T.C.
IŞIK UNIVERSITY
SCHOOL OF GRADUATE STUDIES

DOCTORAL THESIS
DEPARTMENT OF COMPUTER ENGINEERING
PROGRAM

Ali Buğra KANBUROĞLU

LARGE LANGUAGE MODEL BASED AUTOMATED TRANSLATION OF NATURAL LANGUAGE TO SQL

SUPERVISOR
Assoc. Prof. Faik Boray TEK

İSTANBUL, January 2025

T.C.
IŞIK UNIVERSITY
SCHOOL OF GRADUATE STUDIES

DOCTORAL THESIS
DEPARTMENT OF COMPUTER ENGINEERING
PROGRAM

Ali Buğra KANBUROĞLU

(218DCS8078)

LARGE LANGUAGE MODEL BASED AUTOMATED TRANSLATION OF NATURAL LANGUAGE TO SQL

SUPERVISOR

Assoc. Prof. Faik Boray TEK

İSTANBUL, January 2025

T.C.
IŞIK UNIVERSITY
SCHOOL OF GRADUATE STUDIES

DOCTORAL THESIS
DEPARTMENT OF COMPUTER ENGINEERING PROGRAM

Ali Buğra KANBUROĞLU

(218DCS8078)

LARGE LANGUAGE MODEL BASED AUTOMATED TRANSLATION OF NATURAL LANGUAGE TO SQL

Date: 22.01.2025

Thesis Supervisor: Assoc. Prof. Faik Boray TEK / Istanbul Technical University

Jury Members: Prof. Dr. Olcay Taner YILDIZ / Özyeğin University

Assoc. Prof. Yusuf YASLAN / Istanbul Technical University

Asst. Prof. Emine EKİN / Işık University

Asst. Prof. Tuğba ERKOÇ / Işık University

İSTANBUL, January 2025

ÖZET

BÜYÜK DİL MODELİ TABANLI DOĞAL DİLDEN SQL'E OTOMATİK ÇEVİRİ

Doğal dilin SQL sorgularına dönüştürülmesi, son yıllarda önemli ilerlemeler kaydetmiş bir alandır, ancak Türkçe gibi kaynakları sınırlı dillerde hala bazı zorluklar bulunmaktadır. Bu tez, bu zorluklara çözüm getirmek amacıyla üç önemli katkı sunmaktadır. İlk katkımız, 10.809 doğal dil cümlesi ve karşılık gelen SQL sorgusundan oluşan, ilk çok alanlı Türkçe Text-to-SQL veri kümesi olan TUR2SQL'in geliştirilmesi ve açık erişimle araştırmacıların kullanımına sunulmasıdır. Bu veri kümesi üzerinde probleme özgü geliştirilmiş bir derin öğrenme yöntemi olan SQLNet ve en başarılı büyük dil modellerinden (BDM) biri olan ChatGPT modellerinin başarımı karşılaştırılmıştır. Sonuçlar ChatGPT'nin üstün performansını göstermektedir. İkinci temel katkımız, İngilizce'den SQL'e çeviri çalışmalarında sıklıkla kullanılan, çok alanlı geniş Spider veri kümesinin çevrilerek en geniş kapsamlı Türkçe Text-to-SQL veri kümesi TURSpider'ın oluşturulması ve yine açık erişimle paylaşılmasıdır. TURSpider, karmaşık sorgular ve farklı zorluk seviyeleri içermektedir. Türkçe Text-to-SQL görevleri için büyük dil modellerinin (BDM) eğitimini ve karşılaştırmasını kolaylaştırmaktadır. Yaptığımız karşılaştırmalı incelemede, ince-ayar uygulanmış Türkçe BDM'lerin, rekabetçi bir performans sergilediğini ve bazı modellerin, sorgu doğruluğu açısından OpenAI modellerini geride bıraktığını göstermektedir. Performansı daha da iyileştirmek amacıyla, CoF (Chain-of-Feedback, Geri Bildirim Zinciri) metodolojisi uygulanmış ve bunun birden fazla modeldeki etkinliği gösterilmiştir. Son olarak, Text-to-SQL görevleri için açık kaynaklı BDM'lerin performansını artırmayı amaçlayan, birden fazla modelin çıktısını birleştiren MoA (Mixture-of-Agents, Temsilcilerin Karışımı) çerçevesini keşfediyoruz. MoA ve CoF tekniklerini birleştirerek geliştirdiğimiz MoAF-SQL yaklaşımımızın, özellikle karmaşık

sorgularda önemli iyileştirmeler sağladığı gösterilmiştir. Deneylerimiz, MoAF-SQL'in rekabetçi sonuçlar elde ettiğini ve açık kaynaklı BDM'lerin Text-to-SQL başarımını artırma potansiyelini göstermektedir.
Anahtar Kelimeler: Text-to-SQL, BDM, TUR2SQL, TURSpider, MoA

ABSTRACT

LARGE LANGUAGE MODEL BASED AUTOMATED TRANSLATION OF NATURAL LANGUAGE TO SQL

The field of Text-to-SQL, which involves converting natural language into SQL queries, has seen significant advancements, but challenges remain, particularly for low-resource languages like Turkish. This thesis introduces three key contributions to address these challenges. Our first contribution is the development and open-access release of TUR2SQL, the first cross-domain Turkish Text-to-SQL dataset, which consists of 10,809 natural language sentences paired with their corresponding SQL queries. We evaluate the performance of SQLNet, a deep learning model specifically designed for this task, and one of the most successful Large Language Models (LLMs), ChatGPT, on this dataset. The results demonstrate the superior performance of ChatGPT. The second major contribution is the construction and publicly available release of TURSpider, the most extensive Turkish Text-to-SQL dataset. TURSpider is built by translating the widely used cross-domain Spider dataset from English to Turkish. This dataset includes complex queries with varying difficulty levels, facilitating the training and comparison of large language models for Turkish Text-to-SQL tasks. Our comparative analysis shows that fine-tuned Turkish LLMs achieve competitive performance, with some models surpassing OpenAI models in query accuracy. To further enhance performance, we apply the Chain-of-Feedback (CoF) methodology, demonstrating its effectiveness across multiple models. Finally, we explore the Mixture-of-Agents (MoA) framework, which combines outputs from multiple models to improve the performance of open-source LLMs for Text-to-SQL tasks. By integrating MoA with the CoF technique, we propose MoAF-SQL, an approach that significantly improves performance, particularly on complex queries. Our experiments show that

MoAF-SQL achieves competitive results, highlighting its potential to enhance the Text-to-SQL capabilities of open-source LLMs.
Keywords: Text-to-SQL, LLM, TUR2SQL, TURSpider, MoA

ACKNOWLEDGEMENT

Firstly, I would like to express my gratitude to my advisor Assoc. Prof. Dr. Faik Boray Tek for his invaluable guidance and support throughout this thesis.

I would also like to extend my thanks to my mother, Emine Gülgün, and my father, Özer, for their love and encouragement, which have always given me strength.

Finally, I am deeply grateful to my daughter, Eylül Lina, whose birth has filled my life with happiness and positivity, giving me the motivation to complete this thesis. I owe special thanks to my love and my beloved wife, Büşra, for always being by my side and helping me stay motivated throughout this journey.

Ali Buğra KANBUROĞLU

This study is dedicated to my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS LIST

ATIS: Airline Travel Information System

BDM: Büyük Dil Modeli

BERT: Bidirectional Encoder Representations from Transformers

CNN: Convolutional Neural Network

CoF: Chain-of-Feedback

CoT: Chain-of-Thought

CoX: Chain-of-X

CSV: Comma Separated Values

Dev: Development

EM: Exact Matching

EX: Execution Accuracy

IT: Information Technology

LF: Logical Form Accuracy

LFA: Logical Form Accuracy

LLM: Large Language Model

MAS: Microsoft Academic Search

MoA: Mixture-of-Agents

MoAF: A Feedback Driven Mixture-of-Agents Approach

MoE: Mixture-of-Experts

NL: Natural Language

NLIDB: Natural Language Interface to Databases

NLP: Natural Language Processing

NL2SQL: Natural Language to SQL

OOD: Out-of-Domain

PMF: Probability Mass Function

PRISMA: Preferred Reporting Items for Systematic reviews and Meta-Analyses

RNN: Recurrent Neural Network

SQL: Structured Query Language

Train: Training

CHAPTER 1

1. INTRODUCTION

Relational databases are frequently used in the IT industry to organize and maintain data and information from various fields and domains. To retrieve information from these databases, professionals require machine-readable instructions in the form of programs. Structured Query Language (SQL) is the most common language used to access and query data in relational databases. However, experts or professionals with SQL knowledge are required to access database and obtain meaningful information. These experts or professionals must also understand the existing schemas and tables of the database and create appropriate queries.

Obviously, the need for specialized knowledge creates a barrier for non-expert users. Text-to-SQL systems aim to address this challenge by enabling users to query databases using natural language. If successfully applied, envisioned Text-to-SQL systems can help end-users translate natural language input into queries without the need for SQL knowledge to query necessary information from the database. Some data applications, such as ThoughtSpot[1], have already claimed the capability to translate natural language given by users into SQL queries and display or visualize the results.

According to Gartner Inc., chatbots will become the primary customer service channel for roughly a quarter of organizations by 2027. Thus, end-users will be able to access the desired information with the help of assistant applications such as the ones shipped with mobile operating systems. We see that a common trend in human-computer interaction is transforming and chatbots in the form of large language models are getting more capable each new day.

[1] https://www.thoughtspot.com

The Text-to-SQL task focuses on automating the translation of natural language statements into Structured Query Language (SQL) queries. Writing an SQL query requires knowledge of SQL syntax and the database schema, which can be a major challenge for many users. Text-to-SQL systems address this issue by using natural language inputs to produce executable SQL queries.

1.1 STATEMENT OF THE PROBLEM

The process of converting natural language to SQL is referred to in literature with different terms, such as natural language to SQL (NL2SQL), Text-to-SQL, or natural language interface to databases (NLIDB). In this thesis, we use the term Text-to-SQL. Text-to-SQL is the task of automatically translating natural language (NL) into SQL, as shown in Figure 1.1. In Large Language Model-based Text-to-SQL (LLM-based Text-to-SQL), the task involves processing an input that includes a natural language statement (*N*) and corresponding database schema information (*S*). The database schema (*S*) is defined as *S = (T, C)*, where $T = \{t_1, \dots, t_m\}$ represents multiple tables, and $C = \{c_1, \dots, c_n\}$ denotes the columns within those tables.

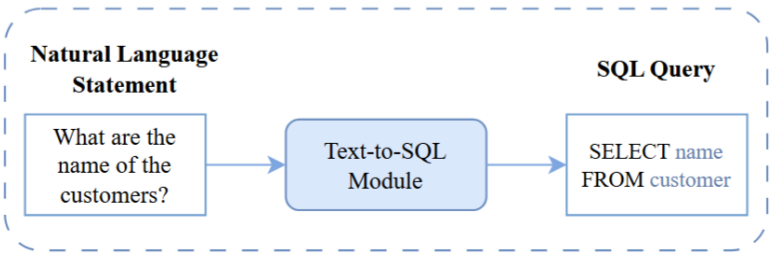

Figure 1.1 A Simple Text-to-SQL Diagram.

The main objective of the Text-to-SQL task is to automatically generate an SQL query (*Q*) based on the given natural language statement (*N*) and the database schema information (S). This requires an LLM (*L*) to interpret the natural language statement (*N*) within the context of the given database schema

(*S*) and subsequently produce the corresponding SQL query (*Q*). The likelihood of an LLM (*L*) generating a SQL query (*Q*) is defined as a conditional probability distribution, where $\mathcal{P}$ is the prompt template, and $|Q|$ is the length of Q. Q_i denotes *i*-th token of SQL query (*Q*). The task can be expressed as in (1.1):

$$\mathbb{P}_L(Q|\mathcal{P}(Q,S))\prod_{i=1}^{|Q|}\mathbb{P}_L(Q_i|\mathcal{P}(Q,S),Q_{<i}) \tag{1.1}$$

The database schema (*S*) provides critical contextual information, enabling the LLM to understand the semantics of (*N*) and generate an accurate SQL query (*Q*) that interacts correctly with the database tables and columns defined in *S*.

A query refers to a question or request for data, which can be expressed in natural language. For example, "Retrieve the ids of all authors." would translate to the SQL query which is "SELECT id FROM author" SQL utilizes keywords that match the given natural language. In this example, SELECT is used to retrieve data from the database, FROM specifies the table being used, and WHERE is used to establish a condition. In the following example, we give a typical SQL query.

SELECT attributes FROM tables WHERE conditions

There are various structures that are used when constructing an SQL query, such as JOIN, NESTED, GROUP BY, and ORDER BY, as in (Date, 1989).

1.2 FROM DEEP LEARNING-BASED APPROACHES TO LLM-BASED APPROACHES

Several deep learning-based studies in the literature have addressed different aspects of the Text-to-SQL problem. For example; Seq2SQL (Zhong et al., 2017), and SQLNet (Xu et al., 2017), focused on the ordering issue that refers to the challenge of correctly identifying the order in which different components of a SQL query should appear when generating the query from natural language text. Also, SyntaxSQLNet (Yu et al., 2018c), and RYANSQL (Choi et al., 2021), addressed the complex and cross-domain Text-to-SQL task.

Additionally, IRNet (Guo et al., 2019), addressed the mismatch problem, which refers to the challenge of accurately aligning the elements of a natural language sentence with their corresponding components in the SQL query. Moreover, GNN (Bogin et al., 2019), RAT-SQL (Wang et al., 2020c), and SADGA (Cai et al., 2021), considered the schema representation problem that refers to the challenge of effectively representing the structure of a database schema in a way that can be understood and used by a model to generate accurate SQL queries.

Recently, the development of large language models (LLMs) such as PaLM (Anil et al., 2023) and GPT-4 (Achiam et al., 2023) has driven substantial progress in the Text-to-SQL domain, establishing LLM-based approaches as a dominant paradigm in this field. These models use advanced techniques to manage the challenges of converting natural language into SQL queries (Zhang et al., 2024). For example, Liu et al. (2023) conducted an extensive analysis of ChatGPT's Text-to-SQL capabilities. In addition, Jiang et al. (2023b) introduced StructGPT, a general framework aimed at improving the zero-shot reasoning abilities of LLMs over structured data. Through experiments on different datasets, they showed that their approach can significantly improve the zero-shot performance of LLMs. Pourreza and Rafiei (2024) proposed a novel approach called DIN-SQL based on few-shot prompting to address the Text-to-SQL task. Their method decomposes the problem into multiple steps: (1) schema-linking, (2) query classification and decomposition, (3) SQL generation, and (4) self-correction, leading to consistent and remarkable performance improvements across different LLMs. Sun et al. (2023) contributed to this evolving landscape with SQL-PaLM, an LLM-based Text-to-SQL model leveraging PaLM-2 (Anil et al., 2023).

1.3 SUMMARY OF CONTRIBUTIONS

This study builds on two conference papers and two journal papers published during my doctoral journey. Here, we list the papers and summarize

the main contributions of each paper. A detailed list of contributions is provided in the corresponding chapters on related research.

- Kanburoğlu, A. B., & Tek, F. B. (2023). TUR2SQL: A cross-domain Turkish dataset for Text-to-SQL. In 2023 8th International Conference on Computer Science and Engineering (UBMK) (pp. 206-211). IEEE. https://doi.org/10.1109/ubmk59864.2023.10286686
- Kanburoğlu, A. B., & Tek, F. B. (2024). Text-to-SQL: A methodical review of challenges and models. Turkish Journal of Electrical Engineering and Computer Sciences, 32(3), 403-419. https://doi.org/10.55730/1300-0632.4077
- Kanburoğlu, A. B., & Tek, F. B. (2024). TURSpider: A Turkish Text-to-SQL Dataset and LLM-Based Study. IEEE Access. https://doi.org/10.1109/access.2024.3498841
- MoAF-SQL: A Feedback-Driven Mixture-of-Agents Approach for Text-to-SQL (*Accepted, In Progress*)
- MoAF-SQL with TURSpider - In Preparation

TUR2SQL – Contributions (Kanburoğlu and Tek, 2023):

- We developed a framework that automates the Text-to-SQL dataset generation process using given database schema and Turkish natural language templates.
- We introduced the TUR2SQL, the first publicly available cross-domain Text-to-SQL dataset for the Turkish language.
- We conducted experiments using SQLNet and ChatGPT models on the TUR2SQL dataset, resulting that SQLNet showed limited performance and ChatGPT achieved superior performance.

A Methodical Review – Contributions (Kanburoğlu and Tek, 2024a):

- We conducted a PRISMA-based methodical review of Text-to-SQL studies, including literature surveys. Additionally, we incorporated a review of studies and datasets related to the conversion of non-English languages to SQL.

