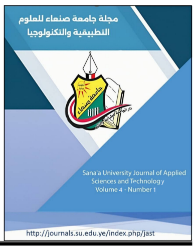
Vol.4, No.1, Page 1507- 1519, 2026 https://doi.org/10.59628/jast.v4i1.2248 ISSN 2958-9568 JAST

مجلة جامعة صنعاء للعلوم التطبيقية والتكنولوجيا
Sana'a University Journal of Applied Sciences and Technology
https://journals.su.edu.ye/index.php/jast/



Enabling Arabic Database Querying via Parameter-Efficient Fine-Tuning of Large Language Models

Mohammed Taj[1], Mohammed Zayed[2] *, Abdulwahid Alhetar[1], Mohammed Rajeh[1], Mohammed Abbas Al-Sharafi[1] and Basem Abdulrhman Munassar[1]

[1]**Department of Information System, Faculty of Computer and Information Technology, Sana'a University, Sana'a, Yemen,**
[2]**Department of Computer Science, Faculty of Computer and Information Technology, Sana'a University, Sana'a, Yemen**

*Corresponding author: mzayed @ su.edu.ye

Abstract

Recent advancements in Natural Language Processing (NLP) and Text-to-SQL systems have enabled easier interaction with relational databases. However, most solutions focus on English, leaving Arabic underrepresented. This study addresses that gap by fine-tuning the Llama-3-SQLCoder model to convert Arabic text into correct and executable SQL, enabling non-technical users to work with databases without learning SQL syntax. We enhanced the model using Low-Rank Adaptation (LoRA) and Unsloth, training it on Arabic questions paired with SQL queries from the Northwind database. To support low-resource environments, the model was converted to the GGUF format, reducing computational requirements while preserving performance. Evaluation results showed an execution accuracy of 90.24% and a validity rate of 97.56%, outperforming the zero-shot baseline (44% and 80%). The model also achieved an Exact Match score of 32% and an F1 score of 0.83, compared to 12% and 0.61 for the baseline. These findings demonstrate that LoRA and Unsloth are effective for adapting SQL-specialized models to Arabic. Despite these improvements, the system still struggles with complex nested queries and dialectal variations, indicating areas for future work. Overall, this study contributes to narrowing the gap between Arabic and other languages in Text-to-SQL research and improves database accessibility for non-technical users.

Article Info

Keywords:
Text-to-SQL, Large Language Models (LLMs), Natural Language Processing (NLP), Parameter-Efficient Fine-Tuning (PEFT), Low-Rank Adaptation (LoRA), GGUF Quantization, Unsloth

Article History:
Received: 14-September-2025,
Revised: 11-October-2025,
Accepted: 28-December-2025,
Available online: 29 January 2026.

1. INTRODUCTION

The use of computers in natural languages has long been a significant objective of artificial intelligence[1]. Recent advances in Natural Language Processing (NLP) and the intersection of computer science, AI, and linguistics have made this objective more realizable [1]. NLP enables computers to comprehend, understand, and meaningfully produce natural language [1]. One of the significant contributors to recent advancements is the introduction of large-language models (LLMs). These deep learning models are trained using a vast amount of text data. Models of this type found within the GPT and Llama series have shown remarkable abilities in tasks such as translation, summarizing, deep reasoning, and code generation [2]. The most useful application of these technologies is database connection. Relational databases contain significant amounts of data pertaining to businesses, research, and daily life. Accessing data is often prohibitive unless one has knowledge of special query phrases such as a Structured Query Language (SQL) [3]. This places a hindrance before analysts, researchers, and managers need information, but possess no programming ability. Text-to-SQL technology bridges this hole by converting natural language questions like أي الموظفين حقق أعلى مبيعات في الربع الأخير؟ and others into exact SQL commands capable of being executed [4]. The development of the text-to-SQL domain has progressed rapidly and is largely due to advances made within the space of large language models (LLMs) [2]. Initial attempts used rule-based approaches or small neural networks [5]. Unlike previous attempts, new models are pre-trained on vast amounts of text and code before being tuned toward the output of the SQL [5]. The current work has focused on increasing the accuracy, complex query handling, and processing of loose phrasing and adaptability toward varying structural databases [4]. For example, beyond simple retrievals such as اعرض جميع أسماء الفئات our model can handle more complex queries like من هو الموظف الذي حقق أعلى مبيعات في الربع الأخير؟ ,which requires joining multiple tables (Employees and Orders), applying aggregation, and filtering by time. Despite these developments, most of these works largely accommodate English only and leave a huge gap in Arabic support. Arabic poses unique challenges in NLP because of its rich morphology, diacritic variations, clitic-based tokenization, and right-to-left writing systems. These characteristics often cause ambiguity and data sparsity when training machine-learning models. In this study, we address these challenges through tailored preprocessing (normalization, diacritic removal, and stop-word filtering) and parameter-efficient fine-tuning (LoRA and Unsloth), enabling the model to effectively handle Arabic inputs. This study aims to fill this gap by fine-tuning the Llama-3-SQLCoder model [6] to produce valid and executable SQL statements from Arabic texts. Llama-3-SQLCoder was selected as the foundational model in preference to alternatives such as GPT or BERT. BERT, which is primarily an encoder-based architecture, is less suitable for generative tasks, such as SQL query generation. By contrast, Llama-3-SQLCoder, which is derived from the Llama-3 family, is a decoder model optimized for text generation and further specialized for database-related tasks. Unlike the GPT series, the Llama-3 models are openly accessible, which enhances the transparency, reproducibility, and opportunities for customization. In particular, the SQLCoder variant provides prior knowledge of the SQL syntax and database concepts [6], giving our study a distinct advantage. Rather than training a general-purpose multilingual model to acquire SQL expertise from the ground-up, we adapted an existing model already specialized in SQL to handle the additional complexity of Arabic. Ethical Compliance: All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards. Ethical Compliance: All procedures performed in this study involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards. Conflict of Interest Declaration: The authors declare that they have no affiliations with or involvement in any organization or entity with any financial interest in the subject matter or materials discussed in this manuscript. Author Contributions: AB and MJ contributed to the design and implementation of the research, JK to the analysis of the results and to the writing of the manuscript. VK conceived the original and supervised the project. Through the use of contemporary fine-tuning [7] methods, such as LoRA [8] and Unsloth [9], and training on a corpus of Arabic questions parallel to SQL queries, this work is proposed to help non-technical Arabic speakers interact highly and fruitfully with relational databases and fill the linguistic gap within the text-to-SQL domain.

2. RELATED WORK

The text-to-SQL field aims to convert natural language questions into structured query language (SQL) commands. This has made significant strides over the years. This process reflects large changes in Natural Language Processing (NLP). The focus has shifted from rule-based systems to statistical models, and now to large-scale deep learning methods. However, this progress has been predominantly centered on English, creating a significant performance gap for other languages, particularly Arabic.

