FINE-TUNING OF SMALL/MEDIUM LLMS FOR BUSINESS QA ON STRUCTURED DATA

Rasha Ragab and Abdulrahman Altahhan

School of Computing, University of Leeds, Leeds, UK

ABSTRACT

Enabling business users to directly query their data sources is a significant advantage for organisations. The majority of enterprise data is housed within databases, requiring extensive procedures that involve intermediary layers for reporting and its related customization. The concept of enabling natural language queries, where a chatbot can interpret user questions into database queries and promptly return results, holds promise for expediting decision-making and enhancing business responsiveness. This approach empowers experienced users to swiftly obtain data-driven insights. The integration of Text-to-SQL and Large Language Model (LLM) capabilities represents a solution to this challenge, offering businesses a powerful tool for query automation. However, security concerns prevent organizations from granting direct database access akin to platforms like OpenAI. To address this limitation, this Paper proposes developing fine-tuned small/medium LLMs tailored to specific domains like retail and supply chain. These models would be trained on domain-specific questions and Queries that answer these questions based on the database table structures to ensure efficacy and security. A pilot study is undertaken to bridge this gap by fine-tuning selected LLMs to handle business-related queries and associated database structures, focusing on sales and supply chain domains. The research endeavours to experiment with zero-shot and fine-tuning techniques to identify the optimal model. Notably, a new dataset is curated for fine-tuning, comprising business-specific questions pertinent to the sales and supply chain sectors. This experimental framework aims to evaluate the readiness of LLMs to meet the demands for business query automation within these specific domains. The study contributes to the progression of natural language query processing and database interaction within the realm of business intelligence applications.

KEYWORDS

NLP, Text-2-SQL, Fine-Tuning, Small/Medium LLM.

1. INTRODUCTION

In the domain of natural language processing, semantic parsing is recognized as a fundamental capability, serving as the conduit through which natural language queries are efficiently transformed into structured query language (SQL) representations. This transformation is facilitated by methodologies such as sequence-to-sequence learning and attention mechanisms, which play essential roles in accurately converting natural language queries into SQL equivalents. The dataset comprising pairs of natural language queries and their corresponding SQL representations serves as a cornerstone resource, enabling the model to acquire and refine its parsing abilities by exposure to diverse linguistic patterns and query structures.

David C. Wyld et al. (Eds): AdNLP, CDKP, NCO, SAI, MLT, Signal, SOFT, ICAITA, CMC - 2024
pp. 43-61, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141005

Moreover, the integration between Test-2-SQL and LLM involves the incorporation of schema-aware parsing techniques to enhance parsing accuracy and relevance in SQL query generation. These techniques leverage schema information derived from the underlying database structure to guide the parsing process. Additionally, the process of fine-tuning constitutes a crucial aspect of model enhancement, aiming to equip the model with domain-specific knowledge and task-oriented proficiency. This endeavour enhances the model's generalization capabilities across various query patterns and specific task contexts.

Therefore, the utilization of text-to-SQL and LLM technologies is a significant capability that empowers business teams lacking proficiency in SQL or possessing limited knowledge of it, as it enables them to directly query their data and receive textual responses. For instance, they can inquire, "What is the customer with the highest sales value in 2024?" and receive a response such as, "Carrefour achieved the highest sales in 2024, valued at 50 million Euros." To obtain this information, the company's database is accessed through an engine equipped with the appropriate SQL statement to retrieve the answer, and subsequently, the LLM transforms the output into the appropriate textual format. Such an approach improves efficiency by eliminating the need to construct intermediary layers, dashboards, or reports that may not sufficiently address the diverse information needs of experienced users seeking insights from raw data. Moreover, it addresses challenges where IT or data teams may face delays in meeting these demands promptly.

Despite these advantages, organizations often harbour concerns regarding data security, especially about potential data leakage risks associated with large-scale LLM providers who control data usage and may record user interactions. Consequently, open-source LLMs present a viable solution by allowing enterprises to fine-tune and deploy their own local LLMs and chat agents on internal cloud platforms, thereby ensuring adherence to organizational data policies and access protocols.

However, leveraging open-source LLMs poses its own set of challenges, particularly in the realm of SQL query generation and dialect compatibility. Various SQL dialects such as Microsoft SQL Server, MySQL, and Oracle necessitate tailored approaches due to differing functions, optimisation techniques, and keywords. This necessitates a task-specific approach rather than a one-size-fits-all solution, requiring continuous evaluation and improvement cycles.

Moreover, the model's ability to comprehend database schemas and execute complex queries aligned with domain-specific requirements underscores the need for domain-specific fine-tuning. To address this, a curated dataset comprising diverse business questions on sales and supply chain schemas spanning from single-table to complex eight-tables has been prepared for model refinement.

Furthermore, the model's capacity to comprehend the correct formula based on the provided schema (table structure) is critical. For example, contemplate a situation in which the aim is to compute the overall sales value, where the quantity sold is stored in one table and the unit price in another. This necessitates joining both tables and multiplying the quantity by the unit price to derive the total sales figure. The intricacy of this undertaking fluctuates based on the schema and the particular formula or Key Performance Indicator (KPI) needed, which is tailored to the domain and must be comprehended by the model before query formulation to ensure precision in results.

Consequently, this study includes the development of a foundational dataset comprising 1,000 diverse business questions related to sales and supply chain schemas and the SQL queries that answer that question on the context schema specified on both the SQL server and My SQL Query syntax. This dataset serves as an initial framework for fine-tuning the models to enhance their domain-specific understanding, thereby optimizing performance in text-to-SQL tasks within the business domain. The dataset will encompass inquiries related to invoices, sales orders, shipments, customer and product master data, payments, etc. Notably, to the best of the author's knowledge, there is currently a lack of freely available domain-specific question-answer sets on sales operations and supply chain.

