

Adapting Large Language Models (LLMs) for Text-to-SQL Generation in Low-Resource Languages: A Case Study of Chichewa

By

John Emeka Eze (john.eze@aims.ac.rw)
African Institute for Mathematical Sciences (AIMS), Rwanda

Supervised by: Doctor Dunstan Matekenya
World Bank Group, Washington DC, USA

June 2025

*AN ESSAY PRESENTED TO AIMS RWANDA IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF
MASTER OF SCIENCE IN MATHEMATICAL SCIENCES*



AIMS

African Institute for
Mathematical Sciences
RWANDA

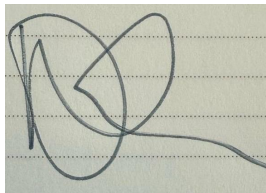
DECLARATION

This work was carried out at AIMS Rwanda in partial fulfilment of the requirements for a Master of Science Degree.

I hereby declare that except where due acknowledgement is made, this work has never been presented wholly or in part for the award of a degree at AIMS Rwanda or any other University.

A handwritten signature in black ink, appearing to be 'John Emeka Eze', written on a light-colored background.

Student: John Emeka Eze

A handwritten signature in black ink, appearing to be 'Doctor Dunstan Matekenya', written on a light-colored background with horizontal lines.

Supervisor: Doctor Dunstan Matekenya

ACKNOWLEDGEMENTS

I extend my deepest gratitude to the individuals whose support made this research and my studies at AIMS Rwanda possible. I sincerely appreciate my supervisor Dr Dunstan Matekenya for his guidance and mentorship throughout the research and Nadine, my thesis tutor for the corrections and constructive suggestions always rendered with exceptional warmth and encouragement. am deeply indebted to all lecturers, tutors, and administrative staff at AIMS Rwanda for creating an environment conducive for learning. Your collective dedication was fundamental to the successful completion of my study and this project. Finally and most importantly, I acknowledge with humble gratitude God's grace that sustained me through this challenging endeavor.

DEDICATION

I dedicate this to my father Late Mr Fidelis Eze, my beloved mother Mrs Martha Eze and to all the researchers whose research silently contribute to the development of society.

Abstract

The rapid evolution of text-to-SQL systems powered by large language models (LLMs) has transformed database accessibility, yet these advances remain confined to high-resource languages like English, leaving speakers of low-resource languages (LRLs) underserved. This study addresses this gap by proposing a novel framework to adapt LLM-based text-to-SQL systems for LRLs, with a focus on Chichewa—a Bantu language spoken by over 12 million people in Malawi and parts of Zambia and Zimbabwe. We identify critical challenges including scarce parallel NL–SQL annotated data, and schema diversity, which hinder direct application of state-of-the-art models. To overcome these, we integrate cross-lingual transfer learning and schema-aware prompt engineering, leveraging the parameter-efficient fine-tuning (PEFT) method Quantized Low-Rank Adaptation (QLoRA) to optimize pre-trained multilingual LLMs. Our approach includes first experimenting on the Spider benchmark getting up to 94.1% Component Match for QLoRA + Few-Shot (random) when using DeepSeek Coder 6.7B Base and up to 63.1% Execution Accuracy for Few-Shot (MiniLM embeddings) when using Llama 3.1 8B. We created a small scale Chichewa dataset and applied Retrieval-Augmented Semantic Parsing (RASP) the multilingual MiniLM sentence transformer to capture linguistic nuances since it was pretrained on LRLs including Chichewa. Results demonstrate that our framework achieves 78.4% Component Match and 15.4% Execution Accuracy on for few shot + RASP but was outperformed by zero-shot on English translation of the Chichewa NL–SQL pairs by up to 87.6%. The study contributes (1) a systematic methodology for LRL text-to-SQL adaptation, (2) a Chichewa NL–SQL benchmark, and (3) empirical insights into the efficacy of prompt engineering versus fine-tuning in low-data regimes. Our findings underscore the potential and limitations of LLMs in democratizing NLP technologies for linguistically marginalized communities.

Contents

| | |
|---|------------|
| Declaration | i |
| Acknowledgements | ii |
| Dedication | iii |
| Abstract | iv |
| 1 Introduction | 2 |
| 1.1 Background of Study | 2 |
| 1.2 Problem Statement | 4 |
| 1.3 Objectives of the Study | 5 |
| 1.4 Significance of the Study | 6 |
| 1.5 Limitations of the Study | 6 |
| 2 Literature Review | 7 |
| 2.1 Overview | 7 |
| 2.2 Evolution of Text-to-SQL Approaches | 7 |
| 2.3 Challenges faced in Text-to-SQL | 8 |
| 2.4 Leveraging Pre-trained LLMs with Fine-Tuning and Prompt Engineering | 11 |
| 2.5 NLP for Chichewa | 12 |
| 3 Research Methodology | 13 |
| 3.1 Theoretical Foundations | 13 |
| 3.2 Research and Experimental Workflow | 19 |
| 3.3 Text-to-SQL Experiments with Spider Benchmark | 21 |
| 3.4 Text-to-SQL Adaptation Strategies for Chichewa | 24 |
| 4 Evaluation and Results | 27 |
| 4.1 Experimental Setup | 27 |
| 4.2 Results from Spider Benchmark Experiment | 27 |

| | |
|--|-----------|
| 4.3 Results from Chichewa Experiment | 29 |
| 5 Conclusion and Recommendation | 32 |
| 5.1 Conclusion | 32 |
| 5.2 Recommendation | 32 |
| References | 38 |

List of Tables

| | | |
|-----|--|----|
| 3.1 | Model Classification for Chichewa Text-to-SQL Adaptation | 25 |
| 3.2 | Case study illustrating a successful and a failed prediction | 26 |

List of Figures

| | | |
|-----|---|----|
| 1.1 | The framework for text-to-SQL systems entails creating a SQL query using the user's question and the database schema that is supplied. The system generates a pertinent SQL query to ask about the database system for the intended outcome based on the user input and the database schema | 3 |
| 3.1 | The Transformer Architecture (Vaswani et al., 2017) | 14 |
| 3.2 | Overview of a Text-to-SQL Architecture. Source: Mind Inventory | 17 |
| 3.3 | LLM Adaptation Workflow for Chichewa Text-to-SQL | 20 |
| 3.4 | Example Data with English and Chichewa Questions, Database Name, SQL Queries and Responses | 24 |
| 4.1 | Performance Metrics | 28 |
| 4.2 | Model Comparison across the Different Metrics | 28 |
| 4.3 | Zero-Shot Performance Directly from Chichewa to SQL | 29 |
| 4.4 | Zero-Shot Performance for English Translations of Chichewa Questions to SQL | 30 |
| 4.5 | Performance for Few-Shot + Retrieval Augmented Semantic Parsing | 31 |

1. Introduction

1.1 Background of Study

Organizations are depending more and more on relational databases in this digital age to handle and examine enormous volumes of structured data. These databases are now essential components of many contemporary systems, ranging from customer relationship management to corporate analytics. In many industries, the requirement to query, retrieve, and interpret data has grown increasingly important as its volume rises. However, using Structured Query Language (SQL), a technical ability, is frequently necessary for database querying. Text-to-SQL systems have emerged as a result of this disconnect between consumers who demand access to data and the specific expertise needed to retrieve it. (Kanburoğlu and Tek, 2024)

Text-to-SQL systems bridge the gap between non-technical users and sophisticated relational databases by converting users' natural language inquiries into executable SQL queries. (Li and Jagadish, 2014) Easy access to structured data is essential for prompt insights and well-informed decision-making in a time when data quantities are skyrocketing across industries, including healthcare, banking, government, and education. Although surveys show that 51.52% professional developers regularly use SQL (Hong et al., 2024), much fewer non-technical stakeholders are able to directly utilize it, therefore writing SQL is still a barrier. This is addressed by text-to-SQL parsing, which democratizes database queries by enabling users to communicate their requirements using natural language instead of formal terminology. When data complexity rises and manual data exploration becomes impractical and ineffective, this skill becomes crucial. (Dong et al., 2023)

Let \mathcal{X} be the space of natural-language utterances and \mathcal{Y} the space of syntactically valid SQL queries over a fixed database schema \mathcal{S} . A text-to-SQL parser is a function

$$f_{\theta} : \mathcal{X} \longrightarrow \mathcal{Y}, \quad \theta = \text{model parameters.} \quad (1.1.1)$$

Given a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N \subset \mathcal{X} \times \mathcal{Y}$, the usual objective is

$$\theta^* = \arg \min_{\theta} \left[- \sum_{(x,y) \in \mathcal{D}} \log p_{\theta}(y \mid x, \mathcal{S}) \right], \quad (1.1.2)$$

i.e., maximum-likelihood (cross-entropy) training of a conditional language model.

Consider, for instance, a relational database with columns for `company_name`, `revenue`, `country` and `fiscal_year` that details the top 500 corporations in the world by consolidated revenue in 2024, as determined by the annual Fortune Global 500, (Fortune, 2024). Let us say we have a natural language question: *“Could you provide the names of the five companies with the highest revenues in 2024, along with their total revenue, and the country where each is headquartered?”* After analyzing this question and determining the user’s purpose, a text-to-SQL system would automatically create the appropriate SQL query to retrieve the relevant data from the database. The SQL query in this instance may resemble this:

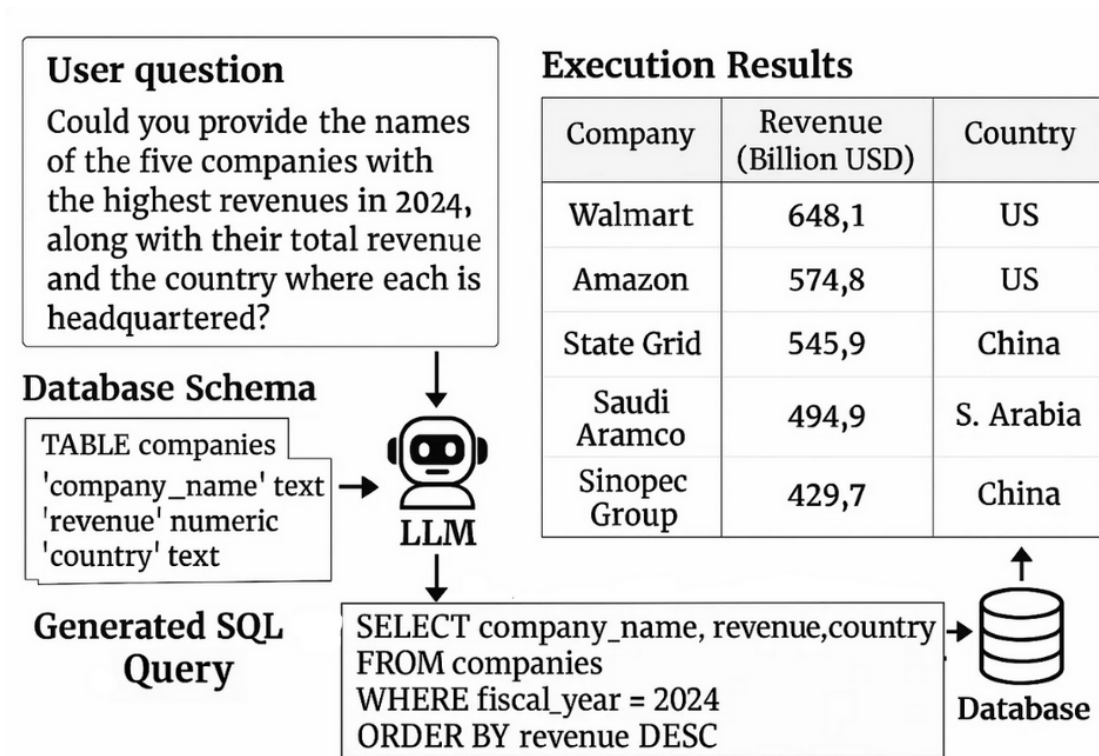


Figure 1.1: The framework for text-to-SQL systems entails creating a SQL query using the user's question and the database schema that is supplied. The system generates a pertinent SQL query to ask about the database system for the intended outcome based on the user input and the database schema

```
SELECT company_name, revenue, country
FROM companies
WHERE fiscal_year = 2024
ORDER BY revenue DESC
LIMIT 5;
```

This SQL query returns the headquarters' name, revenue, and nation. Complex data is made more accessible by the text-to-SQL technology, which allows the user to retrieve this particular information without writing the SQL query.

Text-to-SQL implementation has advanced significantly using earlier conventional techniques. The evolution of these implementations has been a protracted process. The majority of early attempts relied on specifically designed rules and blueprints which were especially appropriate for straightforward database applications (Mahmud et al., 2015; Kang et al., 2013). Due to the increasing complexity of databases, rule-based approaches failed to understand complex schemas giving rise to the data-driven techniques. The ability of text-to-SQL to automatically learn a mapping from the user inquiry to its matching SQL has been enhanced by the creation of deep neural networks (Katsogiannis-Meimarakis and Koutrika, 2021; Kumar et al., 2022). The performance of text-to-SQL systems has since been elevated to a new level by pre-trained language

models (PLMs) with powerful semantic parsing capabilities, which have emerged as the new paradigm (Wang et al., 2023). This topic has improved through incremental research on PLM-based optimization, including pre-training and table content encoding.