2.1. The Evolution of Text-to-SQL and the English-Dominance Paradigm

Early attempts at text-to-SQL were mainly rule-based and relied on semantic parsing [10]. These systems use crafted grammatical rules and lexicons to connect linguistic constructs to the SQL components. Although they were understandable, they were fragile and required extensive expertise. They also struggle with ambiguity and variety of natural languages. The introduction of dataset benchmarks such as WikiSQL [11] and Spider [12] prompted a shift toward machine learning. This resulted in encoder-decoder models with attention mechanisms [13] and specialized architectures such as SQLNet [14] . that learned to encode the natural language question and database schema before decoding them into an SQL query. These models were more robust than their rule-based predecessors but often required task-specific architectures and struggled with the complexity and generalization required for large, multi-table databases. The contemporary revolution in Text-to-SQL is inextricably linked to the rise of Large Language Models (LLMs)

[15]. Models such as OpenAI's GPT series, Meta's Llama series, and specialized variants such as SQL-Coder and Codex, pretrained on vast corpora of text and code, have rendered many earlier task-specific systems obsolete. The prevailing method, known as in-context learning, involves priming a powerful, general-purpose LLM with a database schema and a few question-SQL pairs (few-shot learning) within its context window to generate the desired query [5]. demonstrated outstanding performance on complex English benchmarks such as Spider [12]. A critical limitation persists across nearly all top-performing text-to-SQL systems: a profound bias towards English. The training data for leading LLMs are overwhelmingly English, as are the primary benchmarking datasets [16]. This creates an obstacle for nonEnglish speakers. Although multilingual LLMs such as mT5 [17] and BLOOM [18] are available, their performance on structured tasks, such as SQL generation for low-resource languages, often falls short compared to their English-based models. The specific challenges of the Arabic language, which include its complex structure, right-to-left writing, and various dialects, worsen this issue worse [13]. Directly translating an English-based query for an Arabic question often results in syntax mistakes, schema mismatches, and poor results, because the model's internal understanding of SQL is linked to English schema elements and phrasing.

2.2. The Specific Challenges of Arabic Text-to-SQL

The complexities of using text-to-SQL in Arabic are multifaceted and move beyond translation at a simple level; therefore, the direct use of English-based models is ineffective. The unique barriers include the following. Morphological Complexity: Arabic is a highly inflected and morphologically rich language featuring a rich system of roots, patterns, and affixes, which results in a high out-of-vocabulary rate and tokenization difficulty [19]. Dialectal Differences: The fact that there is a difference between Modern Standard Arabic (MSA), used in written formal contexts, and the different regional dialects (like Egyptian, Levantine, and Gulf) used in daily interactions poses a significant challenge. A model trained only on MSA might not be able to understand questions expressed in a dialect [16]. Right-to-Left (RTL) Script: The right-to-left nature of a script requires special handling in tokenization and model design, which is typically left-to-right language tailored [19]. Data Availability: One of the significant deficiencies is the expansive, high-quality Arabic text-to-SQL datasets available for effective training and thorough evaluation [4]. The direct translation of an Arabic question and the use of an English-centric model often results in syntactic errors, schema mismatch, and suboptimal performance as a result of the model's internal understanding of SQL, which does not match the linguistic configuration inherent in Arabic.

2.3. Prior Work on Arabic and Multilingual Text-to-SQL

Recent studies have addressed this issue. The development of multilingual datasets, such as the Arabic parts of Spider's counterpart, BIRD-SQL [20], provides crucial resources for training and evaluation. Furthermore, parameter-efficient fine-tuning techniques, such as Low-Rank Adaptation (LoRA) [8], have emerged as effective methods to adjust large models for new tasks and languages without the high cost of complete fine-tuning. These methods allow for effective instruction tuning on smaller language-specific datasets. Some studies have spurred efforts to handle the Arabic text-to-SQL task utilizing different types of approaches, as listed in Table 1. Almohaimeed et al. (2024) [4]developed Ar-Spider, an Arabic adaptation of the Spider dataset, and assessed multiple models that primarily focused on English using a translation method and a fundamental fine-tuning technique. Their research revealed a decline in performance when addressing Arabic language tasks directly, and emphasized the necessity for specialized models. Chafik et al. (2025) [16] presented Dialect2SQL as a new set containing Moroccan Darija, with a special interest in overcoming the challenge that dialectal variations represent and that mainly go unnoticed by models trained only on Modern Standard Arabic (MSA). Their analysis reveals that the challenge runs deeper than MSA, reflecting the rich linguistic diversity within the Arab world.Cross-lingual Transfer Learning: Studies like that of Shi et al. (2022) [5] explore cross-lingual semantic parsing with techniques like representation mixup, showing promise for transferring knowledge from high-resource languages (English) to low-resource ones. However, their focus is often general and not specifically optimized for the structural complexities of the SQL generation in Arabic.

2.4. Research Gap and Our Contribution

As listed in Table 1, previous studies have primarily focused on dataset design [4] or illustrating the shortcomings of direct translation. Much literature can be identified as sparse on the effective use of mature, SQL-specialized LLMs that have been fine-tuned especially on Arabic with current and computationally effective fine-tuning paradigms. The current study directly addressed this requirement. We built on these advances by starting with a powerful SQL-aware base model, Llama-3-SQLCoder-8B [6], and systematically adapted it to Arabic. In contrast to earlier studies, we applied a selective PEFT approach with LoRA [8] and the Unsloth framework [9] on a hand-curated Arabic-to-SQL benchmark collection. This allowed us to adequately pair the inter-

Table 1. Summary of Key Studies in Arabic and Multilingual Text-to-SQL

Study (Year)	Methodology	Dataset Used	Key Results / Focus	Identified Limitations
Almohaimeed et al. (2024) [4]	Created Ar-Spider dataset; Evaluated T5-Small & mT5 with fine tuning.	Arabic translation of Spider	Highlighted significant performance drop of models on Arabic compared to English. Demonstrated the need for language-specific resources.	Models used were not state-of-the-art SQL-specialist LLMs. Fine-tuning was basic, not leveraging modern PEFT like LoRA.
Chafik et al. (2025) [7]	Introduced a new dataset for dialectal Arabic; Preliminary model evaluation.	Dialect2SQL	Emphasized the challenge of dialectal variations, a major hurdle for practical deployment in the Arab world.	Focus was on dataset creation; model performance and adaptation strategies were not the primary contribution.
Shi et al. (2022) [6]	Cross-lingual semantic parsing using representation mixup.	Multiple languages from a semantic parsing benchmark.	Showed that cross-lingual transfer from English can be effective for related tasks.	Not specifically focused on Text-to-SQL; does not address the integration with database schemas or complex SQL queries directly.
Our Work	PEFT (LoRA) & Unsloth fine-tuning of a SQL-specialist LLM (Llama-3-SQLCoder).	Curated Arabic Questions + SQL schema.	Directly adapts a SOTA SQL model to Arabic, achieving high execution accuracy (90.24%) and validity (97.56%).	Limited by dataset size; performance on complex nested queries and broad dialectal variations requires further study.

nal SQL-based representations of the model with Arabic linguistic idiosyncrasies without requiring massive computational power or corpora. Our approach demonstrates a concerted effort towards shifting from presenting the problem to having a high-performance, usable solution for Arabic text-to-SQL.

