S3Eval: A Synthetic, Scalable, Systematic Evaluation Suite for Large Language Models

Anonymous ACL submission

Abstract

The rapid development of Large Language Models (LLMs) has led to great strides in model capabilities like reasoning and long-context understanding. However, as LLMs are able to process longer contexts, it becomes more challenging to evaluate whether they have acquired certain capabilities, since the length of text (e.g., 100K tokens) they can process far exceeds what humans can reliably assess in a reasonable duration. In this paper, we propose using complex synthetic tasks as a proxy evaluation method, and present S3EVAL, a **S**ynthetic, **S**calable, **S**ystematic evaluation suite for LLMs evaluation. As a synthetic benchmark, S3EVAL enables the creation of any number of evaluation examples that are theoretically invisible to LLMs, mitigating the test set contamination issue. The synthetic nature of S3EVAL provides users full control over the dataset, allowing them to systematically probe LLM capabilities by scaling text length and varying task difficulty across diverse scenarios. The strong correlation between S3EVAL performance and scores of real-world benchmarks like Big-Bench Hard (BBH) demonstrates the soundness of using S3EVAL for evaluation of LLMs. The in-depth analysis also uncover additional insights, including performance drop when the answer is sparsely distributed or located in the middle context, as well as some counter-intuitive trends of model performance.

1 Introduction

Large Language Models (LLMs) have greatly propelled significant advancements in Natural Language Processing (NLP), such as OpenAI GPT (Brown et al., 2020), Llama (Touvron et al., 2023a,b), StarCoder (Li et al., 2023a), and others. These models perform well in many NLP tasks and claim to have made progress in advanced capabilities such as reasoning, long-context understanding, and so on. However, existing benchmarks (Chang et al., 2023) often fail when it comes to evaluating extremely long-context LLMs or analysing the controllable characteristics and limitations of LLMs.

For long-context understanding, previous work has often evaluated LLMs using the scope of language modeling metrics (i.e., perplexity) (Sun et al., 2021; Peng et al., 2023) or the performance on simple artificial tasks (Li and Roth, 2002; Berant et al., 2013; Mohtashami and Jaggi, 2023). However, these evaluation tasks tend to lack complexity and are narrowly focused on simple comprehension, which is misaligned with the sophistication required for real-world downstream applications. While recent work has made great progress on building evaluation benchmarks at longer context lengths with real-world use cases (e.g, question answering) (Bai et al., 2023b; An et al., 2023), these manually annotated datasets often lack the scale and diversity to thoroughly assess performance on extended context lengths. For example, existing benchmarks struggle to effectively evaluate LLMs that claim an ability to process contexts up to 100K tokens, due to the limited capacity of human annotation for very long text. Developing more scalable and diverse evaluation datasets, potentially leveraging automated supervision, remains an open challenge.

For reasoning analysis (Hendrycks et al., 2021b; Chen et al., 2021a; Suzgun et al., 2023; Zhong et al., 2023), conducting both qualitative and quantitative analysis of answers and reasoning processes provides important insights. However, existing benchmarks lack the ability to precisely control the distribution of the dataset, limiting their utility for in-depth research analysis. In other words, the nature of these benchmarks makes it challenging for developers to identify the specific weaknesses of their LLMs. More configurable and granular benchmarks are needed to enable detailed analysis of model performance. In addition, these benchmarks often draw their evaluation data from NLP tasks that have been extensively studied and are likely to be used in the training corpus of LLMs. The potential data leakage makes the evaluation less convincing.

In this paper, we propose a new evaluation suite called S3EVAL, which addresses the aforementioned issues by using a complex synthetic task - SQL execution - as a proxy for the performance of LLMs on realistic reasoning tasks. As shown in Figure 1, inspired by the work of TAPEX (Liu et al., 2022), S3EVAL is based on the SQL execution task. Specially, given a randomly generated table and a random SQL query, S3EVAL evaluates whether LLMs can return the correct execution results. S3EVAL has three notable characteristics: (1) It is synthetic, with no table or SQL query present in the LLM training corpus. The tasks use complex, grammatically correct SQL syntax, making them very challenging. (2) It is scalable, allowing users to customize the benchmark to any length and difficulty. (3) It is systematic, containing diverse reasoning types and operations. This enables comprehensive evaluation of LLM capabilities.

With these powerful features, developers can extend the context to really long lengths and generate meaningful SQL statements using S3EVAL. As demonstrated in Table 1, developers can modify the configuration settings to conduct focused studies on particular reasoning abilities (e.g., aggregation) of LLMs. We conducted comprehensive multi-perspective experiments on several popular LLMs using S3EVAL. Experimental results demonstrated that the performance of LLMs on S3EVAL aligns closely with their performance on mainstream LLM benchmarks. While LLMs have shown impressive capabilities, our work reveals limitations in their ability to leverage long contexts, since we observe performance degradation of almost all LLMs in long-context settings. By carefully studying experimental results, we can work to pinpoint situations where LLMs tend to fail and summarize valuable insights. For example, LLMs often encounter challenges when the answer lies in the middle of the context, similar to findings from Liu et al. (2023b), or when the answers are sparsely distributed across the input. We believe that these observations from S3EVAL provide valuable guidance for the development of LLMs and dynamic benchmarks.

2 Synthetic Suite: Alignment with Realistic Benchmark

In this section, we describe the details of synthesizing the evaluation data (Section 2.1) and verify the alignment between our synthetic suite S3EVAL and real-world benchmark results.

2.1 Suite Construction

Task Formulation Following previous work (Liu et al., 2022), each example in S3EVAL generally contain an SQL query and a (semi-)structured table T as the input. Each table T consists of M rows $\{r_i\}_{i=1}^{M}$, in which each row r_i contains N cell values $\{c_{\langle i,j\rangle}\}_{i=1}^{M}$. Each cell $c_{\langle i,j\rangle}$ corresponds to a table header h_j. Each SQL query consists of K tokens as $x = x_1, x_2, \cdots, x_K$. Each token x_i originates from SQL keywords, table schema, or table cells. Each multi-step instruction is transformed from SQL query. The task prompts LLM to obtain the execution result A of the SQL on the table T. Our main focus is on analyzing the accuracy of LLM in executing SQL queries.

Random Table Generation All tables in S3EVAL are randomly generated and do not contain any real data or overlap with existing public tables. The tables have M rows and N columns, with adjustable parameters M and N. The column headers are sampled from English nouns (Bird, 2006), falling into three types: TEXT, INT, and DATE. INT columns contain random integers from 1 to 1000, which is an adjustable range. DATE columns have values in year-month-day format. TEXT columns have random strings of length 5 to 12 characters, which is also adjustable. To simulate real-world data where the same value may recur in a column frequently, the data generator includes a parameter to set the probability of duplicating values within a specific column.

Random SQL Generation The SQL language includes a variety of statements to query and manage data. S3EVAL use context free grammar to generate a specific number of examples with controllable attributes. As Table 1 shows, the S3EVAL tool allows configuring several parameters of generated SQL statements, including nesting depth, keywords used, length, coverage of SQL features, computational complexity, and more. For example, *calculate times* can be modified to control the complexity of numerical reasoning for each dataset.

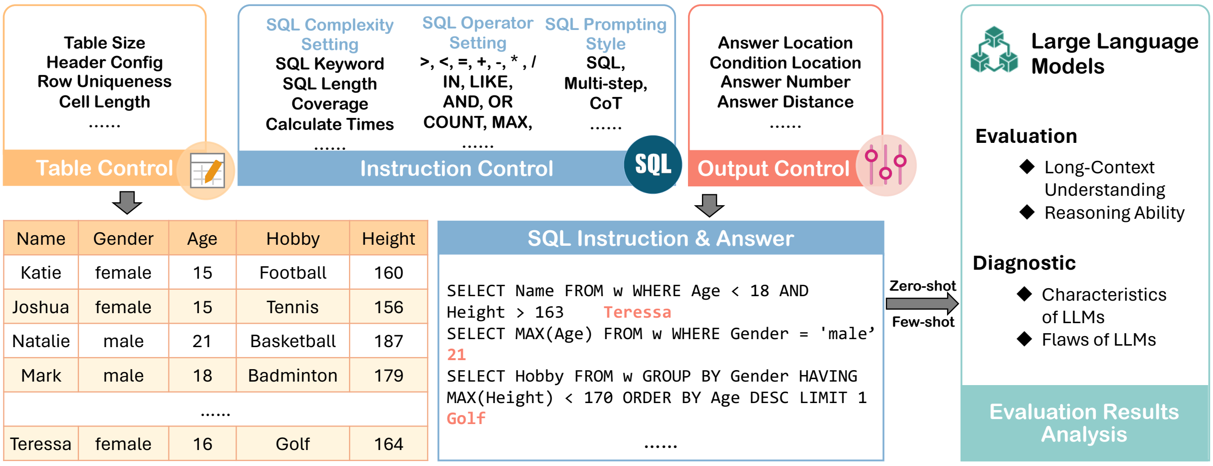


Figure 1: The illustration demonstrates the S3EVAL pipeline, where the capabilities of LLMs are assessed by evaluating their ability to execute SQL queries over randomly generated tables.

Configuration		Description
Table Control	# of Rows	The number of rows in the generated tables
	# of Columns	The number of columns in the generated tables
	Header Type Ratio	The proportion of table column types that are TEXT, INT, DATE
	Cell Uniqueness	The proportion of duplicate cells in each column
	String / Int Length	The string length or numeric range of cell values
Instruction Control	SQL Keywords	SELECT, WHERE, GROUP BY, HAVING, ORDER BY
	SQL Length	The number of tokens after SQL split by space
	Column Coverage	The ratio of columns involved in SQL execution to total columns.
	Row Coverage	The ratio of rows involved in SQL execution to total rows
	Calculate Times	The number of SQL numerical calculations.
	Filter Times	The number of SQL filtering operations.
	Aggregator	COUNT, MAX, MIN, SUM, AVG
	Filter Operator	>, <, =, IN, LIKE
Output Control	Answer Location	The location of SQL answers in the input table
	# of Answer Cells	The number of selected cells in the answer
	Answer Length	The total number of tokens in the answer

Table 1: Our S3EVAL method allows users to customize configuration settings and provides descriptions for each parameter that can be adjusted. More configurations can be found in Appendix D.1.

Except these configures, users can also manually write the specified SQL template to generate fine-grained evaluation data (Appendix C.2).

Evaluation Methods S3EVAL includes both zero-shot and few-shot prompting methods. For each few-shot setting, all examples share one table. N-shot is formalized as $\mathrm{INPUT} = [\mathrm{T}; \mathrm{S}_1; \mathrm{A}_1; \ldots; \mathrm{S}_{\mathrm{n}+1}]$. For the input format of table T, we designed several alternative ways, including markdown, flatten, tapex-style, etc.

To evaluate the performance of LLMs, we use Exact Match (EM) as the evaluation metric. Details are shown in Appendix C.3.

Evaluation Settings Models are evaluated on two settings: **Easy** and **General**. **Easy** is the simplest data that S3EVAL can generate and is used to evaluate LLM's ability to understand the most basic instructions. It contains only one template, "SELECT <col1> WHERE <col2> <op> <value>". **General** is a more difficult setting, containing extensive SQL syntax, and its generating setting is described in Appendix D.2.

General randomly generates SQL queries of varying difficulty from an extensive grammar and provides a comprehensive evaluation of LLM performance. All experiments were run for 3 times, using 1000 randomly generated queries per trial, with tables of 15 rows and 8 columns and an average of 1200 tokens per input. Details on the LLMs are provided in Appendix D.3. Considering *SQL execution* is a difficult task, some models may have a poor understanding of symbolic language, which makes it difficult to execute SQL, so we propose an alternative task *SQL multi-step task* to remove this potential bias. Specifically, it converts an SQL query into a multi-step table operation instruction as shown in Appendix C.5. SQL has a fixed execution flow for the query statement: FROM $\rightarrow$ ON $\rightarrow$ JOIN $\rightarrow$ WHERE $\rightarrow$ GROUP BY $\rightarrow$ HAVING $\rightarrow$ SELECT $\rightarrow$ ORDER BY $\rightarrow$ LIMIT. This is not consistent with the order in which it is written. With this processing, it can also generate chain-of-thought prompting data.