- We identified several significant issues and challenges related to Text-to-SQL studies.
- We examined single-domain and cross-domain benchmarks, related evaluation metrics, and the accuracies of the Text-to-SQL methods on these benchmarks.
- We reviewed recently developed LLM-based Text-to-SQL studies. Additionally, we evaluated and compared three general-purpose LLM models on complex cross-domain Spider dataset.
- We discussed potential future research directions in the Text-to-SQL field.

TURSpider – Contributions (Kanburoğlu and Tek, 2024b):

- We introduced the TURSpider, a large-scale, complex, and cross-domain Text-to-SQL dataset that contains SQL queries in different hardness levels, translated from the original English Spider dataset into Turkish.
- We evaluate state-of-the-art LLMs on the Turkish Text-to-SQL task.
- We fine-tune Turkish LLMs on the TURSpider dataset and assess their performance.
- We compare the performance of fine-tuned Turkish LLMs with GPT-3.5 Turbo and GPT-4 on TURSpider.
- We apply the CoF methodology to enhance LLM performance on Turkish Text-to-SQL.
- We conducted an error analysis based on generated SQL queries to provide valuable insights and directions for future research.

MoAF-SQL – Contributions:

- We introduced MoAF-SQL, an innovative feedback (F) driven approach to the Text-to-SQL task that enhances query accuracy by combining the strengths of multiple large language models within a Mixture-of-Agents (MoA) framework.

- We showed that Mixture-of-Agents (MoA) approach improves the performance of open-source large language models (LLMs) on the Text-to-SQL task.
- We showed that the CoF technique independently improves the performance of open-source LLMs on the Text-to-SQL task.
- We demonstrated that integrating MoA with CoF leads to further improvements.

1.4 ORGANIZATION

In the rest of this thesis, we provide a comprehensive literature review of the challenges, issues, benchmarks, and evaluation metrics in the field of Text-to-SQL research in Chapter 2. Additionally, we delve into the role of large language models (LLMs) in advancing Text-to-SQL systems.

In Chapter 3, we introduce TUR2SQL, a novel cross-domain Turkish dataset for Text-to-SQL tasks. We detail the dataset generation process, including database table collection, survey-based query creation, template-based approaches, and dataset distribution. The chapter concludes with experiments evaluating the dataset and presenting their results.

In Chapter 4, we present TURSpider, a Turkish Text-to-SQL dataset created through a human translation approach. This chapter also discusses the fine-tuning of Turkish language models, experiments, and results, including insights from the Chain-of-Feedback (CoF) technique and an error analysis.

In Chapter 5, we propose MoAF-SQL, a feedback-driven Mixture-of-Agents (MoA) methodology for Text-to-SQL tasks. This chapter explains the MoA methodology, experimental results, and the impact of different model subset combinations on performance.

In Chapter 6, we discuss the methodologies and results presented in Chapters 3, 4, and 5, critically analyzing their implications and limitations.

Finally, in Chapter 7, we summarize the contributions of this thesis and provide directions for future research in Text-to-SQL systems, particularly in the Turkish language context.

CHAPTER 2

2. LITERATURE REVIEW

In this chapter, we provide a methodical review of the Text-to-SQL problem. Firstly, we follow the PRISMA approach to identify surveys and review studies that discussed Text-to-SQL or NL2SQL. Next, we discuss some of the key challenges and issues. Further, we focus on the most commonly used benchmarks and evaluation metrics in Text-to-SQL studies. Also, we provide implementation details of the Text-to-SQL methods. Finally, we explain the LLM-based Text-to-SQL models.

2.1 PRISMA APPROACH FOR SYSTEMATIC REVIEW

We followed the PRISMA approach (Liberati et al., 2009) to identify survey and review studies that discussed Text-to-SQL or NL2SQL. We conducted our search using Web of Science, Scopus, and Google Scholar from 2018 to 2023, using the search terms "(nl2sql OR text-to-sql) AND (survey OR review)".

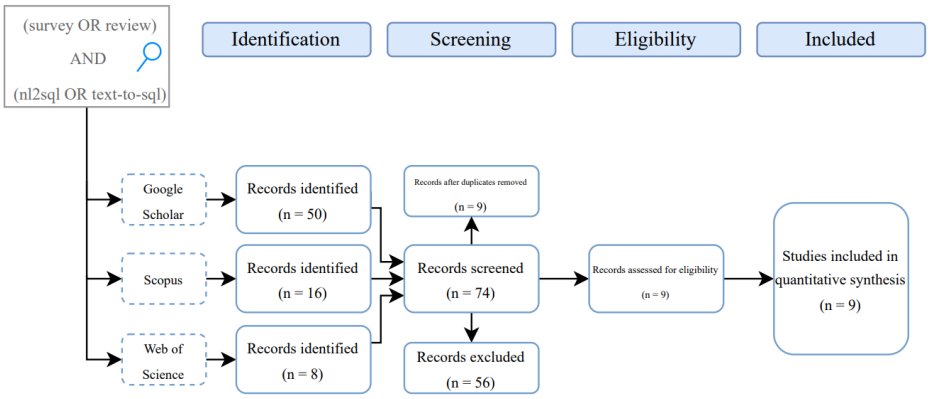

Figure 2.1 The PRISMA Flow Chart for Research Strategy.

Figure 2.1 illustrates the PRISMA flow chart of our review process. As eligibility criteria, we removed duplicates, eliminated studies not available in English, and studies that were not surveys or reviews. As a result, 9 studies remained.

Iacob et al. (2020) delved into architecture decisions for Text-to-SQL models, focusing on encoder and decoder choices, while also drawing insights from classical methods and non-deep learning solutions. Besides, they report the accuracies for different models on the Spider dataset, highlighting the resurgence of natural language interfaces to databases (NLIDB). Kalajdjieski et al. (2020) conducted a comprehensive review of Text-to-SQL methods, models, and datasets, including a wide range of architectures, such as CNNs, RNNs, pointer networks, and reinforcement learning. They introduced various Text-to-SQL datasets and evaluation metrics that consider execution and logical form accuracy. Kim et al. (2020) provided a taxonomy-based review of NL2SQL methods, comparing eleven methods against multiple benchmarks. They introduced a validation tool to measure the quality of NL2SQL models accurately by considering the semantic equivalence of SQL queries. Majhadi and Machkour (2021) presented an overview of NLIDB models, discussing their architectures and experimental results. They emphasized the need for improved accuracy in NLIDB systems. Ahkouk et al. (2021) summarized Text-to-SQL techniques for nested queries, highlighting the potential for generating high-quality queries through human function imitation. Abbas et al. (2022) conducted a detailed review of deep learning-based NLIDB systems, comparing WikiSQL and Spider datasets and identifying challenges in dataset quality and condition accuracy. Baig et al. (2022) categorized NL2SQL frameworks based on implementation techniques and evaluated their performance on the SPIDER dataset, with RAT-SQL using BERT achieving the highest accuracy. Deng et al. (2022) reviewed Text-to-SQL datasets and deep-learning-based methods, categorizing them into different groups and discussing common evaluation metrics. Qin et al. (2022) provided a comprehensive review of deep learning approaches for Text-to-SQL, categorizing them into context-dependent and

context-independent methods. They explored encoding and decoding techniques, including pre-trained language models like BERT, and discussed potential future directions for Text-to-SQL. Katsogiannis-Meimarakis and Koutrika (2023) presented a detailed taxonomy that decomposes the Text-to-SQL problem into different sub-problems and compares various approaches of neural Text-to-SQL methods by explaining the available benchmarks and evaluation methods.

In comparison to existing surveys on Text-to-SQL, our paper contributes a methodical examination of the field by adopting the PRISMA systematic review methodology. We provide a meta-analysis of the existing review papers, categorize Text-to-SQL models based on proposed methodologies, and classify them according to the evaluation metrics and benchmarks used. We provide a detailed analysis of the accuracies achieved by various models on Text-to-SQL datasets and delve into execution-guided evaluation strategies. Furthermore, our study places a specific emphasis on LLM-based approaches for the Text-to-SQL task, evaluating these models on the cross-domain Spider dataset. We extend our investigation to explore the availability of Text-to-SQL datasets in non-English languages. Moreover, we enhance the industrial perspective in our paper by incorporating insights from studies conducted by companies. Besides, we provide valuable insights into the efficiency of Text-to-SQL models by reporting their training times on well-established benchmarks such as WikiSQL and Spider. While existing surveys have provided valuable insights into specific aspects of Text-to-SQL, our paper stands out in its systematic approach, presenting a consolidated view of the field and identifying potential future directions for research in this domain.

Web of Science[2] and Scopus[3] are two major databases used for research references that complement each other. Therefore, we conducted keyword searches in both databases for studies published between 2018 and 2024. The period between 2018 and 2024 was selected for analysis as there was a surge of

[2] https://www.webofscience.com
[3] https://www.scopus.com

interest in Text-to-SQL studies in 2018, which coincided with an increase in deep learning research during those years. Figure 2.2 shows the number of research studies found in Scopus and Web of Science databases using NL2SQL or Text-to-SQL keywords. The number of research papers has significantly increased since 2018, according to the Scopus results between 2018 and 2024.

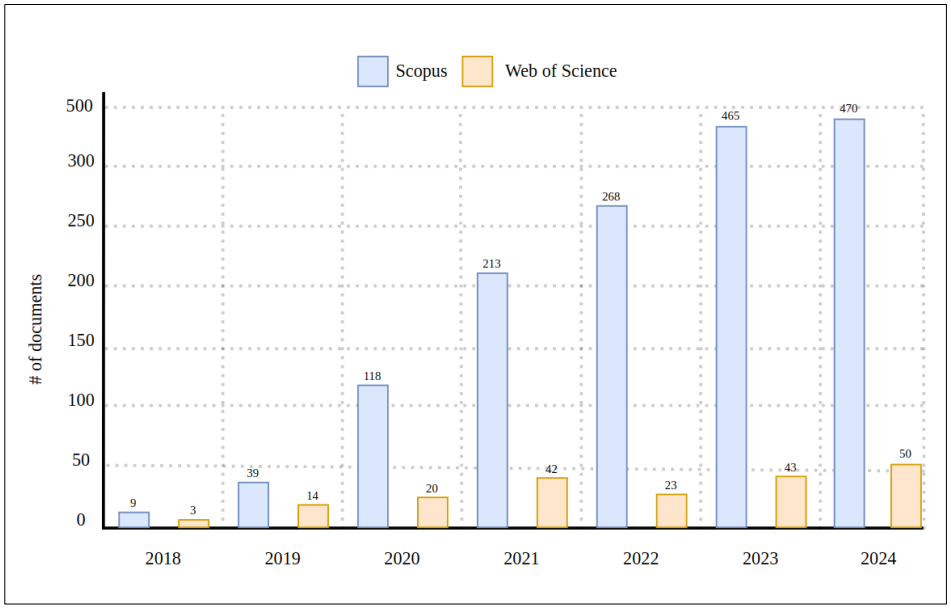

Figure 2.2 Number of Documents Searched from Scopus and Web of Science Databases Including NL2SQL or Text-to-SQL from 2018 to 2024.

2.2 MODELS

We will briefly provide a summary of various Text-to-SQL models. Among these, we will focus on the SQLNet model in detail, because we will use it in next chapter.

- **PRECISE (Popescu et al., 2003):** A database-independent statistical parser that translates natural language queries to SQL queries.

- **NaLIR (Li & Jagadish, 2014):** A generalized and interactive natural language interface that translates given natural languages to SQL statements.
- **ATHENA (Saha et al., 2016):** An ontology-based system for querying relational databases using natural language.
- **SQLizer (Yaghmazadeh et al., 2017):** A sketch-based method that uses domain-specific heuristics to synthesize SQL queries from natural language.
- **Seq2SQL (Zhong et al., 2017):** A deep neural network that uses reinforcement learning with execution rewards to translate natural language questions into SQL queries.
- **SQLNet (Xu et al., 2017):** A sketch-based method that decomposes SQL generation into several sub-modules and performs classification. This model was utilized in our further studies.
- **Coarse2Fine (Dong & Lapata, 2018):** A structure-aware decoding framework for semantic parsing.
- **PT-MAML (Huang et al., 2018):** A meta-learning-based approach to generating SQL queries from natural language.
- **SyntaxSQLNet (Yu et al., 2018c):** A syntax-tree-based model addressing complex and cross-domain Text-to-SQL tasks.
- **TypeSQL (Yu et al., 2018b):** A sketch-based approach treating the Text-to-SQL problem as a slot-filling task using type information to manage rare entities and numbers.
- **STAMP (Sun et al., 2018):** A generative model mapping natural language questions to SQL queries.
- **IncSQL (Shi et al., 2018):** A sequence-to-action parsing approach incrementally filling slot values with predefined actions.

- **DialSQL (Gür et al., 2018):** A dialogue-based structured query generation framework leveraging user interaction to enhance performance.
- **SQLOVA (Hwang et al., 2019):** A sketch-based approach incorporating pre-trained language models.
- **RAT-SQL (Wang et al., 2020):** A framework using relation-aware self-attention to address schema encoding and linking challenges.
- **IRNet (Guo et al., 2019):** A neural approach using schema linking and intermediate representation to address complex and cross-domain tasks.
- **EditSQL (Zhang et al., 2019):** An editing-based encoder-decoder method for cross-domain, context-dependent tasks.
- **TEMPLAR (Baik et al., 2019):** A rule-based system leveraging SQL query logs.
- **GNN (Bogin et al., 2019):** An encoder-decoder parser using graph neural networks to represent database schema structures.
- **X-SQL (He et al., 2019):** A network architecture enhancing structural schema representation with contextual embeddings.
- **BRIDGE (Lin et al., 2020):** A sequential architecture modeling dependencies between questions and databases in cross-domain Text-to-SQL parsing.
- **HydraNet (Lyu et al., 2020):** A hybrid approach using column-wise ranking and pre-trained language models.
- **IE-SQL (Ma et al., 2020):** An extraction-linking approach to Text-to-SQL tasks.
- **SDSQL (Hui et al., 2021):** A multi-task model based on schema dependency.

- **PHOTON (Zeng et al., 2020):** A robust cross-domain model integrating semantic parsing, untranslatable question detection, and human-in-the-loop feedback.
- **LGESQL (Cao et al., 2021):** A line graph-enhanced model capturing relational features without constructing meta paths.
- **SADGA (Cai et al., 2021):** A structure-aware dual graph aggregation network for cross-domain tasks.
- **RYANSQL (Choi et al., 2021):** A sketch-based slot-filling approach for complex and cross-domain Text-to-SQL problems.
- **ATHENA++ (Sen et al., 2020):** An end-to-end system capable of translating nested SQL queries.
- **ValueNet (Brunner & Stockinger, 2021):** An end-to-end system incorporating value candidates during query translation.
- **SeaD (Xu et al., 2021):** A sequence-to-sequence model using schema-aware denoising.
- **ShadowGNN (Chen et al., 2021):** An architecture abstracting schemas at semantic levels with domain-independent representations.
- **UniSAr (Dou et al., 2023):** A unified structure-aware autoregressive model for Text-to-SQL tasks.
- **S²SQL (Hui et al., 2022):** A syntax-enhanced encoder leveraging syntactic dependency information.
- **RASAT (Qi et al., 2022):** A transformer sequence-to-sequence model augmented with relation-aware self-attention.

2.2.1 SQLNet

SQLNet is a neural network designed to generate SQL queries. This neural network uses a sketch-based approach where the sketch contains a dependency graph. In the SQL sketch, tokens like SELECT, WHERE, and AND represent SQL keywords. The tokens starting with "$" indicate the slot to be filled. The

name following the "$" indicates the type of the prediction. For example, the AGG slot can be filled with aggregation functions like SUM or MAX, the VALUE slot with a substring from the natural language (NL) question, the COLUMN slot with a column name, and the OP slot with operations such as; ">", "<", or "=". These predictions are then combined to generate the final SQL query.

In SQLNet, the generation of the query is structured as a dependency graph, where each slot to be predicted is depicted as a box, and the relationships between slots are shown as directed edges. For example, the OP slot depends on both the COLUMN slot and the natural language question. Each slot in SQLNet is handled by a prediction model, and these models do not share their trainable parameters. The WHERE clause is the most complex part of the query to predict. SQLNet first predicts the set of columns that appear in the WHERE clause, and then for each column, it generates the corresponding constraints by predicting the OP and VALUE slots. A column attention mechanism is used during the prediction of column names in the WHERE clause, allowing the model to focus on the most relevant information from the NL question. The OP slot is predicted through a 3-way classification (>, <, =), while the VALUE slot is generated using a sequence-to-sequence structure to generate the relevant substring from the NL question. The SELECT clause, which includes an aggregation function and a column name, is predicted similar to the WHERE clause. However, unlike the WHERE clause, the SELECT clause requires only the selection of one column. Sketch syntax and the dependency of the sketch is illustrated in Figure 2.3.