3. MATERIALS AND METHODS

This section presents the proposed methodology for developing an Arabic-to-SQL system, which involves designing a framework, applying fine-tuning techniques to large language models (LLMs) to improve performance, and converting the model to a GGUF format to ensure efficient performance in settings with limited resources [3].

3.1. Framework Overview

The proposed framework follows a pipeline that processes user inputs in Arabic and generates accurate SQL statements for execution in relational databases. The framework comprises the following stages Figure 1:

1. Input Layer (Arabic Question): The process begins with the user submitting a question in Arabic, such as من هو الموظف الذي حقق أعلى مبيعات في الربع الأخير؟ This question is a natural-language request to be translated into SQL [21].

2. Preprocessing: The input was cleaned and normalized to handle challenges specific to the Arabic language, such as diacritics and tokenization. The stop words were removed to ensure a better understanding of the model.

3. LLM (Llama-3-SQLCoder-8B Arabic → SQL Generator): The preprocessed text is passed to the fine-tuned Llama-3-SQLCoder-8B model, which is trained to generate SQL queries directly from Arabic inputs. This component is the core of the system and leverages fine-tuning with LoRA and Unsloth [8] to improve the performance of Arabic-specific data.

4. SQL Guard + Role-Based Access Control: Before execution, the generated SQL query passes through a safeguard layer that prevents malicious queries. The

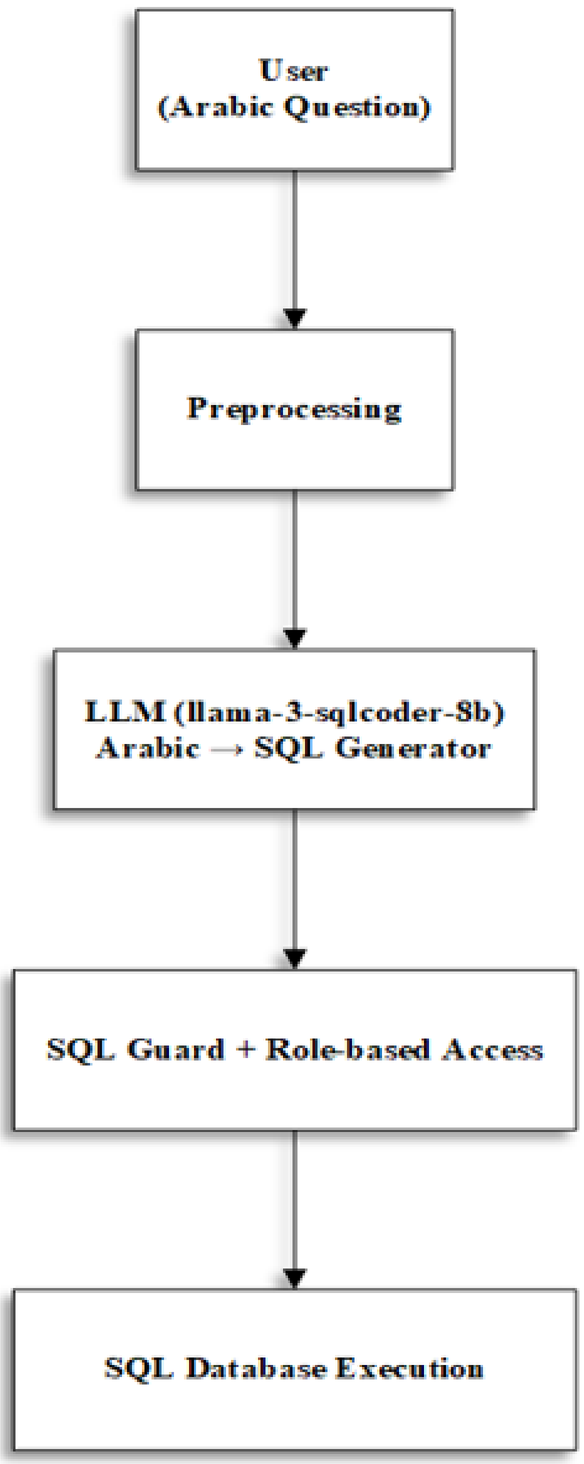


Figure 1. The proposed Arabic-to-SQL system framework.

SQL Guard Filters out potentially harmful operations (e.g., DROP TABLE and DELETE without conditions). Role-based Access Control (RBAC): This ensures that queries respect the user privileges.

5. SQL Database Execution: The validated SQL query is executed against the target relational database. The retrieved results are returned to the user in a structured and readable format.

3.2. Fine-Tuning Strategy

The goal is to create a specialized model that understands natural-language questions in Arabic. To accomplish this cross-lingual transfer [5] efficiently, we used parameter-efficient fine-tuning techniques, specifically Low-Rank Adaptation (LoRA) [8], to tailor the powerful SQL-proficient base model for the Arabic language without the cost of full fine-tuning [7].

3.2.1. Model Selection: Why Llama-3-SQLCoder-8B?

We selected llama-3-sqlcoder-8b for Pre-existing SQL Proficiency: This model is not a general-purpose LLM. This is a version of Meta's Llama 3, which has been fine-tuned using a large collection of SQL-related text and code. This specialized training provides a better understanding of database concepts, schema relationships, and SQL syntax than the base model. This provided a strong foundation for our task [6]. Strong Base Model (Llama 3): Built on the solid Llama 3 8 B parameter foundation [9], which provides a good balance between performance and computational requirements. This makes it possible to fine-tune a single GPU [9], such as Tesla T4 used in the notebook. Large Context Window (8k): The 8192 token context length is essential for this task. The database schemas (CREATE TABLE Statements) can be lengthy. We must provide both the full schema and the user's question to the model within its context window, which this model handles effortlessly [12]. Multilingual Capabilities: While primarily trained on English SQL data, large modern LLMs such as Llama-3 exhibit surprising emergent multilingual abilities [22]. Our fine-tuning process activates and specializes in Arabic. In short, we started with a "SQL expert" and taught it to understand "Arabic," which is more efficient than taking an "Arabic expert" and trying to teach it "SQL". Alternatives Considered Before making our decision, we examined different alternatives: Multilingual encoder-decoder (mT5, mT0, AraT5): good for multilingual comprehension but without SQL-specialized pre-knowledge, leading to weaker schema creation. Large multilingual models (e.g., BLOOM) support good Arabic coverage, but have high computational costs and no SQL specialization. Code-specialized models (CodeGen, StarCoder, and CodeT5+) possess the ability to code language, but not long natural-language-to-schema translation. Closed-source models (e.g., GPT-4) are highly performing, but not suited for open fine-tuning or on-premise deployment. Considering these parameters, Llama-3-SQLCoder-8B has emerged as the optimal choice, possessing a good balance between SQL proficiency, computational speed, multilingual versatility, and open availability.

