Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Datateknik
2024 | LIU-IDA/LITH-EX-A--24/001--SE

Bridging Language & Data: Optimizing Text-to-SQL Generation in Large Language Models

Från ord till SQL: Optimering av text-till-SQL-generering i stora språkmodeller

Niklas Wretblad
Fredrik Gordh Riseby

Supervisor : Oskar Holmström
Examiner : Marco Kuhlmann

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

This thesis explores text-to-SQL generation using Large Language Models within a financial context, aiming to assess the efficacy of current benchmarks and techniques. The central investigation revolves around the accuracy of the BIRD-Bench benchmark and the applicability of text-to-SQL models in real-world scenarios. The research explores why models are not showing significant performance improvements on this benchmark.

The methodology adopted involved a thorough manual analysis of inputs and outputs from two distinct text-to-SQL models: a baseline zero-shot model and the DIN-SQL model, within the financial domain of the BIRD-Bench benchmark. The purpose was to identify and understand the limitations inherent in the dataset and the models themselves.

Findings revealed that the best performing model on the original data was the DIN-SQL model, achieving an accuracy of 40.57%, a performance that raises questions due to its limited efficacy. Upon manual analysis, various types of noise were identified in the dataset, including string capitalization errors, faulty true SQL queries, grammatical errors, mismatches between questions and database schema, and language ambiguities. This led to the curation of two new datasets: one with cleaned questions and SQL queries, and another with only cleaned SQL queries, correcting a total of 52/106 data points.

Subsequent runs of the models on these new datasets showed that data quality significantly impacts model performance. The completely cleaned dataset nearly eliminated the performance gap between the two models, with all models showing a 10%-17% increase in accuracy. Interestingly, on the dataset with only cleaned SQL queries, the performance of the models flipped with the basic zero-shot model now outperformed the DIN-SQL model.

Further analysis of BIRD-Bench's development set across different domains indicated the presence of noise in other areas of the benchmark as well. This suggests that while BIRD-Bench closely resembles real-world scenarios, it falls short in offering a detailed understanding of model performance against different types of noise.

The thesis introduces the concept of classifying noise in natural language questions, aiming to prevent the entry of noisy questions into text-to-SQL models and annotate noise in existing datasets. Experiments using GPT-3.5 and GPT-4 on a manually annotated dataset demonstrated the viability of this approach, with classifiers achieving up to 0.81 recall and 80% accuracy.

Additionally, the thesis explored the use of LLMs for automatically correcting faulty SQL queries. This showed a 100% success rate for specific query corrections, highlighting the potential for LLMs in improving dataset quality.

The implications of these findings are substantial, emphasizing the need for noise-specific benchmarks and enhanced annotations in datasets like BIRD-Bench. This research underscores the importance of addressing specific noise challenges and developing more sophisticated text-to-SQL models.

In conclusion, the thesis offers significant insights into the performance and limitations of text-to-SQL models, setting the stage for future research. This includes creating specialized datasets, enhancing annotations, focusing on identified noise types, developing user-input guardrails, and improving text-to-SQL models overall. Such advancements are expected to significantly improve the functionality and practical application of text-to-SQL technologies across various industries.

Acknowledgments

We begin by expressing our deepest gratitude to our supervisor Oskar Holmström from the NLP group at Linköping University. His passion, humor, and optimism have truly been a beacon of light throughout this journey. His invaluable guidance and expertise in the field of Natural Language Processing have been absolutely instrumental in shaping our research.

We extend our heartfelt thanks to our supervisors from Combient Mix a Silo AI Company, Rahul Biswas and Amin Ahmadi. Their immense patience and welcoming nature have not only introduced us to the intricacies of the corporate world but also provided us with exceptional guidance and support.

Our sincere appreciation goes to Christian Von Koch, whose guidance has also been key. His patience with us, especially on those exhaustive Friday afternoons, has been a source of strength for our tired minds.

We are immensely thankful to Rylan Schaeffer from Stanford University. His advice, kindness, and remarkable willingness to assist us, despite his significant position, have been invaluable to our research.

A special word of thanks to Marco Kuhlmann for graciously accepting the role of our examiner, despite our late submission of the master's thesis application. His understanding and support in this regard have been greatly appreciated, and we thank him for his opinions and insights as well.

Finally, we would like to thank our opponents, Erica and Liv, for their constructive feedback and assistance. Their insights have been critical in refining our thesis and elevating its quality.

Our journey in writing this thesis has been enriched and made possible by the contributions and support of each of these individuals. We are deeply grateful for their involvement in our academic endeavor. A heartfelt thanks to all of you.

Contents

List of Figures

List of Tables

Keywords

- **Chaining**: In the context of language models and machine learning, the process of linking multiple models or tasks in a sequence to achieve more complex objectives or outputs.
- **Classification**: Process in machine learning for recognizing, differentiating, and understanding ideas or objects.
- **Data Quality**: The measure of data's condition, often in terms of accuracy, completeness, reliability, and relevance.
- **Few-Shot Learning**: Machine learning with a very small amount of training data.
- **Language Model**: Models that understand or generate human language.
- **Large Language Model**: Advanced, complex language models with a large number of parameters.
- **Machine Learning**: Field of computer science dealing with algorithms and statistical models for task execution without explicit instructions.
- **Noise**: Unwanted or irrelevant data or disturbances that can affect model performance or data quality.
- **Prompt**: An input or instruction given to a language model or computer program to elicit a response or an action.
- **Prompt Engineering**: The practice of crafting inputs or instructions to effectively interact with and guide language models.
- **SQL (Structured Query Language)**: A domain-specific language for programming and managing data in a relational database management system.
- **Text-to-SQL**: Converting natural language text into Structured Query Language queries.
- **Zero-Shot Learning**: A machine learning approach where a model makes predictions without explicit training on the task.

Abbreviations

- **API**: Application Programming Interface
- **BIRD-Bench**: BIg Bench for LaRge-scale Database Grounded Text-to-SQL Evaluation
- **GPT**: Generative Pre-trained Transformer
- **LLM**: Large Language Model
- **NLP**: Natural Language Processing
- **SQL**: Structured Query Language

1 Introduction

In an era where data-driven decisions play an essential role, the potential of Large Language Models (LLMs) offers exciting opportunities. This research is a joint venture between SEBx, an independent innovation studio initiated by SEB, and Combient Mix, a firm specializing in data science and analytics which is part of Silo AI, Europe's largest private AI lab. It explores the integration of LLMs with structured data sources or more specifically SQL databases. The study assesses the ability of LLMs to communicate with relational databases through generating SQL queries, and what the limitations are for current techniques. It also explores how well current benchmarks and datasets measure the success of such models, and investigates how they can become more robust towards noise in natural language questions. The overarching goal is to explore state-of-the-art technology to transform banking operations and applications. By enabling users to interact with databases through natural language queries, one removes the need for expert knowledge effectively extending the power of data analysis to a broader audience.

1.1 Motivation

In recent years, the emergence of LLMs, including platforms like ChatGPT, has fundamentally altered global perspectives on artificial intelligence and machine learning, reshaping their perceived impacts on society. The versatility of LLMs encompasses a wide array of tasks and domains, encouraging numerous organizations within various industries to explore their potential applications. However, these models exhibit limitations for a number of tasks, including but not limited to arithmetic solutions [1], factual retrieval [2] and updating their world knowledge [3]. Such challenges have lead to the emergence of a distinct research domain dedicated to exploring the potential of integrating LLMs with external tools [2]. Notably, efforts to improve factual retrieval capabilities have culminated in research focusing on the synergy between LLMs and different databases [4, 5, 6].

Established in 1972 and subsequently standardized by ANSI and ISO in 1986, SQL (Structured Query Language) remains the predominant language for relational databases in 2023 [7, 8]. Its pervasive adoption indicates its integration into numerous business systems, highlighting the importance of interfacing LLMs with existing SQL databases. This area presents a promising avenue for enterprises, by empowering users to address questions and queries related to internal organizational data and knowledge stored in structured databases using natural language, removing the need for expert knowledge.

The capabilities of LLMs to understand and generate SQL queries from questions in natural language, commonly referred to as text-to-SQL conversion, represent the key idea behind interfacing LLMs with relational databases. However, the current capabilities of LLMs in text-to-SQL generation are not yet sufficient for a variety of practical applications. This is evident from the benchmark scores of the two most commonly used benchmarks for text-to-SQL, the SPIDER SQL benchmark as well as the BIRD-Bench benchmark, where existing text-to-SQL models struggle to achieve high levels of accuracy [9, 10]. A more concrete illustration of

the difficulties language models sometimes face with text-to-SQL conversion can be found in Figure 1.1. It contrasts an incorrect SQL query produced by an LLM with the correct query for retrieving the number of crimes committed in a specific district in 1995 based on an account ID. The true SQL query includes an inner join between the district and account tables, correctly referencing the account_id, whereas the generated SQL query fails to identify the need for a JOIN and instead incorrectly references the district_id. The figure demonstrates LLMs limitations in text-to-SQL conversion and motivates the need for better techniques.

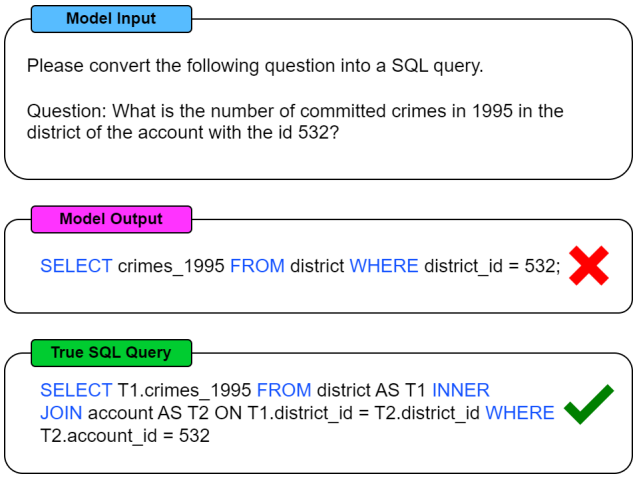

Figure 1.1: An example of a question an LLM struggles in accurately converting.

The focus of this thesis is to investigate the extent to which current benchmarks, such as the SPIDER SQL and BIRD-Bench benchmarks, accurately capture the performance of LLMs in text-to-SQL conversion, particularly given the low accuracies models have demonstrated on these benchmarks. Benchmarks play a crucial role in evaluating and comparing the capabilities of various models, particularly in ensuring they can address a wide array of user queries. However, the inherent ambiguity and complexity of natural language pose significant challenges in developing and evaluating text-to-SQL models that are robust and generalize effectively to real-world scenarios. By thoroughly examining the performance of LLMs against the current text-to-SQL benchmarks, this research aims to provide insights into the strengths and weaknesses of existing models. It seeks to identify specific types of questions and queries where LLMs commonly fail and to understand the underlying reasons for these failures. By pinpointing the shortcomings of current approaches, this research will contribute to the development of more robust and effective text-to-SQL conversion techniques.

1.2 Aim

The primary aim of this thesis is to critically assess and enhance the accuracy of LLMs in text-to-SQL conversion, by evaluating their performance against current benchmarks such as BIRD-Bench. This study will focus on identifying the limitations and gaps in these benchmarks, understanding the specific challenges LLMs face in various query types, and exploring the reasons behind their underperformance in certain scenarios. The objective is to contribute to the development of more robust and effective text-to-SQL conversion techniques, thereby improving the generalizability and reliability of LLMs in interpreting and processing natural language queries into SQL commands. This evaluation will include an analysis of how well these benchmarks account for noise in natural language questions and the quality of data inputs. The research will also explore the models' abilities to identify, classify, and correct data quality issues within the benchmarks, thereby contributing to the development of more robust benchmarks for real-world applications.

Ultimately, this thesis aims to improve the ability of models in integrating with structured data. It strives to provide insights and methodologies that can be leveraged by enterprises, especially in the banking sector, to enhance their operations and decision-making processes through the effective use of LLMs.

1.3 Research questions

1. What is the comparative performance, in terms of accuracy, between a baseline zero-shot text-to-SQL model and the text-to-SQL model DIN-SQL when applied to financial data in the BIRD-Bench benchmark?
2. How does the overall quality of data influence the performance of text-to-SQL models when applied to financial data in the BIRD-Bench benchmark?
3. How do different types of noise and data quality issues impact the performance of text-to-SQL models?
4. To what extent are large language models capable of accurately identifying and categorizing data quality issues within a text-to-SQL benchmark?
5. To what extent can large language models accurately correct data quality issues within a text-to-SQL benchmark?

The quality of data in the research questions refer to grammatical formulation, ambiguity, specificity, and alignment with the database schema. Correct grammar is essential for models to accurately interpret and process queries. Ambiguity in questions can lead to multiple interpretations, while specificity ensures that queries are neither too vague nor overly detailed. A crucial aspect is the alignment of questions with the database schema, as mismatches can result in difficulty in measuring the results. Addressing these elements of data quality is key to enhancing the performance of text-to-SQL models.

1.4 Delimitations

It is important to acknowledge the specific boundaries within which this study operates. The research primarily examines the interaction between selected advanced LLMs and relational SQL databases. This focus limits the study to this particular type of databases, and the findings might not fully extend to other database types.

In assessing the performance of text-to-SQL models, the study makes use of the BIRD-Bench benchmark. The choice of the benchmark is deliberate; however, it means that other available benchmarks in the field are not within the scope of this research. Furthermore, the research is predominantly centered on the financial domain of the BIRD-Bench benchmark due to the collaboration with SEBx and Combient Mix. This domain-specific focus ensures detailed and relevant findings for banking operations but may not encapsulate the challenges or potentials in other domains.

Another key area of this thesis is prompt engineering. While prompt engineering has broader applications outside the text-to-SQL domain, these are not explored extensively in this study. Additionally, the thesis examines how data quality, particularly in terms of grammatical formulation, ambiguity, and specificity, affects model performance. However, it does not delve into other dimensions of data quality such as completeness, consistency, or timeliness.