Large Language Models (LLMs) as BERT (Guo and Gao, 2019), GPT-4 (OpenAI, 2023), Codex (Chen et al., 2021), and LLaMA (Touvron et al., 2023) have lately emerged and completely changed this work. With the help of extensive pre-training on a variety of text corpora, LLMs enable zero- and few-shot learning for Text-to-SQL with little extra training, in contrast to previous rule-based or smaller neural techniques (Chang and Fosler-Lussier, 2024). The precision of query execution has been impressively improved by recent developments in rapid engineering, especially when used on challenging benchmarks like Spider. Even in difficult cross-domain settings, these tactics—which include carefully crafting schema representations, incorporating table content signals, and offering demonstration examples—have proven to raise execution accuracy above the 86.6% threshold (Gao et al., 2024). These in-context learning (ICL) techniques preserve flexibility across a variety of schemas while avoiding intensive fine-tuning (Gao et al., 2024).

Nevertheless, almost all of the study conducted thus far has been on high-resource languages, primarily English. In the Text-to-SQL literature, low-resource languages (LRLs), those having little parallel NL–SQL data and less NLP resources, are still mostly unexplored (Joshi et al., 2020). Hundreds of millions of speakers around the world rely on LRLs to obtain information, but they are unable to take use of database interfaces enabled by LLMs, which is a significant gap. Though it presents special difficulties with timely translation, cross-lingual transfer, and data scarcity, extending Text-to-SQL to LRLs promises to further democratize data access (Yu et al., 2018).

The methods for converting LLM-based Text-to-SQL systems to low-resource languages (LRLs) are examined in this thesis. LLM designs, prompting methods, and fine-tuning approaches are first examined. Using Chichewa, a Bantu language spoken by 12 million people in Malawi and parts of Zambia and Zimbabwe, as a case study, we build upon this foundation by proposing a cross-lingual transfer framework that makes use of English NL–SQL examples, unsupervised translation, and schema-aware prompt templates designed especially for LRLs.

When working with digitized datasets, Chichewa-speaking professionals in the fields of healthcare, agriculture, and education encounter major obstacles that cause bottlenecks in crucial decision-making processes. This study emphasizes the urgent need for inclusive AI systems that empower marginalized linguistic communities by addressing the lack of Text-to-SQL tools for Chichewa.

1.2 Problem Statement

1.2.1 Research Gap. Lack of defined evaluation benchmarks, cross-lingual ambiguity, and data scarcity are the main causes of the complex issue of enabling Text-to-SQL systems for low-resource languages (Yu et al., 2019). Due to the lack of annotated datasets and the high expense of manual curation, parallel natural language–SQL training data—which is a prerequisite for current LLM-based Text-to-SQL systems—is prohibitively scarce for languages like Chichewa. This dependence establishes a fundamental obstacle since current models require substantial retraining in order to

generalize to LRLs (Shu et al., 2023). Schema ambiguity (such as mismatched column names or table linkages) and dialectal variances greatly impede the process of converting knowledge from high-resource languages (HRLs), such as English, to LRLs in the context of cross-lingual transfer (Gan et al., 2021). Models trained on homogenized HRL data are confused by Chichewa's regional lexical and syntactic distinctions between Malawi and Zambia, for example. The lack of standardized evaluation benchmarks specific to LRLs exacerbates these problems. Researchers are unable to compare methods, discover failure modes specific to LRL settings, or rigorously quantify progress in the absence of domain-specific metrics or carefully selected test sets. The prevalence of English-centric benchmarks like Spider, which ignore linguistic and structural subtleties in LRLs, significantly widens this disparity (Yu et al., 2019; Gan et al., 2021).

1.2.2 Implications of Ignoring This Problem. If these gaps are not filled, LRL speakers run the risk of continuing to be excluded. Communities that use languages like Chichewa, which are already underrepresented in digital infrastructure, would be unable to use data-driven decision-making tools and would have to rely on English-language intermediaries or faulty human translation. In industries like healthcare and agriculture, where prompt database access can have a direct impact on results, such exclusion could worsen inequality. For instance, crucial actions may be delayed if Malawian healthcare professionals are unable to access patient databases in Chichewa (Taylor and Kazembe, 2024). Regional scalability is also affected: in the absence of strong Text-to-SQL technologies, Central and Southern African digital infrastructure modernization initiatives would come to a standstill since local stakeholders would no longer be able to access databases. In addition, this disregard immediately compromises attempts to provide fair access to technology on a global scale.

1.3 Objectives of the Study

The general aim of this research is to improve LLM-based Text-to-SQL systems for low-resource languages, using Chichewa as a case study.

The objectives of this research are as follows:

- 1 Develop a cross-lingual transfer framework combining English NL–SQL examples, unsupervised translation, and schema-aware prompt templates
- 2 Construct a small-scale benchmark dataset of Chichewa NL–SQL pairs for evaluation across the metrics Exact Match, Component Match and Execution Accuracy.
- 3 Evaluate prompt engineering (zero-shot and few-shot) vs. parameter-efficient fine-tuning (PEFT) strategies (LoRA and QLoRA) for English and Chichewa, and methods of combining both strategies to improve model performance.

1.4 Significance of the Study

Knowledge and practice in the technological, social, economic, and research realms are advanced by this study. In technical terms, it extends the capabilities of large language models (LLMs) beyond high-resource languages like English by becoming the first to develop cross-lingual adaptation techniques for structured NLP tasks. Socially, it democratizes access to vital data in industries like healthcare and agriculture by enabling Chichewa speakers to communicate directly with databases without the need for SQL knowledge. For businesses that use Chichewa for daily operations, like *Telekom Networks Malawi(TNM)* who provide services for native speakers, the framework lowers operational expenses by reducing reliance on English-language technical intermediaries. From a research standpoint, the study fills in assessment benchmark and dialectal robustness gaps while establishing baseline procedures for low-resource language (LRL) text-to-SQL systems. In addition, this thesis provides a framework that can be reproduced when adapting LLMs to other low-resource languages.

1.5 Limitations of the Study

A major limitation faced in this study is the lack of annotated Chichewa NL–SQL pairs. Also, for the small scale Chichewa dataset that was created was done in the standard central dialect. This method leaves out regional dialectal differences to ensure uniformity. Also, the approach is limited to prompt engineering and parameter-efficient fine-tuning of pre-existing models and due to computational resource constraints LLMs were not trained from scratch. This design decision emphasizes cross-domain comparability but runs the risk of ignoring particular linguistic or structural issues specific to Chichewa databases.

2. Literature Review

2.1 Overview

Text-to-SQL transforms natural language (NL) into syntactically and semantically sound Structured Query Language (SQL) commands that may obtain desired data from relational databases. By bridging the gap between machine-readable interactions with databases and human intent, this task allows non-technical users to query complicated datasets without knowing SQL syntax (Kanburoğlu and Tek, 2024). Fundamentally, text-to-SQL systems need to read unstructured NL input, apply it to provided database schema elements (such as tables, columns, and relationships), and produce executable SQL that complies with database constraints and represents the user's purpose. For instance, the query *"List patients diagnosed with malaria in Lilongwe, Malawi last month"* needs to be converted into a SQL query that picks the right fields, filters by diagnosis type and date, and combines pertinent tables (such as patients, diagnoses, and locations). Because they remove the requirement for SQL expertise and hastens retrieval of information, these text-to-SQL solutions allow non-expert users to interface with databases naturally. Developments in this field are essential for democratizing access to data, especially in low-resource environments where data-driven decision-making is essential but technical literacy is low. Additionally, in the midst of LLM study, text-to-SQL may be able to lessen common hallucination problems by using realistic database information to close knowledge gaps for LLMs (Hong et al., 2024).

2.2 Evolution of Text-to-SQL Approaches

The rule-based systems and grammar-driven parsers used by early Text-to-SQL techniques required a great deal of manual language engineering and had scalability issues. These approaches were limited in their ability to handle sophisticated queries and a variety of schemas, despite being successful in straightforward and domain-specific databases (Mahmud et al., 2015). Although systems like LUNAR (Kang et al., 2013) and NaLIX (Hammami et al., 2021) showed promise in semantic parsing, their scalability and adaptability were hampered by the substantial amount of manual feature engineering they required. Rule-based systems found it difficult to generalize across many domains and databases as queries became increasingly intricate, containing ambiguity and nested structures. Neural network-based models that could learn patterns from training data rather than depending only on preset rules were made possible by this rigidity and performance problems, which led to a move toward data-driven techniques.

Sequence-to-sequence models and semantic parsing techniques gained popularity around 2017 with the introduction of neural networks, providing end-to-end learning capabilities without the need for manually created rules. Large models had to be trained in order to immediately analyze natural language and generate matching SQL queries. Small-scale methods like Seq2SQL and SQL-Net (Katsogiannis-Meimarakis and Koutrika, 2021) that relied on sequence-to-sequence architectures (more especially, LSTM and transformers) (Banitaba et al., 2024) were the first

models in this family. They showed an end-to-end, differentiable architecture that allowed a suitable neural model to translate text into SQL. Compared to earlier text-to-SQL systems, this represented a significant increase in performance, flexibility, and scalability (Katsogiannis-Meimarakis and Koutrika, 2021; Kumar et al., 2022). More sophisticated models later surfaced, using transformer-based designs such as TaBERT (Katsogiannis-Meimarakis and Koutrika, 2021) and BERT (Guo and Gao, 2019), which enhanced the combined comprehension of user intent and database schema.

Pre-trained language models (PLMs) changed natural language processing by moving the paradigm away from task-specific supervised learning and toward generalized pre-training and fine-tuning. This change was sparked by models like BERT (Guo and Gao, 2019) and GPT (Brown et al., 2020), which used extensive unsupervised text corpora to pre-train models that could subsequently be adjusted for particular tasks. Through intensive pre-training, the transfer learning-based PLM idea enabled models to acquire a profound comprehension of natural language, which they could then apply to tasks like text generation, sentiment analysis, and question-answering (Tahir et al., 2024; Wang et al., 2023). By capturing stronger semantic links between user queries and database schema structures, PLMs significantly improved text-to-SQL. PLMs like BERT-SQL made it possible to combine the database schema and the natural language query into a single representation, increasing the accuracy of SQL generation. These approaches tackled issues like as managing intricate queries involving several joins, nested searches, and generalization across domains. PLMs were limited, nonetheless, by the requirement for domain-specific fine-tuning and the inability to comprehend intricate schemas without the aid of extra schema-linking methods (Wang et al., 2023). Although PLMs improved text-to-SQL systems, modern large language models (LLMs) introduced a more scalable method that generalizes even better.

The introduction of large language models (LLMs) like GPT-4 (OpenAI, 2023), Codex (Chen et al., 2021), and LLaMA (Touvron et al., 2023) has greatly increased the capacity of systems handling challenging tasks like text-to-SQL translation. Often without further fine-tuning, LLMs produce more complex and accurate replies by utilizing their extensive pre-training on large datasets (Brown et al., 2020; Rajkumar et al., 2022). LLMs in text-to-SQL applications gain from prompt engineering, which allows them to adapt to various databases and query methods without requiring a great deal of extra training. They also capture more intricate linkages between natural language queries and organized database schemas.

2.3 Challenges faced in Text-to-SQL

2.3.1 General Challenges. Despite advancements in text-to-SQL systems, linguistic complexity and ambiguity, schema comprehension, intricate SQL operations, and cross-domain generalization remain major obstacles (Hong et al., 2024; Mohammadjafari et al., 2023). Each of the aforementioned obstacles may encounter the problems listed below.

1. **Linguistic Complexity:** Text ambiguity and wording diversity are common features of natural language inquiry. For example, in *"Show departments where they hired more than 10 engineers,"* the pronoun *"they"* necessitates distinguishing between departments and orga-

nizations. Strong semantic comprehension is also required for synonyms (such as *"salary"* vs. *"income"*) and paraphrase (*"employees under 30"* vs. *"workers younger than 30"*). Mapping is made more difficult by lexical mismatches between NL words and schema items (for example, a user requesting *"client ages"* when the schema uses *"customer birthdate"*).

2. **Schema Comprehension:** To generate accurate SQL queries from natural language, models must align the user's question with the database structure by pinpointing required tables, columns, and how they connect. Both explicit schema metadata (such as column names and foreign keys) and implicit associations (such as determining that `employee.department_id` refers to `department.id`) must be understood for this. Problems occur when searches cover several tables with intricate linkages or when schemas employ opaque or technical name conventions (for example, `tbl usr attr` rather than `user_profile`).
3. **Complex SQL Operations:** Multi-table joins, sorting (`ORDER BY`), aggregations (`COUNT`, `SUM`), and nested structures (subqueries) are frequently used in real-world queries. To generate correct queries and results from these kinds of operations, the user must be context-aware and obey syntax strictly when feeding the model with a question or request. For instance, the query "Which sales region has the greatest average revenue?" requires sorting, a `GROUP BY` clause, and an aggregation (`AVG(revenue)`). Runtime errors or inaccurate outcomes may arise from erroneous operator placement or clause ordering.
4. **Cross-Domain Generalization:** Because of variations in schema design, vocabulary, and query patterns, models trained on particular domains (like academic databases) find it difficult to adapt to new domains (like healthcare or e-commerce). Without retraining or fine-tuning, a system designed for academic inquiries (such as *"List courses taught by Professor X"*) might not perform well on medical queries (such as *"Find patients with increased hemoglobin levels"*). This restriction is made worse in low-resource languages since there is a lack of domain-specific training data.