3.2.2. Fine-Tuning Techniques: LoRA and Unsloth

To efficiently adapt the large Llama-3-SQLCoder-8B model to the Arabic text-to-SQL task, we used Low-Rank Adaptation (LoRA) [8]. The use of modern optimization techniques (AdamW 8-bit) follows the trend of applying advanced optimization methods to complex problems. Similar to how Al-Hegami and Bin-Ghodel [23] employed Particle Swarm Optimization for medical image classification, our approach leveraged state-of-the-art optimization to efficiently adapt large-language models for specialized tasks. Instead of fine-tuning all eight billion parameters of the model, which would take too long and use too much memory, LoRA introduces a small number of trainable parameters called adapters along with the original fixed model weights [8]. During fine-tuning, only the adapters were updated. This significantly reduces computational demands while maintaining the expressive power of the base model. This method has several advantages. First, it significantly lowers the memory requirement, requiring only approximately 7.4 GB of VRAM compared with over 80 GB for full fine-tuning. Second, it accelerates the training because only a small portion of the parameters are adjusted. Finally, it improves portability. The output is a lightweight adapter file of only a few megabytes, which makes it easy to share and apply to the base model. This is in contrast to the multi-gigabyte checkpoints produced by full fine-tuning [24] In this study, we adopted a configuration with a rank (r) of 16, focusing on the key linear layers of the model, specifically the query, key, value, output, and feed-forward layers [8]. This standard setup effectively balances efficiency and performance. Unsloth is a library that offers optimized versions of popular large language models (LLMs) and their training loops [9]. It enables training that is twice as fast and uses 80% less memory by using custom Triton kernels and techniques such as fused operations [9]. Its main benefits are speed, which greatly reduces the experiment time, cost, and accessibility, as it allows fine-tuning of large models on consumer-grade GPUs, such as the free T4 on Google Colab, while also supporting larger batch sizes. Moreover, Unsloth has a built-in tool for easily converting fine-tuned models into the GGUF format for use with llama.cpp [25], meeting an important project requirement.

3.2.3. Dataset

The dataset was essential for teaching the model how to map Arabic questions to SQL queries. It came from a CSV file and had three main columns: arabicQ, which contained natural language questions in Arabic; schema, which included the CREATE TABLE statements that defined the database structure; and sql, which represented the target SQL queries that answered the questions. During preprocessing, we removed any rows with missing values and renamed the columns in the standard Alpaca format as instruction, input, and output [26]. This ensured compatibility with the training framework. The notebook shows 127 training steps with a batch size of eight, suggesting that the dataset contains approximately 50 examples based on the filename. Although this is sufficient as a proof of concept, a larger dataset would likely improve the strength of the model. The sample questions, like اعرض جميع أسماء الفئات, are written in Modern Standard Arabic (MSA), the usual choice for technical and written tasks [19]. The database schema is based on a standard 'Northwind' structure with tables such as Products, Suppliers, and Categories. Northwinds were chosen because of their status as a popular reference point for SQL-related academic studies, with a comprehensive but simple schema that covers normal business operations, such as managing products, dealing with suppliers, and customer orders. These aspects correspond significantly with the patterns of organizing real-world data, making it suitable for testing a text-to-SQL model's ability to handle real database scenarios. In addition, adopting a standardized schema such as Northwinds ensures consistency and reproducibility in corresponding research efforts.

3.2.4. Preprocessing & Prompt Engineering

The most important step in guiding the behavior of a model is to create a custom prompt template. This template, written in English, arranged the input so that the base model could understand the instructional tasks well [26]. It defines a task as converting a natural language question into a precise T-SQL query. It sets rules for using aliases, casting ratios to float, and responding with "I do not know" for questions that could not be answered. The input was structured by clearly separating the "Question" and the "Schema." It controls the output by requiring only SQL code to be returned within a markdown code block. A tokenizer from the original llama-3-sqlcoder model was used, which handled Arabic subword tokens effectively without any changes. This study focuses mainly on Arabic text-to-SQL, which requires additional preprocessing steps to compensate for the characteristic features present in Arabic text. First, normalization was performed in order to unify multiple orthographic forms (such as transforming , , and to , in addition to the removal of diacritics) and transformed variants of hamza and taa marbouta in the same manner [19]. Subsequently, tokenization was applied using the tokenizer of the original Llama-3-SQLCoder model, allowing for subword-level segmentation of the Arabic language [16]. However, due to Arabic's high level of morphological complexity and its common use of attached clitics (such as و+، ب+، ل+ , a light stemming process was used to reduce words to their root form while preserving their respective meanings [19]. Finally, in an effort to reduce the differences between Modern Standard Arabic (MSA) and conversational Arabic, MSA queries were only used during the training process, acknowledging that dialect management is an enduring problem for future research projects [16]. A key step in preparing the data was to add the End-of-Sequence <|eot_id|> token to each example during the formatting process to signal the end of the training input.

The core prompt template structure was defined as follows:

prompt_template = """### Instructions:

Your task is to convert a natural language question into a precise T-SQL query.

Input:

Generate a SQL query that answers the question: {question}.

This query runs on a database whose schema is represented in this string: {schema}.

Response:

{sql}

"""

3.2.5. Training Setup & GGUF Conversion

Training was performed on a free Tesla T4 GPU using Google Colab with 16GB VRAM. This demonstrates the accessibility of the fine-tuning method. The setup used the AdamW 8-bit optimizer to save memory [27], with a learning rate of 2e-4, an effective batch size of 8 (two per device with four gradient accumulation steps), and a sequence length of 8192 to maximize the capacity of the model. The training lasted for 127 steps and took approximately 22 minutes. The loss curve exhibits a smooth and rapid decline, indicating successful learning. After training, the model was converted to the GGUF format [25] This is a quantized model format designed for efficient use on consumer hardware, such as CPUs and Apple Silicon Macs, with tools like llama.cpp. The LoRA adapters were integrated back into the base model using the unsloth save_pretrained_gguf() function [8]. The full model was first converted to 16-bit GGUF (f16) format and then quantized to q4_k_m format [7]. This quantization reduces the model size from approximately 16GB to 4.6GB while maintaining most of its accuracy. The final output was an "unsloth.Q4_K_M.gguf" file, ready for high-performance local inference without requiring a GPU [9], which successfully combined a carefully selected pretrained model, Llama-3-SQLCoder, with the effective fine-tuning techniques, LoRA [8]and Unsloth [9], on a small but targeted Arabic-to-SQL dataset. The key to its performance is thoughtful prompt engineering [26] to help the model understand its tasks and output formats. The final model was packaged in an efficient GGUF format, making it a practical solution to convert Arabic questions into SQL queries.