This study draws inspiration from Stanford Alpaca's work [1] which showcases seed questions and then generates further similar examples using GPT 4 and uses it to fine-tune the model. This current experiment follows a similar approach and uses GPT 3.5 to generate domain-specific questions and context/schema/table structure to the scope we aim for but not for query generation as this depends on the dialect of the database. So, the SQL query that answers the questions is not generated by GPT 3.5 in the same approach used in the Alpaca paper.

As noted in Alibaba's assessment [2], open-source Large Language Models (LLMs) remain relatively understudied in current research, which predominantly focuses on proprietary LLMs like those developed by OpenAI, particularly in the context of text-to-SQL question-answering tasks. However, recent expansions in open-source LLMs have shown notable progress across various Natural Language Processing (NLP) tasks, although their application specifically in text-to-SQL tasks remains relatively unexplored.

Models such as GPT-3 [3], PaLM, ChatGPT, GPT-4 [4,5], and PaLM-2 [6], categorized as Generative Pre-trained Transformers (GPTs), have demonstrated significant advancements in text-to-SQL capabilities through the utilization of zero-shot and few-shot prompting techniques. These methodologies present the benefit of needing minimal training and exhibiting efficient adaptability to unforeseen data, rendering them highly suitable for query generation tasks, particularly when considering the diverse SQL dialects encountered.

However, the performance of these models may not always be optimal. For instance, research on CodeX [7] and ChatGPT [8] has highlighted successful outcomes using in-context learning for text-to-SQL tasks, although there remains a discernible performance gap compared to fine-tuned alternatives utilizing median-sized LLMs.

This study aims to assess the capability of small and medium-sized open Large Language Models (LLMs) in formulating accurate queries that yield correct responses directly from databases, specifically Microsoft SQL Server and MySQL. Firstly, a dataset comprising 1000 sales and supply chain business questions will be curated, covering various sales operation questions. Subsequently, the performance of selected open LLMs of differing sizes will be evaluated using two methodologies firstly the Zero-shot: Assessing the models' performance without fine-tuning. Secondly, Fine-tuning on Different Dataset Sizes: Examining the impact of fine-tuning using datasets of varying sizes, including 1K, 10K, and 30K Question/Answer/Context (QAC) pairs.

In this study, various Large Language Models (LLMs) were utilized, including meta-llama/Llama-2-7b-chat-hf [9], a small-sized LLM sourced from Meta's official Llama-2 model via Hugging Face with appropriate permissions granted for usage. Additionally, defog/sqlcoder-7b [10], another small-sized LLM, bugdaryan/Code-Llama-2-13B-instruct-text2sql [11], categorized as mid-sized, and gaussalgo/T5-LM-Large-text2sql-spider [12], a large-sized LLM, were employed for analysis and experimentation in the research context. Each of these models was selected to assess their performance and suitability in the specific text-to-SQL tasks under investigation.

2. DESIGN AND METHODOLOGY

This study adopted a rigorous data collection and processing methodology, involving data gathering and model fine-tuning. We elaborate on the different stages of this process in the subsequent sections.

2.1. Model Selection

The experiments utilize specific Hugging Face Large Language Model (LLM) repositories that were specifically tailored and fine-tuned for text-to-SQL tasks and were used for evaluation and finetuning to ensure suitability for the task at hand. Another model was Meta's LIAMA 2 [8], which has demonstrated success in various Natural Language Processing (NLP) tasks.

The selected models are:

- meta-llama/Llama-2-7b-chat-hf [9], Small size LLM.
- defog/sqlcoder-7b [10], Small size LLM.
- bugdaryan/Code-Llama-2-13B-instruct-text2sql [11], mid-size LLM
- gaussalgo/T5-LM-Large-text2sql-spider [12], large-size LLM

2.2. Model Fine-tuning

Initially, fine-tuning entails the adjustment of weights and parameters in a pre-trained model with new data to enhance its performance on a specific task. Employing techniques to optimize the parameters eligible for updating mitigates computational expenses associated with updating all parameters of the model. The research employs parameter-efficient fine-tuning methods, such as Parameter-efficient Fine-tuning (PEFT), which entails the selective freezing of specific layers within the pre-trained model, while fine-tuning only the final layers pertinent to the subsequent task. This strategy preserves computational resources and reduces time consumption in contrast to training the entire model anew. Additionally, QLoRA (Quantized LoRa) [13], an extension of LoRA, addresses memory optimization by employing quantization to map continuous values to a smaller set of discrete ones. This optimization reduces the model's memory consumption without compromising training precision, thus optimizing both computational and memory requirements during the training process.

QLoRA utilizes quantization to reduce a pre-trained language model's precision to 4 bits while employing the "float16" compute type to expedite training. The LoRA attention dimension is configured to 64, with the Alpha parameter set at 16 for scaling, and a dropout probability of 10% is applied to LoRA.

Regarding the training parameters, learning_rate is 2e-4, weight decay is set to 0.001 to be applied to all layers except bias/LayerNorm weights, and the optimizer is "paged_adamw_32bit". The number of epochs is 10 in small models and 5 in with 30 K Row FT dataset.

Our base models will undergo fine-tuning utilizing either the 1K new business-related dataset or the "b-mc2/sql-create-context"[23] dataset sourced from Hugging Face.

The training procedure utilizes the Hugging Face trainer, which initializes the SFT-Trainer class [17] (Supervised Fine-Tuning Trainer), a pivotal step in reinforcement learning from human feedback (RLHF). Hugging Face, through its TRL library, enhances the accessibility of SFT models via straightforward APIs, enabling training on customized datasets with minimal code requirements.

2.3. Models Learning Strategies

In all base models as part of the experimental methodology, zero-shot learning, a machine learning paradigm wherein a model is trained to recognize classes it has never encountered previously, has been utilized. In addition to fine-tuning (FT) the models and evaluating the improvement in the FT model QA ability versus the base model.