These difficulties are magnified in low-resource language settings where dialectal variances, a lack of linguistic resources, and a variety of schemas make accurate translation more challenging. Innovations in cross-lingual transfer, schema linkage, and adaptive decoding are necessary to address these problems; these are fields where LLMs exhibit promise but are still poorly understood for underrepresented languages.

2.3.2 Challenges in Low-Resource Language Settings. Current NLP solutions predominantly serve high-resource languages like English, Spanish, or German, leaving the approximately 3 billion speakers of low-resource languages, primarily in Asia and Africa, underserved (Joshi et al., 2020; Shu et al., 2023).

Here are some of the challenges faced

1. **Scarcity of annotated datasets:** Large NL-SQL corpora are necessary for supervised text-to-SQL, but manual annotation is extremely expensive and time-consuming. For instance, 11 Yale students invested 1,000 man-hours to create the English Spider dataset, which consists of 10,181 question-query pairs (Yu et al., 2018).

2. **Limited unlabelled corpora.** Large text corpora (such as CommonCrawl and Wikipedia) are essential for unsupervised pre-training and base-model learning, but most low-resource languages do not have similar quantities, which makes representation learning ineffective (Conneau and Lample, 2019).
3. **Dialectal variation & code-switching.** Models trained on Modern Standard Arabic find it challenging to generalize to dialectal inputs since informal variations (such Moroccan Darija) lack standardized orthographies and blend several languages (Einolghozati et al., 2021).
4. **SQL dialect & schema diversity.** Variations in SQL functions across database engines (e.g., MySQL's `LENGTH()` vs. PostgreSQL's `CHAR_LENGTH()`) and domain-specific schema conventions cause sharp drops in parsing accuracy under new schemas (Scholak et al., 2021).
5. **Lack of evaluation benchmarks.** Multilingual benchmarks like MultiSpider cover only seven major languages, leaving most low-resource tongues (e.g., Chichewa, Yoruba) unrepresented and making systematic progress measurement difficult (Dou et al., 2022).
6. **Limited SaaS support.** Commercial NLP platforms typically support only major world languages, with low-resource tongues often omitted (Adamopoulou and Moussiades, 2020).

2.3.3 Overcoming the Challenges. Using cutting-edge techniques to democratize Text-to-SQL for low-resource languages by investigating the following are some strategies to address the issues in the preceding section:

1. **Transfer Learning.** By reusing information from high-resource languages, using big pre-trained models in a transfer learning paradigm lessens the need for extensive annotations. (Pan and Yang, 2010).
2. **Multilingual Learning.** Researchers have created new multilingual benchmarks and dialect-specific datasets to help alleviate the data scarcity. MultiSpider has parallel corpora for English, German, French, Spanish, Japanese, Chinese, and Vietnamese, making it the largest multilingual Text-to-SQL semantic parsing dataset currently available. In a similar vein, Dialect2SQL presents an Arabic dialect dataset that uses crowdsourced annotations and synthetic augmentation to capture the syntactic and lexical nuances of Moroccan Darija. ResearchGate for the ACL Anthology. These tools make it possible to perform cross-lingual transfer experiments and emphasize how crucial linguistically and culturally appropriate data is to enhancing model performance (Conneau and Lample, 2019).
3. **Data Augmentation.** The creation of synthetic data has become a viable tactic for growing low-resource corpora. In order to improve model performance in low-resource circumstances, SynQL uses in-domain synthetic data creation, which involves sampling actual natural language utterances and creating corresponding SQL queries using rule-guided templates and small seed sets. To provide pseudo-parallel data without requiring a lot of manual labor, translation-based augmentation uses top-notch translation models to back-translate

pre-existing English Text-to-SQL datasets into target low-resource languages. Even though translation can add noise, careful post-editing and domain adaption frequently result in significant improvements (Wei and Zou, 2019; Sennrich et al., 2016).

4. **Active & Semi-Supervised Learning.** To increase annotation efficiency and improve model robustness in the face of data scarcity, it is important to consider pseudo-labeling unlabeled data and selecting annotating the most instructive examples (Zhu, 2005).
5. **Schema Linking & Constraint-Aware Decoding.** Schema linking and decoding that is conscious of constraints. Syntactic validity is ensured and execution errors are decreased by grounding NL queries in database schemas and applying grammatical constraints during decoding (for example, using PICARD) (Scholak et al., 2021).

2.4 Leveraging Pre-trained LLMs with Fine-Tuning and Prompt Engineering

2.4.1 Prompt Engineering for Pretrained LLMs. Prompt engineering (also called in-context learning) uses natural language task descriptions and in-context examples to guide pre-trained LLMs without requiring gradient changes. The groundbreaking few-shot performance of GPT-3 demonstrated that a 175 B-parameter model may outperform some fine-tuned baselines in a variety of tasks, from translation to arithmetic, by merely conditioning on a small number of exemplars (Brown et al., 2020). Through automated prompt generation and dynamic demonstration selection, later research has improved prompt-based systems to the point where they can achieve up to 30% absolute improvements over standard fine-tuning in low-data regimes (Gao et al., 2021). Despite these developments, performance gaps for many low-resource languages still exist, systematic evaluation techniques and access to reliable native-language datasets continue to be bottlenecks, highlighting the fact that LLMs are not a universal solution and encouraging further research into data-efficient adaptation techniques. By using either zero or a small number of training examples (NL-SQL pairs), in-context learning enables pretrained large language models (LLMs) to execute text-to-SQL. Three popular settings for text-to-SQL in-context learning are introduced in this section.

2.4.2 Fine-Tuning Pretrained Multilingual Language Models. By using extensive unsupervised pretraining on hundreds of languages and then fine-tuning on task-specific data, pre-trained multilingual language models (mPLMs) like XLM-R have shown notable cross-lingual transfer capabilities. This is true even for low-resource languages like Swahili and Urdu, where XLM-R outperforms previous models by more than 15% in accuracy on XNLI (Lample and Conneau, 2019; Conneau et al., 2020). However, in environments with limited resources, it is frequently impossible to fully fine-tune all model parameters. Parameter-efficient fine-tuning (PEFT) techniques fix this by adding a few more parameters while maintaining the majority of the pre-trained weights constant. Using only 3.6% of the parameters per job, adapter modules achieve within 0.8% of full fine-tuning performance on GLUE by inserting lightweight bottleneck layers into each Transformer block (Houlsby et al., 2019). Prefix-tuning and prompt-tuning, two more recent PEFT techniques, optimize continuous prompt vectors or prefixes rather than entire model

weights, producing equal downstream performance with less compute overhead (Li and Liang, 2021; Lester et al., 2021). By breaking down weight updates into low-rank matrices, Low-Rank Adaptation (LoRA) significantly minimizes the trainable parameter footprint and shows that multilingual LLMs may be successfully adapted to languages like Marathi even with sparse data (Hu et al., 2021).

2.5 NLP for Chichewa

2.5.1 Chichewa in Open-Source LLMs and NLP Models. Chichewa (also known as Chewa or Nyanja) is a Bantu language spoken in Malawi, Zambia, and Mozambique. Its support in open-source NLP models and LLMs is limited, as most models prioritize high-resource languages like English. However, some efforts include Chichewa. Meta’s *No Language Left Behind (NLLB)* includes Chichewa (*ny*) in its 200-language coverage for translation tasks, though performance is modest due to sparse training data (Meta AI Research Team, 2022). Chichewa is not directly supported by major monolingual BERT variants, but multilingual models like *AfriBERTa* (covering 11 African languages) exclude it (Alabi et al., 2022). The Masakhane initiative has explored NLP for African languages, including Chichewa, but progress remains early-stage (Nekoto et al., 2020).

2.5.2 Chichewa Dataset Availability. Chichewa datasets are scarce but several notable datasets and sources are available. Some of them are *OSCAR Corpus* containing Chichewa text extracted from Common Crawl, though quality and volume are limited (Suárez et al., 2019). For religious text, the Bible and Jehovah’s Witnesses’ *JW300 Corpus* provide parallel translations (Agyemang et al., 2020). Some individual researchers have worked on creating Chichewa datasets as we can see from (Matekenya, 2022), a dataset containing audio files, transcripts, documentation and metadata about each audio file as well as (Taylor, 2020) which is a spoken corpus. The *NLLB* and *OPUS* repositories include small Chichewa-English parallel datasets (Meta AI Research Team, 2022). Annotated datasets for tasks like POS tagging, NER, or Text2SQL are nearly non-existent.

2.5.3 Text-to-SQL for Chichewa. No published works address **Text-to-SQL** (converting natural language to SQL queries) for Chichewa. Challenges include lack of parallel datasets pairing Chichewa questions with SQL schemas, limited syntactic parsing tools for the language and the case that most Text-to-SQL research focuses on English, with few multilingual extensions (e.g., *mSpider* for Chinese and English; (Dou et al., 2022)). For Chichewa, a feasible approach would involve translating English Text2SQL datasets (e.g., *Spider*; (Yu et al., 2018)) and fine-tuning multilingual LLMs, but this remains unexplored.

3. Research Methodology

3.1 Theoretical Foundations

3.1.1 Structured Query Language (SQL). Structured Query Language (SQL) is a specialized domain-specific language utilized for managing and manipulating relational databases. Developed around the relational database model, SQL enables users to perform data queries, update records, create schema structures, and manage database access. The SQL syntax comprises various key components, including Data Definition Language (DDL), Data Manipulation Language (DML), and Data Control Language (DCL). SQL's strength lies in its declarative nature, allowing users to specify what data to retrieve without detailing how to retrieve it, significantly simplifying database interactions (Codd, 1970). The relational database system is fundamentally based on set theory, where sets S_1, S_2, \dots, S_n are interconnected through relations R . Each set corresponds to a database table within a particular domain. Relationships between tables are established using Foreign Keys. A foreign key is a column whose values can appear across multiple tables, thereby linking them (Date, 2004).

3.1.2 Large Language Models (LLMs). Large Language Models (LLMs) are sophisticated neural networks predominantly based on transformer architectures. They are designed to process, generate, and understand human language. Transformers rely on self-attention mechanisms, allowing them to weigh the relevance of different parts of the input data, facilitating superior context understanding (Vaswani et al., 2017). LLMs, such as GPT-series (Brown et al., 2020) and DeepSeek (Xiao et al., 2024) models, as well as finetuned models like Mixtral (Jiang et al., 2023), achieve high linguistic competence by pre-training on massive text corpora, making them capable of various language-related tasks without task-specific training (Devlin et al., 2018).

3.1.3 The Transformer Architecture and Attention Mechanism.. The Transformer architecture (Vaswani et al., 2017) revolutionized natural language processing by using self-attention mechanisms rather than recurrent or convolutional layers. This architecture consists primarily of encoder and decoder stacks composed of multi-head self-attention and feed-forward neural network layers. The self-attention mechanism allows models to directly capture dependencies between words regardless of their distance within a sequence. Multi-head attention splits the attention mechanism into multiple parallel operations, enabling the model to simultaneously attend to information from different representation subspaces. Additionally, positional encodings are introduced to maintain the sequential order of words, as Transformers do not inherently capture positional information. This architecture forms the backbone of LLMs due to its scalability and effectiveness in capturing contextual relationships.

The attention mechanism, integral to the Transformer architecture, enables models to selectively focus on different parts of the input data based on their relevance to the task at hand. Attention computes weights for each input element relative to others, capturing dependencies across positions irrespective of distance. Mathematically, attention is expressed as

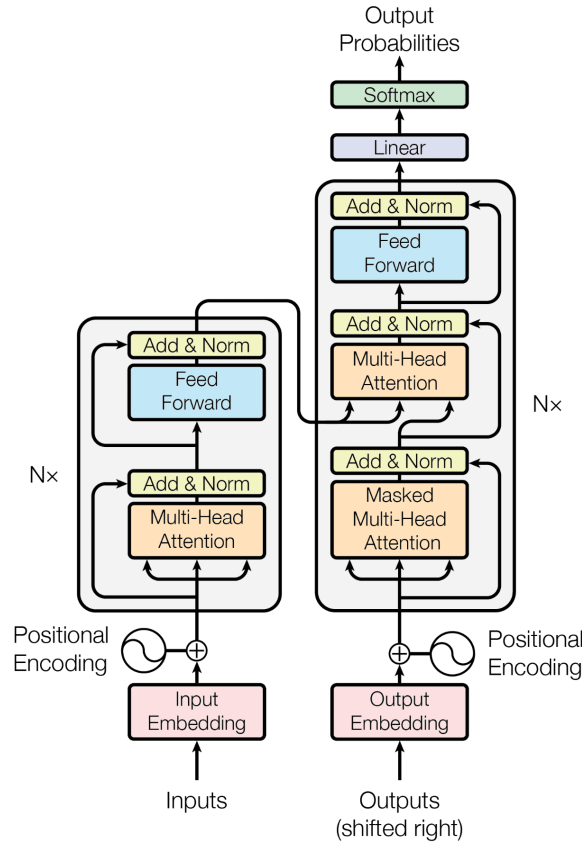


Figure 3.1: The Transformer Architecture (Vaswani et al., 2017)

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \quad (3.1.1)$$

where, Q , the query matrix, represents the query vectors derived from the input data, K , the key matrix, denotes the key vectors that provide context, V , the value matrix, signifies the value vectors containing the information to be combined based on attention scores, d_k , is the dimensionality of the key matrix. Multi-head attention extends this by running several attention operations in parallel, each with its own learnable parameters, allowing the model to jointly attend to information from different representation subspaces. Attention mechanisms provide models with the ability to capture long-range dependencies and intricate contextual relationships within the data.