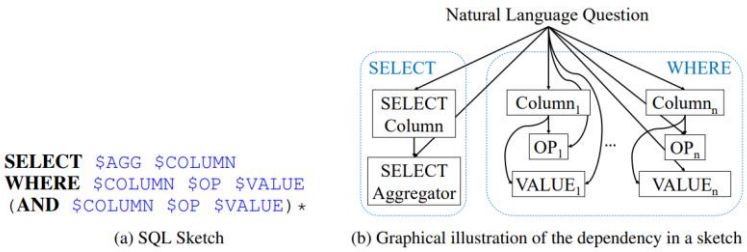

Figure 2.3 Sketch Syntax and the Dependency of the Sketch.

2.3 CHALLENGES AND ISSUES

Text-to-SQL systems face several significant challenges and issues that need to be addressed. Below, we describe some of the key challenges.

A major challenge is the ordering issue, also known as the “order-matters” problem (Xu et al., 2017). In SQL queries, the order of predicates within the WHERE clause does not affect the execution results, allowing the same query to be written in different orders. For instance, the queries “SELECT * FROM Employees WHERE Salary > 50000 AND Department = 'IT' and SELECT * FROM Employees WHERE Department = 'IT' AND Salary > 50000” produce the same result. Solutions such as Seq2SQL (Zhong et al., 2017), SQLNet (Xu et al., 2017), and IncSQL (Shi et al., 2018) address this issue using techniques like reinforcement learning, sequence-to-set models, and sequence-to-action frameworks.

Another critical challenge is handling complex SQL queries across cross-domains, as required by datasets like Spider (Yu et al., 2018a). This complexity necessitates models that can generalize to multiple domains, such as movie databases, geography, and sports. Approaches like SyntaxSQLNet (Yu et al., 2018c) and RYANSQL (Choi et al., 2021) attempt to address this challenge with innovative methodologies. SyntaxSQLNet uses a SQL-specific syntax tree-based decoder and table-aware column attention mechanisms to generate query components through sub-modules that predict operators, keywords, and entities. By considering syntax rules and prediction history, it generates correct and complex SQL queries across different domains. RYANSQL utilizes a sketch-based slot-filling algorithm to generate complex SQL queries by handling nested queries and predicting SELECT statements step-by-step.

A further issue is the lack of information in natural language queries, which often results in inaccuracies due to missing domain-specific knowledge or rare entities. For instance, a query asking for the highest-scoring player in a sports database might fail to specify the sport, leaving the model without enough context to generate an accurate SQL query. Solutions like TypeSQL (Yu et al.,

2018b), which assigns types like PERSON, PLACE, COUNTRY, ORGANIZATION, and SPORT to words to improve the understanding of rare entities and numbers, and DialSQL (Gür et al., 2018), which incorporates user interaction to identify and correct errors, aim to mitigate this problem.

In some cases, a mismatch problem arises (Guo et al., 2019) when SQL column names do not align with their natural language descriptions, leading to confusion, particularly in GROUP BY queries. For instance, a natural language description might refer to “total sales” while the corresponding SQL column name is “revenue”. Approaches like STAMP (Sun et al., 2018) and IRNet (Guo et al., 2019) address this issue by incorporating table structure, SQL syntax, and schema-linking modules to bridge the gap between natural language and SQL representations.

The lexical problem (Guo et al., 2019) occurs in cross-domain benchmarks such as Spider and WikiSQL, where many words in the development set are absent from the training set. This challenge arises because models struggle to represent out-of-domain (OOD) words accurately. For example, a model trained on scientific article datasets may face difficulties when encountering terms specific to movie reviews. IRNet mitigates this problem using schema-linking solutions to enhance the representation for unseen words.

Lastly, the schema representation problem pertains to the challenge of generalizing models to unseen database schemas in cross-domain Text-to-SQL tasks. This requires encoding schema information, such as table and column details, while establishing connections between natural language and database schema. For example, a model trained on e-commerce data must adapt to generating queries for a medical database, which involves understanding entirely different table structures and relationships. Models like GNN (Bogin et al., 2019), RAT-SQL (Wang et al., 2020c), and SADGA (Cai et al., 2021) have been developed to address this challenge by improving schema representation and understanding.

2.4 BENCHMARKS AND EVALUATION METRICS

Early datasets were created within a single-domain, while newer datasets cover multiple domains. Table 2.1 summarizes our comparison of their structures, including JOIN, NESTED, GROUP BY, and ORDER BY. Below, we provide a detailed analysis of these benchmarks.

Table 2.1 Text-to-SQL Datasets and Various Properties.

Dataset	Year	Domain	Join	Nested	Group	Order	# NL	# SQL
ATIS	1990	Single	+	+	-	-	5,280	947
MAS	2014	Single	+	+	+	-	196	196
WikiSQL	2017	Cross	-	-	-	-	80,654	77,840
Spider	2018	Cross	+	+	+	+	10,181	5,693
SparC	2019	Cross	+	+	+	+	4,298	12,726
CoSQL	2019	Cross	+	+	+	+	3,007	15,598
FIBEN	2020	Single	+	+	+	+	300	237

To begin with, we present well-known single-domain benchmarks (Price, 1990; Iyer et al., 2017; Li and Jagadish, 2014; Sen et al., 2020).

The Airline Travel Information System (ATIS) dataset (Price, 1990; Iyer et al., 2017) serves as a benchmark for evaluating Text-to-SQL models in the context of airline travel. It contains 5,280 pairs of natural language and SQL queries, with a database schema of 25 tables representing the ATIS database. This dataset is notable for focusing on a single-domain, presenting unique challenges for Text-to-SQL models in handling domain-specific language and query patterns. It includes simple queries as well as JOIN and NESTED queries, but excludes queries with GROUPING or ORDERING. It has been widely used for Text-to-SQL evaluation. Similarly, the Microsoft Academic Search (MAS) dataset (Li and Jagadish, 2014) consists of 196 queries from academic databases, categorized as easy, medium, and complex, with a schema of 17 tables. The queries include JOIN, NESTED, and GROUPING, but exclude ORDERING, and follow a strict format where natural language queries begin with the word “return”. Lastly, the FIBEN dataset (Sen et al., 2020), designed for finance-

related natural language querying, focuses on financial transactions from public companies. It contains 300 natural language queries paired with 237 distinct SQL targets, including JOIN and NESTED queries.

There are several cross-domain benchmarks available (Zhong et al., 2017; Yu et al., 2018a; Yu et al., 2019a; Yu et al., 2019b). WikiSQL (Zhong et al., 2017) is the largest human-annotated semantic parsing dataset, containing 80,654 examples of natural language questions and SQL queries extracted from Wikipedia tables. However, these queries are relatively simple, lacking advanced features such as JOIN, NESTED queries, GROUPING, or ORDERING. Spider (Yu et al., 2018a) is a more complex, cross-domain semantic parsing dataset, consisting of 10,181 questions and 5,693 unique, complex SQL queries across 200 databases. It includes NESTED queries and ORDERING/GROUPING components. SparC (Yu et al., 2019a) offers 4,298 question sequences and over 12,000 questions and SQL queries, querying 200 complex databases with various SQL structures. Finally, CoSQL (Yu et al., 2019b), a conversational Text-to-SQL corpus, contains 3,007 dialogues with over 30,000 turns and 10,000 expert-labeled SQL queries across 200 databases and 138 domains. This dataset stands out for its extensive dialogue context and large natural language vocabulary compared to others.

Table 2.2 Text-to-SQL Models with Their Datasets and Used Evaluation Metrics.

Dataset	Evaluation Metric	Models
WikiSQL	EX	Seq2SQL, SQLNet, PTMAML, COARSE2FINE, STAMP, TypeSQL, IncSQL, SQLOVA, X-SQL, BRIDGE, HydraNet, IE-SQL, SDSQL, SeaD, RAT-SQL
	LFA	Seq2SQL, SQLNet, PTMAML, STAMP, TypeSQL, IncSQL, SQLOVA, X-SQL, HydraNet, IE-SQL, SDSQL, SeaD, RAT-SQL
Spider	EM	SyntaxSQLNet, EditSQL, GNN, IRNet, ValueNet, RAT-SQL

We also reviewed evaluation metrics in conjunction with benchmark datasets and Text-to-SQL models. Our findings are summarized in Table 2.2, where the evaluation metrics are represented by abbreviations: EX stands for execution accuracy, LFA refers to logical-form accuracy, and EM denotes exact matching.

Execution accuracy (Zhong et al., 2017) evaluates the synthesized (generated) query by comparing its results to those of the ground truth query, typically by executing both queries on the same database and analyzing their outputs. In contrast, *logical-form accuracy* (Zhong et al., 2017) focuses on exact string matching between the synthesized and ground truth queries, penalizing correct results that do not match exactly. Lastly, *exact matching* (Yu et al., 2018a) computes the average exact match between synthesized and ground truth queries across different SQL components, such as SELECT, WHERE, GROUP BY, ORDER BY, and KEYWORDS.

2.5 IMPLEMENTATIONS DETAILS OF THE TEXT-TO-SQL METHODS

Next, we examine the implementation details of the Text-to-SQL methods. First of all, there are rule-based and deep learning-based methods in Text-to-SQL methods. Open-source code resources were searched for all these methods and information was gathered about the programming languages in which the methods were implemented and the frameworks they used. We have identified open-source methods, their frameworks and activity. We listed top 15 Text-to-SQL methods as shown in Table 2.3 where SQLOVA, Seq2SQL and SQLNet models have garnered the highest number of Github stars, indicating their popularity among users. The majority used the PyTorch framework with Python programming language. Table 2.3 summarizes the activity and public attention to the open-source code by using Github stars. GitHub stars is an easy metric to measure how popular an open-source project is. In addition, we assessed the reproducibility of these methods by checking the README files and issues of

the repositories because reproducibility is a crucial aspect of open-source projects, ensuring that others can replicate and verify the results. We examined the GitHub repositories of the models provided in the table. Within the README file of each repository, we conducted searches for the keywords “reproduce, reproducible, reproducibility” to determine whether the results of the mentioned model are reproducible. Additionally, we assessed discussions related to reproducibility by examining both open and closed issues in the repository. For repositories where reproducibility was not explicitly mentioned in the README file but was confirmed through the examination of open and closed issues, we made our determination accordingly.

Table 2.3 The Number of GitHub Stars for Text-to-SQL Models.

Model	Year	Framework	Github Stars	Reproducibility
SQLOVA	2019	PyTorch	621	NO
Seq2SQL	2017	PyTorch	409	NO
SQLNet	2017	PyTorch	409	NO
RAT-SQL	2020	PyTorch	369	YES
UniSAr	2022	PyTorch	326	YES
IRNet	2019	PyTorch	244	NO
BRIDGE	2020	PyTorch	204	NO
EditSQL	2019	PyTorch	189	YES
Coarse2Fine	2018	PyTorch	167	YES
LGESQL	2021	PyTorch	135	YES
GNN	2019	PyTorch	134	YES
PT-MAML	2018	TensorFlow	128	NO
SyntaxSQLNet	2018	PyTorch	125	NO
TypeSQL	2018	PyTorch	107	NO
HydraNet	2020	PyTorch	61	YES

2.6 LARGE LANGUAGE MODEL BASED TEXT-TO-SQL

Recent advancements in the Text-to-SQL field, particularly with large language models (LLMs), have shown significant progress. One key reason is their enhanced performance, as seen in improved execution accuracy (Dong et al., 2023; Pourreza and Rafiei, 2024) on benchmark datasets such as SPIDER.

Additionally, LLMs have introduced a new discipline: prompt engineering. LLMs benefit from their ability to perform prompt engineering, making them adaptable to various settings without requiring additional training. Liu et al. (2023) analyzed ChatGPT's Text-to-SQL capabilities, achieving an execution accuracy of 70.1% on the Spider dataset. Jiang et al. (2023b) introduced StructGPT, a framework designed to enhance LLMs' zero-shot reasoning abilities over structured data. Their approach improved the zero-shot performance of LLMs, achieving 74.8% execution accuracy on the Spider dataset. Pourreza and Rafiei (2024) proposed the DIN-SQL approach, which utilizes few-shot prompting to break down the Text-to-SQL task into multiple steps. This method showed substantial improvements across various LLMs, with the DIN-SQL model reaching 75.6% execution accuracy with CodeX Davinci and 82.8% with GPT-4 on the Spider dataset. (Sun et al., 2023) developed SQL-PaLM, an LLM-based model leveraging PaLM-2 (Anil et al., 2023), which is a Transformer-based model by Google. It achieved 82.7% execution accuracy on the Spider dataset in a few-shot setting. Fine-tuning SQL-PaLM further improved this to 82.8%.

In our study (Kanburoğlu and Tek, 2024b), we evaluated the performance of general-purpose LLMs in Text-to-SQL, measuring execution accuracy (EX). Using Hugging Face’s framework for Llama-2-7b (Touvron et al., 2023) and directly prompting ChatGPT on the Spider development dataset, we found that ChatGPT achieved a remarkable execution accuracy of 73.83. In comparison, we observed that Llama-2-7b achieved a lower execution accuracy of 34.20. However, after we fine-tuned it on the Spider training set, its execution accuracy improved to 46.60. These results provide valuable insights into the effectiveness of these models and the role of fine-tuning in enhancing performance on the complex task of generating SQL queries from natural language. The results from our evaluation of ChatGPT and Llama-2-7b LLMs on Spider development set using zero-shot setting and fine-tuning are presented in Table 2.4.

Table 2.4 Execution Accuracy (EX) of Different LLMs on the Spider Development Set.

Model	Execution Accuracy (EX)
ChatGPT	73.83
Llama-2-7B	34.20
Fine-tuned Llama-2-7B	46.60

ChatGPT demonstrates a strong capability in performing the Text-to-SQL task, achieving an execution accuracy of 73.83%. This outperforms several existing models, including UniSAr (Dou et al., 2023), SADGA (Cai et al., 2021), and RYANSQL (Choi et al., 2021), highlighting its competitive advantage in accurately translating natural language queries into SQL statements.

We explored some challenges encountered in the Text-to-SQL domain, particularly in ChatGPT. First, we examined the ordering issue, where ChatGPT generated multiple variations of a query without affecting the execution result. We explored how these variations might address the ordering problem. Second, we addressed the mismatch problem, where ChatGPT produced queries containing entities not present in the natural language input. While ChatGPT was able to generate variations that did not impact the execution result for the ordering issue, the results were less effective in resolving the mismatch problem.

2.7 CONCLUSION

In conclusion, we provided a methodical review of the Text-to-SQL problem. Firstly, we followed the PRISMA approach to identify surveys and review studies that discussed Text-to-SQL or NL2SQL. Next, we discussed some of the key challenges and issues. Further, we focused on the most commonly used benchmarks and evaluation metrics in Text-to-SQL studies. In addition, we provided implementation details of the Text-to-SQL methods. Finally, we explained the LLM-based Text-to-SQL models.

CHAPTER 3

3. TUR2SQL

Several Text-to-SQL datasets have been created for various languages (Nguyen et al., 2020; Min et al., 2019), especially for English (Iyer et al., 2017; Yu et al., 2019a; Yu et al., 2019b). In the beginning of this thesis study, we conducted an extensive search for Text-to-SQL datasets in Turkish. Although Text-to-SQL research has been growing, we observed that there were no publicly available datasets specifically for Turkish. To address this gap and contribute to ongoing efforts at the national level, we decided to produce a Turkish dataset.

This need is addressed by TUR2SQL, the first publicly available cross-domain Text-to-SQL dataset for Turkish. It includes a set of Turkish natural language queries paired with corresponding SQL queries. The dataset is made publicly available on Github[4] for research purposes, aiming to support the Text-to-SQL community. To construct TUR2SQL, we developed an automated framework that generates query pairs by utilizing existing database tables, columns, and Turkish natural language templates.

3.1 DATASET GENERATION

The core design principle of TUR2SQL was to create a cross-domain Turkish Text-to-SQL dataset containing basic SQL queries, generated automatically using a framework that combines database tables, columns, and Turkish natural language templates. To construct this dataset that pairs Turkish natural language with SQL queries, we developed a framework that automates this process. The framework generates natural language statements and SQL queries based on the provided database tables, columns, and Turkish language

[4] https://github.com/alibugra/TUR2SQL

templates. The dataset generation process is outlined in Figure 3.1. This framework takes Turkish natural language templates, SQL query templates, and database information as input.

