter we insert the generated table and Pandas command as the outputs from the tools in Figures 9 and 10 (see Section 4 for the setups' details). Note that for the Chain-of-Thought (CoT) setups, the "Let's think step-by-step" suffix was added.

22

```
 1   In this task, you should create an appropriate CSV table according to the
         given Instruction and Text.
 2
 3   Example1:
 4   Instruction: Calculate the sum of the year that "To Kill a Mockingbird"
         was published in and the year that it won a prize according to the
         text.
 5   Text: ... It was published in 1960 ... A year after its release, it won
         the Pulitzer Prize ...
 6   Table:
 7   "Event","Year"
 8   "Published","1960"
 9   "Prize Won","1961"
10
11
12   Example2:
13   Instruction: Count the number of achievements that include instructions.
14   Text: 1. We present FLAIR ... 2. How well can NLP models perform? ... 3.
         Pretrained language models have become increasingly prominent ...
15   Table:
16   "Number","Achievement"
17   "1","We present FLAIR"
18   "2","How well can NLP models perform?"
19   "3","Pretrained language models prominence"
20
21
22   Example3:
23   Instruction: Calculate the sum of squares of the stock price increases in
         the text.
24   Text: ... The S&P 500 rose 1.45% ... the Nasdaq Composite popped 1.07%
         ... The Dow Jones Industrial Average led gains, rising 2.12% ...
25   Table:
26   "Increase","Venture"
27   "1.45%","S&P"
28   "1.07%","Nasdaq Composite popped"
29   "2.12%","Dow Jones Industrial"
30
31
32   Now your turn:
33   Instruction: [Your custom instruction here]
34   Text: [Your relevant text spans here]
35   Table:
36   ...
```

Figure 9: Example table generation prompt provided to our evaluated models. The prompt consists of basic instructions, several few-shot in-context examples, and the instance input. The examples demonstrate how to create a CSV table based on given instructions and text. Note that the examples illustrated are slightly modified from the original TACT, for simplicity.

23

```
 1  In this task, you should create an appropriate Pandas command according to
        the given Instruction, Text, and Table, such that the command will
        run on the table and return the correct number answering the
        instruction.
 2
 3  Example1:
 4  Instruction: Calculate the sum of the year that "To Kill a Mockingbird"
        was published in and the year that it won a prize according to the
        text.
 5  Text: ... It was published in 1960 ... A year after its release, it won
        the Pulitzer Prize ...
 6  Table:
 7  "Event","Year"
 8  "Published","1960"
 9  "Prize Won","1961"
10  Pandas Command:
11  df['Year'].astype(int).sum()
12
13
14  Example2:
15  Instruction: Count the number of achievements that include instructions.
16  Text: 1. We present FLAIR ... 2. How well can NLP models perform? ... 3.
        Pretrained language models have become increasingly prominent ...
17  Table:
18  "Number","Achievement"
19  "1","We present FLAIR"
20  "2","How well can NLP models perform?"
21  "3","Pretrained language models prominence"
22  Pandas Command:
23  len(df)
24
25
26  Example3:
27  Instruction: Calculate the sum of squares of the stock price increases in
        the text.
28  Text: ... The S&P 500 rose 1.45% ... the Nasdaq Composite popped 1.07%
        ... The Dow Jones Industrial Average led gains, rising 2.12% ...
29  Table:
30  "Increase","Venture"
31  "1.45%","S&P"
32  "1.07%","Nasdaq Composite popped"
33  "2.12%","Dow Jones Industrial"
34  Pandas Command:
35  (df['Increase'].str.replace('%', '').astype(float) / 100).pow(2).sum()
36
37
38  Now your turn:
39  Instruction: [Your custom instruction here]
40  Text: [Your relevant text spans here]
41  Table: [Your relevant table here]
42  Pandas Command:
43  ...
```

Figure 10: Example Pandas command generation prompt provided to our evaluated models. The prompt consists of basic instructions, several few-shot in-context examples, and the instance input. The examples demonstrate how to create a Pandas command based on given instructions, text, and table. Note that the examples illustrated are slightly modified from the original TACT, for simplicity.

24

```
 1  In this task, you should calculate the numerical answer that is the
        solution for the given instruction and text.  Use the following Table
        in your calculations, by executing the Pandas Command on it.  The
        answer should be numerical, using digits only.
 2
 3  Example1:
 4  Instruction: Calculate the sum of the year that "To Kill a Mockingbird"
        was published in and the year that it won a prize according to the
        text.
 5  Text: ... It was published in 1960 ... A year after its release, it won
        the Pulitzer Prize ...
 6  Table:
 7  "Event","Year"
 8  "Published","1960"
 9  "Prize Won","1961"
10  Pandas Command:
11  df['Year'].astype(int).sum()
12  Answer: 4021
13
14  Example2:
15  Instruction: Count the number of achievements that include instructions.
16  Text: 1. We present FLAIR ... 2. How well can NLP models perform? ... 3.
        Pretrained language models have become increasingly prominent ...
17  Table:
18  "Number","Achievement"
19  "1","We present FLAIR"
20  "2","How well can NLP models perform?"
21  "3","Pretrained language models prominence"
22  Pandas Command:
23  len(df)
24  Answer: 3
25
26  Example3:
27  Instruction: Calculate the sum of squares of the stock price increases in
        the text.
28  Text: ... The S&P 500 rose 1.45% ... the Nasdaq Composite popped 1.07%
        ... The Dow Jones Industrial Average led gains, rising 2.12% ...
29  Table:
30  "Increase","Venture"
31  "1.45%","S&P"
32  "1.07%","Nasdaq Composite popped"
33  "2.12%","Dow Jones Industrial"
34  Pandas Command:
35  (df['Increase'].str.replace('%', '').astype(float) / 100).pow(2).sum()
36  Answer: 0.0558
37
38  Now your turn:
39  Instruction: [Your custom instruction here]
40  Text: [Your relevant text spans here]
41  Table: [Your relevant table here]
42  Pandas Command: [Your Pandas command here]
43  Answer:
44  ...
```

Figure 11: Example of the TACT task prompt provided to our evaluated models, using tables and Pandas commands. The prompt consists of basic instructions, several few-shot in-context examples, and the instance input. The examples demonstrate how to generate the answer to the instruction, based on given instruction, text, table, and Pandas command, and should generate the computed answer. Note that the examples illustrated are slightly modified from the original TACT, for simplicity.

25