It is also crucial to note that the research is conducted within the technological limits and capabilities of LLMs as of 2023. Any advancements in technology that occur post-study are

not considered. This time-bound nature of the research implies that future developments could alter some of the findings or their applicability.

Geographical and cultural differences in language, which can significantly impact the application of natural language processing technologies, are not a central focus of this study. The study focuses on text-to-SQL models applied only on questions in the English language. Thus, the research findings may have limited applicability in varying global contexts. Additionally, while the study acknowledges the importance of ethical considerations and data privacy in the interaction between AI and databases, these aspects are not the primary focus of the research.

By setting these boundaries, the thesis aims to offer a focused and in-depth analysis within its defined areas, contributing valuable insights to the field of AI and machine learning, particularly in how LLMs interface with SQL databases in the financial sector.

2 Theory

This chapter provides an outline of the foundational theory of the project. It begins with a background on SQL, the relational database language, examining its key components. This is followed by an introduction to current state-of-the-art language models, with a particular emphasis on the roles of the attention mechanism and transformer architecture. Further sections describe in-context learning, underscoring its importance in enhancing a model's adaptive capabilities. Continuing with the same topic, the chapter subsequently explains prompt engineering, providing an in-depth examination of different logic and guidance methods. It also covers self-correction techniques for LLMs and introduces the reader to the domain of text-to-SQL generation. It concludes with a review of influential related works, offering a perspective on the field's progression and current standing.

2.1 Structured Query Language (SQL)

This subsection is dedicated to introducing the Structured Query Language (SQL). Originating from Edgar F. Codd's research [11], SQL was developed and introduced by IBM's research department in the 1970s. Today, it remains the most widely used database programming language globally [12, 8].

2.1.1 The Relational Database System

Previous to relational databases, the database systems used were either based on a hierarchical model, where the system was built on a tree based structure, or a network model where the system was connected through pointers and links. Both of these systems had a lack of flexibility and were complex to query data from, which led to Codd's research on relational models where these issues where addressed. The initial concept was released 1970 [11].

The relational database system is grounded in set theory, where sets $S_1, S_2, ..., S_n$ are interconnected through a relation R. These sets correspond to database tables within a specific domain; for example, *Californian Schools* could represent a domain within the broader US School database. Each table in the database contains a *Primary Key*, a unique identifier allowing for the distinct access of each data item row. To establish relationships between tables within the domain, the *Foreign Key* is used. This key is a column whose values may appear in multiple tables. For instance, both the district and account tables might include the foreign key *district id*, indicating that each account is registered in a specific district. In this scenario, *district id* serves as the primary key for the district table and a foreign key for the account table.

The relational database system encompasses various other theoretical concepts, which, however, will not be explored or presented in this project [11]. To demonstrate the structure of a relational database like SQL, a database schema is often utilized. This schema acts as a blueprint for the database's domain, providing either a graphical or textual representation of the data model. It effectively illustrates the interrelationships among different entities, such as through entity-relationship diagrams [13].

2.1.2 The Language of SQL

The actual language of SQL, initially named SEQUEL and later renamed to SQL, was presented by Donald D. Chamberlin and Raymond F. Boyce in 1974 and was based on the relational database system [12]. The SQL language leverages a simple, English-like syntax for data definition, manipulation, and query operations. SQL's syntax consists of declarative statements, enabling users to specify what data is required rather than detailing the procedures for retrieving it. This language is characterized by its use of English keywords and a straightforward structure, making it accessible not only to professional programmers but also to those with less technical expertise. The essence of SQL lies in its ability to interact with relational databases in an inuitive and readable format through various operations like CREATE for creating a database table, SELECT for data querying, JOIN for connecting database columns over different tables, COUNT for counting the number of entries based on a constraint, and SUM for summarizing the numerical values of a particular column.

SQLite is a relational database management system that implements a significant subset of the SQL language. However, SQLite deviates in minor functionality aspects from SQL, resulting in increased simplicity. The data types used in SQLite are *INTEGER* for integer values, *TEXT* for characters and textual strings, *REAL* for decimal numbers, *BLOB* for mixed numerical and textual data, and *None* for empty input [14, 15]. In Listing 1, three examples of SQLite queries are provided, each demonstrating different functionalities:

- Example 1: Creating a User Table with columns for UserID, Username, and Email.
- Example 2: Querying all columns and rows from the Users Table.
- Example 3: Querying all rows from the Employees table where the column value for department is "Marketing."

```
# Example 1.
CREATE TABLE Users (
    UserID INTEGER PRIMARY KEY,
    Username TEXT NOT NULL,
    Email TEXT
);

# Example 2.
SELECT * FROM Users;

# Example 3.
SELECT * FROM Employees
WHERE Department = 'Marketing';
```

Listing 1: Three examples of SQLite-queries.

2.1.3 SQL Queries are not Unique

An intriguing aspect of SQL, especially relevant in the context of text-to-SQL conversion, is the non-uniqueness of SQL queries. In other words there can be multiple, syntactically different SQL queries, all capable of producing the same result set. This non-uniqueness is important in understanding and developing efficient text-to-SQL conversion methodologies.

Central to this trait is the expressive flexibility of SQL. SQL is designed to be a declarative language, focusing on the "what" rather than the "how" of data retrieval. This design

philosophy allows for various ways to specify a desired outcome. For instance, consider a database containing information about employees. To retrieve the names of all employees in a specific department, one could use a simple SELECT statement with a WHERE clause. Alternatively, the same result could be achieved using a more complex JOIN operation if the relevant information is spread across multiple tables.

Furthermore, SQL's support for functions, subqueries, and different types of joins (INNER JOIN, LEFT JOIN, etc.) adds more layers of complexity and flexibility. Different combinations of these elements can lead to queries that look entirely different yet yield identical results. For example, aggregation functions like SUM and COUNT can be used in conjunction with GROUP BY clauses in various ways to produce the same summary information from a data set.

For text-to-SQL conversion, this non-uniqueness presents both challenges and opportunities. From a challenge perspective, it means that a text query could be correctly translated into multiple different SQL queries, making it difficult to define a single, correct conversion. This situation demands robust models capable of understanding the nuances of both natural language and SQL syntax to recognize equivalencies in query semantics.

On the opportunity side, this non-uniqueness allows for a degree of flexibility in designing algorithms for text-to-SQL conversion. Central for a generated query is that it retrieves the correct information, but the non-uniqueness opens the door for optimizing queries not only for correctness but also for efficiency, readability, or adherence to specific syntax styles preferred by an organization. Moreover, it allows the development of systems that can adapt to different database schemas or query optimization strategies, enhancing the versatility and applicability of text-to-SQL technologies.

2.2 Large Language Models

Recently, LLMs have had a significant impact in the fields of Natural Language Processing (NLP) and machine learning, and made a noticeable imprint across a wider range of technological innovations such as text generation, chatbots and coding assistants [16, 17]. This shift started with the introduction of transformer-based architectures, including the Generative Pre-trained Transformers (GPT) [18] and the Bidirectional Encoder Representations from Transformers (BERT) [19], both launched in 2018. These models acted as precursors for later advancements in the domain, leading to the introduction of ChatGPT in November 2022 [16]. This development further elevated the global acknowledgment of LLMs. In the following section, we will explore the core mechanisms that support these models. Specifically, we will examine the contributions of the transformer architecture, the attention mechanism, and in-context learning, which are all integral to the functionality and efficacy of LLMs.

2.2.1 The Transformer Architecture

In 2017, the Transformer architecture was introduced, aiming to enhance the encoder-decoder framework of recurrent language models, which traditionally relied on Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks [20]. The idea behind the Transformer architecture is to completely focus on the attention mechanism and avoid recurrence which enhances the potential for parallelization. The transformer architecture is built on two stacks of encoders and decoders. Each stack contains six identical layers, with the encoder layers containing two sub-layers each. In contrast, the decoder stack is structured with three sub-layers in each of its six layers.

The purpose of the encoder-decoder architecture is for the encoder to process the input data and generate a contextual representation. This representation is then utilized by the decoder to generate or transform data based on the context provided. Before reaching the encoder, in NLP tasks, individual tokens, such as words, are transformed into vectors using

an embedding layer. This layer consists of a matrix of vectors, each corresponding to a token in the vocabulary, which are trained to semantically represent the tokens [21].

The sub layers of the encoder and decoder both consists of a multi-head attention layer and a feed forward layer, but the decoder also has a masked multi-head attention layer which can be seen in Figure 2.1. The masked multi-head attention layer in the Transformer architecture allows the model to selectively attend to specific positions in the input sequence, effectively diminishing the influence of non-targeted values [20].

Because of its performance in NLP tasks such as machine translation, document generation and syntactic parsing and its capacity to attend to long data sequences the transformer architecture was used in the original GPT model [18]. Owing to its efficiency in harnessing parallelism, the Transformer architecture has extended its applicability beyond NLP. It is now utilized in domains like computer vision, specifically in recognition tasks, video processing, multimodal tasks, and three-dimensional analysis [22].

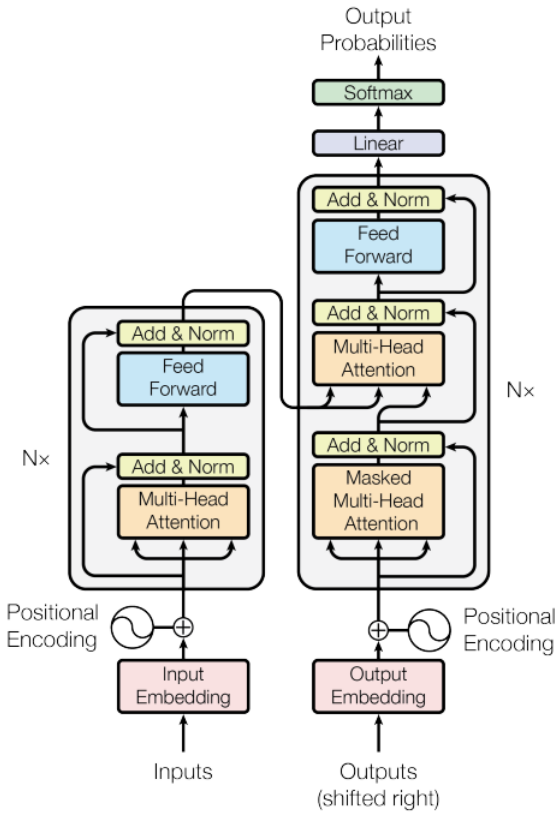

Figure 2.1: The Transformer model architecture [20].

2.2.2 Attention

When the Attention mechanism was introduced, existing models struggled to process long inputs accurately; they tended to focus too much on the values at the beginning and end of the sequence [20]. To solve this problem, the Attention mechanism was introduced which enabled models to focus on different parts of the input sequence, assigning more importance to certain elements while processing others. The mechanism improved their ability to understand context, relationships, and dependencies within the data, thereby enhancing their performance in tasks such as language understanding and translation through amplifying their potential and ability to understand context.

There are various attention mechanisms in the field, but the one predominantly employed in the Transformer architecture combines a method known as scaled dot-product attention within a structure referred to as multi-head attention. The architectures of scaled dot-product attention and multi-head attention are illustrated in Figure 2.2.

In the context of the Transformers architecture, equation 2.1 defines the operation of the scaled dot-product attention mechanism, which involves the manipulation of three distinct vectors referred to as queries (Q), keys (K), and values (V). To explain further:

- Queries (Q) signify the current word or token under examination.
- Keys (K) offer representational context for these queries, signaling the connections between the data values and their corresponding query.
- Values (V), on the other hand, are the data points corresponding to the aforementioned keys.

This architecture is similar to traditional dot-product attention. However, it introduces a scaling factor, represented by $\sqrt{d_k}$, correlating to the dimensionality of the key vector K. The coefficient $\frac{1}{\sqrt{d_k}}$ is introduced to counteract small gradients, ensuring the stability of the softmax function, especially as the magnitude of $\sqrt{d_k}$ increases.

$$\text{Attention}(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Within the multi-head attention framework (equation 2.2), the scaled-dot attention is repetitively employed across each head (h), as depicted to the right in Figure 2.2. In the mechanism, the input is split into multiple parts (or "heads"), and each head performs the attention operation independently of the others which increases the capacity of the attention mechanism without affecting the computational cost. Furthermore, this architecture allows each head layer to learn and focus on different data patterns of the input data, which results in a more expressive and powerful model. Proceeding the multi-head calculations the layers are concatinated and projected through a linear layer [20].

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(head_1, \ldots, head_h)W^O \\ \text{where } head_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.2}$$

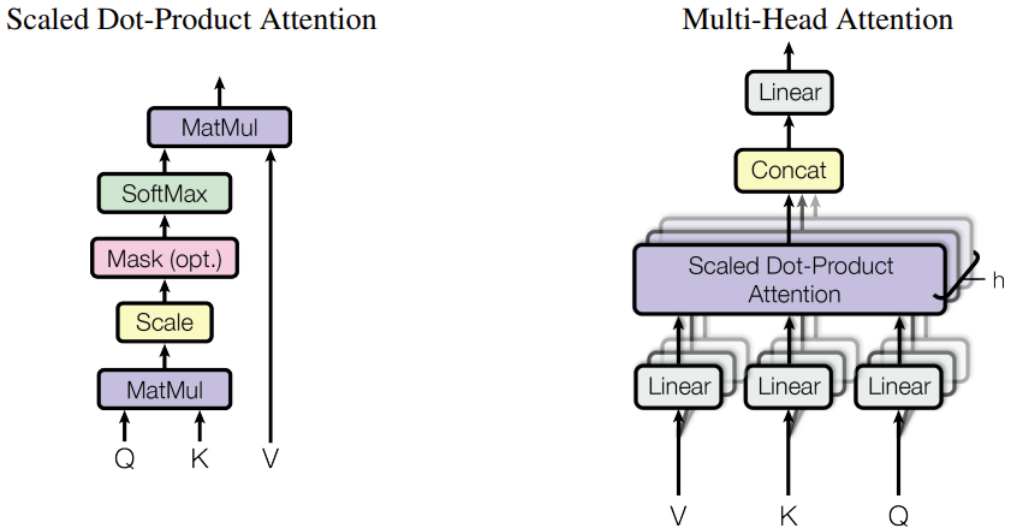

Figure 2.2: Scaled Dot-Product Attention (left). Multi-Head Attention (right) [20].