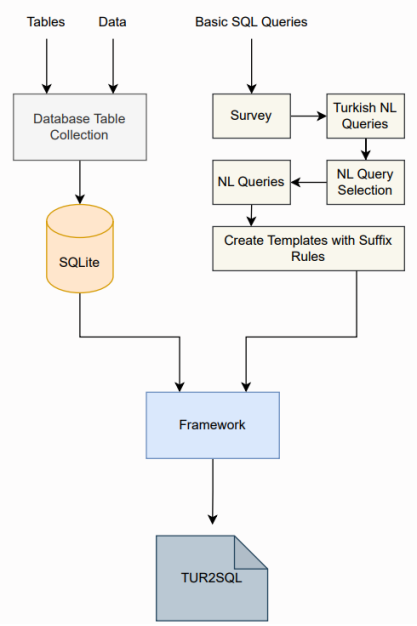

Figure 3.1 TUR2SQL Dataset Generation Flow.

The first step is to create a database table collection (Section 3.1.1). Next, a survey of Turkish natural language query creation is presented (Section 3.1.2). The following sections outline the process of selecting natural language queries (Section 3.1.3) and provide Turkish natural language and SQL templates (Section 3.1.4). Then, suffix rules for natural language templates are introduced (Section 3.1.5). The study proceeds to demonstrate the generation of natural language and SQL pairs (Section 3.1.6), concluding with insights into the dataset distribution (Section 3.1.7).

3.1.1 Database Table Collection

Since the database is one of the inputs for the framework that will automatically generate the dataset, it was created first. The goal was to develop

a cross-domain database by including tables from various domains. Prior to creating these tables, the WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018a) datasets were reviewed, focusing on the table and column structures across different domains. Drawing inspiration from the tables in these public datasets, the tables and columns were manually created. Domains such as cars, nurses, and students were represented through these tables. In total, 47 tables were created manually, all in Turkish.

Since the created tables were empty, it was necessary to generate data for them. Two rows of data were added to each table as examples using the INSERT INTO command. The data were randomly selected by the authors and inserted into the tables. After the tables were created and populated, the database was saved in SQLite format (Hipp, 2020). The sample tables are presented in Figure 3.2.

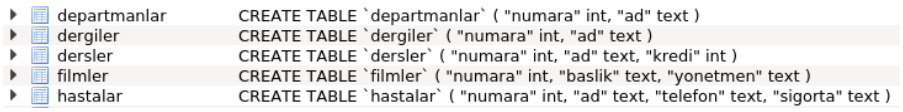

Figure 3.2 Sample Tables from the TUR2SQL Database.

3.1.2 Turkish Natural Language Query Creation Survey

Another input component of the automatic dataset generation framework is the SQL query templates and their corresponding Turkish natural language templates. The first step involves generating the dataset using basic SQL queries. These queries will follow the SELECT-FROM-WHERE structure and will not include GROUP BY, ORDER BY, or NESTED queries. Therefore, five basic SQL queries were manually selected. For example, a student table was created, and the five basic queries were formulated accordingly, as shown in Figure 3.3.

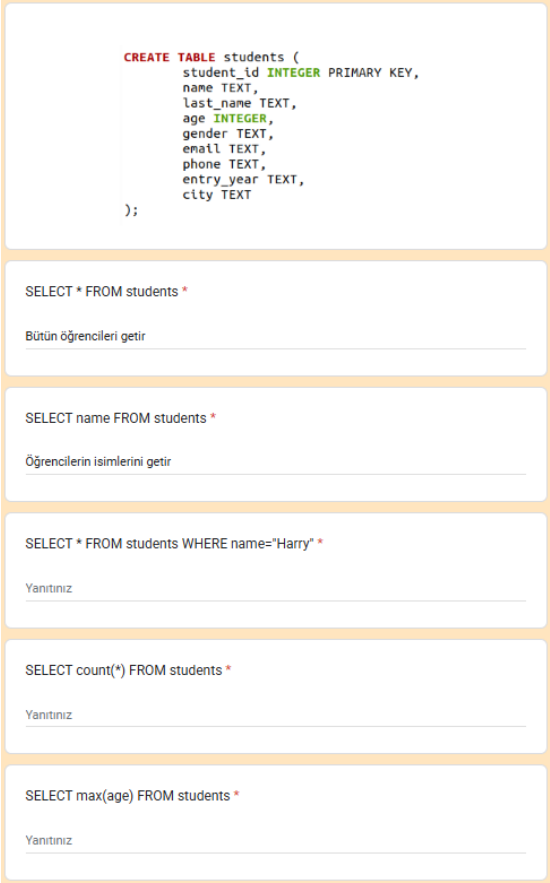

Figure 3.3 Turkish Natural Language Query Creation Survey Form. The First Two Responses Have Been Filled Out as Examples.

As the second step, the survey shown in Figure 3.3 was created on Google Forms[5] using five basic SQL queries and a student table. For each query, participants were asked to write the corresponding sentence in Turkish natural language. The survey link was shared with SQL experts at Huawei Turkey R&D and with students from the Database Systems Course at FMV Işık University, who have basic SQL knowledge. Google Forms allowed the creation of questions and the collection of responses from SQL experts and students. A total of 74 respondents participated in the survey. Since participants provided answers for each SQL query, there were 74 responses per query, totaling over 350

[5] https://www.google.com/forms/about/

answers. The responses could be in the form of either statements or questions. Some of the collected answers are shown in Figure 3.4. All responses were exported as a CSV file from Google Forms.

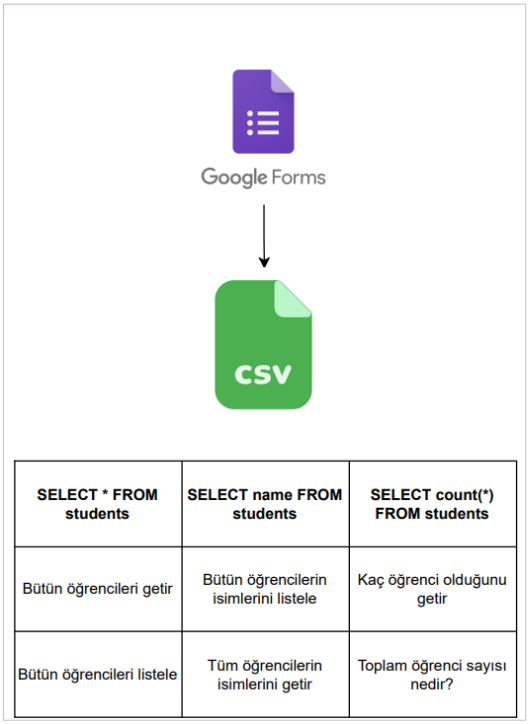

Figure 3.4 Sample Survey Answers.

3.1.3 Natural Language Query Selection

The natural language queries collected through the survey were first processed in a preprocessing step. This included lowercasing the text and removing punctuation. Sentences containing "*" were discarded. English words like "student" were replaced with their Turkish equivalent "öğrenci" to align with the student table. The preprocessing steps are illustrated with three example sentences in Figure 3.5.

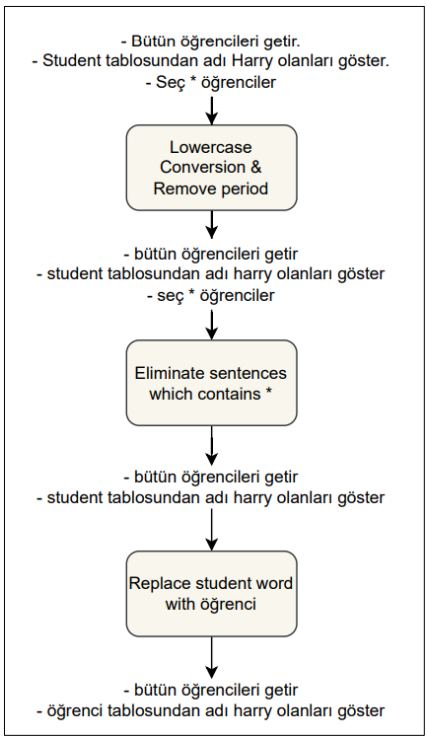

Figure 3.5 Preprocessing Steps.

After completing the preprocessing steps, selection criteria were defined. The natural language sentences collected for each basic query were first organized into a probability mass function (PMF) (Stewart, 2009), based on their frequency of use. The probability of each sentence was represented in this function. Then, using the numpy random choice library (Harris et al., 2020), sentences were randomly selected by their probabilities in the PMF. A total of ten natural language sentences were chosen for each basic query.

3.1.4 Turkish Natural Language & SQL Templates

Five basic SQL queries were identified, and corresponding templates were created for each query group. As mentioned in the previous section, ten natural language queries were selected for each basic query. These queries were structured according to the query type, table name, column name, and values within the selected sentences.

To use the five manually selected SQL queries as templates, table names, column names, and the values in the WHERE clause were defined as placeholders within these queries.

Additionally, to extend the queries involving the WHERE clause, extra templates were generated using the existing natural language and SQL templates. An example of this is shown in Figure 3.6. Templates such as “Veritabanındaki $column_name $value olan $table_name $column_name2 yazdır” were also incorporated into the framework.


```
{
  "question": "Veritabanındaki cinsiyeti Erkek olan sanatçıların numaralarını yazdır.",
  "query_tok": [
    "SELECT",
    "numara",
    "WHERE",
    "cinsiyet",
    "EQL",
    "Erkek"
  ],
  "table_id": "100037",
  "sql": {
    "agg": 0,
    "sel": 0,
    "conds": [
      3,
      0,
      "Erkek"
    ]
  },
  "phase": 1,
  "query": "SELECT numara WHERE cinsiyet EQL Erkek",
  "question_tok": [
    "Veritabanındaki",
    "cinsiyeti",
    "Erkek",
    "olan",
    "sanatçıların",
    "numaralarını",
    "yazdır",
    "."
  ]
}
```


Figure 3.6 A Sample Row from the TUR2SQL Dataset in “*.jsonl” Format.

3.1.5 Suffix Rules for Natural Language Templates

The choice of suffixes for variables in the templates was determined by their equivalents in the natural language. Suffixes for dative, accusative, ablative, plural, possessive, and genitive cases were applied to the variables based on the context of the words they were associated with. These suffixes were added to columns, values, and table names as needed.

For example, in the sentence “Tüm öğrencileri getir” (Bring all students), the corresponding natural language template would be “Tüm $table_name getir.”

In this case, the Turkish word for “student”, “öğrenci” corresponds to the variable “$table_name” Since the plural form of “öğrenci” is “öğrenciler” the plural suffix “-lar / -ler” (Oflazer, 1994) is applied to the table name. Thus, the plural suffix is added to the table name in this example.

3.1.6 The Generation of Natural Language and SQL Pairs

To generate Turkish natural language and SQL query pairs, as described earlier, the data generation framework is provided with database tables, columns, SQL query templates, and Turkish natural language templates. The framework fills the placeholders in these templates with the corresponding database table names, column names, and values. This process resulted in the creation of a dataset containing 10,809 pairs of Turkish natural language and SQL queries.

3.1.7 Dataset Distribution

Table 3.1 shows the distribution of the dataset across the different subsets: development set (dev), test set (test), and training set (train). The development set includes 78 queries with the MAX aggregation function, 54 with the COUNT aggregation function, and 949 queries without any aggregation in the SELECT clause. Additionally, 311 queries in the development set contain a WHERE clause. The test set consists of 167 queries with the MAX aggregation function, 85 with the COUNT function, and 1910 queries without aggregation in the SELECT clause. Moreover, 638 queries in the test set include a WHERE clause. The training set comprises 495 queries with the MAX aggregation function, 331 with the COUNT function, and 6740 queries without aggregation in the SELECT clause. Additionally, 2161 queries in the training set include a WHERE clause.

Table 3.1 TUR2SQL Dataset Distribution.

Dataset	Total	# MAX	# COUNT	None	# WHERE
Dev	1081	78	54	949	311
Test	2162	167	85	1910	638
Train	7566	495	331	6740	2161

We divided the dataset into three parts: Development (dev), Testing (test), and Training (train). The dataset is structured to resemble the WikiSQL (Zhong et al., 2017) dataset, with files saved in the "*.jsonl" format. Figure 3.6 shows a sample row from the dataset, and the following list describes the attributes of each field.

- **question:** represents the natural language query generated by the framework.
- **query_tok:** contains the tokens of the SQL query.
- **table_id:** refers to the identifier of the table addressed in natural language.
- **sql:** categorized into “agg” (aggregation), “sel” (selection), and “conds” (conditions). The “agg” field uses numerical indexes: 1 represents MAX, 3 represents COUNT, and 0 represents none. The “sel” field contains the numerical index of the selected column, while the “conds” field includes the column index, operator index, and condition information.
- **phase:** indicates the phase during which the dataset was generated.
- **query:** stores the SQL query corresponding to the natural language.
- **question_tok:** contains the tokens of the question.

3.2 EXPERIMENTS AND RESULTS

In this study, we used two commonly applied evaluation metrics from the WikiSQL dataset: execution accuracy and logical-form accuracy. In Table 3.2, these metrics are abbreviated as “EX” and “LF” respectively. Execution accuracy measures the similarity between the generated query and the ground truth by executing both queries on the same database and comparing their outputs. This metric calculates the ratio of correctly executed SQL queries to the

total number of examples, indicating how accurately the queries produce the expected results (Zhong et al., 2017). Logical-form accuracy evaluates the similarity between the generated query and the ground truth by checking if the two queries match exactly in their string form (Zhong et al., 2017). We performed experiments on the TUR2SQL dataset using SQLNet (Xu et al., 2017) and ChatGPT models to tackle the Text-to-SQL problem in Turkish. The results are presented in Table 3.2.

Table 3.2 Dev and Test Accuracies on the TUR2SQL Dataset.

Model	Dev		Test	
	LF	EX	LF	EX
SQLNet	42.92	44.49	39.03	40.19
ChatGPT	93.61	98.79	86.72	98.38

We trained the SQLNet model using the implementation from its GitHub repository[6], which employs GloVe (Pennington et al., 2014) for English word embeddings. For our Turkish language requirements, we used Turkish GloVe embeddings obtained from a different repository[7]. To assess SQLNet's performance, we evaluated it on the development (dev) and test sets of the TUR2SQL dataset. On the development set, SQLNet achieved a logical form accuracy of 42.92% and an execution accuracy of 44.49%. In the test set, the logical form accuracy was 39.03%, and the execution accuracy was 40.19%. These results show that SQLNet achieves moderate accuracy when generating and executing SQL queries on Turkish text inputs.

We also used the ChatGPT model to address the Text-to-SQL task on the TUR2SQL dataset. ChatGPT is a large language model-based chatbot developed by OpenAI, built on GPT-3.5 and GPT-4, with users on the free tier having access to the GPT-3.5 version. We leveraged ChatGPT to process Turkish

[6] https://github.com/xiaojunxu/SQLNet

[7] https://github.com/inzva/Turkish-GloVe

natural language statements and generate corresponding SQL queries. The results demonstrated ChatGPT's superior performance over SQLNet. On the development set, ChatGPT achieved a logical form accuracy of 93.61% and an execution accuracy of 98.79%. On the test set, the logical form accuracy was 86.72%, while the execution accuracy reached 98.38%. These results underscore ChatGPT's strong ability to accurately convert Turkish natural language queries into SQL.

3.3 CONCLUSION

We introduced TUR2SQL, the first publicly available cross-domain Text-to-SQL dataset for the Turkish language. To create this dataset, we developed an automated framework that generates data using the provided database tables, columns, and Turkish natural language templates. The TUR2SQL dataset includes 10,809 pairs of Turkish natural language statements and their corresponding SQL queries, covering a range of domains. We evaluated the dataset using SQLNet and ChatGPT models and presented the results. The experiments showed that while SQLNet had limited performance, ChatGPT demonstrated superior performance on the dataset. The TUR2SQL dataset, created specifically for this study, is designed to advance Turkish natural language understanding and encourage further research in the Text-to-SQL field. A conference paper (Demirkiran et al., 2024) utilizing TUR2SQL has been published. This study focuses on evaluating the performance of large language models (LLMs), including GPT-3.5 Turbo, T5, and SQLCoder, on TUR2SQL. The results show that fine-tuning the models with schema context leads to significant improvements in SQL generation accuracy.

CHAPTER 4

4. TURSPIDER

Recently, the advancement of large language models (LLMs) like PaLM (Anil et al., 2023) and GPT-4 (Achiam et al., 2023) has led to significant progress in the Text-to-SQL field. These models utilize advanced techniques to tackle the challenges (Zhang et al., 2024) of translating natural language into SQL queries. The Spider benchmark (Yu et al., 2018a) is commonly used to assess these models, as it features complex queries, including nested structures and advanced SQL clauses like GROUP BY and ORDER BY.