Fine-tune "meta-llama/Llama-2-7b-chat-hf“as follows:

- 10K row from “b-mc2/sql-create-context” hugging face dataset.
- 1K new business QAC plus 9K from “b-mc2/sql-create-context”
- 1K new business QAC plus 29K from “b-mc2/sql-create-context”
- Fine-tune " defog/sqlcoder” on 1K new business QAC pairs.
- Fine-tune " bugdaryan/Code-Llama-2-13B-instruct-text2sql” on 1K new business QAC pairs.

All are fine-tuned for 10 epochs except the 30K row FT where 5 epochs only were applied, as a single epoch on this dataset size took 1.5 hours on the A100 GPU on Google Colab.

Most of these experiment activities took place using Google Colab with T4 GPU and A100 40G GPU and 80G RAM but always the memory usage was low while GPU usage was 20G on average during the model building and QA processing, this varies based on the model size. Also, many preparation tasks for the dataset used Anaconda and Jupyter notebooks. Microsoft SQL server and MY SQL server instances were installed locally to verify the queries and check the results. Moreover, a cloud membership for MSSQL was established to facilitate connectivity from Google Colab to the database, enabling the retrieval of questions and saving model responses for evaluation purposes.

Initially, fine-tuning experiments were conducted on a cloud computational resource provider referred to as Modal [17], following the guidelines provided in a designated tutorial [18]. Subsequently, the final trials were conducted on Google Colab to facilitate closer monitoring, as detailed in a separate tutorial [19], with adjustments tailored to the specific task requirements. Google Colab offers a more structured approach to work, facilitating debugging processes in comparison to Modal. However, Modal exhibits the advantageous capability of maintaining script execution even in the event of client disconnection, rendering it particularly useful for long-run script operations.

The model building, fine-tuning, and QA testing use the LLM libraries provided by Hugging Face like the transformer class, in addition to the tokenizer and peft key classes. Some trials use the LIAMA Index and Lang Chain with GPT 3.5 in the dataset preparation part.

2.4. Datasets

The datasets used for text-to-SQL fine-tuning and zero-shot evaluation are a combination of the following:

- Hugging face b-mc2/sql-create-context [14] dataset. The dataset was constructed by amalgamating elements from both the WikiSQL and Spider datasets. It incorporates creating table statements as contextual information, accompanied by corresponding questions and their respective answers. The answer, serving as the query, utilizes the schema provided in the contextual information to retrieve the information sought in the question. Notably, the questions exhibit simplicity and low complexity, as evidenced by the comprehensive sample check conducted during the experimental phase.

- A novel dataset was generated to address business-specific inquiries on schemas relevant to sales and supply chain operations, encompassing areas such as invoices, payments, shipments, customer master data, and product master data. The creation of this dataset involved employing three distinct methodologies, leveraging GPT-3.5 for schema generation and/or the generation of business domain questions. Furthermore, the dataset construction process drew upon the author's prior professional experiences in the field.

Three methodologies are used for the dataset generation. Initially, publicly accessible datasets (This covers the tables schema and table data) such as the sales history of Rossman, a European drug-store chain, were utilized [16], obtained from Kaggle. Additionally, classic models from a MySQL public database were acquired. Subsequently, manual data analysis was conducted to delineate pertinent business questions, which were then presented to GPT-3.5 using the LLAMA INDEX framework in conjunction with SQL Alchemy. The resultant queries were archived within a database, followed by a meticulous review process to rectify any inaccuracies, before re-executing the queries on the database and saving the finalized outcomes. This method underscores the utilization of pre-existing datasets and the adoption of automated query generation based on formulated business inquiries.

The second approach involved constructing table schemas manually, ranging from one to four tables. GPT-3.5 was then tasked with generating business questions specific to these table schemas, which were subsequently curated. Additional business questions of increasing complexity were manually devised, and queries were constructed and directly verified on the database. The outcomes of these queries were recorded and saved for analysis. This approach underscores a systematic progression in query generation and schema refinement, integrating both manual and automated processes.

In the third methodology, a more intricate schema comprising eight tables was conceived through interactive engagement with GPT-3.5, soliciting an Entity-Relationship Diagram (ERD) tailored to specific requirements. The resultant schema aligns closely with real-world schemas encountered by the author in practical applications. Business inquiries pertinent to this schema were generated by GPT-3.5 and meticulously curated. Subsequently, manual business questions of escalating complexity were introduced, followed by the utilization of GPT-3.5 in conjunction with the LLAMA INDEX framework within SQL Alchemy to generate SQL queries. Analogous to the initial approach, manual scrutiny of queries was undertaken to rectify errors before their execution on the database and subsequent storage of outcomes.

More details about the question generation process are shown below with examples:

Firstly the following question was asked to the Chat GPT 3.5 and many similar questions as well, this is for illustrative purposes.

> For the following schema, prepare 50 questions about sales value and volume growth for the different sales channels product categories brands and customer regions, the set of questions aims to help the data analyst analyse the performance and find the key insights to be used in decision making along the key parts that need quick actions like.
> Create table Customers (...)
> Create table Products (...)
> Create table Customer_Channels (...)

Chat GPT3.5 answer was:

Sales Growth Analysis:

- What is the overall sales growth trend over the past year?
- Are specific sales channels experiencing consistent growth?
- Which product categories have shown the highest sales growth in the last quarter?
- How have sales for a particular brand evolved over the last six months?

.....

Volume Growth Analysis:

- What is the trend in product volume growth for each sales channel?
- How does the volume growth in specific product categories compare to the overall trend?
- Are there notable differences in volume growth by brand?

….

The actual question generation procedure utilizes the outlines given by chat GPT, then applies the following idea that the author came up with to be able to generate a wide range of questions in a short time based on the author's experience and the GPT 3.5 outlines given earlier with the sample question shown beforehand. The idea starts with building question templates and then via Excel formulas or code generating many variants of the same question based on the different parameter changes or via rephrasing the question or using the aliases of the words/conditions. This will result in the quick generation of different questions.

Then based on the schema given, the same question will have a different SQL query that uses the fields and table(s) and joins with the relevant tables to come up with a Query or more that answers the question(s) the user asked for.