2.2.3 GPT

The Generative Pre-trained Transformer (GPT), a type and family of LLMs, is constructed using a decoder-based transformer model and undergoes a two-step training process: unsupervised pre-training followed by supervised fine-tuning. Initially, the model is pre-trained with a vast body of unlabeled text, followed by the fine-tuning phase involving manually annotated training examples [18]. The initial version of GPT, developed by OpenAI, made its debut in 2018, equipped with 110 million parameters [23]. Thereafter, the number of parameters has increased for each model released, where GPT-3 was trained on 175 million parameters (Table 2.3.) [24]. However, the most recent model GPT-4, released in March 2023, has an unconfirmed amount of parameters which has led to inquiries about the sources of its data and parameters. There have been speculations of the model being trained on significantly more parameters than its predecessing models [25]. Although the specific parameters of GPT-4 are not publicly available, studies have compared GPT-4 to its earlier versions. Open AI reports in a study that GPT-4 surpasses GPT-3.5 in few shot scenarios in several recognized benchmarks such as MMLU [26], GSM-8K [27] and HumanEval [28, 29]. Furthermore, using questions from the U.S. Multistate Bar Examination (MBE), a standard test for obtaining a law degree, GPT-4 was subjected to an external evaluation outside OpenAI. Remarkably, it demonstrated the ability to pass this exam with a 75.7% success rate, using only zero-shot prompting. This performance surpasses that of the model implemented in ChatGPT at the time, which scored 49.2% [30]. Additionally, GPT-4's performance was assessed in the medical domain using questions from the United States Medical Licensing Examination (USMLE). The USMLE is a multi-step exam that is essential for medical licensing in the United States. In these tests, GPT-4 attained an overall average accuracy of 83.76% through zero-shot prompting. This result is notably higher than that of the GPT-3.5 model, which scored 49.62% in similar conditions [31].

An identified phenomenon resulting from an increased number of paramters is *emergent abilities* [32], which suggests that an LLM, could gain knowledge of how to perform new tasks without having specifically been fine-tuned at that particular task. This could result in that a model with a significantly higher amount of parameters, potentially such as GPT 4, has the ability to perform tasks at a higher performance than a smaller model such as GPT-3.5, which could one explanation to the performance differences presented in the studies above [32].

Table 2.1: Comparison of GPT Parameters [25, 33].

Models	GPT	GPT2	GPT3	GPT4
Release Year	2018	2019	2020	2023
Parameters	110M	1.5B	175B	Unconfirmed

Another factor that affects how the GPT-models perform is the release of updated versions over time. It is still vague how and when these updates are being launched to the public, and as *Chen et al.* [34] mentions there is a need to frequently measure these differences in performance over time. It has been shown that the performance of GPT 4 in arithmetic tasks, such as deciding if an integer is a prime or not, decreased from an accuracy of 84 % to 51.1 % between the March and June updates in 2023 [34]. At the same time GPT 3.5 had a large improvement from 49.6 % to 76.2 % accuracy in the same task from the same periodical updates. Because of the significant changes over a relatively short time there is a need to continuously evaluate the GPT models over time. In Table 2.2, the most recent versions of the GPT models are presented along with their release notes, release dates, and the dates when their training data were collected. It should be noted that as of December 22, 2023, when accessing the OpenAI API, the *gpt-3.5-turbo* and *gpt-4* endpoints default to the models released in June 2023, specifically *gpt-3.5-turbo-0613* and *gpt-4-0613*, respectively [35, 36].

Table 2.2: Release information for GPT-3.5 and GPT-4 versions [35, 36].

Model	**Version**	**Release Notes**	**Release Date**	**Training Data**
GPT-3.5	gpt-3.5-turbo-1106 [37]	New version of GPT-3.5 Turbo supporting a 16K context window. Improved instruction following, JSON mode, and parallel function calling.	2023-11-06	Up to Sep 2021
	gpt-3.5-turbo-0613 [35]	Snapshot of gpt-3.5-turbo from June 13th 2023. Will be deprecated on June 13, 2024.	2023-06-13	Up to Sep 2021
	gpt-3.5-turbo-0301 [35]	Snapshot of gpt-3.5-turbo from March 1st 2023. Will be deprecated on June 13th 2024.	2023-03-01	Up to Sep 2021
	text-davinci-003 [38]	Better quality and consistency in language tasks than curie, babbage, or ada models. Will be deprecated on Jan 4th 2024.	2022-11-28	Up to Jun 2021
GPT-4	gpt-4-1106-preview (GPT-4-Turbo) [37]	More capable, cheaper GPT-4 Turbo model with a 128K context window.	2023-11-06	Up to Apr 2023
	gpt-4-0613 [36]	Snapshot of gpt-4 from June 13th 2023 with improved function calling support.	2023-06-13	Up to Sep 2021
	gpt-4-0314 [36]	Snapshot of gpt-4 from March 14th 2023 with function calling support. Will be deprecated on June 13th 2024.	2023-03-14	Up to Sep 2022
	gpt-4-32k-0613 [36]	Snapshot of gpt-4-32k from June 13th 2023 with improved function calling support.	2023-06-13	Up to Sep 2023
	gpt-4-32k-0314 [36]	Snapshot of gpt-4-32k from March 14th 2023 with function calling support. Will be deprecated on June 13th 2024.	2023-03-14	Up to Sep 2024

2.2.4 In-Context-Learning

The *Attention* mechanism within LLMs enables more efficient context-based learning in comparison to recurrent neural models, this is due to several factors. Firstly, the attention mechanism allows the architecture to focus on several aspects of the data through parallelization. Secondly, precursor models such as RNN and LSTM networks process data sequen-

tially, resulting in that there may be limitations when processing longer data dependencies, due to vanishing gradients, whereas these limitations are reduced because of parallellization in attention. This increases the capacity for In-Context Learning (ICL) in LLMs. ICL emerges through the mechanism of *prompting*, a process in which user interaction is completed through natural language instructions [39]. ICL refers to the ability of LLMs to learn from the immediate context they are given (in the form of a prompt) without updating their underlying parameters. Essentially, the model adjusts its responses based on the specific context provided within a session or prompt, without requiring traditional training or fine-tuning. This enables LLMs to adapt to new information or specific tasks by simply using relevant context. In the following texts, the concepts of zero-shot and few-shot learning will be explained which suggests two different approaches of giving an LLM a context.

Zero-shot Learning

In an LLM equipped with a substantial number of parameters and trained on an extensive corpus of textual data, the capacity emerges for responding to queries based solely on the question itself [40], without the need for additional contextual information. This ability to perform tasks without task-specific labeled examples is referred to as zero-shot learning [41]. Moreover, the employment of zero-shot learning techniques can offer the advantage of reducing the need for redundant information, thereby mitigating excessive costs associated with token and API usage [42].

Table 2.3: Examples of zero-shot learning, tested on the GPT-2 model [41].

Question	Generated Answer	Accuracy
Who wrote the book the origin of species?	Charles Darwin	83.4%
Panda is a national animal of which country?	China	76.9 %
Who came up with the theory of relativity?	Albert Einstein	76.4 %

Few-shot Learning

In what is called few-shot learning, an LLM is provided with learning examples or in the context of text-to-SQL conversion example questions accompanied by their corresponding accurate answer, thereby establishing a labeled data context. *Brown et al.* proposes that the efficacy of few-shot learning improves more rapidly with model size in comparison to zero-shot learning [40]. This implies that a larger LLM exhibits enhanced capabilities of in-context learning [40].

Studies have shown that the selection of examples is crucial and that it correlates to the performance output of the model. When the examples where selected based on similarity, through nearest neighbor, it has been shown that the performance of the model increased compared to when prompted with random examples [43]. Furthermore, it has been discovered that the ordering of the examples in the prompt affects the performance [44]. An example of few-shot learning can be seen in Table 2.4, where the LLM infers what to respond based on a previous example. The text in italics corresponds to the input prompt, and the bold text the output from the LLM.

Table 2.4: Few-Shot Learning with a Language Model [40].

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is: We were traveling in Africa and we saw these very cute whatpus. A "Burringo" is a car with very fast acceleration. An example of a sentence that uses the word Burringo is: **In our garage we have a Burringo that my father drives to work every day.**

2.2.5 Hallucinations

The rapid improvement of Natural Language Generation (NLG) have enabled LLMs to create well written and grammatical complex texts that often tend to be considered trustworthy. However, studies have shown that the models are prone to generate non-factual texts, *hallucinations*, that deviates from reality and facts [45, 46]. Additionally, detecting hallucinations presents a challenge due to the textual fluency and well-structured nature of the generated content, which can often lead users astray [46]. Even the current state-of-the-art models, GPT 3.5 and GPT 4 from OpenAI, suffers from hallucinations. One solution to this problem could be to connect the LLM with an external knowledge base to ensure that correct facts are available to the model [47].

Table 2.5: Hallucination generated from ChatGPT [47].

Prompt	Where do people smoke more on average than in Russia?
ChatGPT Generation	There are several countries where people smoke more on average than in Russia. According to data from the World Health Organization, the following countries have higher smoking rates: Indonesia: 55.2%, China: 52.9%, Serbia: 47.5%, Serbia and Montenegro: 45.6%, Montenegro: 45.6%, Greece: 44.5%, Bosnia and Herzegovina: 41.8%, Kazakhstan: 41.3%, Kosovo: 41.3%, Moldova: 40.9%. Note that these Figures are estimates and may vary.
Hallucination	The generated answer includes the wrong statistics (compared with statistics from the prevalence of current tobacco use from the World Bank statistics [48]).

2.3 Prompt Engineering

Prompt engineering refers to the practice of crafting, refining, and optimizing inputs (or prompts) to obtain desired outputs from machine learning models, particularly large-scale language models [49]. This has become an area of particular interest with the rise of models like OpenAI's GPT series, where the choice of prompt can significantly influence the quality, accuracy, and relevance of the model's response [50].

The importance of prompt engineering stems from its ability to make use of the huge potential of LLMs. By setting a context or giving explicit directions, prompts act as navigational tools, ensuring that the resulting outputs are aligned with user intentions. The sheer volume of data and knowledge LLMs are trained on necessitates this guidance; without a well-structured prompt, a model might provide generic or off-target responses or might even generate false or fabricated information in the form of hallucinations [51]. In contrast, with effective prompt engineering, the output can be remarkably tailored, sharp, and aligned with the user's intent. Additionally, prompts can be tailored to guide an LLM in ways beyond just determining the output form or refining the input information. With carefully crafted prompts, one can introduce new interaction methods [49]. For instance, an LLM might be directed to draft a synopsis of a historical event or simulate a virtual assistant's conversational flow. Or, more importantly, by crafting better prompts one can significantly increase an LLMs ability to perform complex reasoning [1, 52]. The subsequent parts of this section describe the fundamental components of prompt engineering and the current state-of-the-art reasoning techniques.

2.3.1 Prompt Templates

A prompt template can be conceptualized as a textual string comprising one or several input slots containing dynamic variables. These variables are subject to alteration based on the context and specific requirements of the prompt. In implementation contexts, where the LLM is integrated through an API or a locally stored model, this adaptability of the prompt template significantly enhances usability and scalability for varying coding scenarios. The definition of prompt template engineering can be considered as *"The process of creating a prompting function* $f_{prompt}(x)$ *that results in the most effective performance on the downstream task."* [53].

```
prompt_template = PromptTemplate.from_template(
    "Tell me a {adjective} joke about {content}."
)
prompt_template.format(adjective="funny", content="chickens")

Output:
 'Tell me a funny joke about chickens.'
```

Listing 2: An example of a prompt template [54].

2.3.2 Prompt Chaining

There are complex tasks where LLMs show limited performance, for instance in artithmetic tasks, debugging software code or writing long complete essays in a single run [2, 55]. Therefore, the concept of prompt chaining was initiated where problems are divided into subtasks, where each subtask can be completed by an independent run of an LLM. Moreover, prompt chaining facilitates the incorporation of human feedback between steps, potentially enhancing overall performance. Studies have shown significant benefits of this method: participants reported an 82% improvement in task outcome quality when utilizing chaining. Additionally, it provided enhanced controllability, increased transparency, and improved debuggability. Notably, the subdivided subtasks also served as a safeguard against model hallucinations [55].

2.3.3 Prompt Techniques

A prompt technique constitutes a strategic combination of prompt templates and prompt chaining, aimed at enhancing the performance of LLMs in a variety of tasks. This section will provide an examination of the specific prompting techniques *Chain of Thought* and different types of *Self-Correction* strategies.

Chain of Thought

Chain of thought prompting is a method meant to enhance the reasoning capabilities of LLMs [1]. Although scaling up language models has provided numerous benefits, they still struggle when confronted with tasks demanding more complex reasoning, such as arithmetic or commonsense reasoning. The core idea behind chain of thought prompting is to combine two distinct techniques to better language models abilities on such tasks. The first of these involves the generation of natural language rationales, which act as intermediate steps guiding the model to the answer. The second is in-context few-shot learning through prompting, wherein the model learns a new task only through getting acquainted with a handful of input-output examples that demonstrate said task, thereby avoiding the need for extensive training.