Most research in this area has primarily focused on the English language, with limited attention given to non-English languages. Several datasets exist in languages such as Chinese (Min et al., 2019), Russian (Bakshandaeva et al., 2022), Arabic (Almohaimeed et al., 2024), and Portuguese (Jose and Cozman, 2021), including versions of the popular Spider dataset. However, resources for Turkish remain notably scarce. While a Turkish Text-to-SQL dataset (Kanburoğlu and Tek, 2023) is available, it consists mainly of simple SQL queries with a SELECT statement and a single WHERE clause, lacking more complex nested queries. This underscores the need for a more comprehensive benchmark for Turkish Text-to-SQL tasks. The primary goal of this study is to create such a benchmark dataset to facilitate model evaluation and cross-language comparisons. With the dataset development, we aim to address the following research questions:

- Can inference-based state-of-the-art LLMs already solve the Turkish Text-to-SQL problem? We address this by evaluating state-of-the-art LLMs on the Turkish Text-to-SQL task.
- Can Turkish LLMs fine-tuned on the Turkish Spider (TURSpider) dataset solve the Turkish Text-to-SQL problem? We explore this

by fine-tuning Turkish LLMs on the TURSpider dataset and assessing their performance.

- How do fine-tuned Turkish LLMs perform compared to the GPT-3.5 Turbo and GPT-4 on the TURSpider dataset? We investigate this by comparing the performance of fine-tuned Turkish LLMs with GPT-3.5 Turbo and GPT-4 on TURSpider.
- How can we improve LLM performances for the Turkish Text-to-SQL problem? We approach this by applying the CoF methodology to enhance LLM performance on Turkish Text-to-SQL.

To address these research questions, we developed the TURSpider dataset, which includes complex domain queries and a substantial number of examples, similar to the Spider dataset. Rather than creating an entirely new dataset from scratch, we opted to translate the original Spider dataset to enable cross-language comparisons. To ensure high quality and authenticity, we used a rigorous human translation process instead of relying on machine translation.

Open-source Turkish-supported large language models currently perform poorly on the Turkish Text-to-SQL task. To improve their capabilities, we fine-tuned these models using our TURSpider dataset. Our results show that this fine-tuning significantly enhances the models' performance on Turkish Text-to-SQL tasks.

This study introduces the development and release of the TURSpider dataset, along with an investigation into fine-tuning LLMs to improve their performance on Turkish Text-to-SQL tasks. Our contributions are summarized as follows:

- Construction of the TURSpider Dataset: We created and released the TURSpider dataset, which is specifically designed for Turkish. This provides a benchmark for evaluating Text-to-SQL systems in Turkish, improving the availability of large-scale Turkish Text-to-SQL datasets.

- Fine-tuning Turkish Language Models: Using the TURSpider dataset, we fine-tuned Turkish large language models. By refining these models with the new dataset, we show that one can improve their ability to handle Text-to-SQL tasks, making them more useful in real-world applications.
- Comparative Analysis with Advanced Language Models: Our study compares the fine-tuned Turkish large language models with advanced models like OpenAI GPT-3.5 Turbo and GPT-4. This comparison helps us understand the performance and abilities of Turkish-specific models compared to globally trained ones, providing insights into language modeling across different languages.

4.1 CONSTRUCTION OF THE TURSPIDER DATASET

We developed the TURSpider dataset by manually translating the original Spider dataset from English to Turkish. Instead of creating a new dataset, we opted for translation to support cross-language studies, as the Spider dataset already includes translations in multiple languages, which enables easier comparative analysis. Our initial step was to perform a thorough analysis of the Spider dataset, which consists of 166 databases, each with 684 unique table names. These tables contain various columns with different amounts of data, all stored in SQLite (Owens, 2006) files.

For the translation process, we used a human translation method as shown in Figure 4.1. We worked with three third-year Computer Engineering students from Istanbul Technical University, all of whom are fluent in English and have performed well in database courses. The data to be translated was split into three sections, with each student assigned a portion. The process included a multi-step quality control system: the second student reviewed, corrected, and validated the first student's translation, and the third student then reviewed, corrected, and verified the second student's work. This approach allowed each student to

participate in every round of translation, ensuring thorough cross-checking, correction, and verification.

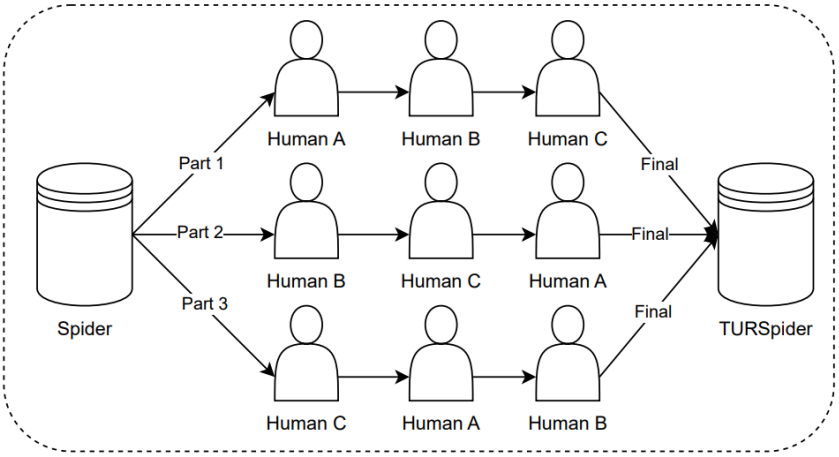

Figure 4.1 TURSpider Human Translation Approach.

The human translation method was used for both the database schema (including database names, table names, and column names) and the dataset (which included questions and queries from the training and development sets). The Spider test set was excluded as it was not publicly available.

We followed several steps in the translation process: First, we translated the English database names into Turkish. Then, we translated the table names within these databases into Turkish as well. Additionally, both the table and column names were translated. While some table names appeared under different databases, we translated them according to their respective domains. For instance, the word “department” was translated as both “departman” and “bakanlık” (meaning “ministry” in English) in different databases. Finally, the questions and queries in the training and development datasets were translated based on the translated database, table, and column names. Figure 4.2 shows the database schema structure along with examples from the English Spider and TURSpider datasets for both training and development data.

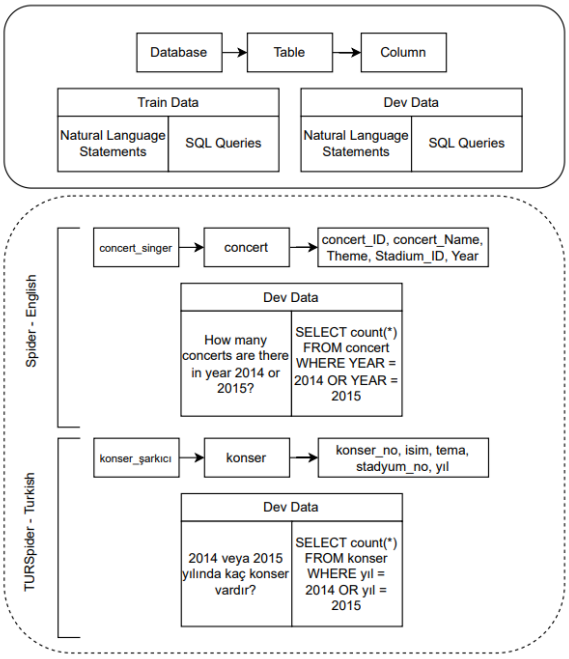

Figure 4.2 TURSpider Database Schema Info.

4.1.1 Man-hours Spent

During the translation of the dataset, we also tracked the total man-hours spent. The first round of translating the database took 52 man-hours. The first round of the development set translation consumed 28 man-hours, while 170 man-hours were required for the first round of translating the training set. In the second round of translations, 72 man-hours were used, and the third and final round took 42 man-hours. In total, 364 man-hours were spent translating the Spider dataset into the TURSpider dataset, as shown in Figure 4.3.

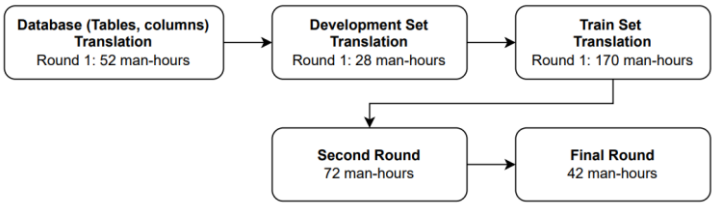

Figure 4.3 Man-Hours Spent Across Translation Phases.

4.1.2 Data Statistics

The TURSpider dataset consists of two primary subsets: a development set and a training set, both structured and scaled similarly to the widely-used Spider dataset. The development set includes 1034 data rows, with 1023 unique questions and 584 distinct SQL queries. The training set contains 8659 data rows, 8506 unique questions, and their corresponding SQL queries. Additionally, the dataset takes into account the difficulty levels of the SQL queries. Figure 4.4 displays examples of both simple and challenging natural language statements and their corresponding SQL queries in English and Turkish. For detailed statistics on question and query uniqueness, as well as difficulty levels, refer to Table 4.1. The size and diversity of TURSpider, with its unique questions and varied SQL queries, indicate that the dataset's findings could be valuable for real-world applications.

Sample 1: Easy NL statement with a basic SQL query.
Original SQL Query: SELECT count(*) FROM farm **Original English NL Statement:** How many farms are there? **Translated Turkish SQL Query:** SELECT count(*) FROM çiftlik **Translated Turkish NL Statement:** Burada kaç tane çiftlik var?
Sample 2: Hard NL statement with a complex SQL query.
Original SQL Query: SELECT T1.city FROM city AS T1 JOIN hosting_city AS T2 ON T1.city_id = T2.host_city GROUP BY T2.host_city ORDER BY count(*) DESC LIMIT 1 **Original English NL Statement:** Find the city that hosted the most events. **Translated Turkish SQL Query:** SELECT T1.Şehir FROM şehir AS T1 JOIN evsahibi_şehir AS T2 ON T1.Şehir_No = T2.Evsahibi_No GROUP BY T2.Evsahibi_No ORDER BY count(*) DESC LIMIT 1 **Translated Turkish NL Statement:** En çok etkinliğe ev sahipliği yapan şehri bul.

Figure 4.4 Example Natural Language Statements with Corresponding SQL Queries in TURSpider.

Table 4.1 TURSpider Dataset Distribution. # Q, # SQL, # E, # M, # H, and # EH Denote the Numbers of Unique Questions, Unique SQL Queries, Easy, Medium, Hard, and Extra Hard Level SQL Queries, Respectively.

Dataset	Total	# Q	# SQL	# E	# M	# H	# EH
Dev	1034	1023	584	248	446	174	166
Train	8659	8506	4736	1983	2999	1922	1755

4.1.3 Evaluation

We evaluated the translated dataset using an inter-annotator agreement measure. Three students independently translated the same set of sentences, and the agreement between their translations was calculated. The inter-annotator agreement was 86.36% for the training set and 81.04% for the development set, showing substantial consistency in the translations.

4.2 FINE TUNING TURKISH LANGUAGE MODELS

We performed experiments on the TURSpider dataset using both inference-only and fine-tuning approaches with the following models:

- **GPT-3.5 Turbo** is an OpenAI model accessible via the OpenAI API, offering a context window of 16,385 tokens, enabling it to process and interpret longer text inputs.
- **GPT-4** is a major advancement in artificial intelligence technology, featuring a context window of 128,000 tokens and approximately 1 trillion parameters. This model is optimized for handling complex tasks with exceptional accuracy and efficiency.
- **Trendyol-LLM-7b-base-v0.1** is a generative model based on the Mistral 7B, trained in both English and Turkish.
- **SambaLingo-Turkish-Base** is a pretrained bilingual model for Turkish and English, adapted from Llama-2-7b through training on 42 billion tokens in Turkish.

- **Turkcell-LLM-7b-v1** is an enhanced version of a Mistral-based Large Language Model (LLM) designed for Turkish. It was trained on a curated Turkish dataset with 5 billion tokens, utilizing the DORA (Liu et al., 2024) and LORA (Hu et al., 2021) methods.

In our fine-tuning approach, we used models with 4-bit quantization and 16-bit floating-point precision for computations. The tokenizer was also configured for right-side padding. For training, we applied the SFTTrainer from Hugging Face with a batch size of 4 on a single A100 GPU, and utilized a cosine learning rate schedule.

For GPT-3.5 Turbo (Ye et al., 2023) and GPT-4 (Achiam et al., 2023) inferences, the default settings, including temperature values, were used within the OpenAI API.

An example of an LLM prompt (in Turkish) used as input for Text-to-SQL translation is shown in Figure 4.5. The English translation of the prompt is: "Write an SQL query for the given natural language statement and table information. Natural language statement: Write SQL query for given natural language statement and table information. Natural language statement: Where is the youngest teacher from? Table Information: teacher (Teacher_ID, Name, Age, Hometown)"

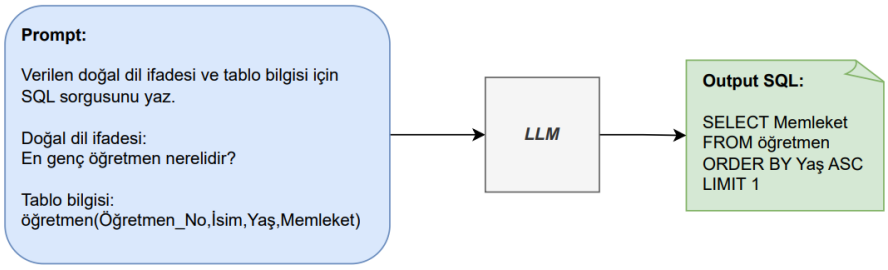

Figure 4.5 LLM Prompt for Text-to-SQL Translation.

4.3 CHAIN-OF-FEEDBACK

Chain-of-Thought (CoT) (Wei et al., 2022) is a widely used prompting technique that enhances the reasoning abilities of large language models by breaking down complex problems into smaller, sequential steps. Building on the CoT approach, several Chain-of-X (CoX) methods (Xia et al., 2024) have been introduced.

A key variant of CoX is the Chain-of-Feedback (CoF), which incorporates feedback from the output of LLMs throughout the generation process to refine responses. In our study, we apply the CoF method to enhance the Text-to-SQL performance of LLMs. Our approach involves an iterative feedback loop where SQL queries are executed, and any errors from the execution are fed back into the system. This error feedback, along with the initial prompt, is then used to regenerate the SQL query.

4.4 EXPERIMENTS AND RESULTS

There is ongoing research on Turkish large language models (LLMs), with new models regularly being developed. In April 2024, we conducted experiments using selected models from Hugging Face (Wolf et al., 2019). For fine-tuning Turkish language models, we focused on popular options such as Llama-2 (Touvron et al., 2023) and Mistral (Jiang et al., 2023a), selecting models of similar sizes, specifically 7B. As a result, we chose Trendyol LLM, Turkcell LLM, and Sambalingo LLM, all of which are industry-specific models from the Hugging Face repository. After fine-tuning, we renamed them TrendyolSQL, TurkcellSQL, and SambaLingoSQL, respectively. We then ran experiments using our proposed TURSpider dataset. For evaluation, we used execution accuracy (EX) (Zhong et al., 2017) across all experiments. Execution accuracy assesses the correctness of predictions by executing the ground truth (G) and predicted (P) SQL queries on the underlying database and comparing the results. A prediction is considered correct if the execution results match. To begin, we

calculated the match score between the ground truth SQL and the predicted SQL, as shown in equation (4.1):

$$match(G,P) = \begin{cases} 1, & G = P \\ 0, & G \neq P \end{cases} \quad (4.1)$$

Then, the execution accuracy (EX) is calculated by equation (4.2):

$$EX = \frac{\sum_{n=1}^{N} match(G,P)}{N} \quad (4.2)$$

We provided 1,034 natural language statements and database schemas from the TURSpider development set to GPT-3.5 Turbo and GPT-4. The execution accuracy of SQL queries obtained through inference-based methods was 55.99 and 57.25, respectively. Fine-tuning Turkish-supported large language models (LLMs) with 8,659 natural language statements and SQL query pairs from the TURSpider training set, along with database schemas, resulted in varying accuracies across different LLMs. The TrendyolSQL model achieved an accuracy of 25.04, SambaLingoSQL reached 30.65, and TurkcellSQL attained 58.22. These results are shown in Table 4.2. The TurkcellSQL model outperformed both inference-based models and other fine-tuned models in terms of accuracy.

Table 4.2 Execution Accuracy (EX) of LLMs on the TURSpider Development Set.