Question template samples are shown below:

- What is the trend in {Analysis Level} { Query } for each {Hierarchy query level} in {Date hierarchy}?

  For example:

  - What is the trend in **<u>product</u>** **<u>volume growth</u>** for each **<u>sales channel</u>** in **<u>2024</u>**?
    {Analysis Level}: product
    { Query }: volume growth
    {Hierarchy query level}: sales channel
    {Date hierarchy}: 2024
  - What is the trend in **<u>category</u>** **<u>sales value growth</u>** for each **<u>region</u>**?
    {Analysis Level}: Category
    { Query }: Sales value growth
    {Hierarchy query level}: region

- What is the {Hierarchy query level} with the {Condition} {Query} in {Date hierarchy}?
-
  - What is the **<u>city</u>** with the **<u>highest</u>** **<u>number of orders</u>** made in **<u>2024</u>**?

- During {date hierarchy}, show the trend of the {Query} in {region_hierarchy_condition} for {product_hierarchy_condition} products.

  - During **<u>Q3 2023</u>**, show the trend of the **<u>sales value</u>** in the **<u>UK</u>** for **<u>Hair care</u>** products.

This way many variants of the questions using the parameters could be generated, and the same parameters set could be used in different questions with paraphrasing/aliases.

The table presented below displays the split of the 1K business questions according to their complexity levels as either easy, medium, or hard. Hard questions encompass formulas and keywords such as "growth rate" and "rolling average," while medium-difficulty questions involve joins between more than two tables and incorporate grouping operations. The evaluation of complexity uses the question and the answer query that references the schema given, so the same question could be easier to answer on a simple one-table reporting schema compared to another operational schema with many tables.
The remaining questions are categorized as easy.

Table 1. New 1K business-related questions, split per complexity.

Complexity/DB	**MSSQL**	**MySQL**	**Total Count**	**Questions%**
Easy	530	35	565	**54%**
Medium	104	65	314	**30%**
Hard	255	59	169	**16%**
Total Count	**889**	**159**	**1048**	

2.5. Performance Metrics

The evaluation methodology relies on Rouge N-grams [20] with n=1, specifically chosen for assessing the similarity of SQL queries. This approach is well-suited for comparing queries due to their specific syntax and known keywords, including table and column names that must align with a predefined ground truth or gold answer. However, the complexity arises from potential variations of queries, such as the use of column aliases or different DB functions for date parts extraction. The evaluation serves as an indicator, recognizing differences in syntax while evaluating word-by-word similarity in a meaningful way. Additionally comparing the resulted SQL queries based on counting the number of joins, group by, and conditions against the golden answer would give an indicator that the resulted query is missing some of the essential components for the answer, and clarify the area of improvement required in the fine-tuning dataset to cover this component, like adding more join or aggregation examples.

Moreover, comparing the resulted records after query execution on the database and comparing the outcome, is often used in other research, but it was avoided here due to potential limitations like handling column name aliases and data sorting variations in the results which may lead to misleading results.

Instead, Rouge N-gram metrics, precision, recall, and F1 score are employed, complemented by syntax correctness checks against SQL Server and MySQL. Rouge-N Recall holds particular significance in this experiment as it evaluates the similarity between n-grams in the answer query and those in the reference gold answer. This metric is calculated by evaluating the proportion of overlapping n-grams in the answer to the total number of n-grams present in the reference gold answer. Furthermore, the analysis involves tallying the primary query keywords present in both the answer and the gold answer to identify errors within a specified threshold.

$$\textbf{Rouge N Recall} = \frac{\textbf{Number of Overlapping N} - \textbf{grams}}{\textbf{Total Number of N} - \textbf{grams in the gold answer}}$$

3. EXPERIMENTAL RESULTS AND DISCUSSIONS

All answers resulting from the queries were validated against both the Microsoft SQL Server and MySQL databases to ensure syntax accuracy and table/column correctness against the schema. Table 2 below presents the results of the model experimental findings, specifically emphasizing the count of queries with valid syntax that were executable on either Microsoft SQL Server or MySQL database. For consistency in evaluation, identical contextual schemas were established on both databases to assess a test set comprising 57 questions, employing both zero-shot and fine-tuning methodologies. The base models underwent zero-shot testing, while the fine-tuned models were evaluated using the same question set initially used for zero-shot testing.

Table 2. The experiment results per model

Model/Results	**QA Count**	**Total Valid structure SQL**		**MSSQL Valid SQL**		**My SQL Valid SQL**		**Rouge N=1 Precision Avg**	**Rouge N=1m Recall Avg**	**Rouge N=1 F1 Avg**
Zero-shot sqlcoder2	57	25	44%	5	9%	25	44%	0.37	0.39	0.37
Fine-tuned csqlcoder2 (With new 1K QA)	**57**	**42**	**74%**	**27**	**47%**	**37**	**65%**	**0.47**	**0.60**	**0.49**
Zero-shot Llama-2-13B-instruct-text2sql	**57**	**42**	**74%**	**19**	**33%**	**40**	**70%**	**0.42**	**0.33**	**0.35**
Fine-tuned Llama-2-13B-instruct-text2sql (With new 1K QA)	57	39	68%	28	49%	33	58%	0.44	0.54	0.46
Zero-shot Llama-2-7b-chat-hf	57	40	70%	20	35%	39	68%	0.36	0.36	0.32
Fine-tuned Llama-2-7b (10k QAC Without new 1K QA)	57	18	32%	0	0%	18	32%	0.20	0.45	0.26
Fine-tuned Llama-2-7b (10k QAC With new 1K QA)	57	27	47%	21	37%	20	35%	0.37	0.56	0.42
Fine-tuned Llama-2-7b (30k QAC With new 1K QA)	57	35	61%	23	40%	26	46%	0.40	0.54	0.45
Zero-shot T5-LM-Large-text2sql-spider	57	3	5%	0	0%	3	5%	0.36	0.25	0.28

Initially, it is important to discern the distinction between valid syntax and a correct outcome. Consider the following SQL query: it exhibits syntactic correctness by featuring accurate table and column names, executing successfully within the database environment. However, this query lacks a crucial condition specifying the year 2023. Consequently, while the syntax is valid by SQL grammar, the query is logically flawed due to the absence of this critical condition. As a result, the query will retrieve sales data for the specified store across all years, rather than solely for the year 2023, rendering the obtained results incorrect and misleading in terms of the intended query objective. You can find below an illustration example that a missing condition resulted in an incorrect answer, however, it has a correct syntax and can run on the database successfully and return a list of values. Table 3 illustrates the correct answers in the different models and the average rouge N1 recall matric value. Moreover, the possible reasons for the incorrect answer are detailed in Table 4.

