SQLong: Enhanced NL2SQL for Longer Contexts with LLMs

Dai Quoc Nguyen, Cong Duy Vu Hoang, Duy Vu, Gioacchino Tangari
Thanh Tien Vu, Don Dharmasiri, Yuan-Fang Li, Long Duong
Oracle Corporation
{dai.nguyen,vu.hoang,duy.vu,gioacchino.tangari}@oracle.com
{thanh.v.vu,don.dharmasiri,yuanfang.li,long.duong}@oracle.com

Abstract

Open-weight large language models (LLMs) have significantly advanced performance in the Natural Language to SQL (NL2SQL) task. However, their effectiveness diminishes when dealing with large database schemas, as the context length increases. To address this limitation, we present SQLong, a novel and efficient data augmentation framework designed to enhance LLM performance in long-context scenarios for the NL2SQL task. SQLong generates augmented datasets by extending existing database schemas with additional synthetic CREATE TABLE commands and corresponding data rows, sampled from diverse schemas in the training data. This approach effectively simulates long-context scenarios during finetuning and evaluation. Through experiments on the Spider and BIRD datasets, we demonstrate that LLMs finetuned with SQLong-augmented data significantly outperform those trained on standard datasets. These imply SQLong's practical implementation and its impact on improving NL2SQL capabilities in real-world settings with complex database schemas.[1]

1 Introduction

The NL2SQL task focuses on translating natural language questions into SQL queries, enabling non-experts to interact with databases seamlessly (Deng et al., 2022). Recent advances leverage LLMs, finetuned on structured input prompts (*e.g., task instructions, database schema, and natural language question*), to achieve state-of-the-art performance (Yang et al., 2024b; Liu et al., 2024) on benchmarks such as Spider (Yu et al., 2018) and BIRD (Li et al., 2023). Despite significant progress, a critical challenge persists: LLMs finetuned on existing benchmarks still struggle with large database schemas due to limited context handling. Current datasets primarily feature small schemas, failing to represent real-world complexities. Additionally, the absence of publicly available large-schema datasets further hinders progress. Addressing this, we propose SQLong, a data augmentation framework designed to enhance LLM performance in long-context NL2SQL tasks by extending schemas to meet predefined context thresholds.

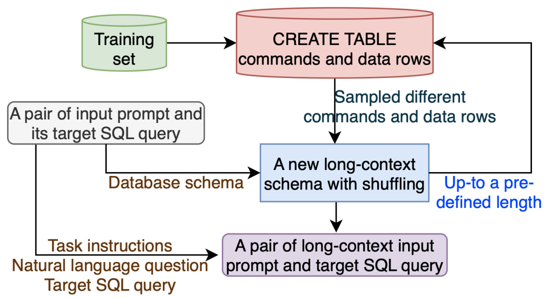

Figure 1: Our proposed SQLong Pipeline.

SQLong constructs augmented data by sampling CREATE TABLE commands and data rows from diverse schemas. These datasets enable LLMs to effectively manage large schemas and maintain robustness in long-context scenarios. Our experiments with *CodeQwen1.5-7B-Chat* (Bai et al., 2023) and *Llama-3.1-8B-Instruct* (Dubey et al., 2024) show SQLong consistently outperforms baseline finetuning, achieving an average accuracy improvement of over 2.2% on benchmarks like `Spider-dev`, `Spider-test`, and `BIRD-dev`.

Moreover, SQLong enables the creation of 45 long-context test sets, with context lengths up to 128k tokens. Models finetuned with SQLong exhibit significant performance gains, achieving an 11% improvement over base models and a 6% improvement over larger-scale models within the same family. These results highlight SQLong's effectiveness in real-world, large-schema scenarios.