Adapting LLMs for specialized tasks such as Text-to-SQL often involves leveraging their ability to learn task-specific nuances through methods like fine-tuning, prompt engineering, and few-shot learning. Adaptation enables LLMs to better align their output with specific domains or applications, enhancing performance in targeted tasks beyond their generalized training.

3.1.4 Tokenization and Vector Embeddings. Tokenization is the process of converting raw text into a sequence of discrete units called *tokens*. Given an input sequence of characters

$S = (c_1, c_2, \dots, c_n)$, tokenization applies a mapping

$$\mathcal{T}(S) = (t_1, t_2, \dots, t_m)$$

where $t_i \in \mathcal{V}$ for a fixed vocabulary \mathcal{V} , and typically $m \leq n$. Tokens can be words, characters, or subword units (e.g., produced by Byte-Pair Encoding). Each token t_i is assigned an integer index $v_i = \text{VocabLookup}(t_i)$.

Vector embeddings assign each token $t \in \mathcal{V}$ a dense vector $\mathbf{e}_t \in \mathbb{R}^d$, where d is the embedding dimension. Let $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ denote the embedding matrix. For a token t with index v :

$$\mathbf{e}_t = E[v, :] \quad (3.1.2)$$

Given a sequence of m tokens, their embeddings form a matrix in $\mathbb{R}^{m \times d}$:

$$\mathbf{E}_{\text{input}} = \begin{bmatrix} \mathbf{e}_{t_1} \\ \mathbf{e}_{t_2} \\ \vdots \\ \mathbf{e}_{t_m} \end{bmatrix} \quad (3.1.3)$$

Embeddings are trained to capture semantic and syntactic relationships, and similarity can be measured by cosine similarity:

$$\text{sim}(\mathbf{e}_i, \mathbf{e}_j) = \frac{\mathbf{e}_i \cdot \mathbf{e}_j}{\|\mathbf{e}_i\| \|\mathbf{e}_j\|} \quad (3.1.4)$$

In transformer models, token embeddings are further processed to produce contextual embeddings sensitive to the input sequence.

3.1.5 In-context Learning (Prompt Engineering). In-context learning (also known as prompt engineering) involves providing an LLM with task-relevant examples and instructions directly within the input prompt to guide its response generation without changing its underlying parameters (Brown et al., 2020). Some prompt engineering techniques used in Text-to-SQL tasks include:

1. **Zero-shot Text-to-SQL:** Here, pretrained LLMs' ability to directly infer the NL-SQL relationship from a table without the need of demonstration examples is assessed. A task instruction and a test question with the matching database are included in the input. The text-to-SQL proficiency of LLMs is directly evaluated using zero-shot text-to-SQL (Rajkumar et al., 2022; Liu et al., 2023).
2. **Single-domain Few-shot Text-to-SQL:** This method is intended for uses or fields where creating examples is simple, including airline reservations and geographic data queries. Using a few in-domain demonstration cases gathered from the same database as the test question, it assesses LLMs' adaptability. Evaluating the LLMs' text-to-SQL performance with little in-domain training data is the aim (Rajkumar et al., 2022).
3. **Cross-domain Few-shot Text-to-SQL:** This environment assesses how well models generalize to new domains by using demonstrations from outside the domain. In this case,

one or more demonstration databases that differ from the test database are represented by the seven demonstration NL-SQL pairings. The cross-domain few-shot text-to-SQL test evaluates LLMs' ability to apply what they have learned from demos to new databases (Chen et al., 2023a; Poesia et al., 2022).

Recent approaches, such as the DIN-SQL and MAC-SQL frameworks, have demonstrated that carefully structured prompts significantly enhance the accuracy of SQL query generation from natural language.

3.1.6 Fine-tuning. Fine-tuning involves training a pre-trained LLM further on task-specific datasets to optimize its parameters specifically for the intended task, such as Text-to-SQL. Fine-tuning typically involves fewer data points than full pre-training and can be more computationally efficient. Techniques like Low-Rank Adaptation (LoRA (Hu et al., 2021)) and Quantized Low-Rank Adaptation (QLoRA (Dettmers et al., 2023)) are employed to improve efficiency and reduce computational demands, facilitating the fine-tuning of even large models on more modest hardware setups. More details about LoRA and quantization are given below.

1. **Low-Rank Adaptation (LoRA).** Low-Rank Adaptation (LoRA) is a parameter-efficient fine-tuning (PEFT) technique that reduces the number of trainable parameters required for adapting large models. Let $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be the original (frozen) weight matrix in a linear or attention layer. LoRA introduces an additive low-rank matrix:

$$W_{\text{adapted}} = W + \Delta W, \quad \Delta W = BA^T \quad (3.1.5)$$

where $A \in \mathbb{R}^{d_{\text{in}} \times r}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$ are trainable matrices, and $r \ll d_{\text{in}}, d_{\text{out}}$ is the rank (typically 8–64). For input $x \in \mathbb{R}^{d_{\text{in}}}$, the layer computes:

$$y = W_{\text{adapted}}x = Wx + B(A^T x) \quad (3.1.6)$$

This approach reduces trainable parameters from $d_{\text{out}} \times d_{\text{in}}$ to $r(d_{\text{in}} + d_{\text{out}})$, achieving significant efficiency without sacrificing adaptation capability.

2. **Quantization** Quantization involves reducing the numerical precision of model parameters from higher-bit representations (such as 32-bit floating-point) to lower-bit formats (e.g., 8-bit integers). Quantization reduces the memory footprint and computational requirements, facilitating efficient model deployment, especially on resource-constrained hardware. A common quantization approach is uniform affine quantization, given by:

$$Q(x) = \text{round} \left(\frac{x - \min(x)}{\max(x) - \min(x)} \right) \cdot (2^b - 1) \quad (3.1.7)$$

where x is the original parameter value, b represents the bit-width, and $Q(x) \in \{0, 1, \dots, 2^b - 1\}$, denotes the quantized value. Quantization, when combined with LoRA, gives birth to QLoRA, a method which allows for a more highly efficient fine-tuning and deployment of large language models.

3.1.7 Retrieval-Augmented Semantic Parsing. Retrieval-augmented semantic parsing (RASP (Zhang et al., 2024)) enhances the process of mapping natural language utterances x to formal logical forms y by incorporating relevant retrieved contexts C_x . The overall process is summarized by

$$y = f(x, \text{Retrieve}(x)) \quad (3.1.8)$$

where f is the semantic parser, and $\text{Retrieve}(x)$ returns the top- k contexts most relevant to x . Given a retrieval corpus \mathcal{D} and a similarity function $s(x, c)$ (often cosine similarity between embeddings), the top- k contexts are:

$$C_x = \arg \max_{c \in \mathcal{D}} s(x, c) \quad (3.1.9)$$

with

$$s(x, c) = \frac{\phi(x) \cdot \psi(c)}{\|\phi(x)\| \|\psi(c)\|} \quad (3.1.10)$$

where ϕ, ψ are neural encoders.

At inference, the parser conditions on both the query and retrieved contexts, which can include examples, schema elements, or documentation, providing improved accuracy and generalization.

3.1.8 The Text-to-SQL Architecture. Text-to-SQL involves translating natural language queries into SQL statements. Mathematically, the task can be formalized as follows:

Given a natural language query Q_{NL} and a database schema S_{DB} , the goal is to generate a syntactically correct SQL query Q_{SQL} such that:

$$\mathcal{F}(Q_{NL}, S_{DB}) \rightarrow Q_{SQL} \quad (3.1.11)$$

where \mathcal{F} denotes the function represented by the model. The process follows the following stages

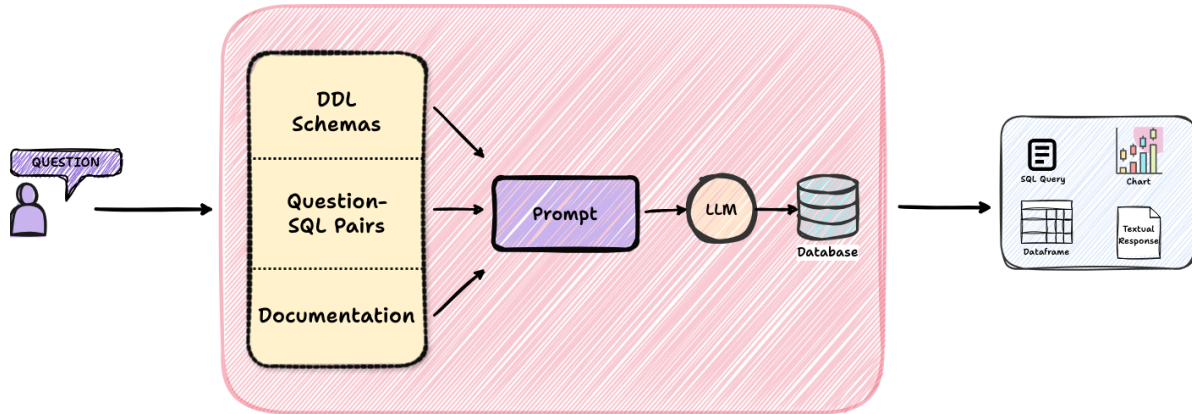


Figure 3.2: Overview of a Text-to-SQL Architecture. Source: [Mind Inventory](#)

- 1 Natural Language Understanding** The process begins with the user inputting natural language questions alongside SQL database schema into the LLM. Sometimes it is necessary to accompany these with documentation in cases where specialized knowledge is needed or even in the case of low resource languages. The LLM processes these inputs with serialized schema to understand the users questions, extract intent and key entities and relations.

- 2 **Schema Linking** The LLM maps entities it extracted from the natural language question to database schema elements (tables/columns). In this stage of the text-to-SQL task, ambiguity resolution is prioritized using schema-aware prompts for both the zero-shot and the few-shot in-context learning.
- 3 **Query Generation** When the schema linking process is finished, the LLM then generates SQL queries autoregressively based on the established semantic relationships with constraints like syntax validation through constrained decoding, clause ordering (SELECT → FROM → WHERE) and cross-domain generalization through fine-tuning.
- 4 **Query Execution and Output** The generated SQL is executed on a benchmark databases and execution results are returned as JSON for accuracy validation. These results could be converted back to natural language to ensure seamless interpretation. The accuracy of the generated SQL query is typically evaluated using metrics like Execution Accuracy (EX), Exact Match (EM), and Component Match, comparing the outputs or query structures against the ground truth queries.

3.1.9 Evaluation Metrics. Evaluation metrics for Text-to-SQL are crucial for quantifying model performance. Commonly used metrics include:

- 1 **Exact Matching (EM).** Exact Matching measures whether the SQL query generated by the model exactly matches the ground-truth SQL query character-for-character. This includes strict adherence to whitespace, punctuation, aliases, and keyword ordering (e.g., SELECT * FROM table vs. SELECT * FROM table). EM evaluates syntactic precision, penalizing even minor deviations such as extra spaces, alternate column orderings, or synonym usage (e.g., JOIN vs. INNER JOIN). While highly stringent, it ensures reproducibility in environments where query syntax must align perfectly with database constraints. In addition, EM fails to account for semantically equivalent but syntactically distinct queries (e.g., WHERE age > 30 vs. WHERE 30 < age) and is overly sensitive to trivial differences, making it less reflective of real-world usability. During evaluation, both predicted and gold-standard (reference) SQL queries are normalized by removing comments, extraneous whitespace and standardizing keyword casing. Mathematically, the Exact-Match score is the Kronecker delta

$$\text{EM}(\text{pred}, \text{gold}) = \delta(N(\text{pred}), N(\text{gold})) = \begin{cases} 1 & \text{if } N(\text{pred}) = N(\text{gold}), \\ 0 & \text{otherwise.} \end{cases} \quad (3.1.12)$$

where N is the a normalization function that standardizes SQL queries by removing quotes, case sensitivity, and whitespace differences, and resolving aliases or syntactic variations.