2.2 Alignment on Scaling Law

Previous work (Kaplan et al., 2020; Hoffmann et al., 2022) shows a positive correlation between the cross-entropy loss of LLMs and the amount of computing resources used for training, as described by the empirical scaling law. To verify whether the scaling law holds for our S3EVAL, we employ a set of checkpoints of Pythia-12B (Biderman et al., 2023) that are open-sourced at different training steps, corresponding to different amounts of compute. Using both General and Easy settings, we observe a consistent pattern as illustrated in Figure 2: the scores show a smooth progression of improvement that aligns with the scaling law with increasing the training steps. The steady, incremental performance gains over time, lacking any spikes, demonstrate S3EVAL's reliability as a evaluation suite. Overall, these experimental results confirm the scaling law's accuracy in forecasting model gains during training across diverse evaluation settings.

2.3 Alignment on Benchmark Performance

In the above, we validated that the LLMs also exhibits the scaling law observed in NL on the S3EVAL suite. A natural question that arises is whether its performance on S3EVAL is correlated with the performance on real-world, NL benchmarks. To examine the hypothesis, we first compare the performance of different LLMs on S3EVAL and on WikiTableQuestions (Pasupat and Liang, 2015), a table question answering dataset consisting of questions and answers. It is worth noting that to align the difficulty, we use the SQL queries from WikiTableQuestions (Shi et al., 2020) as our S3EVAL evaluation set.

To systematically compare the performance, following previous work (Liu et al., 2023a), we consider two correlation measures: the Pearson correlation coefficient (r), which evaluates the linear relationship between model scores on the two benchmarks, and the Kendall rank correlation coefficient (τ), which assesses whether the relative ranking of models is consistent across the benchmarks. The strong correlation between LLMs' performance on the SQL execution task and the table question answering task, as evidenced by the high r (e.g., 99.1) and high τ (e.g., 93.6) in Figure 3.

Although S3EVAL has shown significant correlation with WikiTableQuestions, the fact that they are both tasks on tables may cause one to question whether S3EVAL can serve as a proxy task to evaluate LLMs' capabilities on generic reasoning tasks. Therefore, we also compare the performance on S3EVAL with the results of generic popular benchmarks like BBH (Suzgun et al., 2023) and HumanEval (Chen et al., 2021a). The results depicted in Figure 4a demonstrate a strong correlation between LLM performance on S3EVAL and the BBH benchmark, with BBH performance obtained from the OpenCompass platform using few-shot chain-of-thought prompting (OpenCompass, 2023). Similarly, Figure 4b illustrates the alignment between S3EVAL performance and pass@1 scores on HumanEval (Chen et al., 2021b) for code LLMs. The results demonstrate that S3EVAL serves as a robust proxy task for assessing the reasoning capabilities of LLMs on realistic benchmarks. Concrete experimental results are provided in Table 3.

3 Scalable Suite: Unlimited Evaluation Resources

S3EVAL provides a unique capability to generate infinite number of examples (Section 3.1) with infinite length (Section 3.2).

3.1 Scalable Number of Evaluation Examples

The strength of S3EVAL is its ability to generate unlimited number of examples for evaluation. This stems from two key design choices in S3EVAL: (1) the synthetic table size can be scaled to different number of rows and columns, and (2) the table cells are synthesized from randomly generated strings. Combined with the provided large library of SQL query templates, these features enable the creation of a near-infinite set of unique evaluation examples. This kind of capacity enables the continuous creation of novel examples unseen during training, which helps safeguard test data integrity by preventing leakage of the evaluation set into the training corpus.

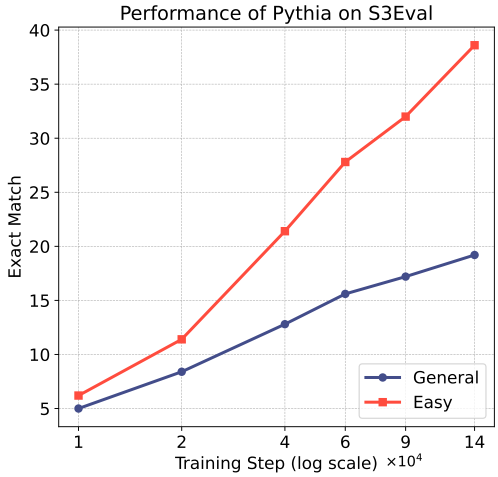


Figure 2: The performance of Pythia-12B on the Easy and General settings of S3EVAL was evaluated across different training steps.

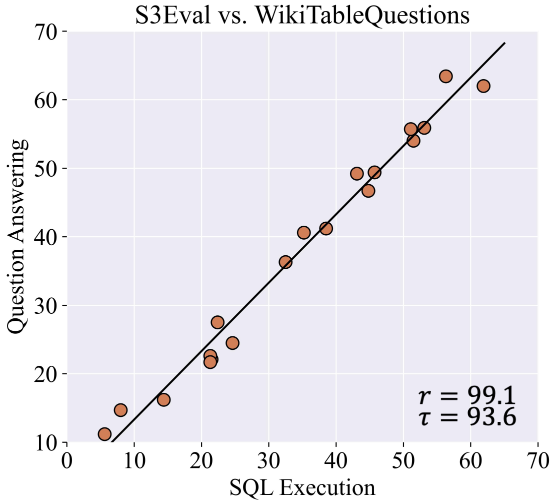
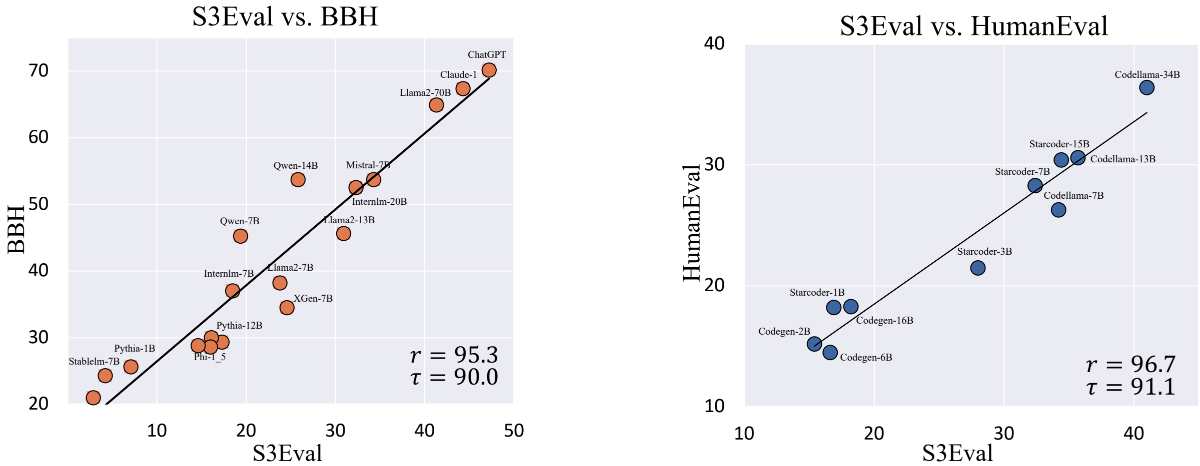


Figure 3: The performance of different LLMs on S3EVAL and WikiTableQuestions.



(a) Performance of large language models

(b) Performance of code large language models

Figure 4: Each point in the scatterplot represents the LLM performance on the benchmarks corresponding to the horizontal and vertical coordinates. The black straight line is the trend line. The larger the values of r and τ, the higher the correlation between the two benchmarks. We consider $\tau > 0.8$ to be high concurrence.

However, the absence of data leakage does not necessarily mean that S3EVAL's performance always represents the model's out-of-distribution generalization ability. It is because the model may perform well on S3EVAL via domain-specific training on the SQL execution task, rather than acquiring more general abilities. To investigate whether LLMs can "hack" S3EVAL via domain-specific training, we fine-tuned StarCoder-1B (Li et al., 2023a), which is not able to solve SQL execution tasks, on a randomly generated dataset of one million examples. The performance of the fine-tuned StarCoder-1B is illustrated in Figure 5, where it is evaluated on three types of test datasets: `Seen Table` (same tables as training), `Unseen Table` (new tables in same format as training tables), and `Unseen Templates` (new SQL query templates). For the unseen table setting, we explore different table shapes, where $(x \times y)$ means the table consists of x rows and y columns.

The experimental results demonstrate that for Unseen Tables with different shapes, regardless of their size, the performance of the fine-tuned StarCoder experiences a substantial decline compared to Seen Tables. Likewise, when faced with Unseen Templates, the performance of the fine-tuned StarCoder exhibits a significant drop. The results indicate that even if LLMs have been heavily trained on SQL execution tasks, their out-of-distribution performance can still be accurately evaluated by

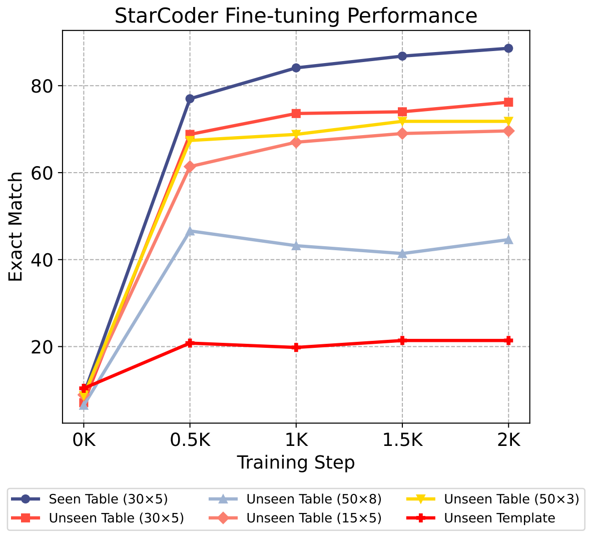


Figure 5: SQL execution training experiments on S3EVAL.

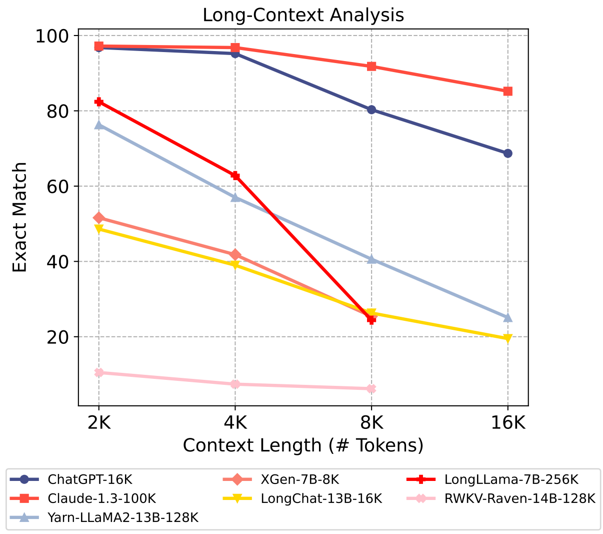


Figure 6: Experiment results of different LLMs on different context lengths.

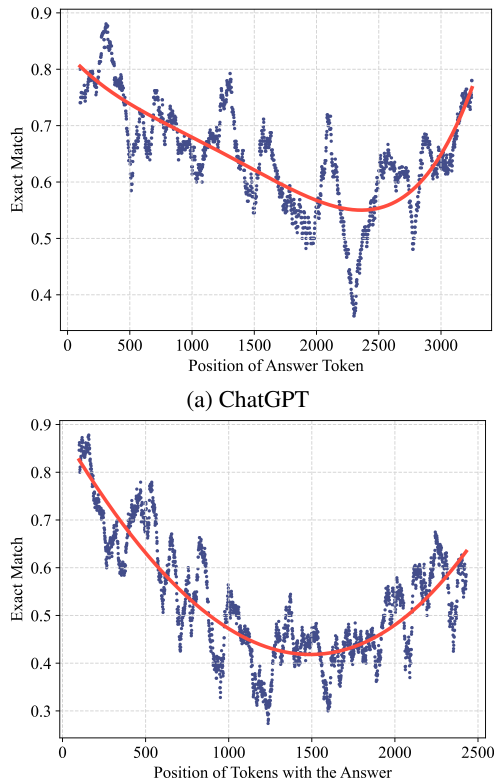
using novel SQL templates. These new SQL templates can be easily generated thanks to the vast grammar of SQL queries. Additionally, evaluating LLMs on larger tables that they were not trained on can also reveal part of their out-of-distribution capabilities.