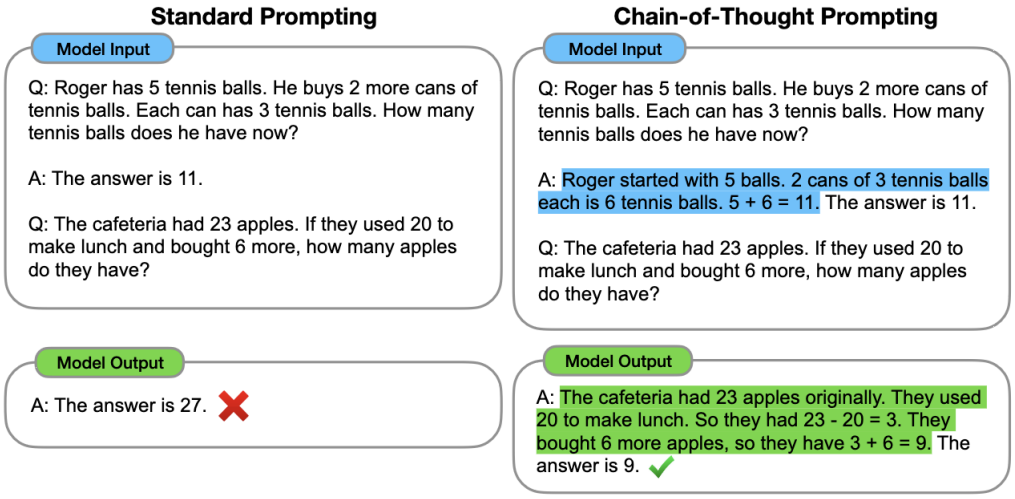

Figure 2.3: The difference of a standard prompt versus a chain of thought prompt [1]. The highlighted blue text is the chain of thought part of the prompt triples used to instruct the model to perform reasoning. The part of the answer highlighted in green is the specific part of the model response corresponding to the chain of thought prompt.

At its core, chain of thought prompting operates by prompting the language model with prompts formulated as triples: these comprise an input, a chain of thought, and an output as can be seen in Figure 2.3. Here, the chain of thought serves as a sequence of intermediate natural language steps, crafted to steer the model towards the final output. The method has excelled in empirical evaluations, showcasing its superiority over standard prompting techniques for reasoning tasks. For instance, the technique broke the previous record on the GSM8K benchmark of math word problems when the original paper was released back in January 2022 and has also performed well on newer text-to-SQL benchmarks [6]. The technique introduced a new standard in prompting for reasoning tasks and surpassed previous records of state-of-the-art performance, and the concepts behind it still serve as a backbone for newer state-of-the-art approaches [56, 57, 58, 59].

Self-Correction for Large Language Models

LLMs efficacy is often undermined by undesired and inconsistent behaviors, such as hallucinations, unfaithful reasoning, and toxic content [45, 3]. A promising approach to address these issues is self-correction, where an LLM is prompted or guided to correct problems in its own output [60]. Self-correction in LLMs like GPT-4 involves them refining their initial responses to enhance accuracy and relevance. Initially, the LLM generates a response based on the given prompt. However, this response may contain errors or inaccuracies. In the next step, the LLM is prompted to revise its response to correct the identified mistakes. This could involve altering phrases, adding information, or changing perspectives based on the nature of the errors. This self-correcting mechanism is used to make LLMs more reliable and accurate in responding to diverse prompts.

2.4 Text-to-SQL

Text-to-SQL is a technique within the field of NLP that focuses on translating natural language statements into SQL queries. At a fundamental level, it allows users to interact with databases using common language rather than technical SQL commands [61]. For instance, if

a user asks, "How many products are in the inventory?", a text-to-SQL system would generate a SQL query like:

```sql
SELECT COUNT(*) FROM inventory;
```

Listing 3: Basic SQL Query.

The mechanism underlying this process faces several complexities [42]. Firstly, understanding the user's intent is central, which involves extracting relevant entities and conditions from the input text. Next, the system confronts the challenge of schema mapping. This phase involves associating elements from the natural language query, such as "products", to their database counterparts, which could be a table of items or a related entity using some other name. Subsequently, after grasping the user's intention and mapping it to the database schema, the system begins the SQL generation phase. The generated query can range from a straightforward SELECT statement to something considerably more complex, potentially containing multiple joins, conditions, and aggregations.

The possible use cases of text-to-SQL are many. It enables non-experts to engage with databases without the need to master SQL. This feature is particularly attractive for chatbots and virtual assistants, which can make use of text-to-SQL to obtain data from databases in response to user queries. Moreover, in the realm of business intelligence, such a technique can empower users to ask data-driven questions in their natural everyday language, streamlining the data analysis process.

With the evolution of deep learning and the arrival of architectures like BERT and GPT, the text-to-SQL domain has witnessed considerable advancements [62, 56, 6, 61]. Furthermore, datasets such as Spider and BIRD-Bench provide benchmarks for text-to-SQL, helping in the evaluation and continual refinement of these systems in ongoing research endeavors [61, 6].

2.5 Classification

Text classification, an important aspect of NLP, has garnered increased attention due to the evolving capabilities of LLMs. The potential of these models in classifying both text and code presents an area for research and evaluation [63].

The definition of classification could be described as *"The subject of classification is concerned with the investigation of relationships within a set of 'objects" in order to establish whether or not the data can validly be summarized by a small number of classes (or clusters) or similar objects"* [64]. Classification has a fundamental role in machine learning, in particular, supervised machine learning, where a model is trained on labeled data to be able to predict and classify new data into these labels

In classification, the outcomes of predictions are categorized into four types: *True Positive* (TP) and *True Negative* (TN), which are correct predictions for the presence and absence of a feature, respectively. *False Positive* (FP) occurs when a feature is incorrectly predicted to be present, while *False Negative* (FN) happens when a feature's presence is incorrectly predicted as absent. Essentially, TP and TN represent correct predictions, whereas FP and FN signify errors in prediction. To visualize these predicted categories a confusion matrix is often used which is displayed in Table 2.6 [65, 66].

There are different metrics of how accurately a model classifies data and this section will describe four of these supervised classification metrics and their purpose.

Table 2.6: The structure of a confusion matrix with two classes [64]

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

2.5.1 Classification Metrics

When using classification in supervised machine learning the aim is to minimize the error when classifying. To measure this the four most common metrics are *Accuracy*, *Recall*, *F-1 Score*, and *Precision*, which will be presented below.

Accuracy

Accuracy is a widely used and straightforward metric for evaluating classification models, as it calculates the ratio of correct predictions to the total number of predictions made [67].

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Observations}} \tag{2.3}$$

Recall

Recall, also known as sensitivity in psychology, represents the ratio of correctly identified positive cases (TP) to the total actual positive cases. This metric is particularly vital in medical diagnostics, as it measures the model's ability to accurately identify true cases of illnesses, emphasizing the importance of not missing any real positive cases [65].

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.4}$$

Precision

In data mining, precision, also referred to as confidence, measures the proportion of actual positive cases (TP) in all predicted positive cases, including false positives (FP). It essentially calculates the model's accuracy in correctly predicting positive instances [65].

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.5}$$

F-1 Score

The F1-score, or F-measure, is a statistical tool that combines precision and recall into a single metric by calculating their harmonic mean. This mean is particularly effective for ratios, as it tends to penalize extreme values. The F1 score is often preferred when both false positives and false negatives are significant, providing a balanced view of a model's performance in handling positive cases. This makes it a usable metric, especially in situations where relying solely on either precision or recall might be misleading. [68, 69].

$$\text{F-1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.6}$$

2.6 Related Work

This section focuses on two central areas in the text-to-SQL domain. The first subsection outlines the primary benchmarks and datasets used for evaluating text-to-SQL systems. The following Subsection, *2.6.3 Text-to-SQL Methods*, presents the various methods that have been proposed to convert natural language queries into SQL statements. Together, these sections

provide a structured overview of the foundational and recent works that inform the discussions and investigations in this thesis.

2.6.1 Text-to-SQL Benchmarks & Datasets

The *Spider-dataset* [61] is a large-scale cross-domain text-to-SQL dataset that was annotated by 11 Yale students in 2018. The dataset consists of 10,181 questions and 5693 unique SQL queries in 200 databases with multiple tables across 138 different domains, the total size of the dataset is approximately 950 MB. Following the release of the WikiSQL dataset in 2017 [70], the Spider dataset emerged as the subsequent significant text-to-SQL dataset and is now among the most frequently employed datasets for benchmarking in this domain [62, 56, 42]. One of the core objectives of the Spider dataset is to increase the complexity of data in comparison to WikiSQL. It goes beyond merely utilizing SELECT and WHERE clauses, incorporating more elaborate queries that encompass GROUP BY, ORDER BY, and HAVING clauses within nested queries [61]. Each SQL-query has been divided into 4 levels of complexity: easy, medium, hard and extra hard. The complexity is decided based on the number of SQL components, selections and conditions. For instance the more clauses of GROUP BY, ORDER BY and INTERSECT or nested sub-queries an SQL query contains, the harder the prompt is considered. The spider paper describes the difficulty of hard as: *"For example, a query is considered as hard if it includes more than two SELECT columns, more than two WHERE conditions, and GROUP BY two columns, or contains EXCEPT or nested queries. A SQL with more additions on top of that is considered as extra hard."* [61].

Another text-to-SQL dataset is BIRD-Bench [6], which aims to be more realistic than WikiSQL and Spider. The data is collected from real-world scenarios and maintains their original form. The dataset contains 12,751 unique question-SQL pairs in 95 different databases over 37 different domains. With its size of 33.4 GB it is substantially larger than its predecessors. Each question of the dataset is labeled within the difficulties of simple, moderate or challenging. The complexity level of each query has been annotated by an English native speaker with degrees above bachelor's level and who has database-related knowledge. The annotators have evaluated the query based on the factors of question comprehension, schema linking, external knowledge acquisition and reasoning. Each question is also labeled with an optional evidence value, which is a string that provides a context of the query that could otherwise be hard for an LLM to understand on its own [6].

2.6.2 A Closer Look at BIRD-Bench

This section offers a more in-depth examination of the BIRD-Bench dataset as it is described by its authors, by exploring its construction methodology, the process employed for question annotation, and the implementation of external knowledge and difficulty grading to enhance its relevance and challenge for real-world applications [6].

Dataset Overview

The BIg Bench for LaRge-scale Database Grounded Text-to-SQL Evaluation (BIRD-Bench) dataset is an extensive and diverse dataset for assessing text-to-SQL capabilities. It contains 12,751 distinct question-SQL pairs, spanning across 95 different databases and accumulating a total size of 33.4 GB. BIRD-Bench covers 37 professional fields, offering a wide variety of domains such as blockchain, hockey, healthcare, and education, thereby providing a comprehensive platform for testing and improving text-to-SQL technologies.

The dataset is divided into a training set with 11,218 data points and a development set with 1,533 data points. The development set encompasses 10 domains, while the training set extends over 27 domains.

Dataset Construction

BIRD-Bench's construction involved a multi-source approach to obtain databases with complex schemas and substantial real-world value distributions. Recognizing the limitations of synthetic databases due to privacy concerns, the creators sourced 32% of the BIRD-Bench's databases from Kaggle, known for its challenging and noisy data. Another 48% were obtained from the CTU Prague Relational Learning Repository, providing multi-relational data for machine learning research. The remaining 20% of databases were constructed similarly to DuSQL, involving the synthesis and standardization of schemas and generation of database constraints. This diverse sourcing led to a collection of 95 databases across 37 domains, ensuring a wide representation of real-world scenarios.

Question Annotation

For question annotation, native English speakers with database knowledge and above bachelor's level degrees were employed. They were provided with ER diagrams and database description files for understanding the databases and were then asked to write questions in natural language regarding the contents of the databases provided. These questions were evaluated by text-to-SQL experts, and only the questions of those receiving at least two votes for validity were included. This rigorous process ensured that the questions were valid and challenging, contributing to 12,751 pairs of text-to-SQL data.

SQL Annotation

The SQL annotation process involved skilled data engineers and database students who were rigorously tested before they were employed. Each annotator was initially asked to write the corresponding queries to 10 predefined questions, and only those who scored a 9 out of 10 for those examples was qualified to continue writing SQL queries for the dataset. A double-blind technique was then used to check the validity of the written SQL queries. This approach included two independent annotators generating SQLs for the same question, after which the generated SQLs were executed against the corresponding database. Those retrieving the same information from the database were included in the dataset. Those who's information diverged where checked and modified by experts until they agreed on the SQLs format. The idea of this approach was to significantly reduce annotation errors.

External Knowledge

The inclusion of external knowledge in BIRD-Bench, as reflected in the 'evidence' entries of the dataset, is a necessity for accurately mapping natural language instructions to database elements according to the authors. This can for example be contextual insights about what a column with an ambiguous name in a database table refers to, or how a financial figure is calculated when mentioned in a question. The creators of BIRD-Bench have systematically categorized this knowledge into four distinct types:

- **Numeric Reasoning Knowledge:** Involves basic math operations and their compositional use, like percentages and ratios.
- **Domain Knowledge:** Pertains to specific knowledge in different domains necessary for generating effective SQL queries.
- **Synonym Knowledge:** Ensures the correct conversion of differently phrased questions to SQL.
- **Value Illustration:** Provides detailed descriptions of database values, their types, categories, and mappings.

Difficulty

The BIRD-Bench dataset introduces a nuanced system for classifying the difficulty of text-to-SQL examples, categorizing them into three levels: simple, moderate, and challenging, making up 30%, 60%, and 10% of the examples respectively. This approach expands on previous methods, such as those used in SPIDER, by incorporating a broader range of factors beyond just SQL complexity.

In this system, SQL annotators evaluate each example across four key dimensions, each scored on a scale from 1 to 3. The first dimension, Question Understanding, gauges how easily the intent of the question can be grasped, with 1 indicating straightforward questions and 3 denoting highly ambiguous ones. The second dimension, Knowledge Reasoning, assesses the extent of external knowledge required to translate the question into SQL. Here, a score of 1 suggests no additional knowledge is needed, whereas a score of 3 indicates a significant need for complex external knowledge.

The third dimension, Data Complexity, examines the complexities of the database schema and the data size. A simple schema and data set would score 1, while a highly complex schema, difficult to understand even with descriptions, would score 3. The final dimension, SQL Complexity, rates the syntactic complexity of the SQL query itself, with 1 being a simple query and 3 being a highly complex one with numerous functions.

Each of these dimensions contributes equally to the overall difficulty assessment of an example. By aggregating the scores across these dimensions, examples are sorted into the respective difficulty categories. This comprehensive method of evaluating difficulty offers a more detailed and multi-faceted analysis for text-to-SQL tasks, allowing for a deeper understanding of the challenges involved in different types of queries.