- 2 **Component Matching (CM):** Component Matching decomposes SQL queries into individual clauses (e.g., SELECT, FROM, WHERE, JOIN, GROUP BY, ORDER BY) and evaluates the accuracy of each component independently. A clause is considered correct only if it exactly matches the corresponding clause in the ground truth. CM identifies

structural weaknesses in the model’s parsing capabilities. For example, a model might excel at SELECT clauses but struggle with nested WHERE conditions. This granularity helps diagnose specific failure modes, guiding iterative improvements. SQL queries are parsed into abstract syntax trees (ASTs) using tools like `sqlglot`. Each AST node is compared to the ground truth, with partial credit awarded for partially correct clauses (e.g., 3 out of 4 correct WHERE conditions). The overlap between the gold-standard and predicted features can be represented mathematically. Let $F(q)$, the lowercase node names and literal values extracted from $\text{AST}(q)$ be the feature set returned by `flatten_ast` for a SQL query q . Then the component-match score is

$$\text{CM}(\text{pred}, \text{gold}) = \frac{|F(\text{pred}) \cap F(\text{gold})|}{|F(\text{gold})|}, \quad (3.1.13)$$

i.e. the fraction of structural tokens from the gold query that also appear in the predicted query.

- 3 **Execution Accuracy (EX).** Execution Accuracy was used to measure the semantic correctness of generated SQL by executing it against the target database and verifying that the returned results match those of the ground-truth query. EX prioritizes functional validity over syntactic rigor. A query passes if it retrieves the correct data, even if its structure diverges from the gold standard (e.g., using a subquery instead of a JOIN). This metric reflects real-world utility, where end-users prioritize accurate results over query structure. We can represent this mathematically. Let $R(q, d)$ be the sorted result set returned by executing SQL q on database d , for a dataset of N items $\{(\text{gold}_i, \text{pred}_i, d_i)\}_{i=1}^N$ the *execution-accuracy* is

$$\text{EA} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}[R(\text{pred}_i, d_i) = R(\text{gold}_i, d_i)], \quad (3.1.14)$$

i.e. the proportion of examples whose predicted query returns *exactly* the same result set (or identical error) as the gold query when run on the same SQLite database.

3.2 Research and Experimental Workflow

The experiments were conducted to systematically evaluate and improve the ability of large language models (LLMs) to perform Text-to-SQL parsing, first on the widely recognized Spider Benchmark and then on a specialized small-scale Chichewa dataset. The initial phase of the research utilized the Spider 1.0 benchmark, which provides a challenging and diverse set of natural language questions paired with complex SQL queries across many distinct database schemas. This allowed the study to test models’ generalization capabilities to unseen database structures and assess their baseline SQL generation performance. During these experiments, models were exposed to a fraction of the Spider training set through parameter-efficient QLoRA fine-tuning and evaluated using subsets of the validation data to balance computational efficiency with model development needs.

Prompt engineering played a critical role in guiding the models to translate natural language questions into SQL statements effectively. Three prompt templates were explored: zero-shot prompting to test the model's inherent ability without examples, random few-shot prompting to provide general demonstration examples, and retrieved few-shot prompting, which incorporated semantically similar training examples selected via embeddings to enhance contextual relevance. A deterministic beam search decoding strategy was employed to ensure consistent and reproducible SQL query generation, avoiding stochastic sampling that could introduce variability detrimental to the evaluation.

The QLoRA fine-tuning procedure allowed efficient adaptation of large models by updating only a small set of adapter parameters, reducing memory demands while maintaining high performance. This method involved sampling a subset of training data, shrinking the model backbone to a compressed 4-bit format, adding low-rank adapters to attention layers, and training over multiple epochs. This strategy enabled rapid fine-tuning on limited resources and facilitated modular deployment by loading the base model and adapter separately.

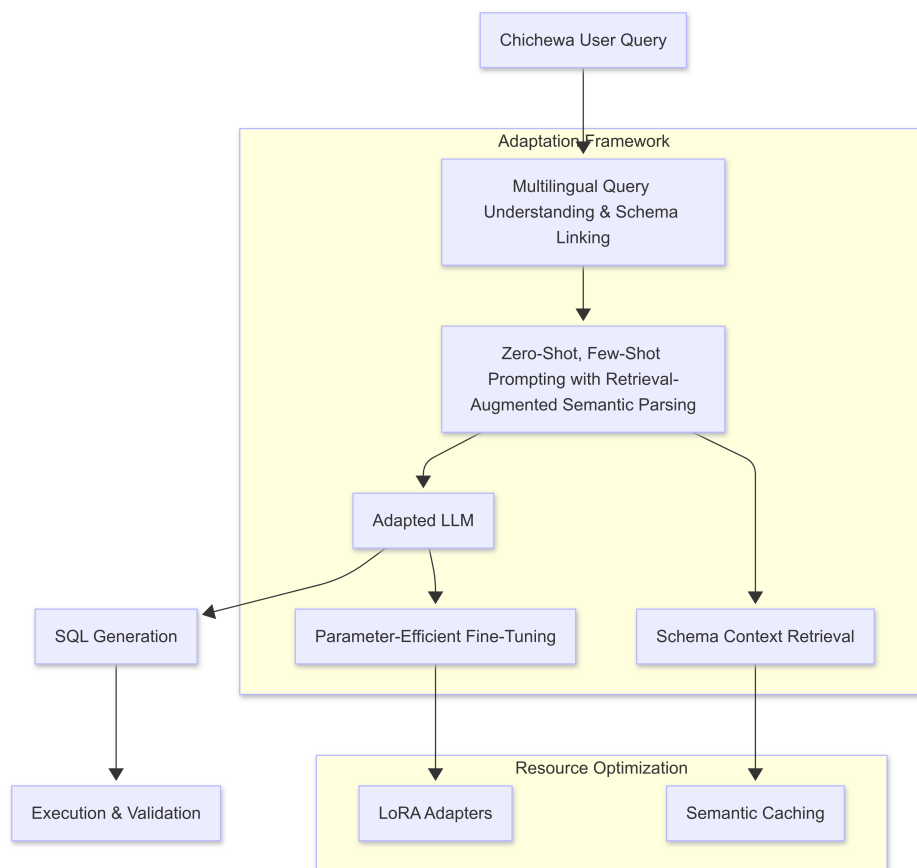


Figure 3.3: LLM Adaptation Workflow for Chichewa Text-to-SQL

Following the Spider experiments, adaptation efforts shifted to the Chichewa Text-to-SQL task as shown in Figure 3.3, which presents unique challenges such as limited data, multilingual complexities, and domain-specific vocabulary. A small benchmark containing three Malawi-related databases and over 200 question-SQL instances was used to evaluate models. Data preparation

ensured schema consistency and query correctness, leveraging schema serialization, normalization, and syntax verification.

A diverse selection of nine models was tested, spanning base and instruction-tuned LLMs, code-specialized models, and multilingual models, to explore different strengths in handling the Chichewa language and SQL generation. Five adaptation strategies were investigated to address linguistic and data scarcity bottlenecks: zero-shot prompting tested baseline model capabilities; English translation pivoting utilized automatic translation to bridge language gaps; multilingual retrieval-augmented semantic parsing leveraged semantically similar examples for contextual cueing; few-shot prompting provided explicit demonstrations to help schema linking; and finally, parameter-efficient QLoRA fine-tuning refined model parameters to capture lexical and structural nuances specific to Chichewa and the target databases.

Throughout all experiments, a strict and uniform evaluation protocol was maintained. Models generated a single SQL query per input question using fixed prompt templates and decoding parameters, and performance was assessed using three complementary metrics: Exact Match measured exact query equivalence, Component Match evaluated structural similarity via abstract syntax trees, and Execution Accuracy tested the correctness of the query outputs on the underlying databases. This comprehensive evaluation framework ensured fair comparisons across models and strategies, and helped isolate the contributions of each adaptation method to overall Text-to-SQL performance in both general and low-resource language contexts.

3.3 Text-to-SQL Experiments with Spider Benchmark

3.3.1 The Spider Benchmark. The research begins with using the Spider 1.0¹ Benchmark, a widely adopted dataset and evaluation framework for Text-to-SQL tasks, designed to test the ability of models to translate natural language questions into complex, domain-agnostic SQL queries.

The Spider dataset comprises 10,181 natural language questions paired with 5,693 unique SQL queries, spanning 138 distinct domains such as healthcare, education, sports, and entertainment. These questions are distributed across 200 databases, each with its own schema containing tables, columns, foreign key relationships, and data types. The dataset is partitioned into three splits: a training set (7000 examples, 70 databases), a validation set (1,034 examples, 20 databases), and a test set (2,147 examples, 40 databases). Crucially, the databases in the test set are entirely distinct from those in the training and validation sets, ensuring that models must generalize to completely unseen schemas.

The SQL queries in Spider range from simple single-table retrievals (e.g., `SELECT name FROM employees`) to highly complex operations involving multi-table joins, nested subqueries, aggregations (e.g., `COUNT`, `SUM`), and conditional clauses (e.g., `HAVING`, `ORDER BY`). For instance, a query like *"Find departments with more than 50 employees earning over \$100,000"* requires joining the employees and salaries tables, applying a `GROUP BY` clause, and filtering with a `HAVING` condition. Below is a concise explanation of how this data was used.

¹<https://yale-lily.github.io/spider>

- 1 **Training Data Usage.** For QLoRA fine-tuning, 20% of the training set (1,400 examples) was used from the HuggingFace rendering of the Spider Benchmark, for the model's exposure to diverse schemas and SQL structures. This ensures robust parameter updates even in resource-constrained settings. Also, during few-shot prompting, 5 examples were taken from this training data, first by simple random sampling and then by using the `all-MiniLM-L6-V2` sequence transformer to get the most semantically relevant examples for finetuning.
- 2 **Validation Data Usage** During development, 20% subset of the validation data (206 instances) was used to evaluate the models across all the methods considered. This subset serves as a lightweight proxy for quicker iteration, allowing you to test prompt designs and QLoRA configurations without incurring high computational costs.
- 3 **SQL Schema and Database Usage.** Database schemas were serialized into flattened strings to integrate with natural language prompts. During training, schemas were dynamically injected into prompts with the Kaggle rendering of the same dataset using HuggingFace mapping utilities. At inference, schemas were preloaded into a SQLite instance, and generated queries were executed via Python's `sqlite3` module. The gold-standard SQL contains the reference queries for which the predicted SQL were measured using Exact Match (EM) and Component Match (Match) metrics while their outputs were measured using Execution Accuracy (EX).

3.3.2 Prompt Engineering. This section explains the interactions made with the language models. It describes the prompt design and decoding choices as well as how they are adapted to the Spider corpus using a parameter-efficient QLoRA procedure. Here is contained a detailed breakdown on how the three prompt templates used in this research. They are as follows:

1. **Zero-shot.** This contains the instruction, schema, and NL question blocks. The instruction block contains a minimal, single-turn instruction as a direct Llama-3 and DeepSeek checkpoints which already contain general instruction-following skills.
2. **Random 5-shot.** This contains five fixed training examples drawn once by simple random sampling with, each providing its own schema, NL question and gold SQL, followed by the target schema and question. Because the examples themselves already illustrate the mapping task in detail, the instruction section is kept short and direct; the five shots supply the contextual cues the model needs to infer the NL-to-SQL pattern.
3. **Retrieved 5-shot.** In this case, the retrieval augmented semantic parsing (RASP) method was used in which k-nearest-neighbour search was performed for every new question using cosine similarity on MiniLM sentence-embeddings (`all-MiniLM-L6-v2`) and the $k = 3$ semantically closest training questions were selected. Their demonstrations, with the same concise prompt instruction as the random version, were then prepended to the target schema and question.

Every prompt terminates with the sentinel `### SQL:` so that generation begins at a predictable token. Generated texts were trimmed by, removing everything after the first sentinel like blank

line, back-ticks and special characters and, keeping the substring before the first semicolon, because Spider queries are single-statement.

3.3.3 Decoding Strategy. For all Spider experiments, SQL queries were generated with *deterministic beam search*. Beam search maintains a small set of the highest-probability partial hypotheses at every decoding step, allowing the model to revise earlier choices while still converging to a single, reproducible output. We keep `do_sample` disabled throughout the study. Stochastic decoding (top-pp, temperature) was not used as it can diversify outputs, but also inject run-to-run variance that complicates controlled ablation studies; moreover, Spider evaluation penalises even benign lexical variants if they alter execution results. A deterministic beam therefore remains the best balance between quality, repeatability, and resource use for low-resource Text-to-SQL setting.

3.3.4 QLoRA Fine-tuning Procedure. To adapt the open-source models we used needing a super-computer, we used QLoRA which trains only a tiny, trainable add-on matrix while keeping the rest of the network frozen and stored in an ultra-compressed format. Below is a detailed breakdown of the process.