3.2 Scalable Length of Evaluation Examples

One advantage of S3EVAL is its scalability and adjustable context length per example. The flexibility allows S3EVAL to rigorously evaluate LLMs that claim capability with long contexts. To clearly expose limitations of current LLMs, we intentionally chose the **Easy** setting in S3EVAL to evaluate their performance. Specifically, we establish table configurations with approximately 2K, 4K, 8K, and 16K tokens, by using different numbers of rows and fixing the number of columns. We generate a dataset consisting of 500 samples for each evaluation setting. The experimental results on up to 16K context length are plotted in Figure 6. As observed, the performance of almost all LLMs, significantly decreases as the context length increases. Of all the models, Claude-1.3-100K is the only one that maintains a relatively strong performance trend. Detailed results can be found in Appendix A.5.

Given the limitation of existing LLMs on long-context tasks, we are curious about the bottleneck of them. By using S3EVAL, we can systematically investigate the long-context modeling capabilities of LLMs by controlling the distribution of answers in the evaluation suite. Specifically, we use the **Easy** setting and fix the number of answers to four cells (i.e., the result of the SQL execution is always spanning four cells). As illustrated in Figure 8, we introduce two distribution patterns, *Dense* and *Sparse* [1] to probe the limitations of current LLMs. The dense mode only requires the model to understand the local context, whereas the sparse mode requires the model to have a broader, global understanding of the context across multiple blocks. The sparse mode intuitively poses more challenges and demands more complex reasoning across a broader scope of the provided context. We conduct experiments on ChatGPT and Yarn-llama2-13B (Peng et al., 2023). The experimental results indicate that both models perform significantly better in dense mode compared to sparse mode, as shown in Figure 8. This indicates that LLMs struggle to retrieve information over long sequences, even though their pre-training included lengthy contexts. This may be caused by the fact that the training data does not contain sufficient examples of long-distance dependencies for the model to learn effectively. Furthermore, the steep drop in performance from 4K to 8K tokens for both ChatGPT and Yarn-Llama2 in dense mode indicates that current length extension techniques may not be as effective as hoped. In summary, we believe that S3EVAL provides a valuable framework for evaluating long-context language models, as it allows testing models on dialogues of arbitrary length. This establishes a solid foundation for advancing research on large language models that can leverage long-term con-

[1] Examples of these two patterns can be found in Appendix C.1.



(a) ChatGPT

(b) CodeLlama

Figure 7: The relationship between LLMs performance and the position of the answer token.

text.

4 Systematic Suite: Controllable Analysis

S3Eval provides a comprehensive framework that empowers developers to synthesize diverse evaluation examples for systematically assessing LLMs from multiple perspectives. In this section, inspired by the work of lost in the middle (Liu et al., 2023b), we first analyze the impact of answer position on performance (Section 4.1). Then we evaluate LLMs from different viewpoints, and we have conducted some initial explorations on the reasoning types analysis (Section 4.2). Last, we provide some insights by analyzing LLMs on three selected SQL templates (Section B.2). These experiments reveal counter-intuitive performance trends and new discoveries that may inspire further research and extension of the work.

4.1 Answer Position Analysis

We investigate the influence of the answer's position on the performance of LLMs, which is generally considered important. Unlike standard NLP benchmarks where it is difficult to control the position of the answer, S3Eval allows for fine-grained control of answer position at the token level. To mitigate the influence of long contexts, we only analyzed answers that fell within a limited context window (i.e., less than 4K tokens).

Echoing the findings of Liu et al. (2023b), "lost in the middle", our results in Figure 7 demonstrate that both ChatGPT and CodeLlama achieve higher performance when the answer is located at the beginning or end of the context, compared to when it appears in the middle. In addition, we found a periodic fluctuation trend in the performance of both models as the position of the answer shifts within the context. For example, the performance of ChatGPT increases from 0 to around 200, then starts to decrease from around 200 to 500. This wave-like pattern in performance appears to correlate with the position embedding approach used by LLMs.

In contrast to previous studies that used long-context question answering tasks (Liu et al., 2023b; Bai et al., 2023b) for analysis and are thus limited to controlling answer positions at the paragraph level, S3Eval provides a more precise approach by focusing on token level. This key difference enables S3Eval to offer fine-grained control and promote the exploration of relevant phenomena.

4.2 Reasoning Type Analysis

S3Eval enables the creation of multiple templates to generate different SQL statements, with each statement representing a distinct reasoning type. We selected six common reasoning types to investigate the reasoning capabilities of LLMs and examined four different LLMs: ChatGPT, Claude, Mistral-7B, and CodeLlama-34B. Following Liu et al. (2022), the six reasoning types [2] we considered are `Filter`, `Aggregate`, `Arithmetic`, `Superlative`, `Comparative`, and `Group`. The example SQL and the experimental results of different LLMs are presented in Table 2. The expressive power of SQL queries enables S3Eval to be used for evaluating diverse scenarios such as numerical reasoning, multi-hop reasoning, complex code understanding, and multi-turn interaction with intermediate execution results.

5 Related Work

Evaluating large language models (LLMs) has garnered significant interest in the NLP community (Chang et al., 2023). This allows us to gain

[2]Detailed templates for each type can be found in Appendix C.2.

SELECT Name FROM table WHERE Country = 'China'

Name	Age	Country
Jean	32	France
John	28	USA
Liu	24	**China**
Chen	31	**China**
Lei	34	**China**
Wang	19	**China**
Yuki	28	Japan
Kim	33	Korea
Raj	32	India
Noah	18	Canada
Luca	20	Italy

Dense

Name	Age	Country
Liu	24	**China**
Jean	32	France
John	28	USA
Chen	31	**China**
Yuki	28	Japan
Kim	33	Korea
Raj	32	India
Lei	34	**China**
Noah	18	Canada
Luca	20	Italy
Wang	19	**China**

Sparse

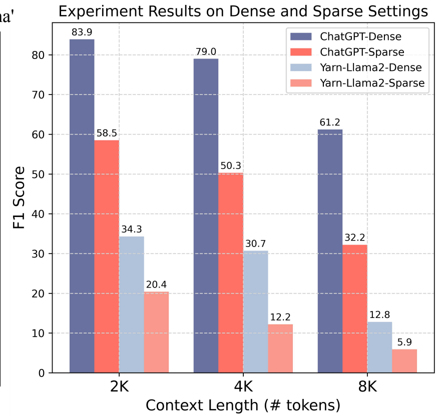


Figure 8: Experiment results of ChatGPT and Yarn-Llama 2 on *Dense* and *Sparse* Settings. *Dense* means that the answer cells (i.e., `Liu`, `Chen`, `Lei`, `Wang`) lie in adjacent rows, and *Sparse* means that the answer cells are separated. The model performs better on local queries which only involves adjacent cells.

Operator	Example SQL	ChatGPT	Claude	Mistral	CodeLlama
Filter	SELECT lyonnais FROM table WHERE farmer = 'mijl' AND lashing >288	79.6	79.2	64.8	72.8
Arithmetic	SELECT synset + refuge FROM table WHERE blender = 'owxdbzjg'	67.2	59.4	5.4	10.6
Comparative	SELECT upsetter < jollity FROM table WHERE kelp = 150	45.2	46.4	44.8	46.6
Aggregate	SELECT MIN(skeptic) FROM table	38.4	39.4	28.4	33.8
Group	SELECT lats FROM table GROUP BY shastan HAVING sum (logbook) = 56	38.1	28.2	31.0	37.8
Superlative	SELECT severity FROM table ORDER BY bierce DESC Limit 1	24.8	41.4	19.2	28.3

Table 2: Reasoning types experiments examples of different LLMs.

a deeper understanding of the specific capabilities and limitations of LLMs while guiding further research. Researchers proposed MMLU (Hendrycks et al., 2021a) to measure the knowledge acquired by a language model during pre-training. In recent years, with the development of LLMs, a series of general evaluation benchmarks have emerged. For instance, BBH (Suzgun et al., 2023) and AGIEval (Zhong et al., 2023) assess the reasoning ablitities. GSM8K (Cobbe et al., 2021) evalutes the math reasoning, HumanEval (Chen et al., 2021a) and MBPP (Austin et al., 2021) measure code capalities. Our work aims to provide an evaluation suite for measuring reasoning ability.

Many previous works on long-text modeling rely on the perplexity (Sun et al., 2021; Peng et al., 2023) or performance on simple artificial tasks (Li and Roth, 2002; Berant et al., 2013; Mohtashami and Jaggi, 2023). Concurrently, ZeroSCROLLS (Shaham et al., 2023), L-Eval (An et al., 2023) and LongBench (Bai et al., 2023b) are proposed as evaluation benchmarks for long-text modeling. However, these benchmarks are built from existing public datasets and have fixed evaluation types. In contrast, S3EVAL can effectively assess comprehension of infinitely long-context. Furthermore, S3EVAL allows customization of settings to generate evaluation data that meets specific needs, enabling effective evaluation of model deficiencies and discovery of new insights into LLMs.

6 Conclusion

In this paper, we have introduced S3EVAL, a novel synthetic evaluation suite for LLMs using SQL execution. S3EVAL represents a scalable and systematic approach to evaluate LLMs on a dynamic task. Our experiments demonstrate strong alignment between S3EVAL and traditional evaluation benchmarks. The key innovations of S3EVAL are its flexibility, allowing unlimited context length and unlimited evaluation examples, and its fine-grained, systematic nature which enables detailed analysis of model capabilities and flaws. We believe S3EVAL can serve as a valuable benchmark for LLM development and shed light on the dynamic synthetic benchmark construction.

Limitations

Our work is still in progress. Besides the features described in this paper, it currently supports complex multi-turn SQL execution task and multi-turn instruction task. Moreover, it also supports multilingual testing, especially for reasoning data generation of low-resource languages, which has not been widely studied by the academic community. However, this paper has not yet conducted a systematic analysis of these complex new features.

In addition, due to the complex and diverse syntax of SQL, the syntax that S3Eval can generate is still relatively limited, which is also what we need to do in our future work. Moreover, there is currently no toolkit that can randomly generate a large number of complex SQLs, which is also a significance of our work.

SQL operations contain many reasoning operations. Currently, we have not yet coupled these symbolic reasoning operations with real reasoning capabilities. In the future, we will test the alignment between various reasoning abilities and corresponding SQL abilities.

Due to space limitations, many valuable experimental results are shown in Appendix B. We analyzed in detail the impact of various types of influencing factors on the results and have drawn other valuable conclusions.

Exploring the treasure contained in synthetic data is our goal for the future, and we believe that this work can bring inspiration to this field.

References

Chenxin An, Shansan Gong, Ming Zhong, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *CoRR*, abs/2307.11088.

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Shivanshu Purohit, Tri Songz, Wang Phil, and Samuel Weinbach. 2021. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021. Program synthesis with large language models. *CoRR*, abs/2108.07732.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023a. Qwen technical report. *CoRR*, abs/2309.16609.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2023b. Longbench: A bilingual, multitask benchmark for long context understanding. *CoRR*, abs/2308.14508.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

Steven Bird. 2006. NLTK: the natural language toolkit. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2023. A survey on evaluation of large language models. *CoRR*, abs/2307.03109.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021a. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021b. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

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. *CoRR*, abs/2110.14168.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *CoRR*, abs/2203.15556.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy V, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Moustafa-Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023a. Starcoder: may the source be with you! *CoRR*, abs/2305.06161.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023b. Textbooks are all you need II: phi-1.5 technical report. *CoRR*, abs/2309.05463.

Nelson F. Liu, Tony Lee, Robin Jia, and Percy Liang. 2023a. Do question answering modeling improvements hold across benchmarks? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13186–13218. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023b. Lost in the middle: How language models use long contexts. *CoRR*, abs/2307.03172.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: table pre-training via learning a neural SQL executor. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. *CoRR*, abs/2305.16300.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. A conversational paradigm for program synthesis. *CoRR*, abs/2203.13474.

Erik Nijkamp, Tian Xie, Hiroaki Hayashi, Bo Pang, Congying Xia, Chen Xing, Jesse Vig, Semih Yavuz, Philippe Laban, Ben Krause, Senthil Purushwalkam, Tong Niu, Wojciech Kryscinski, Lidiya Murakhovs'ka, Prafulla Kumar Choubey, Alexander R. Fabbri, Ye Liu, Rui Meng, Lifu Tu, Meghana Bhat, Chien-Sheng Wu, Silvio Savarese, Yingbo Zhou, Shafiq Joty, and Caiming Xiong. 2023. Xgen-7b technical report. *CoRR*, abs/2309.03450.