Example Data Point

The example entry in Listing 4 demonstrates the dataset's structure. For instance, the question is classified as 'simple'. The 'evidence' field provides external knowledge or specific references to the relevant data points (e.g., "A12 refers to unemployment rate 1995"), while the SQL field contains the query needed to retrieve the answer. This structure demonstrates BIRD-Bench's emphasis on real-world applicability and the need for understanding both the database schema and its values.

```
{
    "question_id": 91,
    "db_id": "financial",
    "question": "The average unemployment ratio of 1995 and
    1996, which one has higher percentage?",
    "evidence": "A12 refers to unemploymant rate 1995; A13
    refers to unemploymant rate 1996",
    "SQL": "SELECT DISTINCT IIF(AVG(A13) > AVG(A12), '1996',
    '1995') FROM district",
    "difficulty": "simple"
},
```

Listing 4: Example of an entry in the BIRD-Bench dataset [9].

2.6.3 Text-to-SQL Methods

This chapter discusses advancements in Text-to-SQL methods, highlighting the development of models like DIN-SQL and DAIL-SQL that enhance the ability of LLMs to generate complex SQL queries.

In the article *DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction* the authors presents an approach to text-to-SQL generation where a complex task is divided into smaller sub-tasks and thereby closing a performance gap between fine-tuned and prompting methods when generating complex SQL-queries with LLMs [56]. Evaluating the prompting technique on the Spider-dataset [61], the authors achieves an execution accuracy of 85.3 % with GPT-4 and 78.2 % using the CodeX Davinci model. In the article it is presented that the major factors where text-to-SQL queries fail with the CodeX Davinci model are Schema-Linking (37 % of failures), JOIN-queries (21 % of failures) and wrong columns (15 % of failures) which often occured when a GROUP BY clause was required. The DIN-SQL model is comprised of four distinct modules: (i) Schema Linking Module, tasked with managing database references and natural language queries; (ii) Classification & Decomposition Module, which categorizes each query into one of three complexity levels—easy, non-nested complex, or nested complex; (iii) SQL-generation Module, further subdivided into three specialized sub-modules each focusing on the generation of queries corresponding to the mentioned complexity levels; (iv) the Self-Correction Module, which instructs the LLM to identify and correct errors within the generated SQL queries [56].

Another, text-to-SQL model is presented in the article *Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation* [62] where zero-shot and few-shot prompting methods, but also fine-tuned models in a text-to-SQL context are evaluated. The article presents the DAIL-SQL model which scored the highest execution accuracy upon publication [62]. The article evaluates five different examples of zero-shot prompting in a text-to-SQL setting; Basic Prompt, Text Representation Prompt, OpenAI Demonstration Prompt, Code Representation Prompt, and Alpaca SFT Prompt. The Code Representation Prompt scored an execution accuracy of 75.6 % together with LLM CODE-DAVINCI-002. The code representation prompt was created through a "Create Table prompt" which can be seen in Figure 5.

The few-shot prompting methods that are evaluated are the following Full-Information Organization, SQL-Only Organization and DAIL-SQL. The DAIL-SQL model employs masking on both the target and example questions to diminish noise and enhance understanding of their general structure. After masking, the questions are converted into embeddings. The Euclidean distance is then used to measure the similarity between the target question and each of the example questions. In addition, DAIL-SQL generates SQL queries through a preliminary model using the target question. The generated query, called s', is an approximation of the desired query s*. Thereafter the Euclidean distance between s' and the true label query s_i is calculated. Subsequently, the example questions with the smallest Euclidean distance are chosen, given that the query distance is below a predefined threshold τ. In doing so, the model effectively organizes question-to-SQL mappings, determining which examples should be provided to the LLM for the creation of the few-shot prompt [62].

The DAIL-SQL model achieved an execution accuracy of 86.2% on the Spider dataset. The model's accuracy was further enhanced to 86.6 % by incorporating self-consistency; however, this improvement was time-consuming, as noted by the authors. The article also showcases results from open-source models that were fine-tuned on the Spider dataset. Notably, when a fine-tuned model was combined with zero-shot prompting, the highest accuracy of 69.1% was attained using the LLaMA-33B model in conjunction with the Code Representation Prompt [62].

```
CREATE TABLE continents (
    ContID int primary key,
    Continent text,
    foreign key(ContID) references countries(Continent)
);

CREATE TABLE countries(
    CountryId int primary key,
    CountryName text,
    Continent int,
    foreign key(Continent) references continents(ContID)
);

/* Answer the following: How many continents are there? */
SELECT
```

Listing 5: Code Representation Prompt [62].

3 Method

This chapter outlines the methods applied throughout the research process. Firstly, Section *3.1 Overview* provides an introduction to the research design and the general approach adopted during the investigation. Next, Section *3.2 Data Collection & Datasets* offers comprehensive insights into the dataset used, as well as the evaluation metrics and validation processes implemented.

Subsequently, in Section *3.3 Text-to-SQL Models*, the practical aspects of the research are explained. This includes a discussion on the choice of LLMs and the specific text-to-SQL models employed in our experiments. Section *3.4 Text-to-SQL Experiments* then elaborates on the experimental methodology. The progression of our methodology is further explained in Section *3.5 Result Analysis*, where we describe the process used to analyze the experimental results. Thereafter, in Section *3.6 Question Classification* the next phase of experiments is explained by examining LLMs capabilities as classifiers. Finally, Section *3.7 Correcting SQL Queries* addresses the methods employed for correcting data points within our research framework.

The prompting templates and SQL-analysis spreadsheets used and created during the method are attached in the project's GitHub repository[1] .

3.1 Overview

An illustration over the research process of this thesis is presented in Figure 3.1. The methodology was initiated with a thorough review of literature in prompt engineering and current text-to-SQL implementations, followed by a search to identify a suitable dataset for the experimental phase. The next stage of the process consisted of developing an evaluation pipeline for the experiments, along with the construction of the text-to-SQL models zero-shot, few-shot, and DIN-SQL. Subsequently, the first series of experiments were conducted, followed by a thorough analysis of the results obtained. Insights from this analysis led to a focus on investigating model performance on cleaned and noise-free versions of the dataset. This approach was then evaluated through a second series of experiments, accompanied by further analysis of the experiment results. The second analysis generated the proposition to use LLMs to improve data quality in text-to-SQL tasks, which led to classification and correction experiments. Finally, when the four different experiment iterations had been finalized, the results were summarized. The findings are presented in chapter *4 Results* and further assessed in chapter *5 Discussion*.

[1] https://github.com/niklaswretblad/Text-to-SQL-Generation

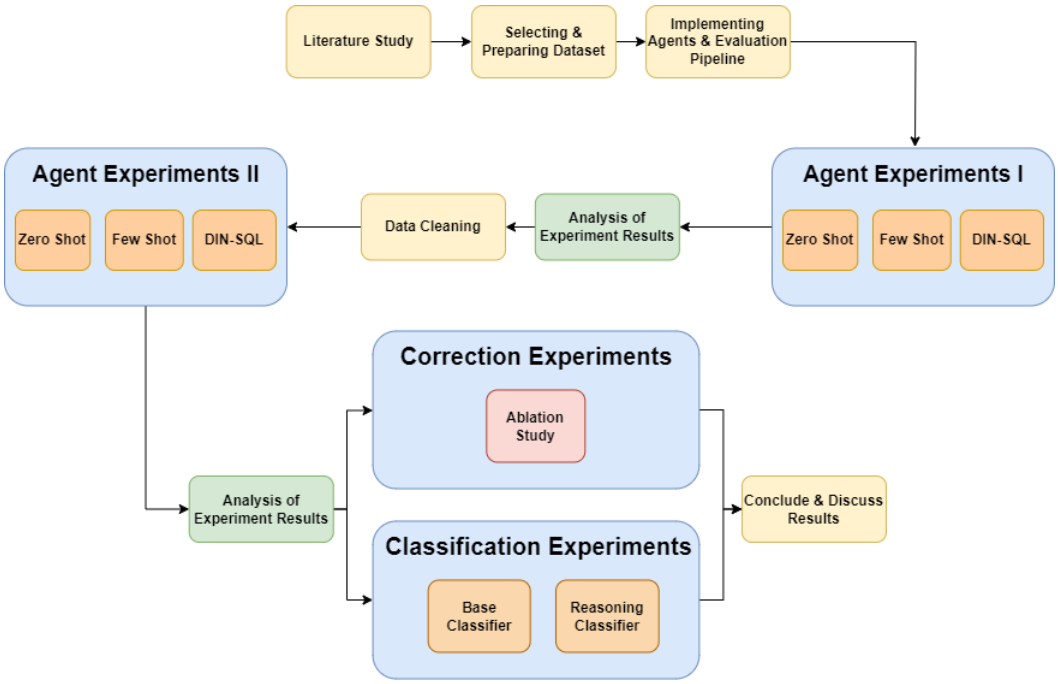

Figure 3.1: The research process of the project.

3.2 Data Collection & Datasets

A comprehensive literature study was conducted to identify the most relevant datasets for text-to-SQL research. The literature study centered around a detailed analysis of the three most popular datasets in the text-to-SQL field: WikiSQL, Spider, and BIRD-Bench, to determine their suitability for this research. The datasets are described in Section *2.6.1 Text-to-SQL Benchmarks & Datasets*, but some of their characteristics will be mentioned again in this Section in order to proberly motivate the choice of dataset and benchmark.

As described in Section *2.6.1 Text-to-SQL Benchmarks & Datasets*, WikiSQL is a prominent dataset in the text-to-SQL domain that was released in 2017, featuring over 24,241 tables and 80,654 natural language questions [70]. Each question is linked to a corresponding SQL query and a Wikipedia table. The primary focus of WikiSQL is to test the ability of models to translate natural language questions into SQL queries based on a table schema, without necessitating complex SQL operations or in-depth understanding of database content. The queries in WikiSQL are therefore fairly simple, only containing simple SELECT and WHERE operations without nested queries or JOIN operations.

Spider, in contrast, presents a more complex scenario with its collection of 200 databases and 10,181 questions. Notable for its cross-domain diversity, Spider challenges models with a variety of SQL structures and keywords across multiple tables. This dataset requires a deeper understanding of database schema and the generation of complex SQL queries. However, similar to WikiSQL, Spider does not emphasize the understanding of the actual content within the databases.

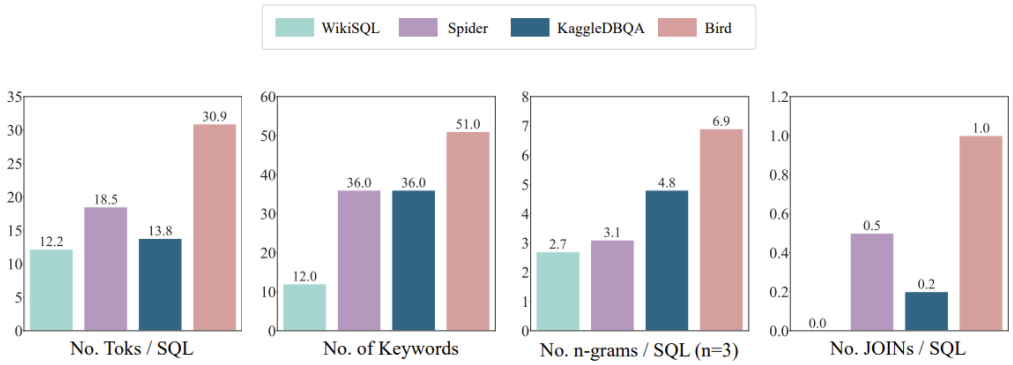

Figure 3.2: The complexity of BIRD-Bench compared to other benchmarks [6].

BIRD-Bench (BIg benchmark for laRge-scale Database) represents a significant advancement in the text-to-SQL field. It comprises 12,751 pairs of text-to-SQL data across 95 databases, with a total size of 33.4 GB and covering 37 professional domains. BIRD-Bench is designed to fill the gap between academic research and real-world applications by emphasizing the comprehension of database values. It introduces challenges such as dealing with dirty database contents, requiring external knowledge to link natural language questions with database contents, and the efficiency of SQL queries in large databases. Despite progress in text-to-SQL models like Codex and ChatGPT, BIRD-Bench underscores ongoing challenges, with even the most advanced models achieving only 40.08% execution accuracy compared to 92.96% for human performance. This dataset is vital for testing and improving the practical applicability and efficiency of text-to-SQL models in real-world scenarios.

Because of its complexity and greater alignment with real-world use cases BIRD-Bench was selected for this research. Its focus on database values and the unique challenges presented by large-scale databases make it an essential dataset for driving forward the practical application of text-to-SQL methods in industry settings.

3.2.1 The Financial Domain in BIRD-Bench

The clients of the project, Combient Mix and SEBx, were exclusively interested in use-cases of text-to-SQL in finance and banking. BIRD-Bench, as part of its development set, hosts a domain specific to finance and banking called *financial*. Rather than investigating a text-to-SQL models ability across all domains in the BIRD-Bench dataset, this research therefore centered around the data found in the financial domain of the benchmark. The domain is composed of 106 question and query pairs and have questions spanning all three types of difficulties found in the benchmark. More specifically, 63 out of the 106 questions were regarded as simple, 37 as moderate and 6 as challenging. The financial domain in BIRD-Bench is based on the BERKA-dataset [71], which is part of the larger *The CTU Prague Relational Learning Repository* [72]. The BERKA-dataset was published in 1999 and consists of financial information from a Czech Bank, containing over 5300 clients and over 1 million transactions [71].

Database Schema

Figure 3.3: Database schema of the financial domain from the BIRD-Bench benchmark [71].