Question: **What is the revenue of Store#10 in 2023?**

Answer Query:

```
Select sum(Sales_Value) as total_sales
from Store_Sales
where storeId = 10;
```

Comment: The query possesses valid syntax; however, it constitutes an incorrect answer as it lacks the condition "year = 2023."

Table 3. How many are correct answers and what is the percentage from the total samples 57 test samples?

Model	**Correct answer**		**Rouge N1 recall Avg**
Zero-shot sqlcoder2	7	12%	0.81
Fine-tuned csqlcoder2 (With new 1K QA)	**31**	**54%**	**0.85**
Zero-shot Llama-2-13B-instruct-text2sql	4	7%	0.56
Fine-tuned Llama-2-13B-instruct-text2sql (With new 1K QA)	27	47%	0.89
Zero-shot Llama-2-7b-chat-hf	6	11%	0.55
Fine-tuned Llama-2-7b (10k QAC Without new 1K QA)	0	0%	
Fine-tuned Llama-2-7b (10k QAC With new 1K QA)	16	28%	0.93
Fine-tuned Llama-2-7b (30k QAC With new 1K QA)	16	28%	0.94

Table 4. The potential sources of error in the generated query. This method compares the count of the SQL keywords (join, group by, where order by) in the answer versus the count in the gold answer and reports any mismatch.

Model/Results	**Valid structure queries**	**Mismatch "Join"**	**Mismatch "Group By"**	**Mismatch "Where"**	**Mismatch "Order By"**
Zero-shot Sqlcoder2	25	0	12	3	10
Fine-tuned Sqlcoder2 (With new 1K QA)	42	0	5	4	7
Zero-shot Llama-2-13B-instruct-text2sql	42	3	18	8	18
Fine-tuned Llama-2-13B-instruct-text2sql (With new 1K QA)	39	1	4	2	15
Zero-shot Llama-2-7b-chat-hf	40	3	21	22	27
Fine-tuned Llama-2-7b (10k QAC Without new 1K QA)	18	0	1	3	5

Fine-tuned Llama-2-7b (10k QAC With new 1K QA)	27	2	1	4	8
Fine-tuned Llama-2-7b (30k QAC With new 1K QA)	35	0	4	1	6
Zero-shot T5-LM-Large-text2sql-spider	3	1	0	0	2

Next, we will discuss each model's results in detail.

3.1. Lama2 13B – Instruct – Text2SQL Model

This model is derived from the LIAMA CODE Model, originally designed for general code synthesis and comprehension, and subsequently fine-tuned using the “bugdaryan/sql-create-context-instruction” dataset, which itself was curated from data sourced from WikiSQL and Spider datasets. During fine-tuning, the model was prompted to encapsulate SQL query answers within specified <sql> </sql> tags, a task it accomplished efficiently compared to alternative models. The base model underwent an initial zero-shot evaluation followed by fine-tuning using a new set of 1,000 business-related Question-Answer-Context (QAC) pairs and subsequently re-evaluated using the same question set. Notably, the test questions were novel to the model, with roughly half of them based on schemas familiar to the model and the remaining half introducing new schemas. Initially, the base model achieved 74% validity in zero-shot testing; however, only 7% of the total test sample produced correct answers (specifically, 4 queries). Post-fine-tuning, the percentage of valid queries decreased to 68%, yet the proportion of correct answers surged to 47%, with 25 queries correctly addressing the questions and 2 queries providing partial responses. Additionally, Rouge N1 recall increased after fine-tuning, indicative of enhanced inclusion of relevant columns, tables, and keywords aligning with the reference "gold" answers. In summary, the introduction of the new dataset notably improved the overall model's accuracy in correctly responding to the test set.

Error analysis:

Following the extraction of generated queries and verification of their validity against either the SQL Server or MySQL databases, an additional layer of validation was implemented by comparing the SQL keywords with those present in the reference "gold" answers. Specifically, counts of keywords such as "Join," "Group by," "Order By," and "Where" were tallied in both the gold answer and the resultant query. Any discrepancies in keyword counts between the two were identified as mismatches.

After the fine-tuning process, significant improvements were observed in the occurrences of "group by" and "where" mismatches, decreasing from 18 issues in the base model to only 4 issues post-fine-tuning. Specifically, mismatches in the "where" component of the fine-tuned answers reduced from 8 to 2. Furthermore, the number of Microsoft SQL Server (MSSQL) valid syntax queries increased to 28 queries post-fine-tuning, whereas valid syntax queries for MySQL decreased from 40 to 33 queries.

One observation was seen in that the base model shows inconsistency in applying the correct date condition, so the model is not capable of interpreting the correct date condition each time it was asked for data in the year 2023, the examples below illustrate this scenario. After finetuning this behaviour becomes more consistent, therefore, the FT model gives the correct answers for both questions below.

Question:

What is the number of shipments made in 2023 from Egypt to Algeria?

Base Model Answer:

```
SELECT COUNT(*) FROM shipments
WHERE From_Country = 'Egypt'
AND To_Country = 'Algeria'
AND shipment_date = '2023-01-01'
```

Comment:

This red condition limits the result to one day not a full year as requested in the question. It is a valid syntax but an incorrect answer.