OpenCompass. 2023. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480. The Association for Computer Linguistics.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon LLM: outperforming curated corpora with web data, and web data only. *CoRR*, abs/2306.01116.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *CoRR*, abs/2309.00071.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. Zeroscrolls: A zero-shot benchmark for long text understanding. *CoRR*, abs/2305.14196.

Tianze Shi, Chen Zhao, Jordan L. Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1849–1864. Association for Computational Linguistics.

Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. 2021. Do long-range language models actually use long-range context? In *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 807–822. Association for Computational Linguistics.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics.

InternLM Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities. `https://github.com/InternLM/InternLM`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *CoRR*, abs/2304.06364.

A Evaluation Experiments Results

A.1 Other Synthetic Task

S3EVAL is a synthetic task that possesses a certain level of difficulty and robustness, which allows for a good assessment of an LLM's overall capability compared to previous works. We choose key-value retrieval task (Liu et al., 2023b), given a key, the goal is to return the associated value. We test several LLMs on this task, and the experiments results are shown in Figure 9. It demonstrates that key-value retrieval task is a simple task which has low correlation with real LLMs reasoning benchmark. S3EVAL, as a complex and robust benchmark, can provide reference for future synthetic data.

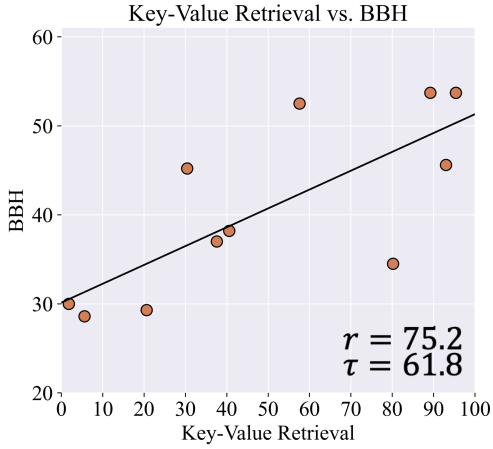


Figure 9: Performance analysis of key-value retrieval task and BBH.

A.2 Overall Performance

The detail performance are shown in Table 3.

A.3 Reliability Experiments

Symbolic Tasks vs. Natural Language Tasks. Another point to prove is that symbolic tasks are consistent with their natural language counterparts. SQL execution is a suitable task because SQL can be intertranslated with an natural question. As can be seen from the "WTQ" column of the Table 3 and Figure 10a, LLM's ability to execute SQL is consistent with its table question answering ability.

Synthetic data vs. Real data. We want to verify if the synthesized SQL is simpler. The tables "SQL-general" and "WTQ-SQL" show the difference in performance between the model on synthetic and real data. We keep the average length of the tables similar, and the experimental results show that the synthetic SQL is more complex than the real SQL.

		Synthetic Task		Realistic Benchmark	
		S3EVAL-Easy	S3EVAL-General	WTQ	Reasoning Task
LLM	GPT-4	**99.4**	**63.1**	70.8	86.7
	ChatGPT	97.0	47.2	62.0	70.1
	Claude-1	98.2	44.3	63.4	67.3
	Llama-2-70B	94.2	41.3	55.9	64.9
	Mistral-7B	87.4	34.3	55.7	53.7
	Llama2-13B	75.0	30.9	49.2	45.6
	InternLM-20B	78.0	32.3	49.4	52.5
	Qwen-14B	71.8	25.8	46.7	53.7
	Llama-2-7B	54.2	23.8	40.6	38.2
	Qwen-7B	56.4	19.4	41.2	45.2
	Xgen-7B	55.2	24.6	36.3	34.5
	Internlm-7B	41.6	18.5	27.5	37.0
	Phi-1_5	27.6	16.1	22.1	30.0
	Stablelm-7B	6.0	4.2	14.7	24.3
	Stablelm-3B	4.2	2.9	11.2	21.0
	Pythia-12B	31.4	17.3	24.5	29.3
	Pythia-6.9B	25.2	16.0	22.6	28.6
	Pythia-2.8B	26.4	14.6	21.7	28.8
	Pythia-1B	8.4	7.1	16.2	25.6
Code LLM	CodeLlama-34B	91.4	41.0	53.9	36.4
	CodeLlama-13B	90.0	35.7	49.9	30.6
	CodeLlama-7B	75.2	34.2	44.9	26.3
	StarCoder-15B	87.2	34.4	39.2	30.4
	StarCoder-7B	88.4	32.4	33.3	28.3
	StarCoder-3B	79.0	28.0	27.5	21.5
	StarCoder-1B	37.4	15.4	21.1	15.2
	CodeGen-15B	36.8	18.2	25.0	18.3
	CodeGen-6B	25.0	16.9	17.8	18.2
	CodeGen-2B	31.4	16.6	20.8	14.5

Table 3: SQL Execution Task Performance on different LLMs.

And Figure 10c shows that, the performance of LLMs on real tables and synthetic tables is very relevant.

Different S3EVAL Settings. As shown in Figure 10b, even if the data settings are very different, LLMs are guaranteed a consistent performance ranking on S3EVAL.

A.4 Other SQL Prompting Styles

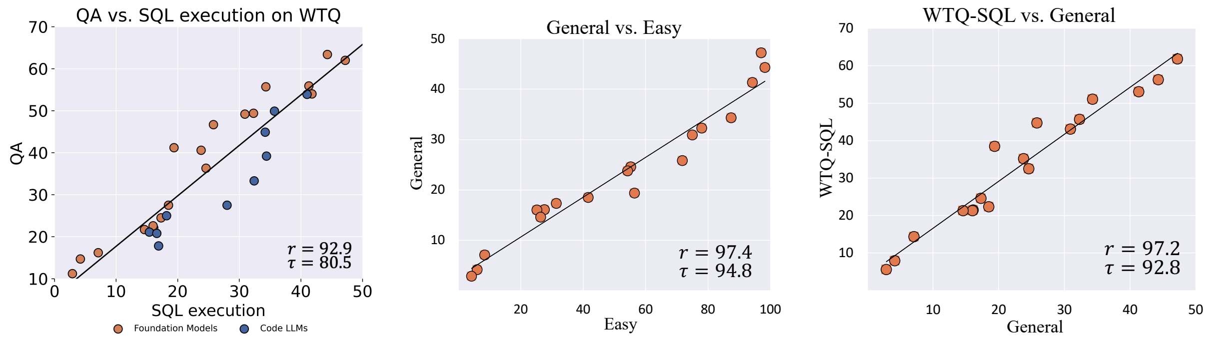
SQL execution task with Chain-of-Thought prompting. SQL is a complex multi-step reasoning task. To verify whether it is a reliable reasoning task, S3EVAL generates multi-step execution instructions for SQL. ChatGPT's performance (markdown) improves from **38.0** to **48.5** when using chain-of-thougnt prompts. The chain-of-thought examples are shown in below. The examples of chain-of-thought prompting are shown in Appendix C.6.

SQL multi-step instruction experiments. SQL multi-step instruction is an auxiliary task. We generate two new datasets using different settings than **Easy** and **General**, named **Data1** and **Data2**. Experiments results are shown in Table 5.

A.5 Long-Context Experiments

Context windows limit the long-context capabilities of LLMs. Previous researchers have proposed many ways to extend the length of context windows, often to 64K, 128K and so on. Existing benchmarks (Bai et al., 2023b; An et al., 2023) collect data from existing NLP communities (which causes data leakage), and more importantly because collecting large amounts of data is difficult. S3EVAL, on the other hand, is easy to collect data with variety and complexity. Existing benchmarks also can't effectively evaluate very long texts, but S3EVAL can evaluate arbitrary lengths.

YaRN (Peng et al., 2023) extend LLaMA2 context windows to 128K, however, they only evaluated the model's perplexity, which we believe is not a true reflection of its long-context understanding capability. So we use S3EVAL to generate table data of different lengths and keep all parameters same to evaluate the performance of yarn-LLaMA2, and the experimental results are shown in Table 4. It shows that, yarn-llama2 has a no-



(a) Alignment between QA task and SQL execution Task on WikiTableQuestions.

(b) Alignment between General and Easy Settings.

(c) Alignment between Synthetic and Real Table SQL execution task

Figure 10: Experimental results of the alignment experiments.

ticeable dip in performance on 20K-80K, which is good for a small number of tasks as well. But compared to ChatGPT (which we can only test 16K length tables), there's a noticeable gap.

B Controllable Analysis Results

B.1 Answer Position Analysis

In addition to the figures in the main text, we also conduct experiments with row level. We use two methods to visualize the results. (1) Sliding windows (Figure 11a,11b). We choose windows=5 and smooth the data to make a dot plot and a trend line. (2) Grouping calculations (Figure 11c,11d). Group neighboring rows together with the granularity of 5, 10, and 20. For example, if granularity is 20, then we group the rows with answers located in 1-20, 20-40, 40-60, 60-80, and 80-100, for a total of five groups, and calculate the average scores.

B.2 Template Controlled Analysis

Each data template in S3EVAL includes corresponding reasoning types, and thus it provides fine-grained control over the evaluation examples. To stimulate new insights and uncover counter-intuitive performance phenomena of LLMs, we present several controlled analysis examples using simple templates as a starting point.

Template1: SELECT [text_col1] FROM table WHERE ([text_col2] = [text2])

We first explore the relationship between the model performance and the locations of [text_col1] and [text_col2]. To begin with, we generated a set of 10×15 tables, each comprising 15 distinct columns. We created 400 unique combinations by pairing each value in text_col1 with each value in text_col2. For each of the 400 pairs, we generated 40 evaluation examples, resulting in a total of 16,000 evaluation examples. After SQL execution experiments, we calculated the scores of each pair and constructed a heatmap, which is illustrated in Figure 16. The heatmap indicates that the performance is overall better when [text_col1] is the previous column. And the model performance is also better when the [text_col1] column is before [text_col2] column. It indicates that the model tends to focus on the beginning of a specific paragraph. Moreover, in multi-hop reasoning, LLMs excel at hopping to the context preceding a intermediate hop, but struggles when it comes to searching backward.

Template2: SELECT [text_col1] FROM table WHERE ([text_col2] = [text2]) $\times$ N

We then investigate the impact of the number of WHERE conditions on LLM performance. Intuitively, more conditions should make it harder for LLM to execute SQL since the instruction becomes more complex. However, the experimental results contradict this intuition, as shown in blue in Figure 14. We speculate that this counter-intuitive result stems from how LLMs actually reason: by looking up string co-occurrences rather than logically considering all conditions.

Template3: SELECT COUNT([text_col]) FROM table WHERE [text_col] = [text] .

We analyze the counting ability of LLMs, which is an important numerical reasoning capability. To avoid potential symbolic effects of SQLs, we also use the instruction style (Section 2.1) to prompt the model (e.g. Please count the number of "[text_col] is [text]"). As shown in Figure 15, whether it is zero-shot or few-shot, SQL style or instruction style, the performance of LLMs is best when the COUNT value is the smallest or the largest. When

Model	Max-Ctx	SQL Execution							
		2K	4K	8K	16K	20K	40K	60K	80K
ChatGPT	16k	96.8	95.2	80.3	68.7	-	-	-	-
Claude-1.3-100K	128k	97.2	96.8	91.8	85.2	-	-	-	-
Yarn-LLaMA2-13B	128k	76.3	57.0	40.6	25.1	20.6	17.6	17.0	12.0
XGen-7B	8k	51.6	41.8	25.4	-	-	-	-	-
LongChat-13B	16k	48.6	39.0	26.3	19.5	-	-	-	-
LongLlaMA-7B	256k	82.4	62.8	24.4	-	-	-	-	-
RWKV-Raven-14B	128k	10.5	7.4	6.2	-	-	-	-	-

Table 4: Long-Context experiments on S3EVAL.