Approach	Model	Execution Accuracy (EX)
Inference-only	GPT-3.5 Turbo	55.99
Inference-only	GPT-4	**57.25**
Fine-tuning	TrendyolSQL	25.04
Fine-tuning	SambaLingoSQL	30.65
Fine-tuning	TurkcellSQL	**58.22**

Table 4.3 shows the execution accuracy results from various LLMs tested against SQL queries of different difficulty levels using the TURSpider development dataset. The analysis reveals notable differences in performance across the LLMs, depending on query complexity. For medium-difficulty SQL

queries, GPT-4 performs the best, demonstrating superior accuracy compared to the other LLMs in the study. This highlights GPT-4's strong ability to interpret and execute moderately difficult SQL queries. TurkcellSQL stands out as the top performer across all difficulty levels—easy, hard, and extra hard—consistently surpassing the other models. Its strong performance emphasizes its robustness in handling a wide range of SQL query challenges.

Table 4.3 Execution Accuracy (EX) of LLMs with Different Hardness Levels on the TURSpider Development Set.

Model	Easy	Medium	Hard	Ex. Hard	All
GPT-3.5 Turbo	80.64	58.29	39.65	30.12	55.99
GPT-4	76.20	**62.10**	41.95	31.92	57.25
TrendyolSQL	40.72	21.97	20.68	14.45	25.04
SambaLingoSQL	50.80	24.43	34.48	13.25	30.65
TurkcellSQL	**82.25**	54.93	**55.74**	**33.73**	**58.22**

Recently, several advanced methods (Kanburoğlu and Tek, 2024b) have demonstrated superior performance on the original Spider dataset compared to the results we obtained on our dataset. For example, custom methods tested on Spider have shown impressive results. The DIN-SQL model, for instance, achieved an execution accuracy of 82.8% using GPT-4, while SQL-PaLM reached an execution accuracy of 82.7% on the same dataset.

4.4.1 CoF Results

Table 4.4 presents a comparison of model execution accuracy (EX) and the number of execution errors for various iterations of language models, including GPT-3.5 Turbo, GPT-4, and TurkcellSQL, both separately and with the Chain-of-Feedback (CoF) method.

Table 4.4 Execution Accuracy (EX) of LLMs with and without Chain-of-Feedback (CoF) on TURSpider Development Set.

Model	Execution Acc. (EX)	# of Execution Errors
GPT-3.5 Turbo	55.99	47
GPT-4	57.25	103
TurkcellSQL	**58.22**	111
GPT-3.5 Turbo + CoF	56.38	28
GPT-4 + CoF	59.57	59
TurkcellSQL + CoF	**60.05**	70

TurkcellSQL achieves the highest execution accuracy of 58.22, with 111 execution errors. However, integrating the CoF approach results in noticeable improvements across all models. GPT-4 + CoF reaches an execution accuracy of 59.57, with 59 execution errors, followed by TurkcellSQL + CoF at 60.05 with 70 execution errors. Interestingly, while GPT-3.5 Turbo + CoF and GPT-4 + CoF show a slight decrease in execution accuracy compared to their unaugmented versions, they exhibit significant reductions in execution errors, suggesting a more refined and precise execution process enabled by the CoF methodology. Overall, the integration of Chain-of-Feedback proves to be an effective enhancement strategy, improving execution accuracy and reducing errors across various language model architectures.

4.4.2 Error Analysis

GPT-3.5 Turbo, GPT-4, and TurkcellSQL were evaluated based on their ability to generate SQL queries against a database schema. The analysis revealed discrepancies between the generated queries and their execution results.

For GPT-3.5 Turbo, out of 1034 generated queries, 579 (55.99%) matched the execution results, while the remaining 455 queries either failed to match or were inaccurately generated. A manual inspection of these 455 queries showed that 408 (89.67%) were syntactically correct SQL queries but did not match the execution results. Additionally, 47 (10.33%) queries were found to be inaccurately generated. Among these inaccuracies, it was observed that 25

queries used non-existent column names, 14 used non-existent table names, 1 had an ambiguous column reference, 1 had the wrong number of arguments for a function, 2 used non-existent functions, and 4 had syntax errors.

Similarly, for GPT-4, out of 1034 generated queries, 592 (57.25%) matched the execution results, leaving 442 queries with discrepancies. Upon manual inspection, 339 (76.70%) of these queries were syntactically correct but did not match the execution results. Additionally, 103 (23.30%) queries were inaccurately generated. Among these inaccuracies, it was found that 26 queries used non-existent column names, 66 used non-existent table names, 2 had ambiguous column references, 1 had the wrong number of arguments for a function, 2 used non-existent functions, and 6 had syntax errors.

Further analysis focused on the TurkcellSQL model. For this model, out of 1034 generated queries, 602 (58.22%) matched the execution results, while the remaining 432 queries had issues. Manual examination showed that 321 (74.31%) of these queries were syntactically correct but did not match the execution results. Additionally, 111 (25.69%) queries were inaccurately generated. Among these inaccuracies, it was observed that 83 queries used non-existent column names, 15 used non-existent table names, 7 had ambiguous column references, 2 had misuse errors, and 4 had syntax errors.

Table 4.5 Error Distribution on Generated SQL Queries for Different LLMs.

Error Type	Models		
	GPT-3.5 Turbo	GPT-4	TurkcellSQL
	Occurrences		
No such table	14	66	15
No such column	25	26	83
No such function	2	2	0
Ambiguous column	1	2	7
Syntax error	4	6	4
Misuse	0	0	2
Wrong # of arguments	1	1	0
TOTAL	47	103	111

We also conducted a more detailed analysis of the error types shown in Table 4.5. Since most errors are caused by “no such table” and “no such column” errors, we focused on these two error types in more detail. We provide examples for “no such table” and “no such column” errors with gold and predicted SQL queries in Table 4.6. We observed that errors mostly arise from Turkish character errors, plural suffix errors, and hallucinations. For instance, in the case of a Turkish character error, if the table name is “öğrenci” (student in English), LLM predicted it as “öğrenci”. Plural suffix errors were seen where “arabalar” (cars in English) was written as “araba” (car in English). For hallucinations, we noticed instances where English words were used instead of Turkish table or column names. There were also made-up words, and underscores were used to separate compound words. The distribution of these errors for GPT-3.5 Turbo, GPT-4, and TurkcellSQL is shown in Figure 4.6.

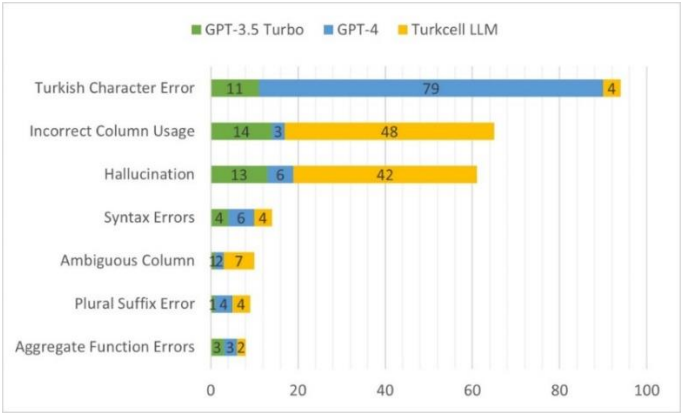

Figure 4.6 Error Detail Distribution.

Figure 4.7 compares the number of errors made by the three models, GPT-3.5-Turbo, GPT-4, and TurkcellSQL, across different difficulty levels. GPT-3.5-Turbo performs well on easy and hard tasks but struggles with medium and extra hard challenges. GPT-4 shows a more balanced distribution of errors, particularly with increases in medium and extra hard tasks. TurkcellSQL does well in easy tasks but has significant difficulties in medium and extra hard levels.

Overall, these findings indicate that while GPT-3.5-Turbo and TurkcellSQL are effective with simpler tasks, all models face challenges with more complex queries.

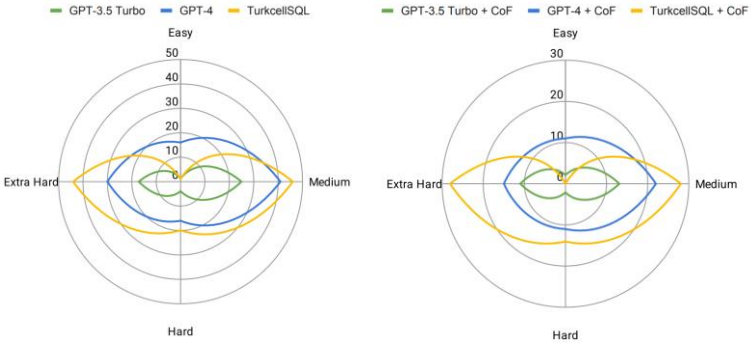

Figure 4.7 Number of Errors Made by LLMs with and without (CoF) Across Different Difficulty Levels.

Table 4.6 Examples of Errors in Generated SQL Queries.

Error Type	Example
No such table	Gold: SELECT isim , soyisim , eposta_adresi FROM Sahipler WHERE eyalet LIKE '%North%' Pred: SELECT isim , soyisim , eposta_adresi FROM sahip WHERE eyalet LIKE '%North%'
No such column	Gold: SELECT max(ağırlık), EvcilHayvanTürü FROM Evcil_hayvanlar GROUP BY EvcilHayvanTürü Pred: SELECT max(ağırlık) , evcil_hayvan_yaşı , evcil_hayvan_türü FROM Evcil_Hayvanlar GROUP BY evcil_hayvan_türü

4.5 CONCLUSION

We presented TURSpider, a large-scale, complex, cross-domain Text-to-SQL dataset containing SQL queries at varying difficulty levels. Translated from the original English Spider dataset into Turkish, TURSpider establishes a strong benchmark for Turkish Text-to-SQL tasks, filling the gap in advanced Turkish

datasets. We also compared the performance of fine-tuned Turkish large language models (LLMs) with inference-based models like GPT-3.5 Turbo and GPT-4, showing that Turkish LLMs are competitive and often outperform these inference-based models. Additionally, we performed an error analysis of the generated SQL queries, offering valuable insights and directions for future research. We believe the public release[8] of TURSpider will serve as an important benchmark for Turkish and cross-lingual Text-to-SQL research and applications.

[8] https://github.com/alibugra/TURSpider

CHAPTER 5

5. MOAF-SQL: A FEEDBACK DRIVEN MIXTURE-OF-AGENTS APPROACH

Large Language Models (LLMs) have shown great potential in the Text-to-SQL task, achieving strong results (Gao et al., 2024; Pourreze and Rafiei, 2024; Kanburoğlu and Tek, 2024) across various benchmarks. However, most previous research has focused on OpenAI LLMs (Pourreza and Rafiei, 2024; Dong et al., 2023), while open-source models have been less studied. Open-source LLMs are more accessible but often face limitations with understanding context and generating responses. Recent advancements in agentic collaboration approaches (Wang et al., 2023; Cen et al., 2024; Xie et al., 2024) have further enhanced the capabilities of LLMs, allowing them more accurate handling of complex queries. These methods, particularly those involving multiple agent collaboration, provide effective ways to improve performance by combining the strengths of different models.

In this study, we draw inspiration by the Mixture-of-Agents (MoA) (Wang et al., 2024) approach, a general framework that enhances the task-solving performance of LLMs through the collaboration of multiple models. We apply MoA to the Text-to-SQL task to improve the translation of natural language statements into executable SQL queries. The collaborative process in MoA helps generate higher-quality SQL queries, leading to better accuracy, particularly for complex queries. Our contributions can be listed as follows:

- Mixture-of-Agents (MoA) improves performance: The MoA approach enhances the performance of open-source LLMs on the Text-to-SQL task, achieving higher accuracy through model collaboration.

- Chain-of-Feedback (CoF) (Xia et al., 2024) contributes additional gains: The CoF technique independently improves model performance by iteratively refining the model's output.
- Combination of MoA and CoF: Integrating MoA with CoF leads to further improvements, showing that these methods can be effectively combined to boost the performance of open-source LLMs on the Text-to-SQL task.

5.1 METHODOLOGY

The Mixture-of-Agents (MoA) methodology (Wang et al., 2024), inspired by the Mixture-of-Experts (MoE) (Shazeer et al., 2017) approach, leverages multiple Large Language Models (LLMs) to collaboratively enhance response quality. During the collaboration process, LLMs categorized into two distinct roles: Proposers and Aggregators. Proposers generate diverse responses, even if they may not be of the highest quality. They form the preliminary context and draft examples, helping to create the final response of the aggregator. Aggregators are models that gather and combine responses generated by Proposers into a single, coherent and informative final response. The MoA structure consists of *l* layers, with each layer *i* containing *n* large language models (LLMs), represented as $A_{i,1}, A_{i,2}, \ldots A_{i,n}$. Each LLM $A_{i,j}$ processes an input text and generates a continuation of it. Given an initial input prompt x_1, the output of the *i*-th, layer in the MoA structure, denoted y_i, is defined as equation (5.1):

$$y_i = \bigoplus_{j=1}^{n} [A_{i,j}(x_i)] + x_1, \qquad x_{i+1} = y_i \tag{5.1}$$

Here, + denotes text concatenation, while $\bigoplus$ represents the application of an Aggregate-and-Synthesize prompt. We take $A_{l,1}(x_l)$ (output from the LLM in the *l*-th layer as the final result and evaluate our metrics based on this output. This structure operates entirely through prompting and generation, without the need for fine-tuning.

In this study, we implemented a two-layer architecture to support the MoA approach. The first-layer consists of multiple LLMs that independently process a given Text-to-SQL prompt, each generating a response based on its interpretation of the query. In the second layer, an aggregation step takes place, where a final LLM synthesizes the outputs from the first-layer models into a cohesive and optimized SQL query. Aggregation step aims to improve the accuracy and quality of the final output by combining the different responses provided by the initial models. As illustrated in Figure 5.1, for example, we utilized three LLMs in the first-layer—Llama3.1 8B (Dubey et al., 2024), Mistral NeMo 12B (AI, 2024), and Qwen2 7B (Yang et al., 2024). In the second layer, the outputs from these models were aggregated by a final LLM, Gemma2 27B (Team et al., 2024a).

With the proposed model we address the following research questions:

- **RQ1:** Does the Mixture-of-Agents (MoA) approach improve the performance of open-source large language models (LLMs) on the Text-to-SQL task?
- **RQ2:** Does the Chain-of-Feedback (CoF) technique enhance the performance of open-source LLMs on the Text-to-SQL task?
- **RQ3:** Does combining the Chain-of-Feedback method with the Mixture-of-Agents approach further improve the performance of open-source LLMs on the Text-to-SQL task?

To evaluate the performance of MoA-SQL, we use the Spider (Yu et al., 2018a) dataset, a comprehensive and widely-used benchmark for Text-to-SQL tasks. The Spider dataset contains a diverse set of queries and corresponding SQL statements, spanning various domains and complexity levels. This enables a rigorous evaluation of our approach against existing models and provides insights into its effectiveness in handling real-world Text-to-SQL challenges.