In this paper, we focus on demonstrating that SQLong-augmented models outperform their unaugmented counterparts across varying context

[1]Table Representation Learning Workshop at ACL 2025

Proceedings of the 4th Table Representation Learning Workshop, pages 47–55

```
Given an input Question, create a syntactically correct
SQLite SQL query to run.
Pay attention to using only the column names that you can
see in the schema description.
Be careful to not query for columns that do not exist. Also,
pay attention to which column is in which table.
Please double check the SQLite SQL query you generate.
DO NOT use alias in the SELECT clauses.
Only use the tables listed below.

CREATE TABLE grades (
   "student_id" INTEGER,
   "student_name" TEXT,
   "subject" TEXT,
   "grade" TEXT,
   PRIMARY KEY ("student_id")
)
/* 3 rows from grades table:
student_id      student_name  subject    grade
1     Alice       math        A
2     Bob         math        B
3     David       science     B
*/

Question: Show me all the students getting an A in math

SELECT student_name FROM grades WHERE subject =
'math' AND grade = 'A'
```

Figure 2: Prompt template for the NL2SQL task.

lengths. While direct comparisons to retrieval-augmented generation (RAG) schema linking are beyond this paper's scope, our findings suggest combining SQLong with RAG could unlock further gains. Our main contributions include:

- **Introducing long-context NL2SQL:** A challenging new task for evaluating LLM performance on large database schemas.
- **SQLong pipeline:** A novel, scalable data augmentation approach for generating long-context training and test datasets.
- **Empirical insights:** Comprehensive experiments validating SQLong's effectiveness in enhancing LLM robustness and accuracy in long-context scenarios.
- **Resource sharing:** Plans to release SQLong datasets and code to support further research.

2 The Proposed SQLong Pipeline

The NL2SQL task aims to translate a natural-language question about a database schema into a corresponding SQL query. Following the standardized prompt template (Rajkumar et al., 2022), we represent the input prompt to LLMs in the format of *(task instructions, database schema, natural language question)*.[2] As illustrated in Figure 2, the database schema is represented by CREATE TABLE commands and three sample data rows for each corresponding table.

Using supervised finetuning (SFT) (Wei et al., 2022), LLMs can be trained on pairs of input prompts and target SQL queries to optimize their performance on the NL2SQL task. Specifically, given a training set $\mathbf{T}$ comprising pairs of input prompts $\mathbf{x}$ and corresponding target SQL queries $\mathbf{s}$, the supervised finetuning process can be formulated as minimizing the log-likelihood loss (Wei et al., 2022), as shown below:

$$\mathbb{E}_{(\mathbf{x},\mathbf{s})\sim\mathbf{T}}\left[\sum_{i=1}^{|\mathbf{s}|}\log p_\theta\left(s_i|\mathbf{s}_{<i},\mathbf{x}\right)\right]$$

wherein $|\mathbf{s}|$ is the length of $\mathbf{s}$, s_i is the i-th token, $\mathbf{s}_{<i}$ is the prefix of $\mathbf{s}$ up to the i-th position, and θ denotes the given LLM's parameters.

In this work, we introduce **SQLong**, a novel approach for constructing long-context finetuning and benchmark datasets, as illustrated in Figure 1. SQLong augments database schemas to enable large language models (LLMs) to effectively handle long-context scenarios in natural language to SQL (NL2SQL) tasks.

The SQLong pipeline has three main steps:

1. Schema Collection. We collect all CREATE TABLE commands and three sample data rows for each table from the training database schemas, compiling them into a comprehensive schema set.

2. Schema Augmentation. For each training pair, consisting of an input prompt (task instructions, database schema, natural language question) and its target SQL query, SQLong randomly samples items from the schema set. These sampled items contain table names distinct from those in the given database schema. The sampled items are combined with the original schema, and the resulting schema is randomly shuffled to produce a new, long-context database schema. This shuffling introduces variability in the positions of the original tables and columns.

3. Long-Context Prompt Generation. SQLong generates an augmented input prompt in the format of task instructions, the long-context database schema, and the natural language question, while keeping the target SQL query unchanged. It ensures that the combined length of the long-context input prompt and the target SQL query does not exceed a predefined context length (e.g., 32k tokens), maintaining compatibility with the model's tokenizer constraints.

[2] In datasets with additional complexity, such as BIRD, the question may be supplemented with extra information, such as evidence. For simplicity, this additional information is omitted in Figure 2.