Model	**SQL Execution**				**SQL Multi-Step Instruction**			
	Zero-Shot		*Few-Shot*		*Zero-Shot*		*Few-Shot*	
	Data1	**Data2**	**Data1**	**Data2**	**Data1**	**Data2**	**Data1**	**Data2**
ChatGPT	96.4	47.0	97.0	49.0	97.9	30.0	98.8	34.8
Codellama-13B	71.2	34.3	90.0	39.8	63.9	12.1	88.0	22.8
StarCoder-15B	52.3	24.7	85.8	37.6	44.4	14.4	84.2	19.2
InternLM-20B	60.4	22.7	78.0	35.0	58.8	14.9	76.6	28.1
InternLM-20B-Chat	71.2	31.3	78.0	34.2	67.6	21.9	74.4	25.4
LLaMA2-13B	68.1	23.2	75.0	32.3	50.5	5.4	74.6	18.2
LLaMA2-13B-Chat	51.6	16.4	71.5	28.3	9.4	1.0	64.2	21.1
Vicuna-13B	57.6	26.8	81.6	35.4	48.9	11.5	78.8	24.2

Table 5: SQL Multi-Step Task performance on different LLMs.

the COUNT value is in the middle, the performance of the model is almost zero.

In the future, developers can employ the S3EVAL suite to analyze the performance of LLMs with various complex SQL queries and discover new insights. They can also investigate more on the multi-step instruction prompting (Section C.5) and chain-of-thought prompting (Section C.6) to better understand LLMs.

B.3 Input Format Analysis

In this section, we focus on comparing two formats of inputting tables, namely *markdown* and *flatten*, to explore their impact on LLMs performance. Figure 12 clearly demonstrates a significant improvement in the model's performance when the *flatten* format is used instead of the *markdown* input format at any experiments settings.

The reason behind this improvement lies in the structure of the SQL template, specifically "select <col1> where <col2> <op> <int2>". In order to execute this template, the model needs to locate the column corresponding to col2 and then identify the row where "int2" is found. This process involves 2-hop reasoning. In *markdown* mode, the challenge lies not only in the LLM's understanding of the table structure but also in how to navigate to another column in the same row. However, in *flatten* mode, redundant columns are added to each row as "Column is value." This additional information simplifies the LLM's understanding of the table structure and facilitates reasoning. As a result, the flatten method proves to be more beneficial for LLM performance due to its enhanced structure comprehension and reasoning capabilities.

B.4 SQL Keywords Analysis

SQL statements follow a specific syntax and are a well-established language in the database domain. We first control SQL statements to contain only specific types of keywords from the perspective of SQL keywords and test the performance of different models on S3EVAL. The experimental results are shown in Figure 13. The change in the performance of LLMs on SQL statements reflects the trend in the difficulty of reasoning.

B.5 SQL Attribute Analysis

S3EVAL has the ability to flexibly modify the properties of generated SQL statements, including the length of the statement, the number of computations, and the quantity of filtering numbers. These features can intuitively impact the complexity of SQL. In our experiments, we set the table size to

15×10 and adjusted the SQL settings for examining the effect of different SQL attributes on model performance. For example, in the analysis of "Calculation Times," we employed 500 samples with 0, 1, 2, and 3 calculation times respectively. The experimental outcomes of all SQL attributes are illustrated in Figure 17. While it might be expected that model performance would decline as these factor values increase, the performance actually fluctuates. Upon combining Column number, Row number, Calculation times, and Filter times in the statistical analysis, we identified a significant downward trend in the model, as demonstrated in Figure 17f.

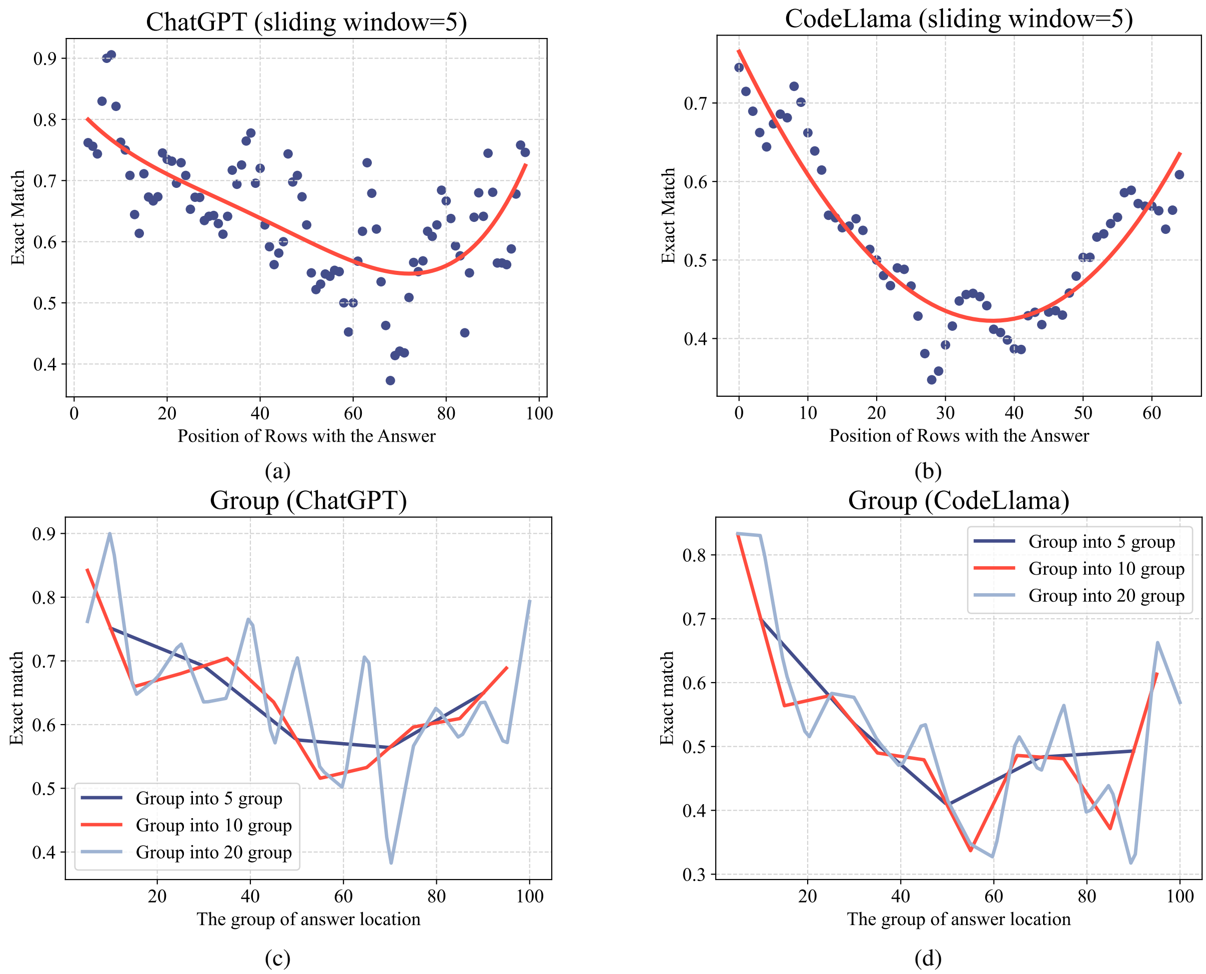


Figure 11: Effect of answer position on model performance. We use two methods to visualize the results. (1) Sliding windows (Figure 11a,11b). We select a window size of 5 and smooth the data to make a dot plot and a trend line. (2) Grouping calculations (Figure 11c,11d). We group neighboring rows with granularities of 5, 10, and 20. For instance, with a granularity of 20, we group rows with answers located in the ranges 1-20, 21-40, 41-60, 61-80, and 81-100, resulting in five groups, and compute the average scores.

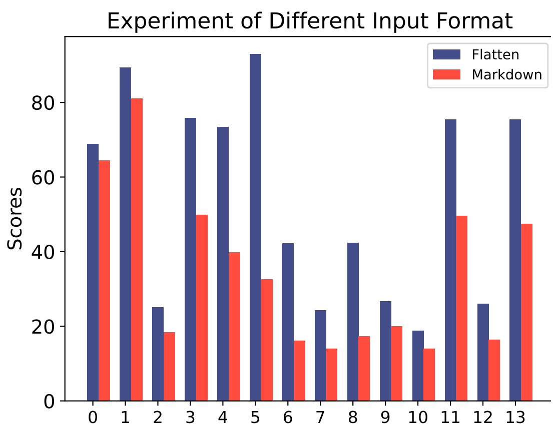


Figure 12: Different input format.

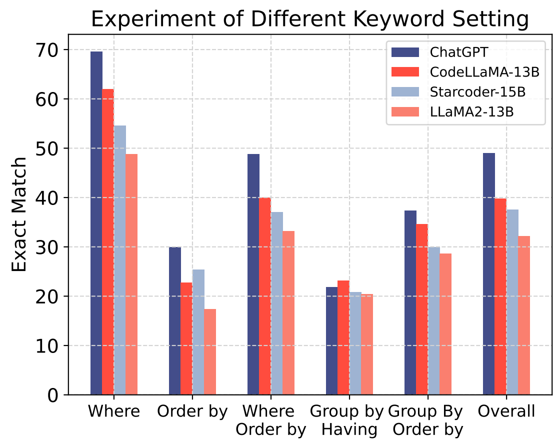


Figure 13: Different keywords setting.

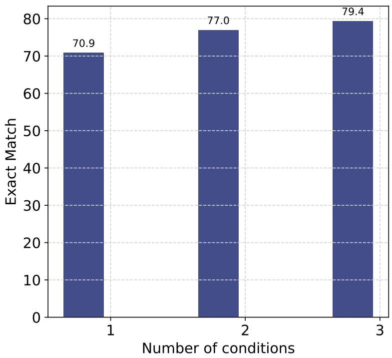


Figure 14: Trend of ChatGPT performance with where condition number using Template2.

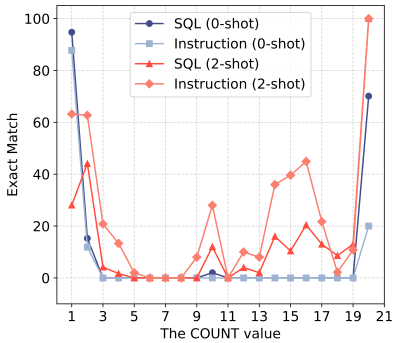


Figure 15: Trend of ChatGPT performance with the COUNT value in Template3. Only when the COUNT value is the largest or smallest, the model have good performance.

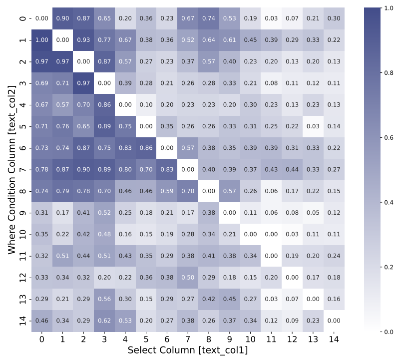
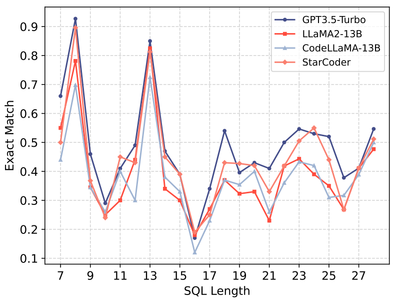
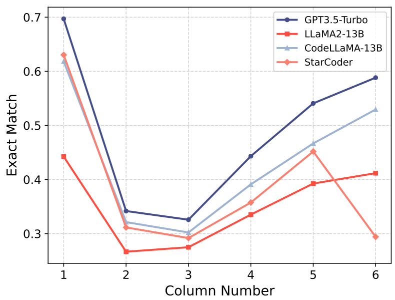


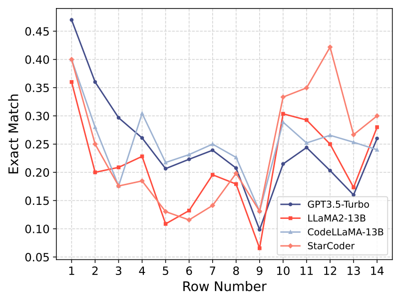
Figure 16: ChatGPT performance with different locations of [text_col1] and [text_col2]. The performance improves when the example has the location of [text_col1] before [text_col2].