Figure 3.3 displays the database schema for the financial domain. This schema contains various tables, such as those for loans, transactions, accounts, cards and clients, all reflecting the financial orientation of the database. Descriptions of what information these tables contain are presented in Table 3.1. The database consists of 55 columns distributed across eight distinct tables. While the majority of the column names are intuitively understandable, some present interpretative challenges, as evident in the schema shown in Figure 3.3. An illustrative example is the district table, which incorporates 16 unique columns. This includes a column titled *district_id* along with 15 other columns, ranging from *A2* to *A16*. The latter columns' names do not readily convey the nature of the data they hold, making them less intuitive to understand. In practice, a database schema will often be accompanied by a data dictionary or documentation that explains each table and column in detail. Such documentation would typically provide the context needed to fully understand the meaning of each element in the schema, the range of possible values for fields with unspecified types, and the

business logic underlying the relationships. Without this additional documentation, fully interpreting and effectively using the database can be challenging as illustrated by the column names in the districts table. In the theoretical framework outlined in Section *2.6.2 A Closer Look at BIRD-Bench* on BIRD-Bench, it is highlighted that the dataset includes a unique feature for each question termed *hint*. This feature is designed to offer insights or supplementary information corresponding to the specifics detailed in such database documentation.

Further, the lines in Figure 3.3 between the tables represent relationships, where the nature of the relationship is indicated by the shape of the tail end of the lines where they connect to each table. A one-to-many relationship is indicated by the line beginning with a single line and the one digit above it, and then ending in a crow's foot (three lines) at the opposite end. For example, an account can have multiple orders, transactions, dispositions, and loans associated with it, but each of those entities is only linked to one account. An account can have many loans, but one loan is exclusively only linked to one account, which makes sense. Further, clients and accounts are related through the disposition table in a many-to-many relationship. An account can have many different clients associated with it, for example, one client listed as the owner of the account and multiple other clients listed as users for the account. This could for example be practical for sharing an account in a family, where one parent could be the owner of the account and then multiple other family members listed as users. A single client can also be related to many different accounts in the other way around.

Table 3.1: Table descriptions of the tables in the database of the financial domain [9].

Table Name	**Description**
loan	Contains details of loans.
order	Holds information about monetary orders.
trans	Represents financial transactions.
account	Contains account information.
disp	Links clients to accounts (dispositions).
card	Contains details about cards issued.
client	Holds client information.
district	Contains details about districts or regions.

3.2.2 Analysis and Comparison of Domains

The BIRD-Bench authors have previously published results that measure the success of models across the entire training and development sets, however, these published results do not provide a granular understanding of how the financial domain compares to the other domains within the benchmark [9]. This lack of domain-specific insight posed a challenge for this research in comprehensively comparing the performance of models within the financial domain with previously gathered results on the development and training sets. It also made it difficult to understand the specific nature of the financial domain because of the lack of available comparisons. The finance domain only amounts to about 0.8% of the total data points in the BIRD-Bench dataset and about 7.5% of the total data points in the development set which further motivates this problem. Therefore, an in-depth analysis and comparison of the different domains in the development set of the benchmark were performed. This analysis involved three key parts to provide a more detailed perspective:

1. Receiving old log files from the author of the DIN-SQL model, where the model inputs and outputs had been logged on the entire development set. Analyzing these logs enabled the measurement and comparison of the accuracy of the DIN-SQL model across each domain in the development set, providing insights into the model's domain-specific strengths and weaknesses.

2. Investigating the databases in the benchmark by collecting and measuring the number of tables and columns in each database. This process also involved a comparative analysis of the database sizes, in terms of both tables and columns, against the accuracy achieved by the DIN-SQL model in each domain from the previous step. The objective was to discern any potential correlation between the size of the databases and the models' accuracy.

3. The analysis included a detailed enumeration of the questions categorized by their difficulty level across each domain. Additionally, the proportion of each difficulty level was evaluated in relation to the accuracy of the DIN-SQL model, as established in the previous step. This methodical approach was aimed at assessing the impact of a domain's overall question difficulty on the performance of text-to-SQL models. Such insights are crucial in understanding the varying challenges posed by different domains and how they influence model accuracy.

3.3 Text-to-SQL Models

This section will present the two text-to-SQL models, *Baseline: Zero-Shot Model* and *DIN-SQL Model*, used in the experiments and what LLM and hyperparameters that were applied to generate the predicted SQL queries. Both models are composed out of different prompts and prompting techniques built on top of openAIs LLM APIs.

3.3.1 Choice of Large Language Model

The core of the research conducted centers around the use of LLMs, and the same is true for the models created. All models described in this section at the core utilizes an LLM in order to convert a question in natural language into a SQL query. The LLMs used in the experiments are GPT-3.5-Turbo and GPT-4, chosen because they are the highest-performing publicly available models in the text-to-SQL benchmarks of BIRD-Bench and Spider [9, 10].

GPT-3.5-Turbo was employed during preliminary testing and in most of the experiments, due to its cost-effectiveness and high performance. GPT-4 was used exclusively for comparing its performance with that of GPT-3.5-Turbo.

The default GPT-3.5-Turbo model was utilized in all experiment runs, except those employing the DIN-SQL prompting technique. For these, the GPT-3.5-Turbo-16k variant was used to accommodate the longer token length of DIN-SQL's prompting template.

In all of our experiments, we consistently set the temperature hyperparameter to 0. This decision was based on observations that increasing the temperature often resulted in a decrease in the accuracy of SQL conversion. Specifically, higher temperature settings were found to compromise the structural integrity of the SQL output, frequently resulting in outputs that did not adhere to correct SQL structure. Moreover, at elevated temperatures, the model tended to include extraneous reasoning and text, thereby disregarding previous instructions and further diminishing the quality and precision of the SQL conversion.

3.3.2 Baseline: Zero-Shot Model

The zero-shot model acts as the baseline of the study since it presents the most basic structure and idea of a text-to-SQL model using LLMs. The idea is to first compose a prompt that integrates the question, the associated database schema, an instruction directing the LLM to generate valid SQL, and the hints or external knowledge sourced from the BIRD-Bench dataset. Subsequently the model is prompted to generate a SQL query. The zero-shot prompt template can be seen in Listing 6. It begins with an instruction that the database schema will be provided in the form of CREATE_TABLE statements, followed by a variable that represents the database schema of the database the question we are trying to convert refers to.

Thereafter, an instruction telling the LLM to use valid SQL syntax to answer the provided question is written. Subsequently, the LLM is informed about the intention of the hint, along with variables representing the actual question and hint. Lastly, to avoid extensive and unnecessary output, the LLM is instructed to generate nothing other than the actual SQL query.

```
Database schema in the form of CREATE_TABLE statements:

{database_schema}

Using valid SQL, answer the following question based on the
tables provided above.

Hint helps you to write the correct sqlite SQL query.
Question: {question}
Hint: {evidence}
DO NOT return anything else except the SQL query.
```

Listing 6: Zero-Shot Prompting Template.

Database Schema

In order for an LLM to generate a correct SQL query when provided with a question, it needs to know the specific details of the database it refers to. For instance, it needs to know the names of the tables inside the database, the names of the columns of the tables, the different relationships between the different entities and so on. Hence, the schema of the database is provided to the LLM within each prompt as can be seen in Listing 6. Further, a database schema can be expressed in a multitude of different formats, but previous studies has shown that expressing the schema in the form of SQL *CREATE_TABLE* statements improved accuracy compared to other formats [73]. This format is demonstrated in Listing 7. The zero-shot model and all other models implemented in this research therefore expressed all database schema in the form of CREATE_TABLE statements.

3.3.3 DIN-SQL Model

DIN-SQL refers to *Decomposed In-Context Learning of Text-to-SQL with Self-Correction*, which is, at the time of writing, the top-performing text-to-SQL prompt engineering method [56]. It addresses challenges in converting natural language queries to SQL by decomposing the process into smaller, manageable sub-problems. The aim of the model structure is that it should be especially effective for complex queries involving multiple joins or nested structures. The methodology consists of four main modules:

1. **Schema Linking Module:** It identifies references to database schema and condition values within natural language queries. This step is crucial for generalizing across domains and synthesizing complex queries.

2. **Classification & Decomposition Module:** This module classifies queries into one of three categories: easy, non-nested complex, and nested complex. It also detects the tables to be joined and any sub-queries for nested queries. The categorization is essential for the subsequent SQL generation process.

3. **SQL Generation Module:** Depending on the query class, different processes are employed. For *easy* class queries, simple few-shot prompting is used. For *non-nested complex* class queries, an intermediate representation, like NatSQL, bridges the gap between

```
CREATE TABLE account
(
    account_id INTEGER default 0 not null primary key,
    district_id INTEGER default 0 not null,
    frequency TEXT not nuLl,
    date DATE not null,
    foreign key (district_id) references district (district_id)

);

CREATE TABLE card
(
    card_id INTEGER default 0 not null primary key,
    disp_id INTEGER not null,
    type TEXT not null,
    issued DATE not null,
    foreign key (disp_id) references disp (disp_id)
);
```

Listing 7: CREATE_TABLE statements corresponding to the account and card tables in the BIRD-Bench dataset [9].

the query and SQL statement. The *nested complex* class involves multiple intermediate steps, including solving sub-queries before generating the final SQL.

4. **Self-correction Module:** This module corrects minor mistakes in the generated SQL queries, such as missing or redundant keywords. Two types of prompts, *generic* and *gentle*, are used depending on the model (CodeX or GPT-4) to direct the correction process.

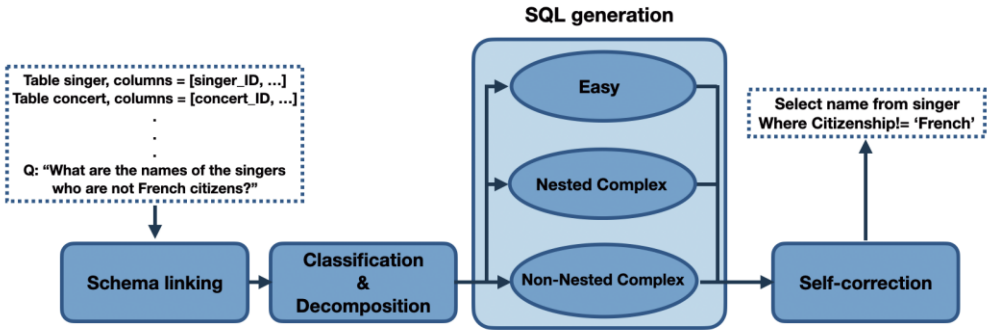

Figure 3.4: The architecture of the DIN-SQL model [56].

These modules are composed of distinct prompting templates, each specialized for each specific task. When a question requires conversion into SQL, it is processed sequentially by a pipeline consisting of these four unique modules. Each template, or submodule, is dedicated to addressing a particular aspect of the larger challenge of transforming a question into SQL format. This methodology uses a small number of carefully selected few-shot learning examples within each prompting template. These examples are drawn from the training datasets of relevant benchmarks, providing targeted guidance to help the conversion process.

3.4 Text-to-SQL Experiments

This section outlines the structure, methodology, and evaluation of the text-to-SQL experiments conducted as part of this study. The core objective of these experiments was the development and assessment of a pipeline for converting natural language questions into SQL queries, primarily leveraging various text-to-SQL techniques based on LLMs. The experimental process commenced with an input question framed in natural language. This question was then run through a systematic pipeline, involving multiple prompts and distinct calls to an LLM, specifically tailored to optimize the conversion into a SQL query. Upon successful conversion, the resulting SQL query was executed against the corresponding database. Due to the non-uniqueness of SQL queries, where different queries can yield identical results, the success of a generated query was determined by comparing the data obtained from the database from the true query with that retrieved by the generated query. This comparison was done automatically through code implementation. A match between the data retrieved was deemed a *success*, and accuracy was determined by the ratio of successful matches to the total number of both successes and failures. The experiments utilized the web-based software *Weights and Biases* for logging results and metrics. This platform is designed for monitoring, visualizing, and optimizing machine learning model hyperparameters [74].

Following the implementation of the evaluation pipeline, which integrated the text-to-SQL models with the LLM API, database, and evaluation functionality, the initial iteration of experiments was conducted. Each experiment recorded a set of data for each data point, encapsulated as:

$$\textit{question, true SQL query, predicted SQL query, success, difficulty} \tag{3.1}$$

In addition to the data in table (3.1), further metrics, including model accuracy and hyperparameters, were documented either iteratively throughout each experiment or as cumulative values at the conclusion of the experiment.

3.5 Result Analysis

After the first iteration of experiments, the results of the data Table (3.1), presented in Section *3.4 Text-to-SQL Experiments*, were analyzed for each model. Every data point was read through to find patterns between the question, true SQL queries, model, and predicted query. The concluded patterns and results of the first iteration can be seen in Section *4.2 First Round of Experiments*. A significant observation was the compromised quality of the data. Parts of the dataset had quality issues due to ambiguous questions or erroneous true SQL queries.

Consequently, the analysis led to a new phase of the methodology where the financial domain of the BIRD-Bench dataset was corrected. The correction process included a detailed analysis of each data point's question and true query, specifically investigating the grammatical and semantic aspects of the questions, as well as assessing the specificity or ambiguity of these questions in relation to the corresponding data retrieved by the true queries. However, there is a significant probability that this process was exposed to cognitive bias, which will be discussed in Section *5.2 Method*. The dataset before and after the first iteration of corrections are attached in the GitHub repository[2].

Subsequent to the data correction process, a further iteration was conducted to evaluate the impact of this enhanced data quality on the SQL query generation by the LLM. The outcomes of this analysis, presenting the relationship between data quality and generated query accuracy, are detailed in Section *4.3 Analysis of Experiment Results*.

3.6 Question Classification

Upon evaluating the models with both the initial and modified datasets, it became evident that dataset variations significantly influenced performance. Consequently, this created the incitament for a more thorough data analysis, leading to an additional methodological phase. This phase involved categorizing through manual question annotation and systematically classifying them using the language model through a sequence of experimental runs.

3.6.1 Annotating Question Quality

During the process of analyzing the results, a pattern of data quality issues was recognized. In a significant portion of the dataset, the questions and SQL queries were clear and unambiguous. However, many data points contained unclear and ambiguous questions, likely impacting the text-to-SQL model's performance. Some questions had subtle ambiguities that might affect the model, though it's uncertain to what extent. Additionally, certain questions, despite being understandable with knowledge of the database structure, were unanswerable due to the database's limitations.