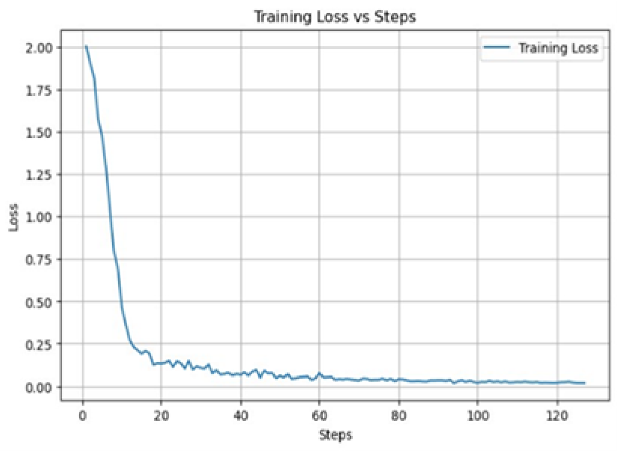


Figure 2. Training Loss vs Steps

4. DISCUSSION AND RESULTS

In this section, we present the results of experiments conducted to evaluate the proposed model. The results were divided into three stages. First, we present the training results and gradual decrease in the loss function. This demonstrated the stability of the fine-tuning process. Next, we outline the evaluation metrics used and the quantitative performance results of the test set [4]. Finally, we provide qualitative examples that demonstrate the model's ability to convert Arabic questions into valid SQL queries.

4.1. Training Results and Convergence Analysis

The fine-tuning process using LoRA [8] and Unsloth [9] demonstrated highly efficient convergence, with the training loss decreasing from 2.0046 to 0.0196 over 127 steps (7.94 epochs), as summarized in Table 2. This represents a 99% reduction in loss, confirming the effectiveness of our parameter-efficient approach for adapting the SQL-specialized model to Arabic.

Table 2. Cumulative outcome of the training procedure

Metric	Value
Initial Loss	2.0046
Final Loss	0.0196
Total Steps	127
Total Epochs	7.94

4.1.1. Training Dynamics and Learning Phases

The training loss curve in Figure 2 indicates three learning phases that offer an understanding of the adaptation process of the model [8].

The first phase (steps 1-20) indicates exponential decay, suggesting high-speed acquisition of core Arabic-to-SQL mapping. This is followed by a refinement phase (steps 20-80) in which the model acquires more subtle linguistic patterns, and lastly, the convergence phase (steps 80-127) in which the stabilization of optimization takes place.

4.1.2. Epoch-wise Convergence Pattern

The epoch-based perspective in Figure 3 validates persistent learning throughout the training epochs, with loss reducing from approximately 2.0 down to less than 0.2 during the first three epochs. This swift early phase progress demonstrates the model's ability to promptly adapt its SQL information to the Arabic language structure [27].

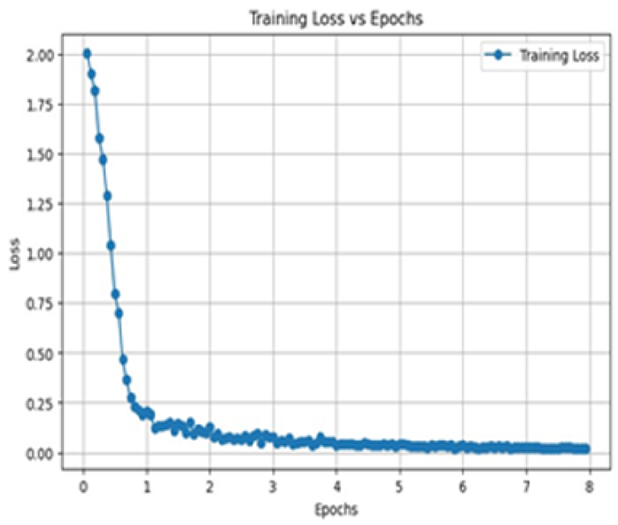


Figure 3. training loss vs epochs

4.1.3. Optimization Stability

The schedule of the learning rate depicted in Figure 4 uses a controlled decay mechanism that allows steady convergence. The high learning rate initially enabled extensive exploration of the landscape loss, whereas the progressive gentle reduction favored accurate adjustment of parameters during subsequent phases, thus avoiding overshooting and ensuring effective optimization.

4.1.4. Smoothed Training Trajectory

The analysis of the moving averages in Figure 5 validates the learning trend at a deeper level by eliminating the batch-level noise. The monotonic smooth decline between the starting and ending loss values is a strong indication of stability-based learning without oscillations or instability at a prominent level [27].

The overall convergence, achieved in a span of only 22 min on a single T4 GPU, also validates the efficient practicality of combining LoRA and Unsloth for the cross-lingual adaptation of large language models [8]. The absence of oscillations owing to overfitting validates that the model learned generalized patterns rather than just memorizing the training dataset, thus providing a strong foundation for subsequent assessment.

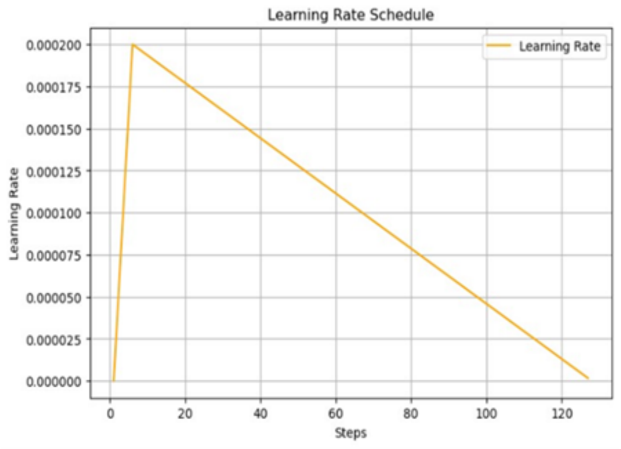


Figure 4. learning rate schedule

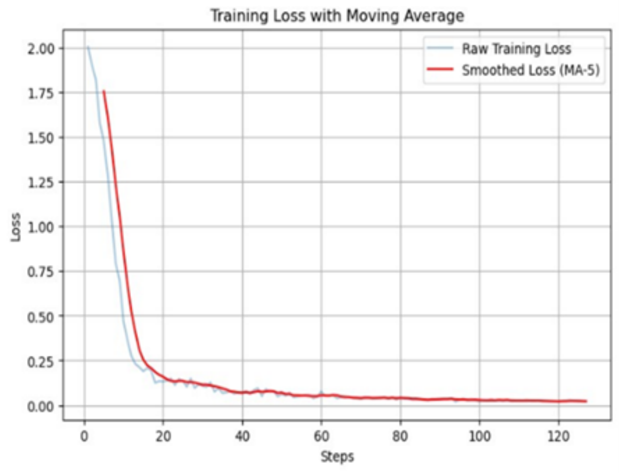


Figure 5. training loss with moving average

4.2. Evaluation Setup

After the optimization, we conducted a controlled experiment to evaluate the performance of the Arabic text-to-SQL model. The reason for this assessment was to verify the capability of the model to generalize beyond the training dataset and to convert Arabic NL queries to SQL queries consistently [4]. We used an 82-question test dataset to conduct this assessment. To analyze the performance of the model in more detail, questions were constructed to span a large number of SQL query types. The following SQL query types were covered: Simple Selection Queries, which are simple retrievals using the SELECT statement; Aggregation Queries, which are functions such as COUNT and SUM; Conditional

Queries, which are constraints specified by the WHERE clause; and Join Queries, which are complex queries that use JOIN to retrieve from multiple tables [9]. We employed the Northwind schema for all experiments. Several tables are mutually related in this schema, for example, Products, Orders, Customers, Employees, Suppliers, and Categories. This is a widely used benchmark schema designed to test queries with varying levels of complexity [12]. To conduct the inference task, we employed a computer with an Intel i5 processor coupled with 8 GB RAM. To avoid employing the unquantized model, we employed a quantized model in the GGUF (q4_k_m) format. This computer setup demonstrates that the model functions efficiently even in environments that are constrained in terms of resources and does not impose any special hardware compatible with GPUs. For comparison, we also added another model, called Base Llama-3-SQLCoder-8B (llama-3-sqlcoder-8b. Q5_K_ M · gguf), respectively. The assessment was conducted directly on Arabic queries without fine-tuning [24].