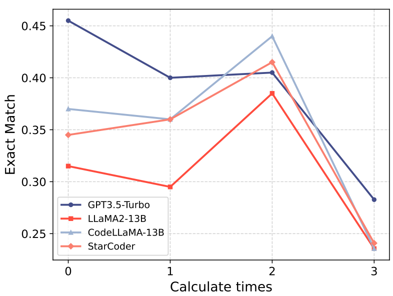
(a) SQL length



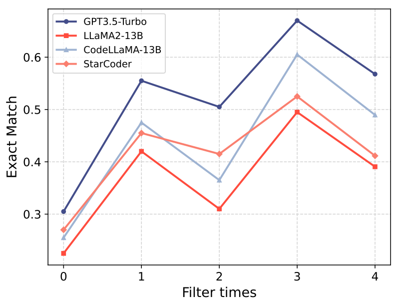
(b) Column number.



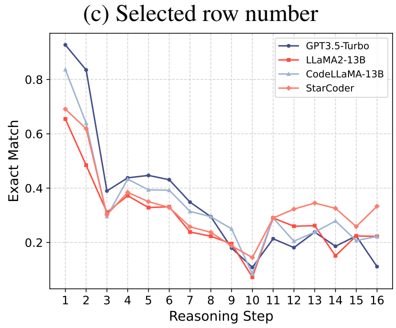
(c) Selected row number



(d) Calculation times



(e) Filter times.



(f) Total analysis.

Figure 17: Effect of SQL Attribute Settings on model performance.

C Data Demostration

C.1 Dense and Sparse Examples

SQL: select boarfish from w where sixties = 'jcrbb'

Answer: ['qxgd', 'lorfaljob', 'qytocp', 'vkfzhqwj', 'xwijyubr']

We can find that *Dense* Setting is better than *Sparse* Setting in all cases.

Sparse Example:

	boarfish	tool	sixties	phoxinus	angling
0	mjdsv	cwzqkdte	tbwqa	yuogpbo	mkxqnrhq
1	nrbmyc	eqciiims	wvfesrtzt	yvvgzj	mkxqnrhq
2	iqdr	ezhuj	bndktpe	yuogpbo	yjblg
3	qxgd	dtfjqfc	jcrbb	haxyaz	yjblg
4	xzrrs	ezhuj	bndktpe	dpimlb	skbpzyhak
5	lorfaljob	eqciiims	jcrbb	jsvbugac	bwxihx
6	pvugxgdju	dtfjqfc	bndktpe	jsvbugac	mkxqnrhq
7	xpkuautv	ezhuj	vyoo	yvvgzj	bwxihx
8	afzrom	jzdra	bndktpe	jsvbugac	mkxqnrhq
9	ivxpmv	eqciiims	bndktpe	jsvbugac	bwxihx
10	ehfvur	ezhuj	tbwqa	yuogpbo	bwxihx
11	bdzsy	ezhuj	bndktpe	yvvgzj	yjblg
12	qruh	ezhuj	bndktpe	dpimlb	skbpzyhak
13	qytocp	jzdra	jcrbb	dpimlb	bwxihx
14	eqaja	ezhuj	bndktpe	haxyaz	yjblg
15	kwvzixe	jzdra	vyoo	jsvbugac	skbpzyhak
16	edmkxm	eqciiims	vyoo	haxyaz	mkxqnrhq
17	fdsdlcpxj	eqciiims	vyoo	dpimlb	blqoislm
18	ipprxzzlv	cwzqkdte	bndktpe	yuogpbo	yjblg
19	gqyxjtbz	eqciiims	tbwqa	dpimlb	yjblg
20	noqfw	ezhuj	vyoo	haxyaz	blqoislm
21	vkfzhqwj	dtfjqfc	jcrbb	yuogpbo	mkxqnrhq
22	konftq	eqciiims	vyoo	dpimlb	bwxihx
23	ymcwhu	jzdra	wvfesrtzt	dpimlb	blqoislm
24	kpygsu	eqciiims	wvfesrtzt	yuogpbo	yjblg
25	tiwfvqgmt	ezhuj	bndktpe	dpimlb	mkxqnrhq
26	ovomhf	dtfjqfc	bndktpe	yuogpbo	blqoislm
27	lokwxn	cwzqkdte	tbwqa	yuogpbo	mkxqnrhq
28	xwijyubr	jzdra	jcrbb	yuogpbo	mkxqnrhq
29	ttonww	dtfjqfc	wvfesrtzt	haxyaz	blqoislm

Dense Example:

	boarfish	tool	sixties	phoxinus	angling
0	mjdsv	cwzqkdte	tbwqa	yuogpbo	mkxqnrhq
1	nrbmyc	eqciiims	wvfesrtzt	yvvgzj	mkxqnrhq
2	iqdr	ezhuj	bndktpe	yuogpbo	yjblg
3	xzrrs	ezhuj	bndktpe	dpimlb	skbpzyhak
4	pvugxgdju	dtfjqfc	bndktpe	jsvbugac	mkxqnrhq
5	xpkuautv	ezhuj	vyoo	yvvgzj	bwxihx
6	afzrom	jzdra	bndktpe	jsvbugac	mkxqnrhq
7	ivxpmv	eqciiims	bndktpe	jsvbugac	bwxihx
8	ehfvur	ezhuj	tbwqa	yuogpbo	bwxihx
9	bdzsy	ezhuj	bndktpe	yvvgzj	yjblg
10	qruh	ezhuj	bndktpe	dpimlb	skbpzyhak
11	eqaja	ezhuj	bndktpe	haxyaz	yjblg
12	kwvzixe	jzdra	vyoo	jsvbugac	skbpzyhak
13	qxgd	dtfjqfc	jcrbb	haxyaz	yjblg
14	lorfaljob	eqciiims	jcrbb	jsvbugac	bwxihx
15	qytocp	jzdra	jcrbb	dpimlb	bwxihx
16	vkfzhqwj	dtfjqfc	jcrbb	yuogpbo	mkxqnrhq
17	xwijyubr	jzdra	jcrbb	yuogpbo	mkxqnrhq
18	edmkxm	eqciiims	vyoo	haxyaz	mkxqnrhq
19	fdsdlcpxj	eqciiims	vyoo	dpimlb	blqoislm
20	ipprxzzlv	cwzqkdte	bndktpe	yuogpbo	yjblg
21	gqyxjtbz	eqciiims	tbwqa	dpimlb	yjblg
22	noqfw	ezhuj	vyoo	haxyaz	blqoislm

```
| 23 | konftq     | eqciiims | vyoo      | dpimlb     | bwxihx    |
| 24 | ymcwhu     | jzdra    | wvfesrtzt | dpimlb     | blqoislm  |
| 25 | kpygsu     | eqciiims | wvfesrtzt | yuogpbo    | yjblg     |
| 26 | tiwfvqgmt  | ezhuj    | bndktpe   | dpimlb     | mkxqnrhq  |
| 27 | ovomhf     | dtfjqfc  | bndktpe   | yuogpbo    | blqoislm  |
| 28 | lokwxn     | cwzqkdte | tbwqa     | yuogpbo    | mkxqnrhq  |
| 29 | ttonww     | dtfjqfc  | wvfesrtzt | haxyaz     | blqoislm  |
```

C.2 SQL Template

General:

```
select <select_condition> from my_table

select <select_condition> from my_table <where_condition>

select <select_condition> from my_table <order_condition>,

select <select_condition> from my_table <where_condition> <order_condition>,

select <select_condition> from my_table <group_condition> <having_condition>,

select <select_condition> from my_table <where_condition> <group_condition> <having_condition>,

select <select_condition> from my_table <where_condition>
<group_condition> <having_condition> <order_condition>,

select <select_condition> from my_table <group_condition> <having_condition> <order_condition>
```

Where Condition:

```
select <text_col1> from my_table where <text_col2> = <text_2>
```

Count:

```
Select Count(<text_col1>) from table where <text_col1> = <text_1>
```

Easy:

```
select <text_col1> from my_table where <int_col1> = <int_1>
select <int_col1> from my_table where <text_col1> = <text_1>
select <int_col1> from my_table where <int_col2> = <int_2>
select <text_col1> from my_table where <text_col2> = <text_2>
```

Filter:

```
select <text_col1> from my_table where <text_col2> = <text_2>
select <text_col1> from my_table where <int_col2> <op2> <int_2>
select <text_col1> from my_table where <text_col2> = <text_2> and <int_col1> <op1> <int_1>
select <text_col1> from my_table where <text_col2> = <text_2> and <text_col3> = <text_3>
select <text_col1> from my_table where <int_col1> <op1> <int_1> and <int_col2> <op2> <int_2>
select <int_col1> from my_table where <text_col1> = <text_1>
select <int_col1> from my_table where <int_col2> <op2> <int_2>
select <int_col1> from my_table where <text_col2> = <text_2> and <int_col2> <op2> <int_2>
select <int_col1> from my_table where <text_col2> = <text_2> and <text_col3> = <text_3>
select <int_col1> from my_table where <int_col2> <op2> <int_2> and <int_col3> <op3> <int_3>
```

Aggregate:

```
select count ( <text_col1> ) from my_table where <text_col2> = <text_2>
select count ( <text_col1> ) from my_table where <int_col2> <op2> <int_2>
select sum ( <int_col1> ) from my_table
select sum ( <int_col1> ) from my_table where <text_col2> = <text_2>
select max ( <int_col1> ) from my_table
select max ( <int_col1> ) from my_table where <text_col2> = <text_2>
select min ( <int_col1> ) from my_table
select min ( <int_col1> ) from my_table where <text_col2> = <text_2>
```

Arithmetic:

```
select <int_col1> + <int_col2> from my_table where <text_col1> = <text_1>
select <int_col1> + <int_col2> from my_table where <text_col1> = <text_1> and <text_col2> = <text_2>
select <int_col1> - <int_col2> from my_table where <text_col1> = <text_1>
select <int_col1> - <int_col2> from my_table where <text_col1> = <text_1> and <text_col2> = <text_2>
```

Superlative:

```
select <int_col1> from my_table order by <int_col1> asc limit 1
select <int_col1> from my_table order by <int_col1> desc limit 1
select <text_col1> from my_table order by <int_col1> asc limit 1
select <text_col1> from my_table order by <int_col1> desc limit 1
select <int_col1> from my_table order by <int_col2> asc limit 1
select <int_col1> from my_table order by <int_col2> desc limit 1
```

Comparative:

```
select ( select <int_col1> from my_table where <text_col1> = <text_1> )
> ( select <int_col1> from my_table where <text_col2> = <text_2> )
select ( select <int_col1> from my_table where <int_col2> <op2> <int_2> )
> ( select <int_col1> from my_table where <int_col3> <op3> <int_3> )
select ( select <int_col1> from my_table where <text_col1> = <text_1> )
< ( select <int_col1> from my_table where <text_col2> = <text_2> )
select ( select <int_col1> from my_table where <int_col2> <op2> <int_2> )
< ( select <int_col1> from my_table where <int_col3> <op3> <int_3> )
select <int_col1> > <int_col2> from my_table where <text_col1> = <text_1>
select <int_col1> < <int_col2> from my_table where <text_col1> = <text_1>
select <int_col1> > <int_col2> from my_table where <int_col3> <op3> <int_3>
select <int_col1> < <int_col2> from my_table where <int_col3> <op3> <int_3>
```

C.3 Table Input Format

Markdown Table:

```
|    |   ercilla |   shucks | liter    | taenia    | dorado    |
|---:|----------:|---------:|:---------|:----------|:----------|
|  0 |        68 |       12 | gcrdvo   | qoath     | katfuw    |
|  1 |       129 |      151 | zmvltkk  | jpcglcjzk | vwqqey    |
|  2 |       248 |      188 | zmdlfbhb | cvhqotsys | wzunmaa   |
|  3 |       267 |      104 | gcrdvo   | ytywunvf  | pjlbo     |
|  4 |       135 |      262 | gcrdvo   | dtnvfp    | ajzpsaoy  |
|  5 |       309 |      119 | zmdlfbhb | klcenmugk | hriunhf   |
|  6 |        25 |      152 | zmvltkk  | cjgcergv  | shrbvrd   |
|  7 |       298 |       18 | zmvltkk  | scvuuc    | ahunvcx   |
|  8 |       321 |      217 | gcrdvo   | ezlp      | hasjaznm  |
|  9 |       139 |      310 | gcrdvo   | ghhjea    | atqvtgoa  |
| 10 |        99 |       34 | zmvltkk  | ecdmpruq  | cfitvz    |
| 11 |       142 |      167 | gcrdvo   | acii      | oenmuezip |
| 12 |       273 |      156 | gcrdvo   | nnvnteh   | tulh      |
| 13 |       197 |       44 | gcrdvo   | pqdbhevkh | dfxuwxz   |
| 14 |       144 |      123 | gcrdvo   | bxrgo     | ccbj      |
```