Question:

Show how many shipments were made in 2023 for each shipment status.

Base Model Answer:

```
SELECT status, COUNT(*)
FROM shipments
WHERE YEAR(shipment_date) = 2023
GROUP BY status
```

Comment:

This example is a correct answer for the same condition of the year 2023. This shows inconsistency across the different answers.

3.2. SQL Coder 2 Model

Utilizing zero-shot prompting with the base SQL Coder 2 model initially yielded SQL queries with correct syntax in 44% of test cases, equating to 25 valid answers, of which only 9% were executable on an SQL Server database. Conversely, all 25 queries were compatible with MySQL databases. Following fine-tuning over 10 epochs using a new dataset of 1,000 business-related Question-Answer-Context (QAC) pairs, the number of structurally valid query answers increased from 25 to 42, representing 74% of the test set. The Rouge N1 recall measure notably improved to 0.6 compared to 0.39 in the base model, both are referencing the gold answer. Notably, as the fine-tuned dataset primarily comprised MSSQL queries, the number of valid MSSQL queries surged by 39%, rising from 5 in the base model to 27 in the fine-tuned (FT) model. Additionally, the count of valid MySQL queries escalated from 25 in the base model to 37 in the FT model.

It is important to emphasize that valid syntax does not necessarily equate to logical correctness matching the gold standard answer. The average Rouge N1 Recall of 0.6 indicates some missing keywords such as "group by," "where," "join," and "order by," as well as potential omissions of column or table names or aliases. Further investigation into this aspect is warranted. Fine-tuning notably boosted the percentage of valid MSSQL database queries to 47%, compared to 9% in the base model. Initially, the base model produced correct answers in 12% of test cases (7 queries) using zero-shot prompting, which significantly increased to 47% (31 queries) following fine-tuning with the new 1,000-business-related dataset. These findings represent a promising improvement in achieving reliable correct answer rates, positioning the fine-tuned SQL Coder model as the most successful outcome within this experimental framework.

Error analysis:

In the SQL Coder model, approximately half of the queries generated by the base model exhibited Group by mismatches, totalling 12 out of 25 queries. However, after fine-tuning, the incidence of Group by issues decreased significantly from 12 to 5 out of 42 valid structured queries in the fine-tuned (FT) model. Concerning the "where" keyword, which serves as a filter condition in queries, 3 out of the 25 total valid queries produced by the base model encountered issues. No issues were observed with the "Join" keyword. Regarding the "Order By" keyword, 10 queries initially exhibited issues, which decreased to 7 queries after fine-tuning the model.
Shown below are two illustration examples for the base model answer and the fine-tuned model answer.

Question:

> What is the promotion revenue of Store#45 in the period between 01/01/2014 and 31/12/2014? show the results per quarter.

Base Model Answer:

```
SELECT CAST(Sales_Value AS FLOAT) / 1000 AS reve-nue_per_quarter,
CAST(Sales_Date AS DATE) AS sales_date
FROM Store_Sales
WHERE StoreId = 45 AND Sales_Date BETWEEN '2014-01-01' AND '2014-12-31'
ORDER BY CAST(Sales_Date AS DATE) ASC;
```

Comment:

> It is a valid syntax query(Query keywords, column, and table names) but an incorrect answer, it doesn't sum the revenue or group by quarter or limits the data to the days that have a promo.

FT Model Answer:

```
SELECT DATEPART(QUARTER, Sales_Date) AS Quarter, SUM(Sales_Value) AS
Total_Sales_Value
FROM Store_Sales
WHERE Has_Promo = 1 AND StoreId = 45
AND Sales_Date >= '2014-01-01' AND Sales_Date <= '2014-12-31'
GROUP BY DATEPART(QUARTER, Sales_Date)
ORDER BY Total_Sales_Value DESC;
```

Comment:

> Valid syntax and correct answer where all conditions are correctly added to the query.

3.3. LIAMA 2 7B Model and its Fine-Tuned Variants

Initially, the base LIAMA 2 – 7B model performed unexpectedly well with zero-shot prompting, yielding 70% of valid structured queries, despite prior uncertainty regarding the model's training on SQL-related datasets. The Llama-2-7b-chat-hf variant demonstrated proficient knowledge of

SQL, successfully constructing queries with correct syntax incorporating operations such as union, case when grouped by, and the ability to interpret date expressions like "H1 2023" to represent the first half of the year. However, this capability exhibited inconsistency, as the model occasion-ally misapplied date ranges even when employing similar statements like "H1 2023." Furthermore, over half of these queries were executable on MySQL databases, whereas approximately half could run on MSSQL platforms, highlighting a promising foundation for subsequent fine-tuning efforts. Despite these strengths, the correctness of answers provided by the model for the test questions was relatively low, with only 11% deemed accurate. Nevertheless, this performance suggests the potential for improvement through fine-tuning, particularly if the syntax and language nuances are well understood and addressed.

Following fine-tuning with 10,000-row pairs sourced from the Spider and WIKISQL datasets, the number of valid queries decreased from 40 in the base model to 18 after this refinement process, with all resulting queries deemed incorrect. Surprisingly, the Rouge N1 recall metric exhibited a slight increase from 0.36 to 0.45. Conversely, when the model underwent fine-tuning using a dataset comprising 1,000 business-related questions supplemented by 9,000 rows from Spider and Wiki SQL, the count of valid syntax queries rose to 27. Although this number is lower than the base model's initial results, it represents an improvement compared to the fine-tuned model without the inclusion of the 1,000 business questions. The fine-tuned model produced correct answers for 16 queries, accounting for 28% of the test set answers.