4.3. Evaluation Metrics

To evaluate the effectiveness of the developed Arabic text-to-SQL system, five widely used evaluation metrics were deployed [9] These metrics are used to test the accuracy, syntactical validity, and computing efficiency of the system. Execution Accuracy (EX) evaluates the percentage of SQL queries that execute correctly against the specified database in producing the right output [7], thus revealing the working correctness of the system. The validity rate represents the percentage of generated SQL queries that are syntactically valid and executable without fault [9]. An exact Match (EM) represents the percentage of generated queries that perfectly match gold-standard SQL queries, representing a stringent test of correctness. The F1 Score examines the token-level intersection and union of the predicted and gold-standard queries and evaluates the measure of both partial correctness and competency in query generation that is both structurally and semantically correct. Response Time measures the average time in seconds taken for the model to translate natural language prompts to SQL queries [12], thus revealing computing efficiency and practicality in real-time applications or resource-prohibitive use cases. Together, these metrics provide an all-round understanding of the model performance in terms of accuracy, robustness, syntactic correctness, and efficiency [9].

4.4. Quantitative Results

A quantitative evaluation was performed to compare the performance of the proposed fine-tuned model with that of a baseline system. This baseline was the base Llama-3-SQLCoder-8B model, which was applied in a zero-shot setting to Arabic inputs [28]. The results are summarized in Table 3, where the optimized model shows a remarkable improvement in all measured metrics. The execution accuracy is twice that of the zero-shot baseline, signifying heightened functional correctness. A high validity rate indicates robust competence in generating syntactically correct questions. Token-level examination reveals more: even if only 32% of the queries have exact matches, the 0.83 F1 reinforces the observation that answered queries largely preserve their primary structure and semantic content. This implies that the model understands the query logic even if it is off by a small amount. Finally, the shorter response time points towards an effective trade-off between accuracy and computational cost, rendering it practical in resource-poor environments. Overall, these outcomes justify the efficacy of fine-tuning through LoRA [8] and Unsloth [9] for fine-tuning Llama-3-SQLCoder for Arabic, and achieving significant improvements in reliability, structure preservation, and accuracy.

4.5. Qualitative Examples

In addition to the quantitative results, we conducted a qualitative evaluation to show how well the proposed model translated Arabic natural-language queries into SQL. Table 4 presents representative examples across different query categories, including simple retrieval, aggregation, and join operations.

The examples show that the fine-tuned model can accurately generate simple queries (e.g., retrieve all category names), produce aggregation queries that correctly compute sales or counts, and handle join queries across multiple tables to obtain semantically correct results. This study further confirmed the ability of the system to be applied to various questions. This shows that it can be useful for Arabic-speaking users who do not possess SQL skills.

4.6. Comparative Analysis and Discussion

The experimental results show that our carefully tuned Llama-3-SQLCoder-8B model achieves a substantial and consistent performance improvement from the zero-shot baseline. This section provides an in-depth analysis of these results, explains the metrics to reveal the model's advantages and shortcomings, and places our work in the broader context of the community.

4.6.1. Interpreting Performance Through Multi-Metric Analysis

The quantitative findings presented in Table 3 provide a complex representation of the model's performance. The principal achievement is evident: the optimized model significantly surpasses the Execution Accuracy (EX) of the foundational model, achieving 90.24%, in contrast to 44%. This outcome illustrates that our LoRA-based modi-

Table 3. cumulative outcome of the training procedure

Model	Execution Accuracy (EX)	Validity Rate	Exact Match (EM)	F1 Score	Avg. Response Time (sec)
Fine-tuned Llama-3-SQLCoder-8B (Arabic)	90.24%	97.56%	32%	0.83	37.05
Base Llama-3-SQLCoder-8B (Zero-Shot Arabic)	44%	80%	12%	0.61	65.09

Table 4. Examples across different query categories

Arabic Question	Generated SQL Query	Execution Result
اعرض جميع أسماء الفئات مع وصفها؟	`SELECT c.CategoryName, c.Description FROM Categories c;`	8 rows (e.g., *Beverages, Condiments, ...*)
من هو الموظف الذي حقق أعلى مبيعات في الربع الأخير؟	`SELECT TOP 1 e.EmployeeID, e.FirstName, e.LastName, SUM(o.Freight) AS TotalSales FROM Employees e JOIN Orders o ON e.EmployeeID = o.EmployeeID WHERE o.OrderDate BETWEEN '1997-10-01' AND '1997-12-31' GROUP BY e.EmployeeID, e.FirstName, e.LastName ORDER BY TotalSales DESC;`	*EmployeeID = 4, Janet Leverling, TotalSales = 2047.93*
كم عدد الموردين الذين يوفرون منتجات من فئة المشروبات؟	`SELECT COUNT(DISTINCT s.SupplierID) FROM Suppliers s JOIN Products p ON s.SupplierID = p.SupplierID JOIN Categories c ON p.CategoryID = c.CategoryID WHERE c.CategoryName = 'Beverages';`	8 suppliers

fication effectively addressed linguistic disparity, allowing a model initially trained on English SQL data to competently comprehend and accurately respond to language inquiries in Arabic.

The addition of Exact Match (EM) and F1 scores provides a critical and deep understanding. The significant disparity in the high EX (90.24%) and low EM (32%) highlights an essential element of the model's operation: it frequently creates queries that are syntactically dissimilar yet semantically similar, which still results in the correct outcome. For instance, the model uses a different column structure, varying table alias, or a logically equivalent WHERE clause in lieu of JOIN. This effect should not be thought of in terms of deficiency; more so, it is an expression of robustness, as it indicates that the model understood the root principles of query creation and not mere pattern retention.

This reading was strongly supported by a high F1 Score of 0.83. This metric confirms that the fine-tuning process significantly improved the model's ability to match Arabic concepts with the right SQL elements the correct choice of columns, tables, and conditions. Although the final query string does not exactly match the gold standard (which is why the EM is lower), its key elements are largely correct, leading to the same correct and precise result.