Flatten Table:

```
Flatten Table Examples:
The table have 5 columns: ercilla | shucks | liter | taenia | dorado
row 1 : ercilla is 68. shucks is 12. liter is gcrdvo. taenia is qoath. dorado is katfuw.
row 2 : ercilla is 129. shucks is 151. liter is zmvltkk. taenia is jpcglcjzk. dorado is vwqqey.
row 3 : ercilla is 248. shucks is 188. liter is zmdlfbhb. taenia is cvhqotsys. dorado is wzunmaa.
row 4 : ercilla is 267. shucks is 104. liter is gcrdvo. taenia is ytywunvf. dorado is pjlbo.
row 5 : ercilla is 135. shucks is 262. liter is gcrdvo. taenia is dtnvfp. dorado is ajzpsaoy.
row 6 : ercilla is 309. shucks is 119. liter is zmdlfbhb. taenia is klcenmugk. dorado is hriunhf.
row 7 : ercilla is 25. shucks is 152. liter is zmvltkk. taenia is cjgcergv. dorado is shrbvrd.
row 8 : ercilla is 298. shucks is 18. liter is zmvltkk. taenia is scvuuc. dorado is ahunvcx.
row 9 : ercilla is 321. shucks is 217. liter is gcrdvo. taenia is ezlp. dorado is hasjaznm.
row 10 : ercilla is 139. shucks is 310. liter is gcrdvo. taenia is ghhjea. dorado is atqvtgoa.
row 11 : ercilla is 99. shucks is 34. liter is zmvltkk. taenia is ecdmpruq. dorado is cfitvz.
row 12 : ercilla is 142. shucks is 167. liter is gcrdvo. taenia is acii. dorado is oenmuezip.
row 13 : ercilla is 273. shucks is 156. liter is gcrdvo. taenia is nnvnteh. dorado is tulh.
row 14 : ercilla is 197. shucks is 44. liter is gcrdvo. taenia is pqdbhevkh. dorado is dfxuwxz.
row 15 : ercilla is 144. shucks is 123. liter is gcrdvo. taenia is bxrgo. dorado is ccbj.
```

C.4 SQL Execution Examples (Few-shot)

```
You are an SQL executor, you need to execute SQL based on the give table and SQL statement
to obtain the execution results.
Only give me the execution results and do not output any other words.
Table:
```

```
|    |   puccoon |   tiepolo |   scope | mutinus    |   intrados | huggins   |   barye |   wear |
|---:|----------:|----------:|--------:|:-----------|-----------:|:----------|--------:|-------:|
|  0 |       171 |       225 |     145 | 2007-04-27 |        322 | yefihroyn |      79 |    207 |
|  1 |       213 |       116 |     319 | 2016-01-15 |        288 | ytyayrvj  |     246 |    272 |
|  2 |       191 |       229 |      95 | 2022-11-08 |        218 | gpmvax    |     167 |     73 |
|  3 |        97 |       155 |     189 | 2013-10-30 |         79 | gpmvax    |      24 |    233 |
|  4 |        56 |        11 |     295 | 2018-12-10 |         81 | yefihroyn |     187 |    198 |
|  5 |       285 |       304 |     168 | 2017-03-24 |         75 | gpmvax    |     111 |     77 |
|  6 |       233 |       325 |      31 | 2014-01-22 |        114 | ytyayrvj  |      20 |    219 |
|  7 |        19 |       146 |     164 | 2021-12-07 |        311 | ytyayrvj  |     188 |      3 |
|  8 |       112 |       255 |      30 | 2015-12-07 |        214 | gpmvax    |      16 |    271 |
|  9 |       175 |        62 |     181 | 2012-04-21 |        182 | gpmvax    |     105 |     76 |
| 10 |       200 |        90 |     101 | 2008-04-28 |        168 | gpmvax    |      70 |    119 |
| 11 |        31 |       180 |      95 | 2004-06-23 |         62 | yefihroyn |     314 |     97 |
| 12 |       297 |       251 |     249 | 2022-02-02 |        185 | yefihroyn |     278 |    313 |
| 13 |        36 |        17 |      67 | 2016-04-14 |        243 | ytyayrvj  |     213 |      4 |
| 14 |        45 |       215 |     182 | 2012-06-15 |        251 | yefihroyn |     221 |     83 |
Now you need to execute SQL based on the given table and SQL statement to obtain the execution result.
Only give me the result and do not output any other words or SQL statement.
The following are some examples.

SQL:select avg ( intrados ) from my_table where tiepolo > 146 group by huggins
having count ( huggins ) > 1 order by count ( tiepolo ) asc limit 1
Answer:146.5
SQL:select wear from my_table where huggins = 'gpmvax' group by huggins
having wear < 83 order by count ( distinct barye ) asc limit 1
Answer:73
SQL:select mutinus from my_table where tiepolo > 116 group by huggins
having max ( wear ) > 119 order by count ( huggins ) asc limit 1
Answer:2014-01-22
SQL:select tiepolo from my_table where puccoon < 191 and intrados < 79 group by huggins
having intrados < 81 and tiepolo < 255 order by count ( barye ) asc limit 1
Answer:180
SQL:select tiepolo from my_table where scope > 31 group by huggins
having min ( tiepolo ) = 62 order by count ( distinct mutinus ) asc limit 1
Answer:62
SQL:select wear from my_table where huggins = 'ytyayrvj' group by huggins
having count ( huggins ) < 5 order by count ( distinct mutinus ) desc limit 1
Answer:
```

C.5 Multi-step Instruction (Few-shot)

```
You need to obtain the final answer based on the table and instructions.
Only give me the result and do not output any other words.
Table:
|    |   puccoon |   tiepolo |   scope | mutinus    |   intrados | huggins   |   barye |   wear |
|---:|----------:|----------:|--------:|:-----------|-----------:|:----------|--------:|-------:|
|  0 |       171 |       225 |     145 | 2007-04-27 |        322 | yefihroyn |      79 |    207 |
|  1 |       213 |       116 |     319 | 2016-01-15 |        288 | ytyayrvj  |     246 |    272 |
|  2 |       191 |       229 |      95 | 2022-11-08 |        218 | gpmvax    |     167 |     73 |
|  3 |        97 |       155 |     189 | 2013-10-30 |         79 | gpmvax    |      24 |    233 |
|  4 |        56 |        11 |     295 | 2018-12-10 |         81 | yefihroyn |     187 |    198 |
|  5 |       285 |       304 |     168 | 2017-03-24 |         75 | gpmvax    |     111 |     77 |
|  6 |       233 |       325 |      31 | 2014-01-22 |        114 | ytyayrvj  |      20 |    219 |
|  7 |        19 |       146 |     164 | 2021-12-07 |        311 | ytyayrvj  |     188 |      3 |
|  8 |       112 |       255 |      30 | 2015-12-07 |        214 | gpmvax    |      16 |    271 |
|  9 |       175 |        62 |     181 | 2012-04-21 |        182 | gpmvax    |     105 |     76 |
| 10 |       200 |        90 |     101 | 2008-04-28 |        168 | gpmvax    |      70 |    119 |
| 11 |        31 |       180 |      95 | 2004-06-23 |         62 | yefihroyn |     314 |     97 |
| 12 |       297 |       251 |     249 | 2022-02-02 |        185 | yefihroyn |     278 |    313 |
| 13 |        36 |        17 |      67 | 2016-04-14 |        243 | ytyayrvj  |     213 |      4 |
| 14 |        45 |       215 |     182 | 2012-06-15 |        251 | yefihroyn |     221 |     83 |
Now you need to get the answer based on the instruction,
only give me the result and do not output any other words.
The following are some examples.

Instruction:Please filter the rows by the column conditions, which need to be met:
The value of column tiepolo needs to be greater than 146.
The rows are then grouped according to the value of the huggins in the remaining rows.
```

```
Then filter some groups by the following condition:the number of column huggins is greater than 1.
Select the average of values of intrados column in filtered rows.
Sort the obtained values in ascending order of the number of tiepolo
and select the smallest value to get the answer.
Answer:146.5

Instruction:Please filter the rows by the column conditions, which need to be met:
The value of column huggins is 'gpmvax'.
The rows are then grouped according to the value of the huggins in the remaining rows.
Then filter some groups by the following condition:the column wear is less than 83.
Select values of wear column in filtered rows.
Sort the obtained values in ascending order of the number of non-repeating barye
and select the smallest value to get the answer.
Answer:73

Instruction:Please filter the rows by the column conditions, which need to be met:
The value of column huggins is 'ytyayrvj'.
The rows are then grouped according to the value of the huggins in the remaining rows.
Then filter some groups by the following condition:the number of column huggins is less than 5.
Select values of wear column in filtered rows.
Sort the obtained values in descending order of the number of non-repeating mutinus
and select the largest value to get the answer.
Answer:
```

C.6 Chain-of-Thought SQL Execution Prompting Examples

```
You are an SQL executor, you need to output the execution process and final answer based on table and SQL.
Table:
|    | masthead   | laertes   |   boo | bothrops   | height   | scraper   | trouser   |   lozenge |
|---:|:-----------|:----------|------:|:-----------|:---------|:----------|:----------|----------:|
|  0 | case       | araeswrid |    41 | lyucg      | urbsmxiv | vgxrh     | esauw     |       281 |
|  1 | case       | araeswrid |   138 | lyucg      | tbvg     | oerigocb  | stevw     |       177 |
|  2 | case       | zncmrrvg  |   303 | loclzoglg  | tbvg     | vgxrh     | stevw     |       234 |
|  3 | thyngfwts  | araeswrid |   288 | loclzoglg  | tbvg     | vgxrh     | esauw     |       224 |
|  4 | thyngfwts  | mrehctv   |   177 | loclzoglg  | urbsmxiv | vgxrh     | esauw     |       228 |
|  5 | case       | araeswrid |   163 | loclzoglg  | urbsmxiv | oerigocb  | stevw     |        60 |
|  6 | thyngfwts  | mrehctv   |    45 | loclzoglg  | cidufm   | oerigocb  | esauw     |       289 |
|  7 | thyngfwts  | zncmrrvg  |    42 | loclzoglg  | tbvg     | ffljyxb   | stevw     |       296 |
|  8 | case       | araeswrid |   275 | lyucg      | cidufm   | vgxrh     | stevw     |       172 |
|  9 | case       | mrehctv   |    20 | loclzoglg  | tbvg     | vgxrh     | esauw     |       147 |
| 10 | thyngfwts  | araeswrid |   302 | lyucg      | urbsmxiv | vgxrh     | stevw     |       297 |
| 11 | thyngfwts  | zncmrrvg  |   137 | loclzoglg  | tbvg     | vgxrh     | esauw     |        63 |
| 12 | case       | araeswrid |   186 | loclzoglg  | cidufm   | ffljyxb   | esauw     |       268 |
| 13 | case       | araeswrid |   194 | loclzoglg  | cidufm   | vgxrh     | esauw     |        98 |
| 14 | case       | araeswrid |   234 | lyucg      | urbsmxiv | vgxrh     | stevw     |       276 |
Now you need to get the answer based on the instruction,
only give me the intermedium results and the final answer.
SQL:
select masthead from my_table where height = 'tbvg' group by masthead order by count ( laertes ) desc limit 1
Execution process:
You need to execute 3 steps.
Step 0:
Please filter the rows by the column conditions, which need to be met: The value of column butcher is 'jxys'.
Intermediate results 0:
|    | encyclia   | butcher   | bowdler   | nuthatch   |   cachexia | claret   |   cortina |   strombus |
|---:|:-----------|:----------|:----------|:-----------|-----------:|:---------|----------:|-----------:|
|  0 | adnh       | jxys      | cxjvfz    | clmb       |          2 | oqmdmbfg |       251 |        184 |
|  1 | xvoxfjbm   | jxys      | cxjvfz    | clmb       |        275 | oqmdmbfg |       140 |        303 |
|  2 | adnh       | jxys      | eohdpivo  | clmb       |        298 | oqmdmbfg |       142 |         28 |
|  3 | adnh       | jxys      | eohdpivo  | rcyixdl    |        153 | oqmdmbfg |        50 |        306 |
|  4 | xvoxfjbm   | jxys      | eohdpivo  | rcyixdl    |        315 | rxbttbm  |       201 |         86 |
Step 1: Select values of strombus column in filtered rows.
Intermediate results 1:
184,303,28,306,86
Step 2: Sort the obtained values in ascending order of claret and select the smallest value to get the answer.
Answer: 184
```