Subsequently, further fine-tuning with an expanded dataset of 30,000 rows, inclusive of the 1,000 business questions and additional Spider/Wiki SQL data, did not lead to an increase in the number of correct answers, which remained unchanged. Similarly, the Rouge N1 recall metric remained consistent. However, the count of valid syntax queries increased to 35, compared to 27 in the previous fine-tuned model using only 10,000 rows. Nevertheless, this figure remains lower than the base model's zero-shot syntax valid responses. These findings suggest that augmenting the fine-tuning dataset beyond a certain threshold offers limited improvements to the model's performance.

Shown below is an example of a good answer by the base model with zero shot, which other models failed to answer.

Question:

What is the monthly growth rate in 2023 versus 2022 in the total billing volume?

Base Model Answer:

```
SELECT SUM(Case When Doc_Date between '2023-01-01' and '2023-12-31' THEN
Billing_Doc.Doc_Value ELSE 0 END) /SUM(Case When Doc_Date between '2022-01-
01' and '2022-12-31' THEN Billing_Doc.Doc_Value ELSE 0 END) * 100 AS
Monthly_Growth_Rate
FROM Billing_Doc WHERE Doc_Date between '2023-01-01' and '2023-12-31'
```

Comment:

Valid syntax and correct answer.

3.4. T5-LM-Large-Text2sql-Spider

Despite being fine-tuned on the Spider dataset, the T5 base model exhibited the ability to answer only three valid structured queries, none of which yielded correct answers. Consequently, this model was not selected for further fine-tuning due to its limited performance in generating accurate query responses.

3.5. GPT 3.5 samples checking

During the development of the 1,000-question dataset, GPT-3.5 was utilized in conjunction with the LIAMA Index and SQL Alchemy Python frameworks to generate an initial draft of SQL query answers that required subsequent refinement and validation against the database engine. The accuracy of these responses achieved a rate of 37% when employing zero-shot prompting, which closely corresponds to the performance observed in small and medium-sized LLMs tested in this study post-fine-tuning.

In scenarios testing for negative outcomes, such as when a queried column is absent (e.g., querying for products in a schema lacking product information), the model tends to construct a query using the missing column as an alias to produce a result, rather than acknowledging the absence of relevant data within the schema. This behaviour was observed across all models in our experiment, including GPT-3.5.

3.6. An Example of the Results from the Top-Performing Models

Question:

What is the total quantity and weight of products exported from Egypt to France in 2023?

Reference golden answer:

```
Select sum(Quantity_In_Cartons) total_quantity,
sum(Quantity_In_Cartons * Unit_weight) as total_weight
From Product_Export
Where From_Country='Egypt' and To_Country = 'France' and Year(Export_date) = 2023
```

Table 5. An example of the resulting queries from the different top models

Model	The answer SQL query	Evaluation	Reason

Answer by sqlcoder2 Zero-shot	SELECT SUM(Quantity_In_Cartons) AS total_quantity,
	SUM(Unit_weight) AS total_weight
	FROM Product_export
	WHERE From_Country = 'Egypt' AND To_Country = 'France' AND Export_date BETWEEN '2023-01-01' AND '2023-12-31';
		Incorrect	Incorrect total weight (should be multiplied by the quantity in carton)

Answer by Fine-tuned csqlcoder2 (With new 1K QA)	select sum(Quantity_In_Cartons) as Total_Quantity, sum(Unit_weight) as Total_Weight from Product_export where From_Country = 'Egypt' and To_Country = 'France' and year(Export_date) = 2023 Group by From_Country, To_Country	Incorrect	Incorrect total weight, unnecessary group by.

Answer by Llama-2-13B-instruct-text2sql Zero-shot	SELECT SUM(Quantity_In_Cartons) AS
	Total_Quantity, SUM(Unit_weight) AS Total_Weight
	FROM Product_Export
	WHERE From_Country = 'Egypt' c
	AND To_Country = 'France'
	AND Export_date = '2023-01-01'
		Incorrect	Incorrect total weight and date condition.

Answer by FT Llama-2-13B-instruct-text2sql (With new 1K QA)	SELECT SUM(Quantity_In_Cartons),
	SUM(Unit_weight*Quantity_In_Cartons)
	FROM Product_Export
	WHERE From_Country = 'Egypt'
	AND To_Country = 'France'
	AND year(Export_date) = 2023
	ORDER BY sender_company ASC;
		Correct	

Answer by Llama-2-7b-chat-hf Zero-shot	SELECT SUM(Quantity_In_Cartons) as
	Total_Quantity, SUM(Unit_weight *
	Quantity_In_Cartons) as Total_Weight
	FROM Product_Export
	WHERE Export_date = '2023-01-01' AND
	From_Country = 'Egypt' AND To_Country = 'France'
	GROUP BY 1
		Incorrect	Incorrect date condition, incorrect group by

4. CONCLUSION

The experimental findings reveal promising outcomes, exemplified by the fine-tuned SQL Coder 2 achieving a 54% correct answer rate and the Fine-tuned LIAMA 2 13 B Instruct-Text2SQL model attaining a 47% correct answer rate. These results suggest potential avenues for further enhancement through additional fine-tuning iterations. Nonetheless, despite these advancements, the current correct answer rates fall short of meeting the expectations of business users, primarily due to the pivotal role of data in facilitating decision-making processes. It is imperative to note that while queries may yield data, inaccuracies in the conditions employed can render the overall result incorrect. Addressing these challenges necessitates the involvement of knowledgeable users in a pilot phase, coupled with iterative fine-tuning procedures.

The utilization of a fine-tuning dataset comprising 1K pairs of question, answer, and context demonstrates a notable enhancement in the performance of the top two models. Such improvements offer valuable implications for organizations across diverse industries, facilitating the development of tailored fine-tuning datasets aligned with their specific business requirements, schemas, database types, and interests. It is essential to underscore that this process is iterative, with ongoing evaluation and fine-tuning serving to augment the efficacy and robustness of these models over time.