To organize these patterns, four distinct quality categories were established to classify the questions presented in Table 3.2.

[2]https://github.com/niklaswretblad/Text-to-SQL-Generation

Table 3.2: Annotated question quality classes.

Category	Description
0	Correct question. May still contain minor errors in language or minor ambiguities that do not affect the interpretation and generation of the SQL query.
1	The question is unclear, ambiguous, unspecific, or contains grammatical errors that surely are going to affect the interpretation and generation of the SQL query
2	Indecisive, badly formulated questions that could or could not affect the text-to-SQL conversion
3	The question is wrongly formulated when considering the structure of the database schema. The information that the question is asking for is not possible to accurately retrieve from the database.

After establishing the categories, the financial domain of the BIRD-Bench dataset underwent an annotation process. Two graduate students within AI and Machine Learning initially annotated the dataset independently, based on the concluded categories. Their annotations were then cross-examined to agree upon a definitive set of labels, which were subsequently applied to the dataset.

3.6.2 Classification Models

Following the annotation of question quality, a series of classification experiments were conducted. In these experiments the LLM was prompted to sort the questions into the predefined categories. The experiments examined a single prompt template classifier, but also a reasoning model where chaining was implemented. This section will elaborate on these experiments.

Base Classifier

The classification experiments were initiated with a less complex approach where the LLM was prompted to categorize the questions into two categories. The first class (0) was a correct question representing the first category presented in Section *3.6.1 Annotating Question Quality*. The second class (1) was an incorrect or ambiguous question representing categories 1-3 of the concluded categories. To give the LLM more detailed instructions and context, few-shot learning was employed, wherein the language model was primed with fabricated question examples—crafted by the annotators—for each category to establish a foundational context. The examples were created to mirror the issues identified in the patterns, based on research presented in Section *2.2.4 Few-shot Learning* that emerged from the analysis of the results. Additionally, experiments were conducted using examples that embodied both class 0, which denotes correct questions, and class 1, indicating incorrect questions. These experiments included examples representing each class individually, as well as examples that encompassed both class 0 and class 1. Additionally, the database schema, BIRD-Bench hint, and lastly the question of each data point was provided. The results of these experiments are presented in Section *4.5 Question Classification* and the prompts used are provided on the project's GitHub repository[3].

When the LLM's ability to classify two categories had been examined, the ability to classify the four individual categories was evaluated, see Table 4.3. In these experiments the same prompt and information was provided, only that the categories were presented with their number. These results are also provided in Section *4.5 Question Classification*.

[3]https://github.com/niklaswretblad/Text-to-SQL-Generation

Considering the results of these experiments featured true positives, false positives, as well as true negatives, the applied evaluation metrics encompassed accuracy, recall, F1-score, and precision.

Reasoning Classifier

The initial classifier solely produced a numerical grade as its output, which could be viewed as limiting since research has indicated that instructing an LLM to explicitly articulate its reasoning or thoughts in responses can enhance its logical capabilities [1]. The second classifier was designed to leverage this approach. It involved a two-step process: first, the LLM was prompted to think and reason about the potential grade for a question. Second, the output from the first step was used as input to the second step, where the LLM was instructed to assign a grade to the question based on the reasoning it had formulated in the prior step. In both steps, the LLM was also provided with the database schema corresponding to the given question in addition to the classification scheme found in Table 3.2. The full prompt templates used can be found on the project's GitHub repository[3].

3.7 Correcting SQL Queries

During the result analysis described in Section *3.5 Result Analysis*, a subset of the questions and queries from the BIRD Bench dataset was identified as being incorrect or poorly formulated. Therefore, a small study was performed to investigate the capability of an LLM in correcting faulty SQL queries. If this would be possible, one could automatically remove the noise in the form of incorrect SQL queries from the data by using an LLM. The study therefore aimed to assess the effectiveness of an LLM in enhancing the accuracy of text-to-SQL datasets by measuring its success in correcting incorrect SQL queries.

The data consited of incorrect SQL queries, along with their corresponding questions and accurate counterparts. Analysis of the results revealed 16 true SQL queries that were formulated incorrectly. Notably, seven of these queries shared a common error type. Therefore 10 incorrect SQL queries were used which contained 10 unique errors in total.

After the data had been collected, a prompt was constructed with the purpose of instructing the LLM to pinpoint and correct SQL queries that were either syntactically incorrect or logically flawed in their interpretation of database relations. To ease the correction process, the prompt was also provided with three additional pieces of information:

- Database Schema: The schema for the relevant database was given in detail in the form of *CREATE_TABLE* statements. The schema is essential for understanding the structural relationships within the database, and so is crucial for the LLM to accurately correct and formulate the queries.
- Natural Language Question: Together with each SQL query the corresponding natural language question was given. The question sets the context for what information the query is intended to retrieve, guiding the correction process.
- Hint: The evidence provided for each of the data points from the BIRD dataset used for this study was also provided, aiding the LLM in correctly interpreting the database schema and relations.

The prompt ended with an instruction telling the LLM to correct the given SQL query. After some initial experiments, it was shown that the LLM often wanted to correct given dates, values, names and numbers even though they were correctly stated in the query. An instruction telling the LLM to assume that all dates, values, names and numbers in the questions and SQL queries are correct were therefore added to the prompt as well.

To evaluate the effectiveness of the LLM in correcting SQL queries, the data fetched by both the generated and the true SQL query were manually compared. If the two queries returned exactly the same data, the generated query was considered as good.

4 Results

This section presents findings from the experimental studies. It begins with exploring the performance of the implemented text-to-SQL models in their first evaluation with the original dataset. This is followed by a quantitative analysis of the revised dataset, featuring selected data point examples from the correction process. Thereafter, results from the second iteration of experiments are presented, where the text-to-SQL models have been assessed on the corrected dataset.

The next part of the result chapter includes performance metrics, presented through plots and heatmaps, from the classification experiments, emphasizing the accuracy of both the base and reasoning classifiers. The section culminates in the presentation of findings from the ablation study. For additional resources including plots, the model's prompting templates, and the project code and plots are available at the GitHub repository[1].

4.1 Comparison Between Domains in BIRD-Bench

This section presents the results of the analysis and comparison of the different domains in the development set of BIRD-Bench as described in Section *3.2.2 Analysis and Comparison of Domains*. The chart in Figure 4.1 serves as a visual representation of a comparative analysis between the domains. It displays the number of tables and columns across the various domains in the development set, with the finance domain highlighted. Additionally, the plot captures the accuracy rates achieved by the DIN-SQL prompt engineering model across the different parts of the development set as denoted by the green bars. As can be seen in the Figure, the financial domain is not an outlier when it comes to any of the three values depicted, but holds close to the averages. The mean accuracy of the DIN-SQL model across all the domains is 37%, whereas the accuracy on the financial domain is 32%. The average amount of tables across the development set is 7.3 and the average amount of columns is 78.3. The financial domain in turn consists of 55 columns distributed across eight distinct tables. Further, no clear relationship between the number of columns and tables and the accuracy of the DIN-SQL model can be observed.

[1] https://github.com/niklaswretblad/Text-to-SQL-Generation

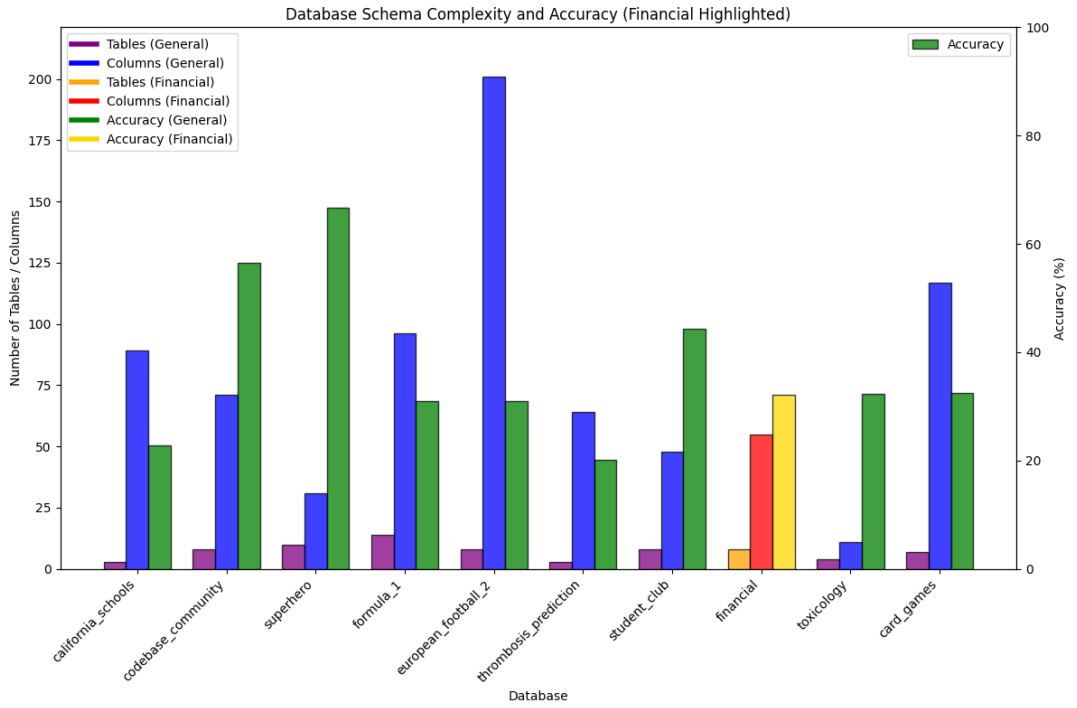

Figure 4.1: Table counts, column counts and DIN-SQL model accuracy across the domains in the BIRD-Bench development set with the financial domain highlighted in red and orange.

The bar chart presented in Figure 4.2 displays the distribution of question difficulties across the different domains of the development set in BIRD-Bench. Each domain is represented by a bar segmented into three color-coded parts, symbolizing the different levels of question difficulty: green for simple, yellow for moderate, and red for challenging. The length of each colored segment within a bar corresponds to the proportion of questions at that difficulty level, allowing for an at-a-glance assessment of the complexity mix within each domain.

Observing the bar chart, we notice significant variations in the difficulty distribution across different domains. For instance, the domain *codebase_community* has a higher proportion of simpler questions. Conversely, the domain *thrombosis_prediction* display a more balanced mix, with a noticeably larger representation of challenging and moderate questions compared to the other domains. A common trend across most domains is the smaller proportion of challenging questions, although there are notable exceptions where these more difficult questions form a significant part of the domain.

The model accuracy is overlayed on the question difficulties as blue bars. It compares the complexity of questions in a domain and the corresponding accuracy achieved.

The distribution of question difficulties and accuracies across domains follows the intuitive trend that a larger proportion of simpler questions leads to higher accuracy, but there are some outliers to this trend. For example, although *codebase_community* has a larger proportion of simple questions than *superhero*, but DIN-SQL performed better on the *superhero* domain suggesting that an abundance of simpler questions doesn't automatically lead to better performance. Another example is the low performance on the *formula_1* domain despite the large proportion of simple questions. The same goes for the domains *student_club* and *card_games*, and especially notable is the *california_schools* domain with its low performance.

To summarize, the findings reveal a general correlation between the complexity of questions in a domain and the model's performance, but this is not always evident challenging the assumption that simpler questions always lead to higher accuracy or that a predominance of difficult questions results in lower accuracy.

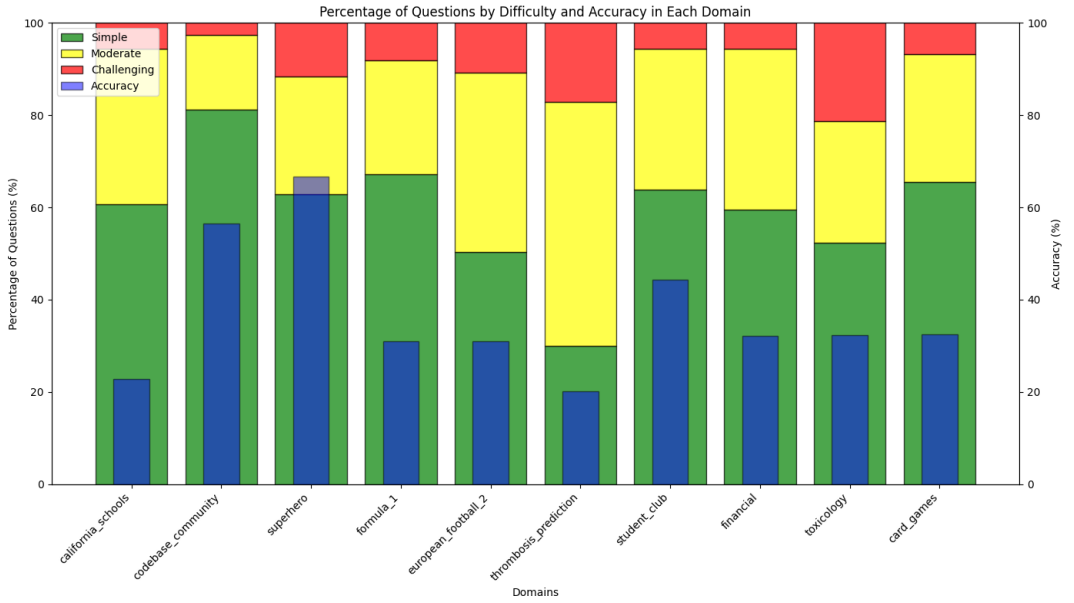

Figure 4.2: Percentage of questions by difficulty and DIN-SQL model accuracy across the domains in the BIRD-Bench development set.