C.7 Real Table SQL Execution (Few-shot)

```
You are an SQL executor, you need to execute SQL based on the give table
and SQL statement to obtain the execution results.
```

```
Only give me the execution results and do not output any other words.
Table:
|    |   id |   agg |   rank | nation              |   gold |   silver |   bronze |   total |
|---:|-----:|------:|-------:|:--------------------|-------:|---------:|---------:|--------:|
|  0 |    1 |     0 |      1 | soviet union        |     50 |       27 |       22 |      99 |
|  1 |    2 |     0 |      2 | united states       |     33 |       31 |       30 |      94 |
|  2 |    3 |     0 |      3 | east germany (gdr)  |     20 |       23 |       23 |      66 |
|  3 |    4 |     0 |      4 | west germany (frg)  |     13 |       11 |       16 |      40 |
|  4 |    5 |     0 |      5 | japan               |     13 |        8 |        8 |      29 |
|  5 |    6 |     0 |      6 | australia           |      8 |        7 |        2 |      17 |
|  6 |    7 |     0 |      7 | poland              |      7 |        5 |        9 |      21 |
|  7 |    8 |     0 |      8 | hungary             |      6 |       13 |       16 |      35 |
|  8 |    9 |     0 |      9 | bulgaria            |      6 |       10 |        5 |      21 |
|  9 |   10 |     0 |     10 | italy               |      5 |        3 |       10 |      18 |
Now you need to execute SQL based on the given table and SQL statement to obtain the execution result.
Only give me the result and do not output any other words or SQL statement.
The following are some examples.

SQL:select nation from table where rank = 1
Answer:Soviet Union
SQL:select nation from table where nation != 'bulgaria'
and total = ( select total from table where nation = 'bulgaria' )
Answer:Poland
SQL:select nation from table order by bronze limit 1
Answer:Australia
SQL:select nation from table order by bronze limit 1
Answer:Australia
SQL:select silver from table order by gold desc limit 1
Answer:
```

C.8 Real Table Question Answering (Few-shot)

```
You need to obtain the final answer based on the table and questions.
Only give me the answer and do not output any other words.
Table:
|    |   id |   agg |   rank | nation              |   gold |   silver |   bronze |   total |
|---:|-----:|------:|-------:|:--------------------|-------:|---------:|---------:|--------:|
|  0 |    1 |     0 |      1 | soviet union        |     50 |       27 |       22 |      99 |
|  1 |    2 |     0 |      2 | united states       |     33 |       31 |       30 |      94 |
|  2 |    3 |     0 |      3 | east germany (gdr)  |     20 |       23 |       23 |      66 |
|  3 |    4 |     0 |      4 | west germany (frg)  |     13 |       11 |       16 |      40 |
|  4 |    5 |     0 |      5 | japan               |     13 |        8 |        8 |      29 |
|  5 |    6 |     0 |      6 | australia           |      8 |        7 |        2 |      17 |
|  6 |    7 |     0 |      7 | poland              |      7 |        5 |        9 |      21 |
|  7 |    8 |     0 |      8 | hungary             |      6 |       13 |       16 |      35 |
|  8 |    9 |     0 |      9 | bulgaria            |      6 |       10 |        5 |      21 |
|  9 |   10 |     0 |     10 | italy               |      5 |        3 |       10 |      18 |
Now you need to get the answer based on the question,
only give me the answer and do not output any other words.
The following are some examples.

Question:which country was first in rank at the 1972 olympics ?
Answer:Soviet Union
Question:which country won the same amount of medals as bulgaria in these olympics ?
Answer:Poland
Question:which nation won the least number of bronze medals ?
Answer:Australia
Question:which nation received the least bronze medals
Answer:Australia
Question:what number of silver medals was won by the nation with the most gold medals ?
Answer:
```

D Experiments Settings Details

D.1 Setting Description

Table Config

```
"col_min": 5, // the min number of cols
"col_max": 8, // the max number of cols
"row_min": 15,  // the min number of rows
```

```
”row_max”: 40,  // the max number of rows
”text_int_date”: [0.55, 0.35, 0.1], // text,int,date  header ratio
”text_int_date_fix”: [”TEXT”, ”TEXT”, ”INT”, ”INT”, ”INT”], // Specify the type of each header
// Probability of duplicate values in each column
”value_repeat_ratio”: [0, 0.2, 0.3, 0, 0, 0, 0, 0, 0.2, 0.5],
”value_repeat_ratio_fix”: [”random”, ”random”], // Specify the duplicate values of each column
```

SQL Config

```
”nest”: [1],  // Number of SQL nestings. options: [1], [2], [1,2],[1,2, 3]
”keywords_setting”: { // if a Keyword is False, then no SQL containing this Keyword is generated.
”select”: true,
”where”: true,
”group by”: true,
”having”: true,
”order by”: true
},
”length_setting”: {      // control the length of sql
”is_available”: false, // To enable this setting, you need to adjust ”is_available” to true first.
// 'value' can be set to specific values, such as [13,14,15],
// if value is null, then the range is used [min, max]
”value”: [],
”min”: 6,
”max”: 16
},
”column_ratio”: {        // Controlling the ratio of columns involved in SQL
”is_available”: false,  // To enable this setting, you need to adjust ”is_available” to true first.
// 'value' can be set to specific values, such as [1,2], Control the number of columns involved in SQL
”value”: [],
// if value is null, then the range is used [min, max], it's the used ratio = (used columns) / (all columns)
”min”: 0.1,
”max”: 0.3
},
”select_row_ratio”:{     // Controlling the ratio of rows involved in select keyword
”is_available”: false,  // To enable this setting, you need to adjust ”is_available” to true first.
// 'value' can be set to specific values, such as [1,2,3,4], Control the number of rows involved in SQL
”value”: [],
// if value is null, then the range is used [min, max], it's the used ratio = (select rows) / (all rows)
”min”: 0.1,
”max”: 0.2
},
// Controlling the calculate times of the sql ['+','-','*','/','sum','count','min','max','avg']
”calculate_times”: {
”is_available”: false,   // To enable this setting, you need to adjust ”is_available” to true first.
”value”: [1,2,3,4]      // 'value' can be set to specific values, means the calculate times
},
// Controlling the filter times of the sql ['=','>','<','in','like']
”filter_times”: {
”is_available”: false,   // To enable this setting, you need to adjust ”is_available” to true first.
”value”: [1,2,3,4,5]  // 'value' can be set to specific values, means the calculate times
},
// Controlling the location of answer in the table, usually used in long-context understanding
”answer_location”: {
”is_available”: false,   // To enable this setting, you need to adjust ”is_available” to true first.
”value”: null,
”min”: 0.1, // if value is null, then the range is used [min, max],
means that  0.1 <  (Row where answer is located ) / (Row number) < 0.9
”max”: 0.9
},
// usually remains 1 in this repo, we often just test the sql whose answer is from one cell.
”answer_cells_number”: 1,
”include”: [],
”exclude”: [],
”n_shot”: 5
```

D.2 General Setting

Table Config

```
”col_min”: 5,
”col_max”: 5,
”row_min”: 30,
```

```
"row_max": 30,
"text_int_date": [0.5, 0.45, 0.05],
"value_repeat_ratio": [0, 0.2, 0.3, 0, 0, 0, 0, 0, 0, 0.5]
```

SQL Config

```
  "nest": [1,2,3],
  "select_grammar": [],
  "keywords_setting": { "select": true,
  "where": true,
    "group by": true,
    "having": true,
    "order by": true
  },
  "length_setting": {
    "is_available": false,
    "value": [],
    "min": 6,
    "max": 16
  },
  "column_ratio": {
    "is_available": false,
    "value": [],
    "min": 0.1,
    "max": 0.3
  },
  "select_row_ratio":{
    "is_available": false,
    "value": [],
    "min": 0,
    "max": 0.2
  },
  "calculate_times": {
    "is_available": false,
    "value": [0]
  },
  "filter_times": {
    "is_available": false,
    "value": [0]
  },
  "answer_location": {
    "is_available": false,
    "row_value": [],
    "column_value":[0],
    "min": 0,
    "max": 1
  },
  "answer_cells_number": 1,
  "multi_test": false,
  "include": [],
  "exclude": [],
  "n_shot": 5
```

D.3 LLMs Used In This Paper

LLMs. LLaMA2 (Touvron et al., 2023a), Qwen (Bai et al., 2023a), InternLM (Team, 2023), Mistral, XGen (Nijkamp et al., 2023), Falcon (Penedo et al., 2023), phi-1_5 (Li et al., 2023b), StableLM (Andonian et al., 2021), Pythia (Biderman et al., 2023), CodeLlama (Rozière et al., 2023), StarCoder (Li et al., 2023a), CodeGen (Nijkamp et al., 2022).

We all use the official model weight from the Huggingface Models[3]. Above we used the model's abbreviation, we list the model's huggingface official label in Table 6.

D.4 Markdown vs. Flatten Setting Experiments

```
"0": Size: 100 * 5, Template: Easy, Model: GPT-3.5
"1": Size: 50 * 5,  Template: Easy, Model: GPT-3.5
"2": Size: 20 * 6,  Template: Count, Model: GPT-3.5
"3": Size: 40 * 10, Template: Where Condition Text, Model: GPT-3.5
```

[3] https://huggingface.co/models

Model	Name
Mistral-7B	mistralai/Mistral-7B-v0.1
Llama-2-13B	meta-llama/Llama-2-13b-hf
InternLM-20B	internlm/internlm-20b
Qwen-14B	Qwen/Qwen-14B
Llama-2-7B	meta-llama/Llama-2-7b-hf
Qwen-7B	Qwen/Qwen-7B
XGen-7B	Salesforce/xgen-7b-8k-base
Internlm-7B	internlm/internlm-7b
Phi-1_5	microsoft/phi-1_5
Stablelm-7B	stabilityai/stablelm-base-alpha-7b
Stablelm-3B	stabilityai/stablelm-base-alpha-3b
Pythia-12B	EleutherAI/pythia-12b
Pythia-6.9B	EleutherAI/pythia-6.9b
Pythia-2.8B	EleutherAI/pythia-2.8b
Pythia-1B	EleutherAI/pythia-1b
Llama-2-70B	meta-llama/Llama-2-70b-hf
CodeLlama-34B	codellama/CodeLlama-34b-hf
CodeLlama-13B	codellama/CodeLlama-13b-hf
CodeLlama-7B	codellama/CodeLlama-7b-hf
StarCoder-15B	bigcode/starcoderbase
StarCoder-7B	bigcode/starcoderbase-7b
StarCoder-3B	bigcode/starcoderbase-3b
StarCoder-1B	bigcode/starcoderbase-1b
CodeGen-15B	Salesforce/codegen-16B-multi
CodeGen-6B	Salesforce/codegen-6B-multi
CodeGen-2B	Salesforce/codegen-2B-multi
Yarn-LLaMA2-13B	NousResearch/Yarn-Llama-2-7b-64k
LongChat-13B	lmsys/longchat-7b-16k
RWKV-Raven-14B	lmsys/longchat-7b-16k

Table 6: LLMs used in our experiments and their corresponding names in Huggingface Hub.

```
"4": Size: 10 * 20, Template: Where Condition Text, Model: GPT-3.5
"5": Size: 10 * 15, Template: Where Condition Text, Model: GPT-3.5
"6": Size: 50 * 5,  Template: Easy, Model: Llama-2-13B
"7": Size: 100 * 5, Template: Easy, Model: Yarn-Llama-2-13B
"8": Size: 50 * 5,  Template: Easy, Model: Yarn-Llama-2-13B
"9": Size: 25 * 7, Template: General, Model: Llama-2-13B
"10": Size: (15~40) * (6~9), Template: General, Model: Llama-2-13B
"11": Size: (15~40) * (6~9), Template: General, Model: Llama-2-13B
"12": Size: (15~40) * (6~9), Template: Easy, Model: Llama-2-13B
"13": Size: (15~40) * (6~9), Template: Easy, Model: Llama-2-13B
"14": Size: (15~40) * (6~9), Template: Easy, Model: Llama-2-13B
```