By systematically extending and diversifying the context, SQLong enhances the robustness and effectiveness of LLMs in handling long-context NL2SQL tasks. We summarise the steps involved in SQLong in Algorithm 1 in Appendix A.1.

3 Evaluation

We assess the effectiveness of our proposed SQLong model in enhancing NL2SQL performance in both short-context and long-context scenarios.

3.1 Experimental Setup

Datasets For the short-context evaluation, we utilize widely adopted benchmark datasets, including Spider (Yu et al., 2018), Spider-realistic (Deng et al., 2020), Spider-syn (Gan et al., 2021), and BIRD (Li et al., 2023). [3] It is noted that Spider-Syn is manually created based on Spider training and development sets using synonym substitution in the original questions, while Spider-realistic is created based on Spider development set by manually removing the explicit mention of column names in the original questions. The BIRD-test set is not publicly available.

For the long-context evaluation, we extend each of the Spider-dev, Spider-test, Spider-realistic, Spider-syn, and BIRD-dev datasets by applying SQLong with a pre-defined context length. Specifically, we generate augmented long-context test sets for nine context lengths: 8k, 16k, 24k, 32k, 40k, 48k, 56k, 64k, and 128k. This process results in a total of 45 long-context test sets, constructed in accordance with the tokenizer of the base model.

Importantly, the long-context test sets are constructed with distinct database schema alignments. To build Spider-based long-context test sets, we use the database schemas from the BIRD training set, whereas for the BIRD-dev long-context test sets, we use the database schemas from the Spider training set. This ensures a robust evaluation across diverse schema configurations and context lengths. The data statistics of the experimental datasets are presented in Figure 3 and Tables 1 and 2.

Baseline Models and Evaluation Metrics We evaluate SQLong using two powerful base models: CodeQwen1.5-7B-Chat (Bai et al., 2023), which supports a context length of up to 64k, and Llama-3.1-8B-Instruct (Dubey et al., 2024), which supports a context length of up to 128k. Following Yu et al. (2018), we report execution-match accuracy on both the original short-context test sets and the augmented long-context test sets.

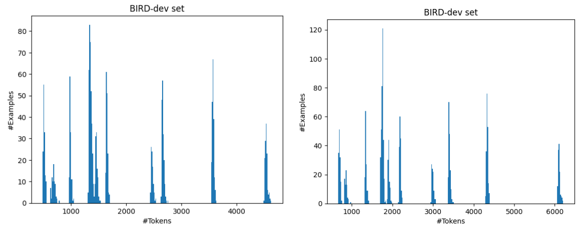

Figure 3: Statistics of input prompt lengths with respect to Llama-3.1-8B-Instruct's tokenizer (left) and CodeQwen1.5-7B-Chat's tokenizer (right) on the original BIRD-dev set. Similarly, the maximum input prompt lengths for the original Spider-related sets are approximately 2,000 tokens for Llama-3.1-8B-Instruct's tokenizer and 2,500 tokens for CodeQwen1.5-7B-Chat's tokenizer.

Dataset	#DB	#tables	#training	#dev	#test
Spider	200	5 ± 3	6,712	1,034	2,019
Spider-syn	200	5 ± 3	6,712	1,034	–
Spider-realistic	200	5 ± 3	6,712	508	–
BIRD	98	7 ± 3	9,428	1,534	–

Table 1: Statistics of the experimental datasets. **#DB** denotes the number of databases. **#tables** denotes the mean and standard deviation of numbers of tables in the databases.

Length	CodeQwen1.5-7B-Chat		Llama-3.1-8B-Instruct	
	Spider-related	BIRD-dev	Spider-related	BIRD-dev
8k	37 ± 4	35 ± 8	48 ± 5	48 ± 8
16k	72 ± 6	76 ± 8	94 ± 7	102 ± 9
24k	107 ± 7	118 ± 8	141 ± 8	157 ± 9
32k	142 ± 8	159 ± 9	186 ± 8	211 ± 9
40k	177 ± 8	200 ± 9	233 ± 9	269 ± 9
48k	212 ± 9	242 ± 9	279 ± 9	320 ± 10
56k	247 ± 9	283 ± 9	326 ± 9	374 ± 9
64k	283 ± 9	324 ± 9	372 ± 8	429 ± 9
128k	551 ± 4	639 ± 7	725 ± 9	843 ± 8

Table 2: Mean and standard deviation statistics of the numbers of tables in input prompts for our augmented long-context test sets with respect to each model's tokenizer.