While the initial progress is indeed promising, it is imperative to acknowledge the multitude of subsequent steps outlined in the roadmap. Primarily, further domain-specific fine-tuning is warranted, particularly within sectors such as sales and supply chain management, necessitating a model equipped with comprehensive financial acumen to comprehend intricate concepts like profit and loss, taxation, and formulaic conditions. Additionally, the development of security-conscious models emerges as a critical imperative, encompassing measures such as user authorization protocols to ensure data access is strictly aligned with authorized permissions. This entails the implementation of row-level security mechanisms, integration of user-centric information into queries, and adaptation of security modules tailored for seamless deployment within enterprise environments.

Moreover, the challenges associated with deploying a local model on a private cloud entail the responsibility of resource management as organizational demand escalates, including considerations regarding the scalability of cost-effective models. Addressing how to synchronize the base model with the latest updates from the market version, while also balancing the necessity of re-conducting fine-tuning on new data to accommodate newly introduced scenarios and customer feedback, alongside managing the frequency of updates and tracking the evolution of scenarios covered, intensifies the maintenance pressure to ensure relevance.

Additionally, optimising query performance on extensive datasets poses an additional challenge. It extends beyond mere query correctness to encompass the generation of optimized queries that leverage table indices and minimize the retrieval of unnecessary data. This aspect warrants further scrutiny and targeted attention to enhance efficiency.

Additional studies are required to evaluate the language model's ability to interpret the question along with the resultant data following the execution of SQL queries on the database, to prepare the final user answer that explains the results in addition to returning the rows.

Emphasizing the significance of taking into account ethical and legal implications during the deployment of machine learning models is crucial. Key concerns encompass biases inherent in training data, which may result in discriminatory outcomes. Privacy is another crucial area, with sensitive personal data that may be handled in queries necessitating robust protection measures. Additionally, misuse of models, such as generating queries that breach privacy or ethical guidelines, must be vigilantly monitored and addressed. Strategies such as bias detection, privacy-preserving techniques, and responsible use guidelines are essential components to integrate into the model development and deployment lifecycle. Ethical oversight and stakeholder engagement are vital to ensure these models serve organizational interests responsibly.

REFERENCES

[1] Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., & Hashimoto, T.B. (2023). “Stanford Alpaca: An instruction-following LLaMA model. “ https://github.com/tatsu-lab/stanford_alpaca

[2] Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., & Zhou, J. (2023). “Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. “ ArXiv. /abs/2308.15363

[3] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., . . . Amodei, D. (2020). “Language Models are Few-Shot Learners. “ ArXiv. /abs/2005.14165

[4] OpenAI. “ChatGPT.” Chat.openai.com, OpenAI, 2023, chat.openai.com/chat

[5] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., . . . Zoph, B. (2023). “GPT-4 Technical Report. “ ArXiv. /abs/2303.08774.

[6] Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J. H., Shafey, L. E., Huang, Y., Mishra, G., Moreira, E., Omernick, M., Robinson, K., Ruder, S., . . . Wu, Y. (2023). “PaLM 2 Technical Report. “ ArXiv. /abs/2305.10403

[7] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., . . . Zaremba, W. (2021). “Evaluating Large Language Models Trained on Code. “ ArXiv. /abs/2107.03374

[8] Liu, A., Hu, X., Wen, L., & Yu, P. S. (2023). “A comprehensive evaluation of ChatGPT's zero-shot Text-to-SQL capability. “ ArXiv. /abs/2303.13547

[9] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., . . . Scialom, T. (2023). “Llama 2: Open Foundation and Fine-Tuned Chat Models. “ ArXiv. /abs/2307.09288

[10] Defog, Inc. (2023). “Defog SQLCoder. “ https://huggingface.co/defog/sqlcoder-7b

[11] Meta. (2023). “Code-Llama-2-13B-Instruct-Text2sql.“ https://huggingface.co/bugdaryan/Code-Llama-2-13B-instruct-text2sql

[12] Gauss Algorithmic. (2023). “T5-LM-Large-Text2sql-Spider. “ https://huggingface.co/gaussalgo/T5-LM-Large-text2sql-spider

[13] Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., & Hajishirzi, H. (2022). Self-Instruct: Aligning Language Models with Self-Generated Instructions. ArXiv. /abs/2212.10560 Dettmers, Tim, et al. “QLoRA: Efficient Finetuning of Quantized LLMs.” ArXiv.org, 23 May 2023, arxiv.org/abs/2305.14314.

[14] B-Mc2. (2023). Dataset “Sql-Create-Context“ https://huggingface.co/datasets/b-mc2/sql-create-context

[15] huggingface.co (2023) “Supervised Fine-Tuning Trainer.” https://huggingface.co/docs/trl/main/en/sft_trainer

[16] FlorianKnauer, Will Cukierski. (2015). Rossmann Store Sales. Kaggle. https://kaggle.com/competitions/rossmann-store-sales

[17] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., & Fedus, W. (2022). “Emergent Abilities of Large Language Models. “ ArXiv. /abs/2206.07682 “Modal.” Modal.com, modal.com/. Accessed 23 Dec. 2023.

[18] liu, Jerry. (2023) “Finetuning LLaMa + Text-To-SQL.” LlamaIndex https://github.com/run-llama/modal_finetune_sql

[19] Awan, A. A. (2023). “Fine-tuning llama 2: A step-by-step guide to customizing the large language model.“ DataCamp. https://www.datacamp.com/tutorial/fine-tuning-llama-2

[20] Lin, C.-Y. (2004). “ROUGE: A Package for Automatic Evaluation of Summaries. “ Aclanthology.org. https://aclanthology.org/W04-1013/

AUTHORS

Rasha Ragab Ahmed , IT Business Partner and Integration lead at Unilever, just earned my MSc in AI from the University of Leeds UK, and I have an Engineering bachelor’s degree in Computer and Control from Port Said University. www.linkedin.com/in/rasharagab21

Dr. Abdulrahman Altahhan – Coauthor, Senior Teaching Fellow in Artificial Intelligence, school of computing at the university of Leeds. Developing, leading and managing postgraduate and undergraduate computer science HE programs in the areas of AI and Data Science.

© 2024 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.