1. **Training Data Sampling and Prompt Construction** The process begins by drawing a simple random sample of 20% of the Spider training questions (1,400 examples) and the data was converted into full a single-turn instruction-plus-answer strings to serve as supervised input for QLoRA fine-tuning that mirrors the format the model will see at inference time
2. **Shrink the frozen backbone.** The finetuning process continues by loading the best performing baseline model which is Deepseek Coder 6.7 Base model in NF4 4-bit with BitsAndBytesConfig. This slashed GPU memory from roughly 60 GB to about 16 GB while still letting the model run at full precision for calculations.
3. **Add a lightweight adapter.** Instead of rewriting all of the frozen weights, we attached two small rank-64 matrices (the LoRA adapter) to each attention layer with the configurations `r = 64`, `lora_alpha = 128`, target modules: `{q_proj, k_proj, v_proj, o_proj}`, `dropout = 0.05`. During fine-tuning the model only learns the values in these thin matrices—less than 1 % of the original parameter count—so training is fast and the final file is a few hundred megabytes rather than many gigabytes.
4. **Model Training.** The 20% sample training questions were passed through the model for five passes (five epochs) and tracked by steps recording each 50 steps. A global batch of 16 examples was simulated with micro-batches of 2, accumulating gradients to stay within the GPU’s memory limits. Activations were checkpointed so they could be recomputed on the fly, halving peak memory at the cost of a small speed hit. Learning rate was set to 2×10^{-4} with AdamW and training was done using SFTTrainer.
5. **Save only what changed.** After training, we stored just the LoRA adapter and its configuration. At inference time we reload the original 4-bit model and “snap on” the adapter, which keeps peak GPU usage below the 40 GB available on a single NVIDIA A100 card.

3.3.5 Metrics and Evaluation Methods. All evaluation runs (zero-shot, random 5-shot, retrieved 5-shot) were done without finetuning and repeated using the fine-tuned adapters (QLoRA + prompt) without changing the prompt structure or decoding parameters, allowing a clean evaluation and comparison. The metrics used were exact match, component match and execution accuracy.

3.4 Text-to-SQL Adaptation Strategies for Chichewa

3.4.1 Benchmark Dataset. A small scale Chichewa benchmark which contains containing 3 databases, namely; Food Agriculture, Prices and Food Insecurity as well as a training and validation dataset with over 200 instances combined. Contained in each instance is an English question (EN), a Chichewa question (NY), the database name, the SQL query and response as is shown in Figure 3.4. The Food Agriculture DB contains crops, the districts in Malawi where they were produced and their seasonal yeild. The Prices DB contains information about the commodity, their prices, the particular date they were harvested along with where they were harvested. Finally, the Food Insecurity DB contains metrics like analyzed population, insecurity level, description of insecurity as well as districts where these insecurities are observed.

```
1 {"Question_EN": "Which district produced the most Macademia in the 2023-2024 season?", "Question_NY": "Ndi boma lanji yomwe idakolola magede ochuluka mu chaka ya 2023-2024?", "SQL": "SELECT District, MAX(Yield) as Max_Yield FROM production WHERE Crop = 'Macademia' AND Season = '2023-2024';", "database_name": "food_agriculture", "SQL_Response": "Mangochi (4998.0)"}
2 {"Question_EN": "How much Apples were harvested in Dowa during the last season?", "Question_NY": "Ndima appples ochuluka bwanj amene adakololedwa ku Dowa munyengo yathayi?", "SQL": "SELECT Yield FROM production WHERE District = 'Dowa' AND Crop = 'Apples';", "database_name": "food_agriculture", "SQL_Response": "0.0"}
3 {"Question_EN": "What was the total yield of Velvet beans in the 2023-2024 season?", "Question_NY": "Ndi nyemba zochuluka bwanji zimene zidakololedwa mu nyengo ya 2023-2024?", "SQL": "SELECT SUM(Yield) as Total_Yield FROM production WHERE Crop = 'Velvet beans' AND Season = '2023-2024';", "database_name": "food_agriculture", "SQL_Response": "18054.0"}
4 {"Question_EN": "How much Wheat was produced in Nkhotakota?", "Question_NY": "Ndi tiligu ochuluka bwanj adakololedwa mu boma la Nkhotakota?", "SQL": "SELECT Yield FROM production WHERE District = 'Nkhotakota' AND Crop = 'Wheat';", "database_name": "food_agriculture", "SQL_Response": "0.0"}
```

JSON ▼ Tab Width: 8 ▼ Ln1, Col 348 ▼ INS

Figure 3.4: Example Data with English and Chichewa Questions, Database Name, SQL Queries and Responses

3.4.2 Data Preparation. Chichewa questions, English translations (where available), and SQL queries were loaded from a line-delimited JSON file. All SQL queries were validated to confirm that referenced tables and columns matched the database schemas. Database schemas were programmatically extracted from SQLite files by listing table and column names. These schemas were converted to textual form and included in all model prompts to provide contextual grounding for SQL generation. Both SQL queries and natural language questions were normalized by trimming whitespace and collapsing multiple spaces into single spaces. Additionally, SQL comma

spacing was standardized with exactly one space after each comma to ensure consistency during exact match evaluation. SQL statements were parsed using the `sqlglot` library to verify syntactic correctness. Queries failing parsing were either corrected or excluded. The resulting abstract syntax trees (ASTs) were used in component-based similarity metrics to assess structural query alignment. Each SQL query was executed against its respective SQLite database to ensure it ran successfully and returned results without errors. Queries producing errors were flagged and fixed if possible.

3.4.3 Model Selection. Considering the task at hand, which is to adapt LLMs to Chichewa in the Text-to-SQL scenario, we chose different baseline models, some were base models, instruction-tuned models (which can easily be guided by in context learning and finetuning as they are tuned to follow instructions), models trained predominantly on code (which is presumed to have good SQL prediction capabilities) and multilingual models (which were trained on different low resource languages including Chichewa). Table 3.1 contains a detailed classification of the 9 open-source LLMs considered for this task. They are all publicly available on HuggingFace².

Table 3.1: Model Classification for Chichewa Text-to-SQL Adaptation

| Model Key | Model Family | Model Type | Primary Capability | Language Focus |
|------------------------------|--------------|------------|--------------------|----------------|
| mistral_7b_base | Mistral | Base | General NLP | English |
| mistral_7b_instruct | Mistral | Instruct | General NLP | English |
| llama3_8b_base | Llama 3 | Base | General NLP | English |
| llama3_8b_instruct | Llama 3 | Instruct | General NLP | English |
| deepseek_coder_6.7b_base | DeepSeek | Base | Code generation | English |
| deepseek_coder_6.7b_instruct | DeepSeek | Instruct | Code generation | English |
| sqlcoder_7b | Defog | Instruct | SQL generation | English |
| codellama_7b_instruct | CodeLlama | Instruct | Code generation | English |
| bloom_7b | BLOOM | Instruct | Multilingual NLP | Multilingual |

3.4.4 Proposed Text-to-SQL Adaptation Strategies. To ensure that the system can robustly parse Chichewa questions into executable SQL, five complementary adaptation routes are explored. Each tackles a different bottleneck, ranging from language coverage to or data scarcity—and, in combination, they form a layered experimentation grid. Below are the different approaches employed to make this adaptation possible.

1. **Zero-Shot Prompting.** First, zero-shot prompting over the LLMs listed in table 3.1 was done with only an *instruction*, the *database schema*, and the raw Chichewa question. The models are tested whether their latent knowledge are enough to produce syntactically valid queries. Results were recorded and improved upon with the subsequent methods.
2. **English Translation Pivot.** In a bid to improve results from baseline zero-shot prompts, an automatic translation was used to convert the Chichewa utterance into English before SQL generation using the NLLB-200 3.3 B open multilingual translation model which was

²<https://huggingface.co/>

pretrained on low resource languages including Chichewa. The translated text is then fed to a model with the original schema.

3. **Multilingual Retrieval-Augmented Semantic Parsing (RASP).** This method improves the LLM ability to interpret rare vocabularies. The LLMs are kept unchanged but its prompts are edited to retrieve the k most similar Chichewa examples from the training set. Similarity is computed with paraphrase-multilingual-MiniLM-L12-v2, a 13-layer, 118M-parameter encoder fine-tuned on 50+ languages. For each retrieved question we insert its gold SQL and the shared schema, giving the LLM concrete lexical cues.
4. **Few-Shot Prompting.** We then prepend $k = 3$ demonstration blocks, each containing (*schema*, *Chichewa question*, *gold SQL*). Few-shot in-context learning helps the model can learn domain-specific patterns (crop names, district filters, seasonal groupings) without any parameter updates. Inlining the schema inside every shot is critical: it forces the model to repeatedly see how column names map onto the low-resource language tokens, thereby improving schema linking.

Table 3.2: Case study illustrating a successful and a failed prediction

| Aspect | Example A – Success | Example B – Failure |
|---------------|---|---|
| Chichewa Q | Ndi mbeu yanji idali ndi zokolola zochu-luka ku Ntchisi mu chaka cha 2023-2024? | Kodi zokolola za <i>lupa</i> mu nyengo ya 2022-2023 zidali zingati ku Dowa? |
| Gold SQL | SELECT Crop, Yield FROM production WHERE District='Ntchisi' AND Season='2023-2024' ORDER BY Yield DESC LIMIT 1; | SELECT SUM(Yield) FROM production WHERE Crop='Lupin' AND Season='2022-2023' AND District='Dowa'; |
| Predicted SQL | same as gold (Bloom-QLoRA) | missed crop filter, returned maize totals |
| EM / EA | 1 / 1 | 0 / 0 |
| Diagnostic | Model copied pattern from retrieved demo correctly. | Retrieval demo used maize; model over-generalised. |

3.4.5 Evaluation Metrics and Inference. Table 3.2 indicates which prediction is flagged as successful or failed. All evaluation runs including zero-shot, few-shot, translation-pivot, retrieval-augmented, or fine-tuned, were conducted on the same held-out set of Chichewa questions using identical prompt structures and decoding settings. In every case, the model was asked to emit a single valid SQL statement, and no further prompt tweaking was allowed between runs. For each model and adaptation strategy, the average Exact Match, average Component Match, and average Execution Accuracy were reported over all Chichewa questions in the evaluation set. By holding prompts, decoding settings, and post-processing constant, this uniform protocol ensures a fair comparison of all methods.

4. Evaluation and Results

4.1 Experimental Setup

- **Computing environment.** Experiments were run in Google Colab Pro on $1 \times$ NVIDIA A100–40 GB GPU.
- **Model.** Only open-source models hosted on HuggingFace as listed in Table 3.1 were used throughout this study.
- **Benchmark Dataset.** All evaluations were made on the validation split of the Spider 1.0 Text-to-SQL benchmark and the small-scale Chichewa dataset created. To keep runtime reasonable, 20 % of the the training split $n = 1400$ of Spider was used in finetuning QLoRA and also 20% of the validation split ($n = 206$) for every different method considered. 31 instances were considered from the Chichewa Dataset with DB name being Food Agriculture. The data was fixed to seed=42 for the sake of reproducibility.
- **Database access.** A lightweight reflection routine produced a compact *schema string* for every database ID so as to enforce schema linking. All Spider benchmark SQLite databases were obtained from a Kaggle mirror dataset¹. For the Chichewa case the databases
- **Project Repository.** The Chichewa benchmark dataset and all notebooks containing data preprocessing, prompt engineering, QLoRA fine-tuning, and evaluation are publicly available in our [GitHub repository](#).²

4.2 Results from Spider Benchmark Experiment

The results from the evaluation of the baseline (zero-shot), few-shot, and QLoRA configurations demonstrate how model performance improves with more advanced techniques. Figure 4.1 focuses on the metric performance while Figure 4.2 focuses on Model comparison. Here is a detailed breakdown on the results and inferences made from them.

4.2.1 Zero-Shot Learning. In the zero-shot case, where the models were expected to generate SQL queries without prior examples, Llama-3.1 8B performed quite poorly with an Exact Match of only 9.2% and DeepSeek Coder 6.7B Base did better with 18.4%. This suggests that while the model can capture some aspects of SQL query structure, it struggles to match the exact wording of the gold-standard SQL. The Component Match was higher at 67.3% for Llama and 83.9% for DeepSeek, indicating the model understood the structure of SQL better than it generated the exact query. Additionally, the Execution Accuracy was 44.6% for Llama and 61.2% for DeepSeek, showing that the model was somewhat successful at executing the queries but had room for improvement.

¹<https://www.kaggle.com/datasets/jeromeblanchet/yale-universitys-spider-10-nlp-dataset/data>

²<https://github.com/johnemekaeze/LowResource-Text2SQL-Chichewa>

4.2.2 Few-Shot (Random Examples). When we moved to few-shot learning, where the model was trained on 5 randomly selected examples, the results showed an interesting trend. The Exact Match improved to 12.1% for Llama but dropped to 15.5% for DeepSeek, which suggests that the random selection of examples didn't align well with the validation set, and the model struggled to learn patterns from irrelevant examples. However, the Component Match improved slightly to 85.1% for Llama and 90.4%, and the Execution Accuracy rose slightly to 53.4%, suggesting that while the model performed better in understanding SQL structure, it still faced challenges with exact query generation.