Training Protocol For each original training set, we use SQLong to create an augmented *long-context finetuning* dataset with context lengths of up to 32k.[4] The augmented dataset is combined with the original training set to form the final

[3]We use the latest BIRD-dev dataset, updated on June 27, 2024. The BIRD-test set is not publicly available.

[4]Due to computational constraints, we limit finetuning to context lengths of up to 32k. Specifically, for each training example, the context length is randomly sampled from a range starting at 4,096 and increasing by 512 increments up to 32,768.

Model	Spider-dev	Spider-realistic	Spider-syn	Spider-test	BIRD-dev	Average
Qwen2-72B-Instruct	82.7	80.7	73.0	82.9	53.7	74.6
CodeQwen1.5-7B-Chat	76.4	70.1	62.7	75.1	44.3	65.7
Finetuned without SQLong	81.9	76.2	68.7	79.6	51.4	71.6
Finetuned with SQLong	**83.4**	**79.7**	**71.2**	**81.3**	**53.3**	**73.8**
Llama-3.1-70B-Instruct	80.7	78.0	73.0	83.7	61.5	75.4
Llama-3.1-8B-Instruct	71.1	63.8	61.0	65.7	40.9	60.5
Finetuned without SQLong	79.2	76.4	69.6	80.4	51.9	71.5
Finetuned with SQLong	**83.2**	**78.0**	**73.1**	**81.8**	**53.3**	**73.9**

Table 3: Execution-match accuracy results (in %) across different datasets and model configurations. Finetuning with SQLong consistently improves performance, with the best results highlighted in **bold**.

dataset used for finetuning the base models.[5]

We experiment with two base models: CodeQwen1.5-7B-Chat (Bai et al., 2023), which supports a 64k context length, and Llama-3.1-8B-Instruct (Dubey et al., 2024), which supports a 128k context length. Finetuning is performed with a batch size of 1, gradient accumulation steps of 8, a learning rate chosen from $1 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}$, and up to 5 epochs on 8×H100 80GB GPUs.

We use Huggingface's TRL (von Werra et al., 2020) for supervised finetuning, employing 8-bit AdamW (Dettmers et al., 2021), Flash Attention v2 (Dao, 2023), and DeepSpeed ZeRO-3 Offload (Ren et al., 2021). For a fair comparison, we also finetune the base models on the original training set (i.e., without SQLong) under the same settings.

Inference Protocol We utilize vLLM (Kwon et al., 2023) for the inference process. For long-context test sets, we employ dynamic NTK RoPE scaling (Peng et al., 2023) to extend support up to a 128k context length for CodeQwen1.5-7B-Chat and its finetuned variants.

3.2 Main Results

Performance on Original Datasets Table 3 summarizes the results on the original development and test sets, comparing base models with larger LLMs such as Llama-3.1-70B-Instruct (Dubey et al., 2024) and Qwen2-72B-Instruct (Yang et al., 2024a). Models finetuned using long-context augmentation via SQLong consistently outperform their counterparts finetuned on original contexts. On average, SQLong delivers an absolute improvement of over 2.2% across five benchmark datasets. Additionally, SQLong-finetuned models achieve performance comparable to much larger LLMs on specific datasets, showcasing the scalability and efficiency of the approach.