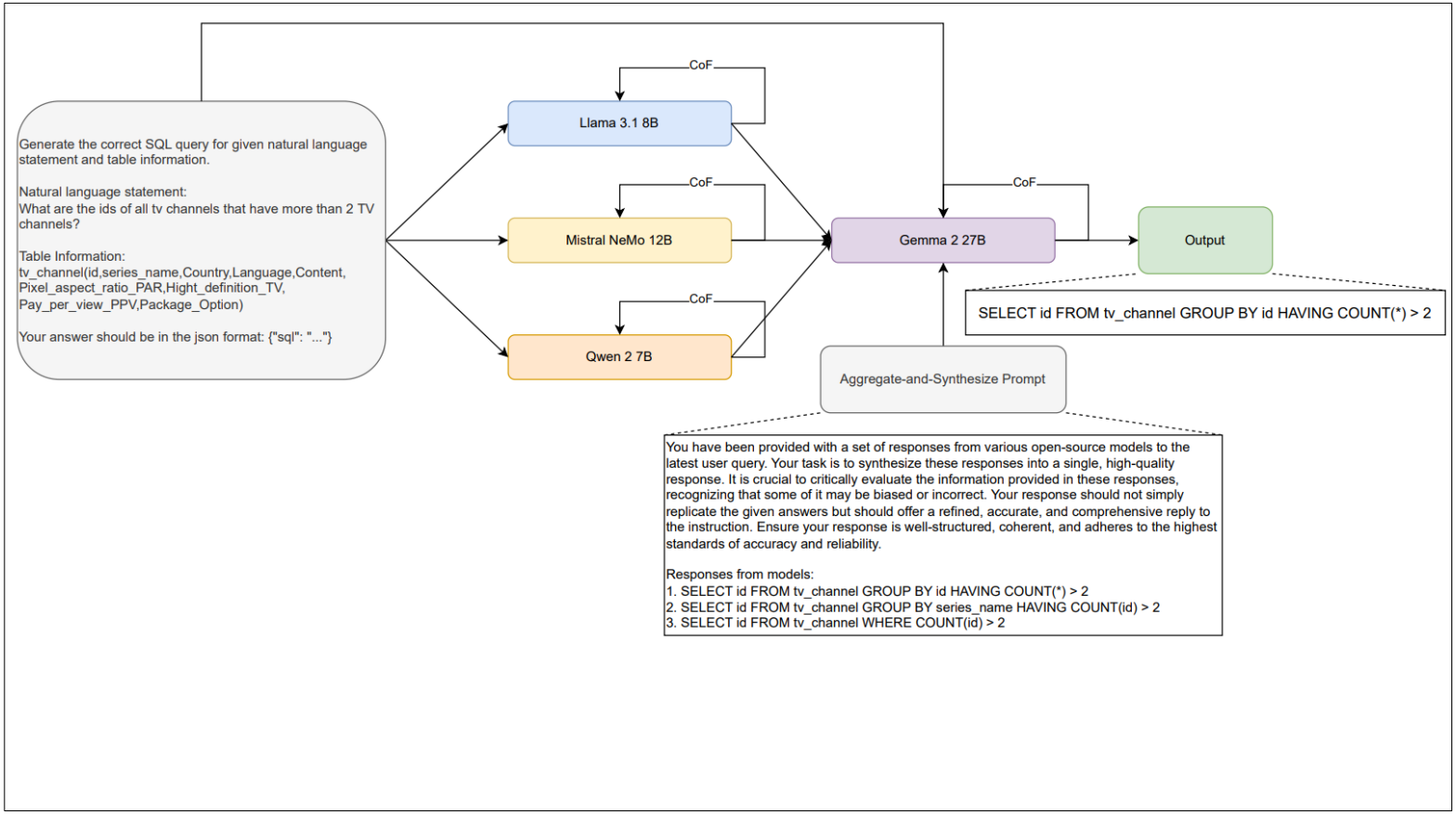

Figure 5.1 Illustration of the Feedback-Driven Mixture-of-Agents (MoAF) Approach. The Example Shows the Three Initial LLMs and the Aggregator LLM, Applied to a Text-to-SQL Task.

We used Execution Accuracy (EX) (Zhong et al., 2017) as an evaluation metric to assess the performance of the predicted SQL queries. EX measures the correctness of the execution results, determining if the generated SQL produces the correct output for the given input.

5.2 EXPERIMENTS AND RESULTS

To implement our experiments, we used Ollama (Ollama, 2024), an open-source framework that simplifies management of large language models. We utilized Ollama's library of models alongside Google Colab (Google, 2024) to download open-source models and perform inference on them. This setup allowed us to efficiently execute our MoA-SQL pipeline and conduct experiments across different model combinations. All models were used with their default parameters to ensure a consistent baseline without parameter modifications to fairly evaluate the impact of the Mixture-of-Agents and Chain-of-Feedback techniques on Text-to-SQL performance.

5.2.1 RQ1: Performance of Mixture-of-Agents Approach

In this section, we compared the performance of the Mixture-of-Agents (MoA) approach to the baseline model.

Table 5.1 Execution Accuracy (EX) of LLMs on the Spider Dev Set.

Approach	Model	Execution Accuracy (EX)
Inference-only	Gemma 2 27B	73.98
Inference-only	Mistral NeMo 12B	70.69
Inference-only	Gemma 2 9B	70.40
Inference-only	CodeGemma 7B	70.30
Inference-only	Llama 3.1 8B	69.82
Inference-only	Qwen 2 7B	64.79
MoA	Gemma 2 27B	76.59
MoAF	Gemma 2 27B	**77.27**

For the aggregator model in MoA, we selected the Gemma 2 27B as the baseline. Based on this, we constructed our MoA architecture using three different open-source language models. When comparing the performance of the MoA architecture to the baseline, we observed that the MoA approach outperforms the Gemma 2 27B model. This comparison is detailed in Table 5.1, demonstrating the effectiveness of the MoA setup in improving performance relative to the baseline model.

5.2.2 RQ2: Effectiveness of Chain-of-Feedback Technique

We evaluated the execution accuracies of five different open-source LLM models on the Spider development set, both with and without the use of the Chain-of-Feedback (CoF) technique. As illustrated in Figure 5.2, the CoF framework begins with the LLM generating an SQL query based on a natural language input. This generated query is executed by a SQL executor. If the query executes successfully, the output is accepted as the final result. However, if the execution fails, the error message provided by the SQL executor is utilized as feedback and fed back into the LLM. The LLM uses this feedback to refine and regenerate the SQL query.

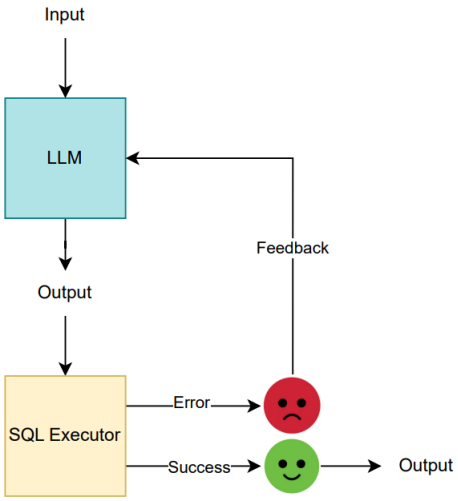

Figure 5.2 Illustration of Chain-of-Feedback (CoF) for Text-to-SQL Task.

The execution accuracy values and the number of execution errors are presented in Table 5.2. The results indicate that applying the CoF technique improves execution accuracy and reduces the number of execution errors for the models. The Llama 3.1 8B model exhibits comparable execution accuracy compared to other models; however, initially it has a higher number of execution errors. The application of CoF significantly improves its performance, resulting in better execution accuracy than other models after the technique is applied.

Table 5.2 Execution Accuracy (EX) of LLMs with and without Chain-of-Feedback (CoF) on Spider Dev Set.

Model	Execution Accuracy (EX)	# of Execution Errors
Qwen 2 7B	64.79	149
Llama 3.1 8B	69.82	103
CodeGemma 7B	70.30	49
Gemma 2 9B	70.40	56
Mistral NeMo 12B	70.69	58
Qwen 2 7B + CoF	67.50	100
Llama 3.1 8B + CoF	73.11	47
CodeGemma 7B + CoF	71.37	27
Gemma 2 9B + CoF	71.08	38
Mistral NeMo 12B + CoF	72.05	29

5.2.3 RQ3: Integration of Chain-of-Feedback and Mixture-of-Agents

In this section, we evaluate whether integrating the Chain-of-Feedback (CoF) technique with the Mixture-of-Agents (MoA) approach provides additional performance gains for open-source LLMs on the Text-to-SQL task. By combining MoA with CoF, we observe further improvements. The CoF technique, an iterative refinement process, introduces a mechanism for further accuracy gains by allowing models to iteratively refine their outputs based on feedback. Table 5.1 illustrates the improvements in execution accuracy when CoF is applied alongside MoA, forming the MoAF approach. These results

demonstrate that the combination of MoA and CoF yields higher performance across varying query difficulty levels as illustrated in Table 5.3.

Table 5.3 Execution Accuracy (EX) of LLMs Across Difficulty Levels on the Spider Dev Set.

Model	Easy	Medium	Hard	Extra Hard	All
Gemma 2 27B	94.35	73.54	59.77	59.63	73.98
MoA-SQL	92.74	76.68	68.96	60.24	76.59
MoAF-SQL	92.74	**76.90**	**69.54**	**63.25**	**77.27**

5.2.4 MoAF for Turkish Study

MoA and MoAF setups on the TURSpider dev dataset show effective performance using open-source LLM models with Turkish language support. The selected models include Aya Expanse 8B, Gemma 2 9B, and Mistral NeMo 12B as proposer models, with Gemma 2 27B as the aggregator model. Table 5.4 shows that both the MoA and MoAF structures perform better than the open-source Gemma 2 27B model, achieving execution accuracies of 57.25 and 60.63, respectively. The MoAF approach, in particular, achieves competitive performance compared to the fine-tuned TurkcellSQL model on the TURSpider dataset.

Table 5.4 Execution Accuracy (EX) of LLMs on the TURSpider Dev Set.

Approach	Model	Execution Accuracy (EX)
Inference-only	Gemma 2 27B	53.77
Inference-only	Mistral NeMo 12B	36.75
Inference-only	Gemma 2 9B	52.32
Inference-only	Aya Expanse 8B	43.13
MoA	Gemma 2 27B	57.25
MoAF	Gemma 2 27B	**60.63**

5.2.5 Impact of Model Subset Combinations

One related research question is whether the achieved results generalize over the various LLMs. To explore this, we conducted 10 experiments by selecting unique subsets of three models from a pool of five distinct models, assessing their impact on execution accuracy in a Text-to-SQL task. This selection process yields 10 unique combinations, calculated using the following formula for combinations:

$$ {}^{n}C_{k} = \frac{n!}{k!\,(n-k)!} \tag{5.2} $$

where:

- n is the total number of models,
- k is the number of models to choose.

The Mixture-of-Agents (MoA) approach utilized a pool of five distinct models: Llama3.1 8B (Dubey et al., 2024), Qwen2 7B (Yang et al., 2024), Mistral NeMo 12B (AI, 2024), Gemma2 9B (Team et al., 2024a), and CodeGemma 7B (Team et al., 2024b).

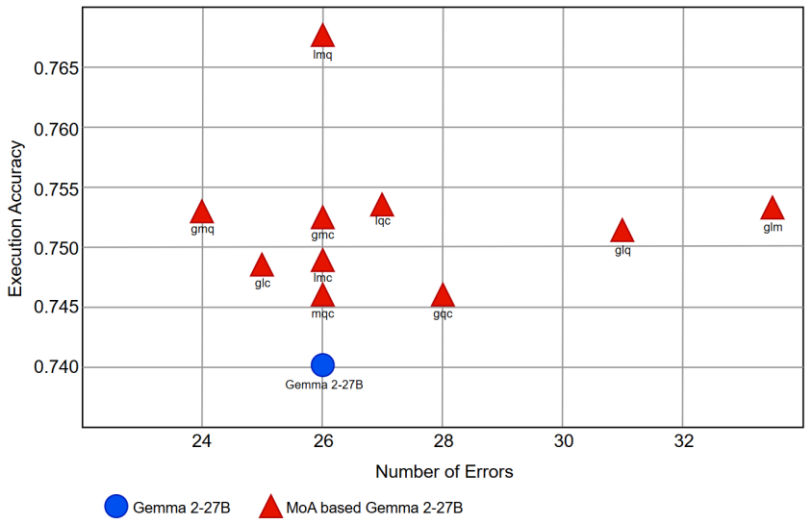

Figure 5.3 Comparison of Errors and Execution Accuracy between the Baseline Model, Gemma 2-27B, and the Mixture-of-Agents-Based Gemma 2-27B. The Subsets, Indicated by Red Triangle Points, Show Varying Levels of Improvement over the Baseline Model, Represented by the Blue Circle Point.

Execution accuracy for each subset was compared to that of the baseline Gemma 2-27B (Team et al., 2024a) model. As shown in Figure 5.3, the subsets, represented by red triangle points, demonstrated varying levels of improvement over the baseline model, represented by the blue circle point. Each subset is labeled according to the constituent models, where “l” represents Llama3.1 8B, “q” represents Qwen2 7B, “m” represents Mistral NeMo 12B, “g” represents Gemma2 9B, and “c” represents CodeGemma 7B. For example, the label “lmq” denotes the subset consisting of Llama3.1 8B, Mistral NeMo 12B, and Qwen2 7B.

The results indicate that the MoA structure yields performance improvements, with subsets achieving diverse execution accuracies and varying numbers of errors. The variation in the number of errors among subsets suggests that specific combinations of LLMs can contribute to more robust performance, particularly in scenarios requiring lower error rates.

5.3 CONCLUSION

We introduced MoAF-SQL, an innovative approach to the Text-to-SQL task that enhances query accuracy by combining the strengths of multiple large language models within a Mixture-of-Agents (MoA) framework. With a two-layer architecture, comprising individual response generation and final output aggregation, MoAF-SQL achieved notable accuracy and robustness in translating natural language prompts into SQL queries. Evaluations on the Spider and TURSpider benchmarks demonstrated that MoAF-SQL achieves competitive results and its potential tackling complex Text-to-SQL conversions.

CHAPTER 6

6. DISCUSSION

This section synthesizes the contributions, limitations, and future directions for each study—TUR2SQL, TURSpider, and MoAF-SQL.

The TUR2SQL study represents an important step forward in Text-to-SQL research for Turkish by introducing the first publicly available cross-domain dataset in this space. It provides a strong starting point for researchers and practitioners. Our experiments showed that ChatGPT significantly outperformed SQLNet, showcasing the power of large language models for this task. However, the dataset's focus on basic queries limits its applicability to more complex, real-world scenarios. Expanding TUR2SQL with more diverse and complex query types could make it even more valuable. Additionally, fine-tuning Turkish LLMs using TUR2SQL could help us better understand their potential.

The TURSpider study extends the scope by introducing a large-scale, complex, and cross-domain Text-to-SQL dataset for Turkish, translated from the English Spider dataset. With varying levels of SQL query complexity, TURSpider establishes a benchmark for Turkish Text-to-SQL tasks and demonstrates the competitive performance of fine-tuned Turkish LLMs compared to inference-based models like GPT-4. Its contributions include not only a valuable dataset but also an error analysis offering directions for improvement. However, translation bias and translation ambiguity limit its utility for purely Turkish applications. Addressing these issues in future work could involve creating native Turkish datasets and optimizing Turkish LLMs for reasoning and coding tasks. Additionally, prompt engineering could further enhance the performance of these models in handling complex queries. In our TURSpider study, we preferred the human translation approach because machine translation methods are not precise enough. Studies like CSpider, PAUQ, and ViText2SQL found that human translation approach is more

accurate and reliable than machine-translated data. However, a comparison with the machine translation method could still be conducted to provide additional insights. According to Sun et al. (2023), GPT-4 achieved 72.9 execution accuracy on the original Spider dataset in a zero-shot setting. Similarly, Liu et al. (2023) reported a 65.1 execution accuracy for ChatGPT on the Chinese CSpider dataset under the same conditions. In contrast, we observed a 57.25 execution accuracy for GPT-4 on the TURSpider dataset in a zero-shot setting, highlighting the importance of our dataset. TURSpider could also support multilingual Text-to-SQL research by serving as a resource for studying language transfer and model fine-tuning. Adding TURSpider to multilingual benchmarks would help researchers explore how Turkish data can improve model performance in other languages.

Finally, the MoAF-SQL study explores a novel approach by combining multiple large language models within a Mixture-of-Agents (MoA) framework to tackle Text-to-SQL tasks. This method proved effective, achieving competitive results on the Spider benchmark. The framework shows promise, but its reliance on open-source models and the lack of optimization in the aggregation prompt are areas where improvement is needed. Future studies could build on this work by experimenting with a three-layer architecture, integrating state-of-the-art models, and refining the aggregation prompts.

In summary, these studies collectively contribute to advancing Text-to-SQL research for Turkish. While each study presents unique contributions, their limitations highlight opportunities for further improvements. Moreover, our experiments utilized open-source and closed-source models in a zero-shot setting. Future research could explore one-shot or few-shot paradigms to evaluate the performance of these models on TUR2SQL and TURSpider datasets. These directions offer a promising pathway for future research to build on the foundations established by this work.

CONCLUSION AND SUGGESTIONS

This thesis contributes to the field of Text-to-SQL with several impactful studies, addressing different challenges and providing new resources and methodologies. Across four distinct areas of work, we have aimed to enhance understanding, establish new benchmarks, and propose innovative approaches to improve Text-to-SQL systems. Below, we summarize the findings and implications of each study and their collective contribution to advancing this domain.

We began with a comprehensive survey of the Text-to-SQL field, examining its challenges, methods, and benchmarks. By presenting a detailed analysis of existing approaches, comparing evaluation metrics, and studying performance across popular datasets, we provided a clear and structured overview of the field. In addition, we showed the transition from deep learning to large language models (LLMs).

One of the main challenges was the lack of a Turkish dataset for Text-to-SQL. We addressed this challenge with the creation of the TUR2SQL dataset. As the first publicly available cross-domain Text-to-SQL dataset in Turkish, TUR2SQL fills a critical gap in resources for this language. The automated framework used to generate the dataset ensures scalability and diversity across domains. Experiments with SQLNet and ChatGPT highlight the challenges and opportunities within this dataset.