4.6.2. Positioning within the Arabic NLP and Text-to-SQL Landscape

Although direct comparisons are difficult because of the lack of widely comparable Arabic text-to-SQL models, our approach should be contextualized alongside alternatives. vs. Multilingual LLMs (mT5, BLOOM): Our model differs from applying a large, general-purpose multilingual model in a zero-shot setting. Models such as mT5 [17] and BLOOM [18] lack inherent specialization in SQL. Our study demonstrates that specializing in a powerful SQL-proficient model (Llama-3-SQLCoder) for a target language via fine-tuning is a more effective strategy than relying on the emergent SQL capabilities of a broader multilingual model. Compared to Arabic-Specific Pre-trained Language Models, such as AraBERT [19], our method is vastly different. Although these models are customized to understand the Arabic language, they lack internal knowledge of SQL or programming. Thus, if these had to be customized for text-to-SQL tasks, they would have involved significant training from scratch. In contrast, our model leverages a strong SQL base and efficiently adapts it to the Arabic language, an approach that has proven to be highly effective and economical in terms of the resources used. Thus, this study fills a distinct and promising gap. It is a hybrid method that uses state-of-the-art SQL-savvy LLMs and brings them within reach for Arabic via parameter-efficient fine tuning.

4.6.3. A Critical Analysis of Limitations

Regardless of the robust performance of the model, the measurements and qualitative analysis indicated shortcomings. Complexity Barrier: This rather low Exact Match rate is partly due to the model's difficulty in dealing with highly complex queries, specifically those involving nested subqueries. Although it is proficient in dealing with joins and aggregates, as in Table 4, queries that involve NOT EXISTS or nested SELECT statements sometimes trigger syntactic errors or logical simplifications that, if runnable at all, do not conform to the gold standard. This is a direct consequence of the extremely small and highly constrained complexity of the fine-tuning set. Dialectal Arabic and Robustness: Another significant and unchecked limitation is that performance is not in dialectal Arabic (Ammiyya) despite our training and test sets being in Modern Standard Arabic (MSA) form. Performance should severely suffer with dialectal input due to variability in word choice and syntax, and would negatively impact all the metrics, but specifically the component-level understanding that is measured in terms of F1. Computational Efficiency Trade-off: Although the average inference time of 37.05 seconds, despite being better than the benchmark and taken on an average consumer-grade CPU, is too long for applications requiring interactivity, it is essentially a trade-off in which we sacrificed real-time processing throughput for accuracy and accessibility on limited hardware. Finally, the optimized model represents an important milestone in the Arabic text-to-SQL domain. Incorporating strong Execution Accuracy, a high F1 Score, and a high validity rate affirms the productivity of our adaptation approach. However, its performance shortcomings are clearly defined by the challenges in processing high-complexity queries, dialect variations, and the need for fast inference, which presents a critical path for future work.

4.7. Limitations and Future Work

Despite our tuned model's excellent performance, predominantly beating the zero-shot baseline for all metric measurements, this research includes multiple deficiencies that describe an unmistakable and feasible paradigm for subsequent research.

4.7.1. Dataset Scale and Complexity Constraints

The principal limitation of this study is the scale and intricacy of the dataset used for fine-tuning and assessment. The capacity of the model for generalization was inherently limited by approximately 50 training instances and 82 test queries. This restricted scale is the most plausible rationale for the notable contrast between the elevated Execution Accuracy (90.24%) and the diminished Exact Match rate (32%). The model has demonstrated the ability to generate semantically accurate queries that produce correct outcomes. However, it has not encountered a sufficient number of examples to reliably reproduce the exact syntactic format of gold-standard queries. This shortcoming is particularly apparent in the processing of complex nested queries that are inadequately represented in the training dataset. Future Work: Of primary interest is the drastic increase in the size of the training set. Our goal is to fine-tune and test the model based on wide, publicly available Arabic text-to-SQL test sets such as ArSpider [4]. Furthermore, we seek to implement curriculum learning techniques, whereby the model is first exposed to simple queries and then to more complex structures that involve subqueries and set operations.

4.7.2. Dialectical Arabic and Real-World Robustness

This research was conducted only in Modern Standard Arabic (MSA). One significant yet unresolved issue is the model's performance when dealing with dialectal Arabic (Ammiyya) spoken in practical interactions throughout the Arab world. The need for accurate MSA syntax and vocabulary, revealed in its performance in the test on level-component matching (F1), would mean that it would noticeably suffer in performance when dealing with the divergent vocabulary, grammatical elements, and scripts that are seen in dialects such as Egyptian or Levantine. This remains and continues to remain a primary deterrent for practical applications. Future Work: We plan to incorporate dialectal sets systematically into the training process. This work will involve using resources such as Dialect2SQL [16] data and exploring efficient fine-tuning techniques, such as multidialect fine-tuning or incorporating a dialect-to-MSA transformer layer as an initial processing step, to enhance the resilience of the model to linguistic variability.

4.7.3. Computational Efficiency for Interactive Applications

37.05 seconds average inference time, measured on a resource-limited CPU and revealing an improvement over the non-optimized baseline, is too high for interactive real-time applications. This was chosen intentionally in favor of high accuracy and validity levels for environments lacking GPUs. Future Work: This study intends to investigate methods for achieving inference times below one second. This work includes the analysis of more demanding model quantization methods (such as the q4_k representation used in GGUF), working with dedicated inference optimization tools such as ONNX Runtime, and compiling the model onto platforms that use modern GPUs, thereby markedly reducing the computation time for real-world applications.

4.7.4. *Cross-Domain Generalization*

The model was fine-tuned and evaluated using only the norwind database schema. Its zero-shot generalizability to unknown database schemas from other domains (such as medical, banking, or e-commerce) is unchecked and limited to all probabilities. This "cross-domain" problem is an acknowledged text-to-SQL problem. Future Work: In the future, we will systematically test the model in multiple domains using benchmarks such as BIRD-SQL [20]. In addition, we will investigate methods such as schema-agnostic encoding and instruction-tuning on multi-domain sets towards enhance the model's capacity to generalize to a new database architecture without wholesale re-training. By addressing these challenges, it is contended that the performance, resilience, and practicality of Arabic text-to-SQL systems would notably increase, and therefore, in the long term, eradicate the accessibility gap for Arabic speakers in the case of data engagement. Future work could explore hybrid optimization approaches, combining techniques such as Particle Swarm Optimization [27] with modern gradient-based methods to further improve training efficiency and model performance across different domains.

5. CONCLUSION

In this study, an effective Arabic text-to-SQL approach is described using parameter-efficient Llama-3-SQLCoder-8B model fine-tuning with LoRA and Unsloth. Putting forward is an effective path toward the adaptation of SQL-specialized large language models for low-resource languages and thus fills an important gap in Arabic data interaction tools. The results of the experimental study demonstrated the reliability of the proposed methodology. The optimized model markedly surpassed the zero-shot baseline, attaining an execution accuracy of 90.24% and validity rate of 97.56%. An assessment using an Exact Match and F1 Score validates that the model generates SQL queries that are both semantically accurate and syntactically appropriate. Notably, the quantized model was effectively implemented in an environment reliant solely on CPU, underscoring its practicality for application in resource-limited contexts, including small enterprises, educational establishments, and governmental agencies, where technical proficiency may be restricted. The model is versatile in that it can support a broad range of query structures, from simple retrievals to sophisticated joins and aggregations. Thus, its characteristics give it significant value in decision-support systems, portals for business intelligence, and learning systems. Furthermore, in aiding Arabic speakers in their interactions with relational databases in the absence of SQL savvy, the system manifests pronounced practical value and potential for deployment in real-time. Despite these successes, several key issues remain to be resolved. It is necessary to expand the model to accommodate dialectal Arabic to provide fair access for all Arabic users. Furthermore, it is necessary to increase the training data to larger, more diverse benchmarks, and more sophisticated query patterns to achieve production-level robustness. Future work should entail creating larger sets of Arabic text-to-SQL, exploring cross-dialect and cross-lingual transfer learning methods, and applying the model to deployable real-time query generation scenarios. With these avenues, this work opens the groundwork for enhancing the accessibility of tabular data to Arabic speakers and, hence, an inclusive and efficient avenue towards data-informed decision-making.