Performance on Long-Context Datasets Figure 4 illustrates the experimental results on long-context test sets. The full details are presented in Tables 4 and 5 in Appendix A.2. Across all datasets, models finetuned with SQLong demonstrate superior performance compared to those trained without SQLong. For instance, on the Spider-test datasets with 8k and 24k context lengths, the Llama-3.1-8B-Instruct model achieves outstanding results of 77.1% and 72.3%, reflecting absolute gains of 7.2% and 13.3%, respectively. Notably, the SQLong-finetuned Llama-8B model outperforms the larger Llama-70B model on 41 out of 45 long-context test sets, with minor exceptions on Spider-realistic 8k and BIRD-dev 8k, 16k, and 24k sets. Similar performance trends are observed with the Qwen models.

On average, SQLong finetuning delivers an 11% absolute improvement over models without SQLong and a 6% advantage over 70B models within the same model family. These results underscore the efficacy of SQLong in handling long-context scenarios and advancing the performance of NL2SQL systems.

Positional robustness We conduct an experiment wherein each original database schema is placed at different positions within the input prompt, assessing the models' ability to detect it regardless of its location.

We select a set of 124 samples from Spider-dev, Spider-realistic, and Spider-syn, ensuring each sample has a maximum input prompt and target SQL query length of 384 tokens according to CodeQwen1.5-7B-Chat's tokenizer. Using SQLong, we augment this set to a 64k context

[5]For Spider, we finetune the base models on the Spider training set and evaluate performance on Spider-dev, Spider-test, Spider-realistic, and Spider-syn.

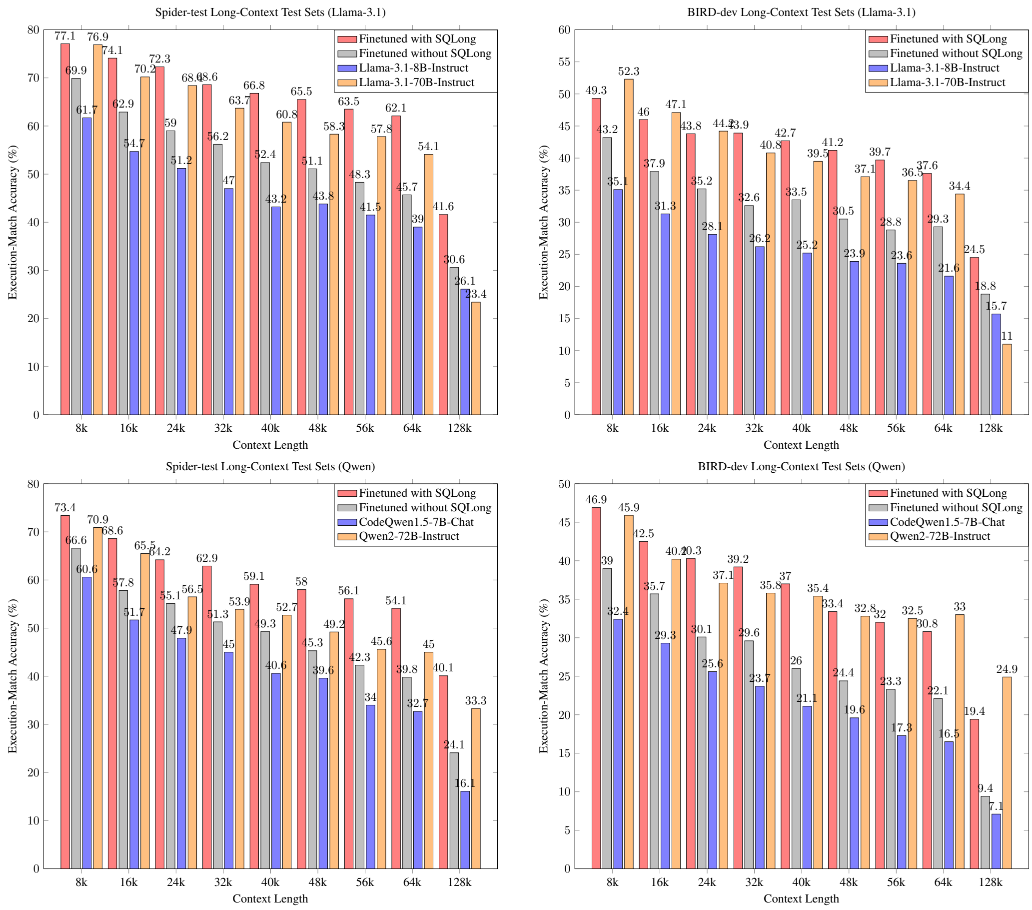

Figure 4: Execution-match accuracy (in %) for Llama-3.1 (top) and Qwen (bottom) families on Spider-test (left) and BIRD-dev (right) long-context test sets.