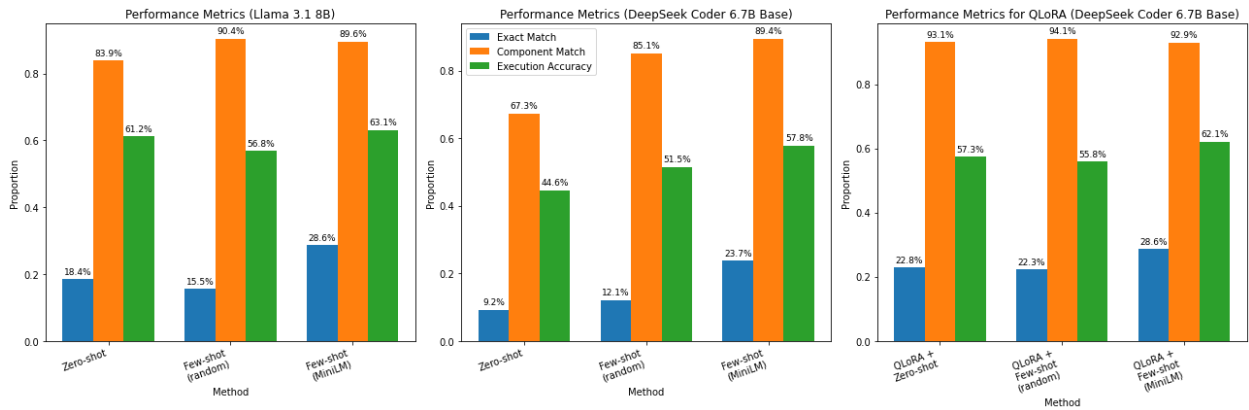


Figure 4.1: Performance Metrics

4.2.3 Few-Shot (MiniLM Embeddings). In contrast, when we used few-shot learning with examples selected using MiniLM embeddings (semantic retrieval), there was a clear improvement in performance. The Exact Match improved to 23.7% for Llama and 28.6% for DeepSeek, indicating that using semantically relevant examples led to better alignment between the question and SQL query. Component Match improved to 89.4% for Llama but dropped slightly to 89.6% for DeepSeek, and Execution Accuracy increased to 57.8% for Llama and 63.1% for DeepSeek. These results highlight the benefit of using semantically similar examples for few-shot learning, as they help the model better capture the patterns in the data.

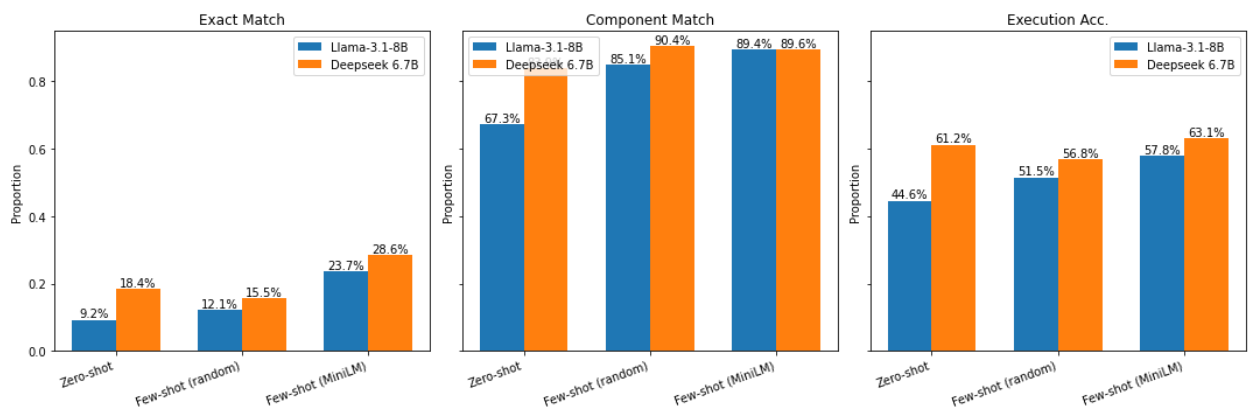


Figure 4.2: Model Comparison across the Different Metrics

4.2.4 QLoRA Fine-Tuning with Prompt Engineering. Since DeepSeek outperformed Llama across all the different metrics as shown in Figure 4.2, we finetuned it alone and the result is shown in the third panel of Figure 4.1. The Exact Match for QLoRA + Zero-Shot and QLoRA + Few-shot (random 5 examples) reached 22.8% and 22.3% respectively, and further increased to 28.6% for QLoRA + Few-Shot (MiniLM embeddings), a jump from the previous configurations, suggesting that fine-tuning allows the model to generate more accurate and relevant SQL queries. The Component Match rose to 93.1% and 94.1% but dropped a little to 92.9%. This indicates that fine-tuning not only improved exact matching but also made the model better at understanding SQL structures. The Execution Accuracy also improved to 61.2%, showing that fine-tuning with relevant examples boosts the model’s ability to execute SQL queries correctly. There was not a considerable increment for Execution accuracy.

Overall, the results showed that QLoRA fine-tuning combined with semantically relevant few-shot examples provides the best performance across all metrics, particularly in Exact Match and Execution Accuracy. Fine-tuning significantly boosts the model’s ability to generate correct SQL queries, making it more reliable for tasks like Text-to-SQL. This analysis suggests that fine-tuning with more focused data and using models that leverage semantic retrieval (such as MiniLM) can greatly enhance the performance of zero-shot and few-shot models, making them more effective for complex tasks like Text-to-SQL

4.3 Results from Chichewa Experiment

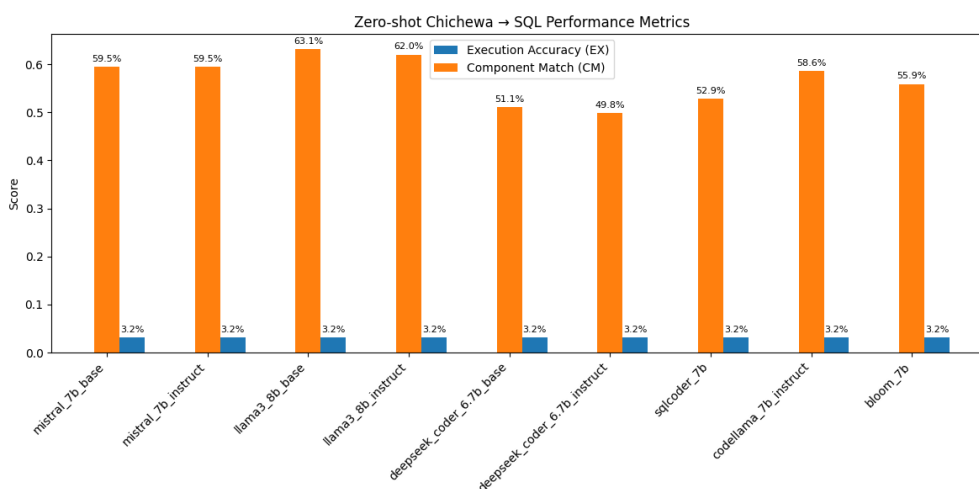


Figure 4.3: Zero-Shot Performance Directly from Chichewa to SQL

4.3.1 Zero-Shot Chichewa-to-SQL. Figure 4.3 presents the zero-shot performance of the 9 different models used for the task of converting Chichewa questions into SQL queries. We see that Component Match (CM) is fair across most models, with values ranging from 49.8% to 63.1%. This suggests that the models are capable of identifying the key components of SQL queries, such as tables and columns, even if they fail to produce the exact syntax. However, Execution Accuracy (EA) remains very low across all models at 3.2%, which indicates that while

the models are able to understand the components of the question and the database schema, they are struggling with generating SQL queries that actually execute correctly in the database. This can be attributed to small syntax errors or mismatches that prevent the generated SQL from returning the expected results. The Exact Match (EM) is also consistently low, which implies that the models fail to generate SQL queries that exactly match the reference SQL. This is due to the inherent difficulty of SQL generation in low-resource languages, where the models are able to recognize patterns but not consistently replicate them in the exact form expected. Models such as Llama3 and Mistral, both base and instruct show relatively better Component Match but still suffer from poor Execution Accuracy and Exact Match, pointing to the need for more specialized training or fine-tuning.

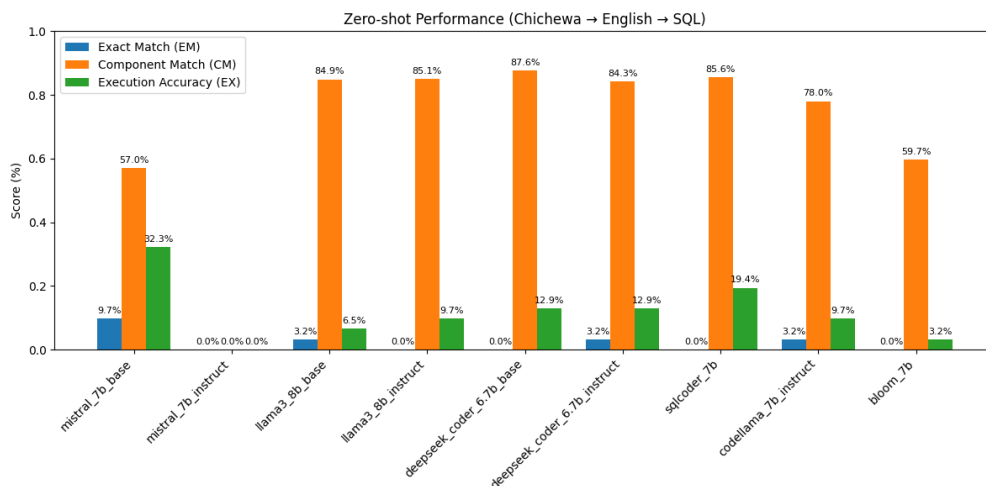


Figure 4.4: Zero-Shot Performance for English Translations of Chichewa Questions to SQL

4.3.2 Zero-Shot English Translations to SQL. From Figure 4.4, the Component Match (CM) is strong across most models, showing high values between 47.0% and 87.6%, except for Mistral Instruct. This suggests that despite issues with exact match, the models understand the structural components of SQL, like tables, columns, and basic operators quite well. The Execution Accuracy (EA) is generally low, with values mostly around 3.2% and 32.3%. This reflects that while the models can identify SQL components, they fail to generate syntactically valid SQL that matches the gold-standard SQL when executed on the database. Exact Match (EM) is particularly low for most models, suggesting that even though the models understand SQL structure well, they are not producing the exact query syntax expected. One notable thing is that although Mistral Base performs the poorest in Component Match, It outperforms the other models in both Exact Match and Execution Accuracy.

4.3.3 Few-shot + RASP Chichewa-to-SQL. Figure 4.5 compares the Few-shot + RASP performance on the test set, showing a much more stable trend across models. A uniform performance can be seen across all models because of the limited availability of data. This caused the models to learn the same thing across all the metrics. The Component Match (CM) is strong across all models, showing high values 78.4%, indicating that the models are good at identifying SQL components like tables, columns, and basic operators. The Execution Accuracy (EA) is generally low, with values at 15.4%. This reflects that while the models can identify SQL

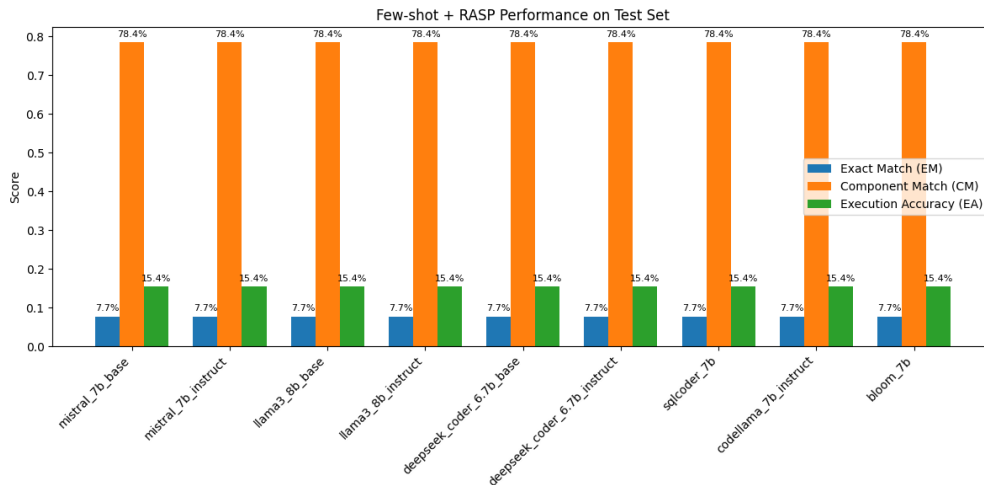


Figure 4.5: Performance for Few-Shot + Retrieval Augmented Semantic Parsing

components, they fail to generate syntactically valid SQL that matches the gold standard when executed on the database. The Exact Match (EM) is at 7.7% across all models, indicating that while few-shot learning and retrieval augmentation can help with the overall structure of SQL queries, they do not significantly improve the exact matching of queries to the ground truth.