REFERENCES

[1] e. a. Rayhan, “Natural language processing: Transforming how machines understand human language,” in *Proceedings of the Conference on the Development of Artificial General Intelligence*, 2023.

[2] D. Gao et al., “Text-to-sql empowered by large language models: A benchmark evaluation,” *Proc. VLDB Endow.*, vol. 17, no. 5, pp. 1132–1145, Jan. 2024. DOI: 10.14778/3641204.3641221.

[3] X. Zhu, Q. Li, L. Cui, and Y. Liu, *Large language model enhanced text-to-sql generation: A survey*, arXiv preprint, Oct. 2024. DOI: 10.48550/arxiv.2410.06011.

[4] S. Almohaimeed, S. Almohaimeed, M. Ghanim, and L. Wang, *Ar-spider: Text-to-sql in arabic*, arXiv preprint, Feb. 2024. DOI: 10.48550/arxiv.2402.15012.

[5] P. Shi et al., “Cross-lingual text-to-sql semantic parsing with representation mixup,” in *Findings of the Association for Computational Linguistics: EMNLP 2022*, Association for Computational Linguistics, 2022, pp. 5296–5306. DOI: 10.18653/v1/2022.findings-emnlp.388.

[6] A. Mohammadjafari, A. Maida, and R. Gottumukkala, *From natural language to sql: Review of llm-based text-to-sql systems*, arXiv preprint, Oct. 2024. DOI: 10.48550/arxiv.2410.01066.

[7] A. Agrahari, A. Gautam, P. K. Ojha, and P. Singh, *Sft for improved text-to-sql translation*, MDPI Preprints, Feb. 2024. DOI: 10.20944/preprints202402.0693.v1.

[8] E. Hu et al., *Lora: Low-rank adaptation of large language models*, arXiv preprint, Jun. 2021. DOI: 10.48550/arxiv.2106.09685.

[9] A. Holtzman, A. Pagnoni, L. Zettlemoyer, and T. Dettmers, *Qlora: Efficient finetuning of quantized llms*, arXiv preprint, May 2023. DOI: 10.48550/arxiv.2305.14314.

[10] T. Shi, K. Tatwawadi, K. Chakrabarti, Y. Mao, O. Polozov, and W. Chen, *Incsql: Training incremental text-to-sql parsers with non-deterministic oracles*, arXiv preprint, Sep. 2018. DOI: 10.48550/arxiv.1809.05054.

[11] V. Zhong, C. Xiong, and R. Socher, *Seq2sql: Generating structured queries from natural language using reinforcement learning*, arXiv preprint, Aug. 2017. DOI: 10.48550/arxiv.1709.00103.

[12] T. Yu et al., “Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Jan. 2018. DOI: 10.18653/v1/d18-1425.

[13] A. Al-Hegami, “A framework for incremental mining of interesting temporal association rules,” *Int. J. Comput. Appl.*, vol. 131, no. 8, pp. 28–33, Dec. 2015. DOI: 10.5120/ijca2015907433.

[14] X. Xu, C. Liu, and D. Song, *Sqlnet: Generating structured queries from natural language without reinforcement learning*, arXiv preprint, Nov. 2017. DOI: 10.48550/arxiv.1711.04436.

[15] A. Vaswani et al., *Attention is all you need*, arXiv preprint, Jun. 2017. DOI: 10.48550/arxiv.1706.03762.

[16] S. Chafik, S. Ezzini, and I. Berrada, *Dialect2sql: A novel text-to-sql dataset for arabic dialects with a focus on moroccan darija*, arXiv preprint, Jan. 2025. DOI: 10.48550/arxiv.2501.11498.

[17] L. Xue et al., *Mt5: A massively multilingual pre-trained text-to-text transformer*, arXiv preprint, Oct. 2020. DOI: 10.48550/arxiv.2010.11934.

[18] E. Manjavacas et al., *Bloom: A 176b-parameter open-access multilingual language model*, arXiv preprint, Nov. 2022. DOI: 10.48550/arxiv.2211.05100.

[19] W. Antoun, F. Baly, and H. Hajj, *Arabert: Transformer-based model for arabic language understanding*, arXiv preprint, Feb. 2020. DOI: 10.48550/arxiv.2003.00104.

[20] N. Wretblad, F. Riseby, R. Biswas, A. Ahmadi, and O. Holmström, *Understanding the effects of noise in text-to-sql: An examination of the bird-bench benchmark*, arXiv preprint, Feb. 2024. DOI: 10.48550/arxiv.2402.12243.

[21] Z. Hong et al., *Next-generation database interfaces: A survey of llm-based text-to-sql*, arXiv preprint, Jun. 2024. DOI: 10.48550/arxiv.2406.08426.

[22] J. Lee et al., *Emergent abilities of large language models*, arXiv preprint, Jun. 2022. DOI: 10.48550/arxiv.2206.07682.

[23] A. S. Al-Hegami and A. S. Ghodel, “A particle swarm based approach for classification of cancer based on ct scan,” *Int. J. Comput. Appl.*, vol. 178, no. 12, May 2019.

[24] L. Xu et al., *Parameter-efficient fine-tuning methods for pre-trained language models: A critical review and assessment*, arXiv preprint, 2023. DOI: 10.48550/arxiv.2312.12148.

[25] K. Kamahori, T. Tang, Y. Gu, K. Zhu, and B. Kasikci, *Fiddler: Cpu-gpu orchestration for fast inference of mixture-of-experts models*, arXiv preprint, Feb. 2024. DOI: 10.48550/arxiv.2402.07033.

[26] S. Parida et al., “Olive: An instruction following llama model for odia language,” in *Proceedings of the IEEE SILCON Conference*, Institute of Electrical and Electronics Engineers, Nov. 2023, pp. 1–7. DOI: 10.1109/silcon59133.2023.10404195.

[27] T. Dettmers, S. Shleifer, M. Lewis, and L. Zettlemoyer, *8-bit optimizers via block-wise quantization*, arXiv preprint, Oct. 2021. DOI: 10.48550/arxiv.2110.02861.

[28] F. Koto et al., *Llama-3.1-sherkala-8b-chat: An open large language model for kazakh*, arXiv preprint, Mar. 2025. DOI: 10.48550/arxiv.2503.01493.