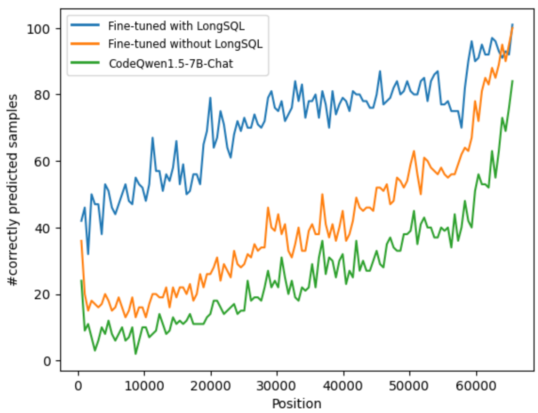

Figure 5: Robust impact of fine-tuned models.

length. In each augmented set, the original database schemas are positioned at specific offsets, starting from 512 and incrementing by 512 up to 64k. This results in 125 new test sets, each containing 124 samples with a 64k context length, corresponding to a distinct schema position.

We compute the number of correctly executed samples for each test set, as shown in Figure 5. The results demonstrate that the long-context fine-tuned model with SQLong is significantly more robust compared to the model without fine-tuning.

4 Conclusion and Future Work

Handling large database schemas poses a significant challenge for NL2SQL models. In this paper, we introduce long-context NL2SQL generation, a novel task that reflects real-world scenarios, and propose SQLong, a simple yet effective augmentation approach for creating long-context finetuning and benchmark datasets. Experiments show that LLMs finetuned with SQLong significantly outperform their counterparts on benchmarks like Spider, BIRD, and our long-context test sets (up to 128k context length).

Future work includes leveraging a RAG-based schema linking approach to retrieve relevant schema elements, enabling more concise and efficient inputs for SQLong-tuned models.

References

Jinze Bai, Shuai Bai, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Naihao Deng, Yulong Chen, and Yue Zhang. 2022. Recent advances in text-to-SQL: A survey of what we have and what we expect. In *Coling*, pages 2166–2187.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2020. Structure-grounded pretraining for text-to-sql. *arXiv preprint arXiv:2010.12773*.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R Woodward, Jinxia Xie, and Pengsheng Huang. 2021. Towards robustness of text-to-sql models against synonym substitution. In *ACL-IJCNLP*, pages 2505–2515.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *SOSP*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *NeurIPS 2023 Track on Datasets and Benchmarks*, 36.

Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuyu Luo, Yuxin Zhang, Ju Fan, Guoliang Li, and Nan Tang. 2024. A survey of nl2sql with large language models: Where are we, and where are we going? *arXiv preprint arXiv:2408.05109*.

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

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. {Zero-offload}: Democratizing {billion-scale} model training. In *USENIX ATC*, pages 551–564.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

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

An Yang, Baosong Yang, and 1 others. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Jiaxi Yang, Binyuan Hui, Min Yang, Jian Yang, Junyang Lin, and Chang Zhou. 2024b. Synthesizing text-to-sql data from weak and strong llms. *arXiv preprint arXiv:2408.03256*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*, pages 3911–3921.

A Appendix

A.1 The algorithm steps in SQLong

Algorithm 1: The algorithm steps involved in the proposed SQLong.

Input: A training set $\mathbf{T}$ of pairs of input prompts and target SQL queries: $\mathbf{T} = \{((instructions_i, database\_schema_i, question_i), target\_sql_i)\}_{i=1}^{N}$, wherein each $database\_schema_i$ is a set of CREATE TABLE commands and three data rows for each corresponding table; a set $\mathcal{T} = \{((instructions_j, database\_schema_j, question_j), target\_sql_j)\}_{j=1}^{M}$; the base model's tokenizer tk, a starting number s_n (default 4096), an ending number e_n (default 32768), an increasing number i_n (default 512), and a pre-defined number p_n (default 8192).