The overall conclusion from is that while the models perform reasonably well in identifying the structure of SQL queries, they still struggle with exact syntax generation and execution accuracy. Fine-tuning on task-specific data and utilizing advanced techniques like retrieval-augmented few-shot learning and LoRA/QLoRA fine-tuning can help improve performance for Exact Match (EM) and Execution Accuracy (EA).

5. Conclusion and Recommendation

5.1 Conclusion

This research aimed to bridge the gap between high-resource language models and low-resource languages (LRLs) by focusing on Chichewa, a Bantu language spoken in Malawi, Zambia, and Zimbabwe. The study centered around adapting large language models (LLMs) for Text-to-SQL tasks, which allows non-technical users to query relational databases using natural language.

Through a series of experiments, we demonstrated that despite challenges such as the scarcity of Chichewa data, dialectal differences, and the absence of standardized NLP resources for the language, it is feasible to develop a system that can accurately translate Chichewa natural language queries into SQL. We utilized various techniques, including zero-shot prompting, multilingual retrieval-augmented semantic parsing, few-shot learning, and parameter-efficient fine-tuning methods such as QLoRA. These strategies allowed us to tackle issues such as schema linking, linguistic complexity, and data scarcity while enhancing the performance of LLMs in low-resource settings.

The experiments conducted on the Spider benchmark and the Chichewa Text-to-SQL dataset show that models can be adapted to new languages, achieving improved performance metrics in both exact match, component match, and execution accuracy. The results indicate that the proposed cross-lingual transfer framework, combining unsupervised translation and schema-aware prompts, can be applied effectively to Chichewa, offering a strong foundation for extending similar approaches to other underrepresented languages.

5.2 Recommendation

A key limitation of this study was the limited availability of annotated Chichewa-NL-SQL pairs. To address this, future work should focus on creating comprehensive, human-annotated datasets that will improve the quality of training data and enhance the generalization ability of the models. Such efforts could be driven through community-driven initiatives or collaborations with local institutions, which would also provide more authentic data that reflects real-world usage.

Another significant challenge identified in this study was the dialectal variability within Chichewa, especially between Malawi and Zambia. Future research should aim to capture these regional differences to improve the model's robustness and ensure it can handle diverse inputs accurately. This could involve the development of region-specific datasets that reflect the different dialects, which will be crucial for building systems that cater to the linguistic diversity within the Chichewa-speaking population.

While parameter-efficient fine-tuning techniques, such as QLoRA, demonstrated promising results, there is potential for improving performance by expanding these techniques to include larger portions of the model. Fine-tuning more significant parts of the model could enhance

the system's ability to process more complex queries. Further exploration of how to fine-tune large language models with limited computational resources would be beneficial, particularly in resource-constrained settings where such models are most needed.

In addition, this study primarily focused on relational database schemas in the context of Chichewa. Expanding the research to include other domains, such as healthcare or agriculture, would offer valuable insights into how these models perform in varied real-world contexts. Cross-domain generalization remains a challenge for Text-to-SQL tasks, and further research should explore strategies to develop adaptable models that can be applied effectively across multiple industries.

The approach demonstrated in this research has the potential to extend to other African languages and low-resource languages (LRLs), promoting digital inclusivity across the continent. By developing a multilingual framework capable of handling a range of languages, we can help bridge the digital divide and empower marginalized communities to make data-driven decisions in their native languages.

Lastly, future efforts should explore how these Text-to-SQL systems can be implemented in practical, user-centric applications. Integrating them into real-world platforms within sectors like healthcare, agriculture, and education can enhance their impact. Developing interfaces that allow non-technical users to interact with databases in their native languages will further enable these systems to serve as valuable tools for communities with limited technical expertise. Agentic approaches could be explored as well. For instance, an AI agent that iteratively plans, drafts, executes, and validates SQL, using a tool loop (LLM \leftrightarrow database) to self-correct errors. Such agent-based chains, combined with retrieval and code execution, may close the current gap in execution accuracy without requiring massive task-specific fine-tuning.

References

- E. Adamopoulou and L. Moussiades. Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2020.
- A. Agyemang et al. Jw300: A wide-coverage parallel corpus for low-resource languages. In *Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC 2020)*, pages 335–345, 2020.
- J. Alabi et al. Adapting pre-trained language models to african languages via multilingual adaptive fine-tuning. 2022. URL <https://arxiv.org/abs/2204.06487>. arXiv preprint arXiv:2204.06487.
- F. S. Banitaba, S. Aygun, and M. H. Najafi. Late breaking results: Fortifying neural networks: Safeguarding against adversarial attacks with stochastic computing. In *arXiv preprint*, 2024.
- T. B. Brown, B. Mann, N. Ryder, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 2020.
- S. Chang and E. Fosler-Lussier. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings. *NeurIPS 2023 Table Representation Learning Workshop*, 2024.
- M. Chen, J. Tworek, H. Jun, et al. Evaluating large language models trained on code. *arXiv preprint*, 2021.
- X. Chen, M. Lin, N. Schärli, and D. Zhou. Teaching large language models to self-debug. *arXiv preprint*, 2023a.
- Xiang Chen et al. Evaluating generalization in text-to-sql parsing, 2023b. arXiv:2305.11589.
- Edgar F Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.
- A. Conneau and G. Lample. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems*, 2019.
- A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and A. Joulin. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.
- C.J. Date. *An Introduction to Database Systems*. Addison Wesley, 2004.
- Tim Dettmers et al. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

- X. Dong, C. Zhang, Y. Ge, et al. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint*, 2023.
- Z. Dou, D. Sherry, and M. Johnson. Multispider: A massively multilingual text-to-sql benchmark. *Findings of EMNLP*, 2022.
- P. Einolghozati, M. Azizi, and S. R. Joty. Dialect2sql: Bridging dialectal arabic and sql parsing. *North American Chapter of the Association for Computational Linguistics*, 2021.
- Fortune. Fortune global 500 2024, 2024. URL <https://fortune.com/ranking/global500/>. Retrieved 2024.
- Y. Gan, X. Chen, and M. Purver. Exploring underexplored limitations of cross-domain text-to-sql generalization. *arXiv preprint*, 2021.
- D. Gao, H. Wang, Y. Li, et al. Text-to-sql empowered by large language models: A benchmark evaluation. *Proceedings of the VLDB Endowment*, 17(5):1132–1145, 2024. doi: 10.14778/3641204.3641221.
- T. Gao, A. Fisch, and D. Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.
- N. Goyal, C. Gao, V. Chaudhary, et al. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538, 2022. doi: 10.1162/tacl_a_00474.
- T. Guo and H. Gao. Content enhanced bert-based text-to-sql generation. *arXiv preprint*, 2019.
- L. Hammami, A. Paglialonga, G. Pruneri, et al. Automated classification of cancer morphology from italian pathology reports using natural language proc@miscli2023bird, title=BIRD: Benchmarks for Individualized Realistic Databases, author=Li, Jiayi and others, year=2023, how-published=arXiv preprint arXiv:2306.01678, note=Accessed: 2024-05-01 essing techniques. *Journal of Biomedical Informatics*, 116:103712, 2021.
- Z. Hong, Z. Yuan, Q. Zhang, et al. Next-generation database interfaces: A survey of llm-based text-to-sql. *arXiv preprint*, 2024.
- N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.
- E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint*, 2021. URL <https://arxiv.org/abs/2106.09685>.
- Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L  lio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak,

- Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts: Sparse mixture of experts for efficient language modeling. <https://mistral.ai/news/mixtral-of-experts/>, 2023. Accessed: 2024-05-01.
- P. Joshi, S. Santy, A. Budhiraja, et al. The state and fate of linguistic diversity and inclusion in the nlp world. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.
- A. B. Kanburoğlu and F. B. Tek. Text-to-sql: A methodical review of challenges and models. *Turkish Journal of Electrical Engineering and Computer Sciences*, 32(3):403–419, 2024. doi: 10.14778/3641204.3641221.
- N. Kang, B. Singh, Z. Afzal, et al. Using rule-based natural language processing to improve disease normalization in biomedical text. *Journal of the American Medical Informatics Association*, 20(5):876–881, 2013.
- G. Katsogiannis-Meimarakis and G. Koutrika. A deep dive into deep learning approaches for text-to-sql systems. *Proceedings of the 2021 International Conference on Management of Data*, pages 2846–2851, 2021.
- A. Kumar, P. Nagarkar, P. Nalhe, and S. Vijayakumar. Deep learning driven natural languages text to sql query conversion: A survey. *arXiv preprint*, 2022. URL <https://arxiv.org/abs/2208.04415>.
- G. Lample and A. Conneau. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.
- F. Li and H. V. Jagadish. Constructing an interactive natural language interface for relational databases. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 78–82, 2014.
- H. Li. Deep learning for natural language processing: Advantages and challenges. *National Science Review*, 5(1):24–26, 2018.
- Jiayi Li, Lianhui Qin, Xiao Lu, Jialong Xu, Yankai Xu, Zonghai Yang, Yuwei Wang, and Minlie Huang. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL <https://arxiv.org/abs/2306.01678>.
- X. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.
- A. Liu, X. Hu, L. Wen, and P. S. Yu. A comprehensive evaluation of chatgpt’s zero-shot text-to-sql capability. *arXiv preprint*, 2023.

- T. Mahmud, K. A. Hasan, M. Ahmed, and T. H. C. Chak. A rule based approach for nlp based query processing. *2015 2nd International Conference on Electrical Information and Communication Technologies (EICT)*, pages 78–82, 2015.
- D. Matekenya. Chichewa speech dataset, 2022. URL <https://doi.org/10.5281/zenodo.6595625>. Zenodo.
- Meta AI Research Team. No language left behind: Scaling human-centered machine translation. *arXiv preprint*, 2022.
- B. Min, H. Ross, E. Sulem, et al. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2), 2023.
- A. Mohammadjafari, A. S. Maida, and R. Gottumukkala. From natural language to sql: Review of llm-based text-to-sql systems. *arXiv preprint*, 2023. doi: 10.48550/arXiv.2410.01066.
- W. Nekoto et al. Participatory research for low-resource machine translation: A case study in african languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 195–210, 2020.
- OpenAI. Gpt-4 technical report. *arXiv preprint*, 2023.
- S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- G. Poesia, O. Polozov, V. Le, et al. Synchromesh: Reliable code generation from pre-trained language models. *arXiv preprint*, 2022.
- N. Rajkumar, R. Li, and D. Bahdanau. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint*, 2022.
- P. Scholak, Y. Elazar, and Y. Goldberg. Picard: Pretrained autoregressive constrained decoding for text-to-sql parsing. *EMNLP*, 2021.
- R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. *ACL*, 2016.
- P. Shu, J. Chen, Z. Liu, et al. Transcending language boundaries: Harnessing llms for low-resource language translation. *arXiv preprint*, 2023.
- P. J. Suárez et al. Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In *Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC)*, pages 33–43, 2019.
- M. H. Tahir, S. Shams, L. Fiaz, F. Adeeba, and S. Hussain. Benchmarking pre-trained large language models' potential across urdu nlp tasks. *arXiv preprint*, 2024. URL <https://arxiv.org/abs/2405.15453>.
- A. Taylor. Spokenchichewacorporus (1.0), 2020. URL <https://doi.org/10.5281/zenodo.3731994>. Zenodo.

- A. Taylor and P. Kazembe. Assessing language barriers in health facilities in malawi. *BMC Health Services Research*, 24(1):1393, 2024. doi: 10.1186/s12913-024-11901-4.
- H. Touvron, T. Lavril, G. Izacard, et al. Llama: Open and efficient foundation language models. *arXiv preprint*, 2023.
- Ashish Vaswani et al. Attention is all you need. In *Advances in neural information processing systems*, 2017.
- H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun. Pre-trained language models and their applications. *Engineering*, 25:51–65, 2023.
- J. Wei and K. Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6381–6387, 2019. URL <https://aclanthology.org/D19-1670/>.
- B. Xiao, C. Deli, C. Guanting, C. Shanhuang, D. Damai, D. Chengqi, D. Honghui, D. Kai, D. Qiushi, F. Zhe, G. Huazuo, G. Kaige, G. Wenjun, G. Ruiqi, et al. Deepseek llm: Scaling open-source language models with longtermism. <https://huggingface.co/deepseek-ai>, 2024.
- T. Yu, R. Zhang, V. Zhong, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *EMNLP*, 2018.
- T. Yu, R. Zhang, H. Y. Er, S. Li, E. Xue, B. Pang, X. V. Lin, Y. C. Tan, T. Shi, Z. Li, et al. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *arXiv preprint*, 2019. URL <https://arxiv.org/abs/1909.05378>.
- X. Zhang, Q. Meng, and J. Bos. Retrieval-augmented semantic parsing: Using large language models to improve generalization. *arXiv preprint*, 2024. URL <https://arxiv.org/abs/XXXX.XXXXX>. Submitted on 13 Dec 2024.
- W. X. Zhao, K. Zhou, J. Li, et al. A survey of large language models. *arXiv preprint*, 2023.
- X. Zhu. Semi-supervised learning literature survey. 2005.