However, we found that TUR2SQL was too simplified for real-world queries. To improve this, we developed the TURSpider dataset, which is more complex and better for cross-domain Text-to-SQL tasks. By translating and adapting the well-known Spider dataset, we provided a resource that not only enriches Turkish NLP but also facilitates cross-language research. Our evaluation of fine-tuned Turkish language models versus inference-based models revealed promising results, further emphasizing the dataset’s value for the community.

LLMs performed well in zero-shot settings, there was still room for improvement. MoAF-SQL approach introduced in this thesis demonstrates how innovative frameworks can improve Text-to-SQL accuracy. By leveraging a Mixture-of-Agents architecture, MoAF-SQL combines the strengths of multiple large language models to handle complex query translations with enhanced robustness and precision. Evaluations on the Spider and TURSpider datasets validated the effectiveness of this method, showcasing its potential for advancing Text-to-SQL systems.

In conclusion, Text-to-SQL has improved significantly, but it is not yet fully prepared for widespread industrial applications. There is still much to do, especially for languages like Turkish. Developing datasets, analyzing errors, and improving models will help move this field forward and make Text-to-SQL systems more robust and useful in the future.

REFERENCES

Abbas, S., Khan, M., Lee, S.J., Abbas, A., & Bashir, A. (2022). A Review of NLIDB with Deep Learning: Findings, Challenges and Open Issues. *IEEE Access*.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., & others (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ahkouk, K., Mustapha, M., Khadija, M., & Rachid, M. (2021). A review of the Text to SQL Frameworks. In *Proceedings of the 4th International Conference on Networking, Information Systems & Security*, 1–6.

AI, M. (2024). *Mistral Nemo*. In Mistral AI | Frontier AI in your hands. https://mistral.ai/news/mistral-nemo/

Almohaimeed, S., Almohaimeed, S., Ghanim, M., & Wang, L. (2024). Ar-Spider: Text-to-SQL in Arabic. *arXiv preprint arXiv:2402.15012*.

Anil, R., Dai, A., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., & others (2023). Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Baig, M., Imran, A., Yasin, A., Butt, A., & Khan, M. (2022). Natural Language to SQL Queries: A Review. *Technology, 4*(1), 147–162.

Baik, C., Jagadish, H., & Li, Y. (2019). Bridging the semantic gap with SQL query logs in natural language interfaces to databases. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 374–385.

Bakshandaeva, D., Somov, O., Dmitrieva, E., Davydova, V., & Tutubalina, E. (2022). PAUQ: Text-to-SQL in Russian. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2355–2376.

Bogin, B., Berant, J., & Gardner, M. (2019). Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4560–4565.

Brunner, U., & Stockinger, K. (2021). Valuenet: A natural language-to-sql system that learns from database information. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2177–2182.

Cai, R., Xu, B., Zhang, Z., Yang, X., Li, Z., & Liang, Z. (2018). An encoder-decoder framework translating natural language to database queries. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 3977–3983.

Cai, R., Yuan, J., Xu, B., & Hao, Z. (2021). SADGA: Structure-Aware Dual Graph Aggregation Network for Text-to-SQL. *Advances in Neural Information Processing Systems, 34*, 7664–7676.

Cao, R., Chen, L., Chen, Z., Zhao, Y., Zhu, S., & Yu, K. (2021). LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2541–2555.

Cen, J., Liu, J., Li, Z., & Wang, J. (2024). SQLFixAgent: Towards Semantic-Accurate SQL Generation via Multi-Agent Collaboration. *arXiv preprint arXiv:2406.13408.*

Chen, Z., Chen, L., Zhao, Y., Cao, R., Xu, Z., Zhu, S., & Yu, K. (2021). ShadowGNN: Graph projection neural network for text-to-SQL parser. *arXiv preprint arXiv:2104.04689.*

Choi, D., Shin, M., Kim, E., & Shin, D. (2021). Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Computational Linguistics, 47*(2), 309–332.

Date, C. J. (1989). *A Guide to the SQL Standard*. Addison-Wesley Longman Publishing Co., Inc.

Demirkiran, F., Coşkun, A. K., Kömeçoğlu, Y., Kömeçoğlu, B. B., & Güven, R. (2024, October). Enhancing Text-to-SQL Conversion in Turkish: An Analysis of LLMs with Schema Context. In *2024 9th International Conference on Computer Science and Engineering (UBMK)*, 1-5. IEEE.

Deng, N., Chen, Y., & Zhang, Y. (2022). Recent Advances in Text-to-SQL: A Survey of What We Have and What We Expect. In *Proceedings of the 29th International Conference on Computational Linguistics*, 2166–2187. https://aclanthology.org/2022.coling-1.190.

Dong, L., & Lapata, M. (2018). Coarse-to-Fine Decoding for Neural Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 731–742.

Dong, X., Zhang, C., Ge, Y., Mao, Y., Gao, Y., Chen, L., Lin, J., & Lou, D. (2023). C3: Zero-shot Text-to-SQL with ChatGPT. *arXiv preprint arXiv:2307.07306.*

Dou, L., Gao, Y., Pan, M., Wang, D., Che, W., Lou, J.G., & Zhan, D. (2023). UniSAr: a unified structure-aware autoregressive language model for text-to-SQL semantic parsing. *International Journal of Machine Learning and Cybernetics, 14*(12), 4361-4376.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., & others. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., & Zhou, J. (2024). Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proceedings of the VLDB Endowment*, *17*(5), 1132–1145. https://doi.org/10.14778/3641204.3641221

Google. (2024). *Google Colaboratory*. Retrieved November 24, 2024, from https://colab.research.google.com/

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J.G., Liu, T., & Zhang, D. (2019). Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4524–4535. Association for Computational Linguistics.

Gür, I., Yavuz, S., Su, Y., & Yan, X. (2018). Dialsql: Dialogue based structured query generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1339–1349.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362.

He, P., Mao, Y., Chakrabarti, K., & Chen, W. (2019). X-SQL: reinforce context into schema representation. *Microsoft Research: Artificial Intelligence*.

Hipp, R. (2020). SQLite, 2020. *URL https://www.sqlite.org/index.html*, 35.

Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Huang, P.S., Wang, C., Singh, R., Yih, W.t., & He, X. (2018). Natural Language to Structured Query Generation via Meta-Learning. In *Proceedings of the*

2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). https://doi.org/10.18653/v1/N18-2115.

Hui, B., Shi, X., Geng, R., Li, B., Li, Y., Sun, J., & Zhu, X. (2021). Improving text-to-sql with schema dependency learning. *arXiv preprint arXiv:2103.04399*.

Hui, B., Geng, R., Wang, L., Qin, B., Li, Y., Li, B., Sun, J., & Li, Y. (2022). S2SQL: Injecting Syntax to Question-Schema Interaction Graph Encoder for Text-to-SQL Parsers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1254-1262.

Hwang, W., Yim, J., Park, S., & Seo, M. (2019). A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Iacob, R., Brad, F., Apostol, E.S., Truică, C.O., Hosu, I., & Rebedea, T. (2020). Neural approaches for natural language interfaces to databases: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, 381–395.

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., & Zettlemoyer, L. (2017). Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 963–973.

Jiang, A., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D., Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., & others (2023a). Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Jiang, J., Zhou, K., Dong, Z., Ye, K., Zhao, X., & Wen, J.R. (2023b). StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 9237–9251. Association for Computational Linguistics.

Jose, M., & Cozman, F. (2021). mRAT-SQL+ GAP: a Portuguese text-to-SQL transformer. In *Intelligent Systems: 10th Brazilian Conference, BRACIS 2021, Virtual Event, November 29–December 3, 2021, Proceedings, Part II 10*, 511–525.

Kalajdjieski, J., Toshevska, M., & Stojanovska, F. (2020). Recent Advances in SQL Query Generation: A Survey. *17th International Conference on Informatics and Information Technologies*.

Kanburoğlu, A., & Tek, F. (2023). TUR2SQL: A Cross-Domain Turkish Dataset

For Text-to-SQL. In *2023 8th International Conference on Computer Science and Engineering (UBMK)*, 206–211.

KANBUROĞLU, A., & TEK, F. (2024a). Text-to-SQL: A methodical review of challenges and models. *Turkish Journal of Electrical Engineering and Computer Sciences, 32(3)*, 403–419.

Kanburoğlu, A. B., & Tek, F. B. (2024b). TURSpider: A Turkish Text-to-SQL Dataset and LLM-Based Study. *IEEE Access*, vol. 12, 169379-169387.

Katsogiannis-Meimarakis, G., & Koutrika, G. (2023). A survey on deep learning approaches for text-to-SQL. *The VLDB Journal*, 1–32.

Kim, H., So, B.H., Han, W.S., & Lee, H. (2020). Natural language to SQL: where are we today?. *Proceedings of the VLDB Endowment, 13(10)*, 1737–1750.

Li, F., & Jagadish, H. (2014). Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment, 8(1)*, 73–84.

Liberati, A., Altman, D., Tetzlaff, J., Mulrow, C., Gøtzsche, P., Ioannidis, J., Clarke, M., Devereaux, P., Kleijnen, J., & Moher, D. (2009). The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: explanation and elaboration. *Journal of clinical epidemiology*, *62*(10), e1–e34.

Lin, X., Socher, R., & Xiong, C. (2020). Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 4870–4888.

Liu, A., Hu, X., Wen, L., & Yu, P. (2023). A comprehensive evaluation of ChatGPT's zero-shot Text-to-SQL capability. *arXiv preprint arXiv:2303.13547*.

Liu, S.Y., Wang, C.Y., Yin, H., Molchanov, P., Wang, Y.C., Cheng, K.T., & Chen, M.H. (2024). DoRA: Weight-Decomposed Low-Rank Adaptation. *arXiv preprint arXiv:2402.09353*.

Lyu, Q., Chakrabarti, K., Hathi, S., Kundu, S., Zhang, J., & Chen, Z. (2020). Hybrid ranking network for text-to-sql. *arXiv preprint arXiv:2008.04759*.

Ma, J., Yan, Z., Pang, S., Zhang, Y., & Shen, J. (2020). Mention Extraction and Linking for SQL Query Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6936–6942.

Majhadi, K., & Machkour, M. (2021). The history and recent advances of Natural Language Interfaces for Databases Querying. In *E3S Web of Conferences*, Vol. 229, 01039.

Min, Q., Shi, Y., & Zhang, Y. (2019). A Pilot Study for Chinese SQL Semantic Parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3652–3658.

Nguyen, A., Dao, M., & Nguyen, D. (2020). A Pilot Study of Text-to-SQL Semantic Parsing for Vietnamese. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 4079–4085.

Oflazer, K. (1994). Two-level description of Turkish morphology. *Literary and linguistic computing*, *9*(2), 137–148.

Ollama. (2024). Ollama. Retrieved November 24, 2024, from https://ollama.com/

Owens, M. (2006). *The definitive guide to SQLite*. Springer.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Popescu, A.M., Etzioni, O., & Kautz, H. (2003). Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, 149–157.

Pourreza, M., & Rafiei, D. (2024). Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, vol. 36, 36339-36348.

Price, P. (1990). Evaluation of spoken language systems: The ATIS domain. In *Proceedings of the workshop on Speech and Natural Language (HLT '90)*. Association for Computational Linguistics, USA, 91–95. https://doi.org/10.3115/116580.116612

Qi, J., Tang, J., He, Z., Wan, X., Cheng, Y., Zhou, C., Wang, X., Zhang, Q., & Lin, Z. (2022). RASAT: Integrating Relational Structures into Pretrained Seq2Seq Model for Text-to-SQL. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 3215–3229.

Qin, B., Hui, B., Wang, L., Yang, M., Li, J., Li, B., Geng, R., Cao, R., Sun, J., Si, L., & others (2022). A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions. *arXiv preprint arXiv:2208.13629*.

Saha, D., Floratou, A., Sankaranarayanan, K., Minhas, U., Mittal, A., & Özcan, F. (2016). ATHENA: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment*, *9*(12), 1209–1220.

Sen, J., Lei, C., Quamar, A., Ozcan, F., Efthymiou, V., Dalmia, A., Stager, G., Mittal, A., Saha, D., & Sankaranarayanan, K. (2020). Athena++ natural language querying for complex nested sql queries. *Proceedings of the VLDB Endowment*, *13*(12), 2747–2759.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538.*

Shi, T., Tatwawadi, K., Chakrabarti, K., Mao, Y., Polozov, O., & Chen, W. (2018). Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Sun, Y., Tang, D., Duan, N., Ji, J., Cao, G., Feng, X., Qin, B., Liu, T., & Zhou, M. (2018). Semantic Parsing with Syntax- and Table-Aware SQL Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 361–372. Association for Computational Linguistics.

Sun, R., Arik, S., Nakhost, H., Dai, H., Sinha, R., Yin, P., & Pfister, T. (2023). SQL-PaLM: Improved Large Language ModelAdaptation for Text-to-SQL. *arXiv preprint arXiv:2306.00739*.

Stewart, W. (2009). *Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling*. Princeton university press.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., & others. (2024a). Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118.*

Team, C. (2024b). Codegemma: Open code models based on gemma. *arXiv preprint arXiv:2406.11409.*

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., & others (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wang, B., Shin, R., Liu, X., Polozov, O., & Richardson, M. (2020). RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for*

Computational Linguistics, 7567–7578.

Wang, B., Ren, C., Yang, J., Liang, X., Bai, J., Zhang, Q.-W., Yan, Z., & Li, Z. (2023). Mac-sql: Multi-agent collaboration for text-to-sql. *arXiv preprint arXiv:2312.11242*.

Wang, J., Wang, J., Athiwaratkun, B., Zhang, C., & Zou, J. (2024). Mixture-of-Agents Enhances Large Language Model Capabilities. *arXiv preprint arXiv:2406.04692*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q., Zhou, D., & others (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, *35*, 24824–24837.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., & others (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Xia, Y., Wang, R., Liu, X., Li, M., Yu, T., Chen, X., McAuley, J., & Li, S. (2024). Beyond Chain-of-Thought: A Survey of Chain-of-X Paradigms for LLMs. *arXiv preprint arXiv:2404.15676*.

Xie, W., Wu, G., & Zhou, B. (2024). Mag-sql: Multi-agent generative approach with soft schema linking and iterative sub-sql refinement for text-to-sql. *arXiv preprint arXiv:2408.07930*.

Xu, X., Liu, C., & Song, D. (2017). Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Xu, K., Wang, Y., Wang, Y., Wen, Z., & Dong, Y. (2021). Sead: End-to-end text-to-sql generation with schema-aware denoising. *arXiv preprint arXiv:2105.07911*.

Yaghmazadeh, N., Wang, Y., Dillig, I., & Dillig, T. (2017). SQLizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, *1(OOPSLA)*, 1–26.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., & others. (2024). Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Ye, J., Chen, X., Xu, N., Zu, C., Shao, Z., Liu, S., Cui, Y., Zhou, Z., Gong, C., Shen, Y., & others (2023). A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *arXiv preprint arXiv:2303.10420*.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., & Radev, D. (2018a). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3911–3921. Association for Computational Linguistics.

Yu, T., Li, Z., Zhang, Z., Zhang, R., & Radev, D. (2018b). TypeSQL: Knowledge-Based Type-Aware Neural Text-to-SQL Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 588–594. Association for Computational Linguistics.

Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., & Radev, D. (2018c). SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1653-1663.

Yu, T., Zhang, R., Yasunaga, M., Tan, Y., Lin, X., Li, S., Er, H., Li, I., Pang, B., Chen, T., Ji, E., Dixit, S., Proctor, D., Shim, S., Kraft, J., Zhang, V., Xiong, C., Socher, R., & Radev, D. (2019a). SParC: Cross-Domain Semantic Parsing in Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4511–4523. Association for Computational Linguistics.

Yu, T., Zhang, R., Er, H., Li, S., Xue, E., Pang, B., Lin, X., Tan, Y., Shi, T., Li, Z., & others (2019b). CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 1962–1979.

Zeng, J., Lin, X., Hoi, S., Socher, R., Xiong, C., Lyu, M., & King, I. (2020). Photon: A Robust Cross-Domain Text-to-SQL System. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 204–214.

Zhang, R., Yu, T., Er, H., Shim, S., Xue, E., Lin, X., Shi, T., Xiong, C., Socher, R., & Radev, D. (2019). Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 5338–5349. Association for Computational Linguistics.

Zhang, B., Ye, Y., Du, G., Hu, X., Li, Z., Yang, S., Liu, C., Zhao, R., Li, Z., &

Mao, H. (2024). Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation. *arXiv preprint arXiv:2403.02951*.

Zhong, V., Xiong, C., & Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

CURRICULUM VITAE