Output: The augmented long-context set $\mathcal{T}'$.

$schema\_set \leftarrow$ collect_unique_commands_and_data_rows($\{database\_schema_i\}_{i=1}^{N}$)

$table\_names \leftarrow$ get_table_names($schema\_set$)

$item\_lengths \leftarrow \{\}$

for $item \in schema\_set$ **do**

  $item_lengths \leftarrow item_lengths \cup \{$get_length($item$, tk)$\}$

$\mathcal{T}' \leftarrow \{\}$

$diverse\_lengths \leftarrow$ range(s_n, $e\_n + 1$, i_n)

for $((instructions, database\_schema, question), target\_sql) \in \mathcal{T}$ **do**

  $original\_length \leftarrow$ get_length($instructions + database\_schema + question + target\_sql$, tk)

  $certain\_length \leftarrow$ randomly_select_value($diverse\_lengths$) `// This aims to construct long-context fine-tuning data with` $\mathbf{T} = \mathcal{T}$`. Otherwise,` $certain\_length$ `is set to` p_n `to construct long-context benchmark data.`

  $local\_table\_names \leftarrow$ get_table_names($database\_schema$)

  $augmented\_schema \leftarrow \{\}$

  for $idx \in$ shuffle_list(range(0, get_size($schema\_set$))) **do**

    if $schema\_set[idx] \notin database\_schema$ and $table\_names[idx] \notin local\_table\_names$ and $original\_length + item\_lengths[idx] < certain\_length$ **then**

      $original\_length \leftarrow original\_length + item\_lengths[idx]$

      $augmented\_schema \leftarrow augmented\_schema \cup \{schema\_set[idx]\}$

  $augmented\_long\_context\_schema \leftarrow$ shuffle_list($augmented\_schema \cup database\_schema$)

  $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{((instructions, augmented\_long\_context\_schema, question), target\_sql)\}$

A.2 Full execution-match accuracy results for all long-context test sets

Model	Context length	Spider-dev	Spider-realistic	Spider-syn	Spider-test	BIRD-dev	Average across 45 sets
		Dataset					
Llama-3.1-8B-Instruct	8k	61.9	53.5	45.1	61.7	35.1	
	16k	58.5	47.0	38.9	54.7	31.3	
	24k	53.2	43.1	32.7	51.2	28.1	
	32k	49.6	42.9	29.9	47.0	26.2	
	40k	48.7	38.4	28.4	43.2	25.2	37.2
	48k	46.9	35.8	24.9	43.8	23.9	
	56k	45.5	32.1	23.8	41.5	23.6	
	64k	42.6	33.1	22.5	39.0	21.6	
	128k	28.0	17.9	10.3	26.1	15.7	
Our model fine-tuned Without SQLong	8k	71.7	63.4	49.3	69.9	43.2	
	16k	66.6	54.7	39.9	62.9	37.9	
	24k	63.6	52.4	35.5	59.0	35.2	
	32k	59.4	48.0	33.1	56.2	32.6	
	40k	57.0	45.1	30.2	52.4	33.5	43.8
	48k	55.9	43.7	28.0	51.1	30.5	
	56k	52.5	40.4	25.7	48.3	28.8	
	64k	51.4	40.9	25.3	45.7	29.3	
	128k	34.7	23.6	13.5	30.6	18.8	
Our model fine-tuned With SQLong	8k	**77.4**	67.1	**61.7**	**77.1**	49.3	
	16k	**75.2**	**66.1**	**53.4**	**74.1**	46.0	
	24k	**71.8**	**64.2**	**50.0**	**72.3**	43.8	
	32k	**68.3**	**61.6**	**46.5**	**68.6**	**43.9**	
	40k	**67.5**	**62.8**	**44.9**	**66.8**	**42.7**	**54.8**
	48k	**66.9**	**56.7**	**40.2**	**65.5**	**41.2**	
	56k	**63.3**	**52.6**	**38.4**	**63.5**	**39.7**	
	64k	**61.3**	**52.2**	**39.3**	**62.1**	**37.6**	
	128k	**43.0**	**33.7**	**21.7**	**41.6**	**24.5**	
Llama-3.1-70B-Instruct	8k	73.9	**67.3**	55.0	76.9	**52.3**	
	16k	67.7	59.4	48.9	70.2	**47.1**	
	24k	62.4	54.9	43.8	68.4	**44.2**	
	32k	60.9	49.6	41.7	63.7	40.8	
	40k	59.0	52.6	37.4	60.8	39.5	48.5
	48k	57.6	46.9	35.0	58.3	37.1	
	56k	55.3	46.3	32.3	57.8	36.5	
	64k	55.0	43.9	31.7	54.1	34.4	
	128k	28.0	25.6	12.3	23.4	11.0	

Table 4: Execution-match accuracy results (in %) on the augmented long-context test sets with respect to the Llama-3.1 model family.

Model	Context length	Dataset					Average across 45 sets
		Spider-dev	Spider-realistic	Spider-syn	Spider-test	BIRD-dev	
CodeQwen1.5-7B-Chat	8k	61.7	49.6	38.1	60.6	32.4	
	16k	55.9	42.1	30.7	51.7	29.3	
	24k	51.5	37.8	27.9	47.9	25.6	
	32k	48.0	30.9	22.8	45.0	23.7	
	40k	46.7	28.9	21.0	40.6	21.1	31.7
	48k	42.4	27.8	18.7	39.6	19.6	
	56k	36.4	24.0	17.5	34.0	17.3	
	64k	36.4	21.3	15.8	32.7	16.5	
	128k	19.2	7.9	6.4	16.1	7.1	
Our model fine-tuned Without SQLong	8k	68.9	57.1	39.5	66.6	39.0	
	16k	62.6	51.4	31.8	57.8	35.7	
	24k	57.6	49.0	29.3	55.1	30.1	
	32k	53.0	41.5	25.6	51.3	29.6	
	40k	53.7	38.4	23.5	49.3	26.0	37.8
	48k	48.7	34.6	22.3	45.3	24.4	
	56k	44.5	33.1	20.9	42.3	23.3	
	64k	43.8	30.3	18.4	39.8	22.1	
	128k	26.1	15.6	9.2	24.1	9.4	
Our model fine-tuned With SQLong	8k	**75.9**	**65.7**	**53.2**	**73.4**	**46.9**	
	16k	**72.9**	**62.6**	**46.6**	**68.6**	**42.5**	
	24k	**68.9**	**58.5**	**43.0**	**64.2**	**40.3**	
	32k	**67.5**	**54.3**	**40.0**	**62.9**	**39.2**	
	40k	**63.4**	**53.7**	**37.4**	**59.1**	**37.0**	**50.2**
	48k	**63.9**	**52.8**	**35.3**	**58.0**	**33.4**	
	56k	**60.3**	**51.0**	**33.6**	**56.1**	32.0	
	64k	**60.6**	**52.4**	**31.0**	**54.1**	30.8	
	128k	**43.4**	**33.7**	**19.4**	**40.1**	19.4	
Qwen2-72B-Instruct	8k	70.6	63.4	47.2	70.9	45.9	
	16k	69.1	58.7	40.6	65.5	40.2	
	24k	60.9	53.3	34.1	56.5	37.1	
	32k	59.6	45.5	31.1	53.9	35.8	
	40k	55.8	45.7	29.5	52.7	35.4	44.2
	48k	52.3	43.7	27.8	49.2	32.8	
	56k	50.8	39.4	27.6	45.6	**32.5**	
	64k	47.3	34.6	25.1	45.0	**33.0**	
	128k	36.8	28.3	18.6	33.3	**24.9**	

Table 5: Execution-match accuracy results (in %) on the augmented long-context test sets with respect to the Qwen mdoel family.