4.2 First Round of Experiments

This section presents the results from the first iteration of experiments. The graph attached in Figure 4.1 presents the execution accuracy results from the evaluation of the text-to-SQL models, presented in *3.3 Text-to-SQL Models*, on the financial domain of the BIRD-Bench dataset. The Figure displays the results of the three models: *Zero-Shot with GPT-3.5*, *DIN-SQL with GPT-3.5* and *Zero-Shot with GPT-4*. The chart indicates a similar level of performance across all models. However, it's important to highlight the performance gap observed between the zero-shot and the DIN-SQL model. The zero-shot model using GPT-3.5 achieved an execution accuracy of 36.19%, while the DIN-SQL model showed an almost five percent larger accuracy of 40.57%. Additionally the zero-shot model based on GPT-4 reached a similar accuracy of 38.09%.

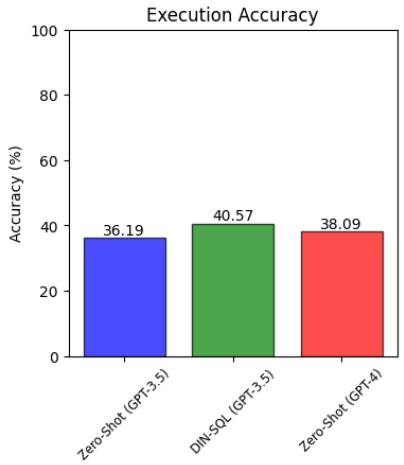

Figure 4.3: Execution accuracy of the implemented models on the financial domain of the BIRD-Bench dataset.

4.3 Analysis of Experiment Results

This section details the corrections applied to the BIRD-Bench dataset's financial domain, following the analysis introduced in *3.5 Result Analysis*. Table 4.1 quantifies the enhancements, including the number of revised unique data points, questions, and true standard queries, improving the dataset's accuracy and reliability for future text-to-SQL model training and evaluation. The errors corrected are divided into six categories. *Semantic Errors* - Questions are phrased in ways that mislead their interpretation. *Vauge/Ambiguous questions* - Questions are posed in ways that complicate making an unambiguous interpretation. *Incorrect SQL (True)* - The true SQL queries fail to retrieve the intended data. *Question and True Query Diverging* - The phrasing of questions and their corresponding true queries do not match to retrieve the same data. *Synonyms* - Word choices in questions may cause confusion or be misconstrued as SQL functions. *String Capitalization*, Mismatches in the capitalization of questions and database entries affect query execution due to SQL's case sensitivity.

Table 4.1: Statistics over erroneous data points and type of errors found during the result analysis.

Statistic	**Value**
Unique data points corrected	52/106
Questions corrected	44
True queries corrected	26

Type of Error	**# of Errors Found**
Semantic Errors (Poor English)	23
Vague/Ambiguous Questions	17
Incorrect SQL (True)	19
Question and True Query Diverging	5
Synonyms	2
String Capitalization	7

4.3.1 Result Analysis Examples

This section will provide examples of data points that were of poor quality and in different ways affected the generation of the SQL query. First, an example of a question that is ambiguous and unspecific will be presented, along with an explanation of how these formulations affect prediction accuracy. Subsequently, an example of a poorly formulated true query and its suggested correction will be displayed. Lastly, an example is provided of a question that does not comply with the structure of the database.

Example 1: Ambiguity in Natural Language Interpretation

In this example, we explore how ambiguities in natural language can significantly impact the interpretation of a language model, leading to divergences between the predicted SQL query and the intended true query. The challenge lies in the natural language question's ambiguity, specifically in the phrase *"List all the withdrawals..."* This ambiguity revolves around determining which columns to return when executing the SQL query.

The phrase is interpreted differently by the language model and the true query. The true query interprets the request as needing to return only the IDs of all transactions (i.e., a specific column from the transactions table). In contrast, the text-to-SQL model interprets the same instruction as a command to return all columns, using the star command **('*')**, for all relevant transactions. This discrepancy highlights a fundamental challenge in text-to-SQL conversion: natural language often lacks the specificity required for unambiguous SQL query formulation. It raises an essential question: *Which columns should be returned when the command is to list all transactions?*

Question:

```
List all the withdrawals in cash transactions that
the client with the id 3356 makes.
```

Listing 8: Question from the entry with id 159 from the financial domain of BIRD-Bench.

True Query:

```
SELECT T4.trans_id
FROM client AS T1
INNER JOIN disp AS T2 ON T1.client_id = T2.client_id
INNER JOIN account AS T3 ON T2.account_id = T3.account_id
INNER JOIN trans AS T4 ON T3.account_id = T4.account_id
WHERE T1.client_id = 3356
AND T4.operation = 'VYBER'
```

Listing 9: True query from the entry with id 159 from the financial domain of BIRD-Bench with the interpreted column to be returned highlighted.

Predicted Query:

```
SELECT *
FROM trans
INNER JOIN disp ON trans.account_id = disp.account_id
WHERE disp.client_id = 3356
AND trans.operation = 'VYBER'
```

Listing 10: Predicted query with id 159 from the financial domain of BIRD-Bench with the interpreted column to be returned highlighted.

Example 2: Data Quality Issue

The second example provides a further illustration of data quality issues within the financial domain of the BIRD-Bench dataset, viewed in Listing 11 to 13. It details a question alongside its initially formulated incorrect true query and the revised true query, which has been corrected following the data result analysis process outlined in section *3.5 Result Analysis*. To understand the SQL code presented in the listings correctly, it is recommended to review the database schema presented in Figure 3.3. The true query provided in the BIRD-Bench dataset uses an *INNER JOIN* SQL command over the *district_id* column in the *account* and *client* tables. This is however not the correct column value to join over since there is no unique database relation between client and account through *distict_id*, one client could have multiple accounts and a certain account could have multiple clients as users and each client and account have their own district_id. Therefore, this would possibly result in duplicate values being incorrectly selected. To modify this into a correct query, in Listing 12, an *INNER JOIN* is done through the *disp* table instead. Through which each account and its corresponding client has a unique relation.

Question:

```
What is the average loan amount by male borrowers?
```

Listing 11: Question from the entry with id 132 from the financial domain of BIRD-Bench.

True Query:

```
SELECT
  AVG(T3.amount)
FROM
  client AS T1
  INNER JOIN account AS T2 ON T1.district_id = T2.district_id
  INNER JOIN loan AS T3 ON T2.account_id = T3.account_id

WHERE
  T1.gender = 'M'
```

Listing 12: True query from the entry with id 132 from the financial domain of BIRD-Bench with the faulty *INNER JOIN* commands highlighted.

Corrected true Query:

```
SELECT
  AVG(T1.amount)
FROM
  loan AS T1
  INNER JOIN account AS T2 ON T1.account_id = T2.account_id
  INNER JOIN disp AS T3 ON T2.account_id = T3.account_id
  INNER JOIN client AS T4 ON T3.client_id = T4.client_id
WHERE
  T4.gender = 'M'
```

Listing 13: Corrected version of the query from the entry with id 132 from the financial domain of BIRD-Bench with the corrected *INNER JOIN* commands highlighted.

Example 3: Database Schema Non-Alignment

The third example, shown in Table 4.2, illustrates a question improperly aligned with the database schema's structure, leading to the LLM lacking the necessary information for a valid prediction. This particular question overlooks the possibility of clients having multiple dispositions, as well as multiple disposition IDs. The structure of the database schema can be viewed in Figure 3.3.

Question Feasibility

Table 4.2: Feasibility of the posed question based on the current database schema, highlighting the one-to-many relationship between client and disposition.

Question	Feasibility
What is the disposition id of the client who made $5100 USD transaction on 1998/9/2?	Not feasible with current schema due to one-to-many relation between client and disposition.

4.4 Experiments on the Cleaned Datasets

In this section, the results of our experiments with text-to-SQL conversion on the cleaned version of the financial dataset described in Section *4.3 Analysis of Experiment Results* are presented. The focus was on assessing the impact of dataset quality on the performance of various models. Three variations of the financial domain dataset from BIRD-Bench was used: the original dataset with noise in both questions and SQL queries, a cleaned version of the dataset where noise was removed from both components, and a partially cleaned version where only the SQL queries were corrected.

The analysis involved three models: *Zero-Shot with GPT-3.5*, *DIN-SQL with GPT-3.5* and *Zero-Shot with GPT-4*. The results, visualized in the accompanying bar plot in Figure 4.4, indicate significant findings regarding the robustness and adaptability of these models in varying conditions of data cleanliness.

Interestingly, on the original noisy dataset, the zero-Shot approach using GPT-3.5 showed approximately 5% lower accuracy compared to the DIN-SQL model. However, when the data was cleaned of noise, the performance gap between the zero-Shot GPT-3.5 and DIN-SQL models narrowed considerably, with the zero-Shot model trailing by a mere 0.5% in accuracy. Remarkably, in the scenario where only the SQL queries were corrected, the zero-Shot GPT-3.5 model excelled, achieving an accuracy of 42.45%, which is slightly higher than the 41.51% accuracy of the DIN-SQL model.

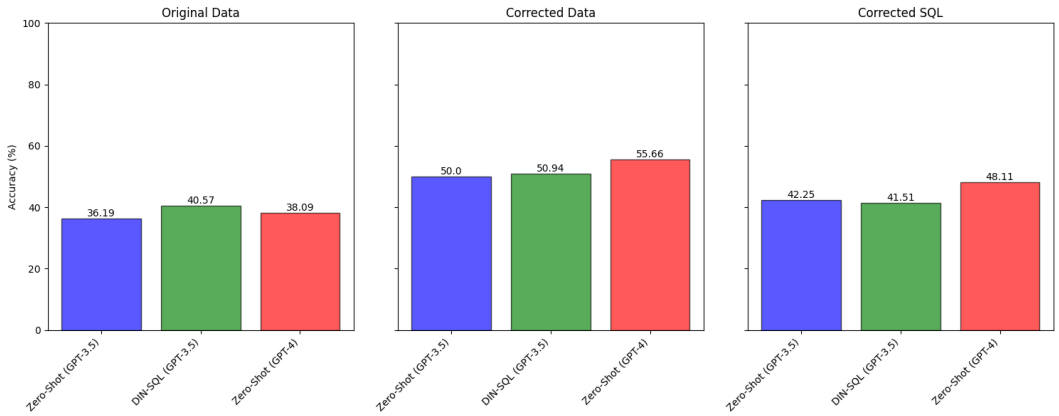

Figure 4.4: Execution accuracy of the different models on the different datasets.

Furthermore, the experiments revealed that GPT-4's zero-Shot model benefited most significantly from the noise removal, although all models substantially improved their performance. The models demonstrate improved performance on both the dataset with cleaned questions and queries and the dataset with solely cleaned SQL queries.

4.5 Question Classification

This section details the outcomes of the question classification experiments discussed in Section *3.6 Question Classification*. It includes graphical representations of the data from both the base and reasoning classifiers.

4.5.1 Question Annotation

The annotations of questions in the Financial domain of the BIRD-Bench development dataset are displayed in Table 4.3. Out of a total of 106 questions, 64 were determined to have no significant effect on how the LLM and prompting models generate SQL queries. In contrast, 42 questions were identified as potentially or definitively influencing the SQL query generation process.

Table 4.3: Number of annotations in each classification category. The definition of each category can be viewed in Table 3.2.

Category	No. Annotations
0	64
1	25
2	5
3	7
1 & 3[2]	5

4.5.2 Base Classifier

As detailed in section *3.6.2 Base Classifier*, the base classifier was subjected to two distinct experiments, the outcomes of which are discussed in this section. The first experiment, utilizing the categories defined in Table 3.2, grouped questions into two broad classifications: those with no significant impact on the LLM (categorized as 0), and those whose formulations could potentially mislead the LLM's interpretation (categorized as 1). The second experiment focused on assessing the classifier's ability to categorize questions into the original four categories as presented in Table 3.2. This section elaborates on the performance and findings of these two experimental evaluations. When reviewing and analyzing the results of the classifiers it should be taken into consideration that the baseline model, of only classifying questions into class 0 as a correct question, has an accuracy of 60.3%. This is based on the 64 questions that have been annotated as correct into class 0, presented in Table 4.3, and the total number of questions is 106.

Classification - Two Categories

This section contains the results from three distinct experiments where questions were categorized by the base classifier as either correct (0) or incorrect/ambiguous (1), the latter incorporating the three different types of faults or errors detailed in section *3.6.1 Annotating Question Quality*. Figure 4.5 and 4.6 presents a comparison of the experiments across the four key metrics — accuracy, precision, recall, and F-1 score. The first Figure, 4.5, displays the experiments titled *GPT-3.5-Turbo with Correct Examples*, *GPT-3.5-Turbo with Incorrect Examples* and *GPT-3.5-Turbo with Correct & Incorrect Examples*. Whereas the second Figure 4.6, the same three different experiments when tested on the GPT-4 LLM.

In the experiments, completed with the GPT-3.5-Turbo model, when provided with examples annotated as correct (0), achieved a modest 54% accuracy rate. This rate improved

[2]Of all the annotations there were 5 questions that were considered to be of both categories 1 and 3.

to 66% when the model was fed examples of only questions annotated as incorrect or ambiguous (1), whereas when provided with correct and incorrect examples the model's performance in terms of accuracy reduced to 56%. In regards to precision, the GPT-3.5-Turbo showed a precision of 42% with annotated correct examples, marginally increasing to 52% with annotated incorrect examples, and reduced to 47% when provided with correct % and incorrect examples.

The recall metric varied noticeably. GPT-3.5-Turbo achieved a high recall of 81% with correct examples but saw this figure drop to 41% with incorrect examples, and increase again to 68% when provided with correct and incorrect examples. Further, the GPT-3.5-Turbo obtained an F1-Score of 55% with correct examples, which decreased to 45% with incorrect examples, and increased to 52% when provided with correct and incorrect examples.

In summary, the GPT-3.5-Turbo model's performance fluctuated based on the type of examples used. It demonstrated higher accuracy and recall rates with incorrect examples, but these gains were not consistent across other metrics like precision and F1-score. The balanced use of correct and incorrect examples did not necessarily lead to better overall performance, highlighting the complexity and variability in the model's response to different prompting examples.

GPT-3.5-Turbo

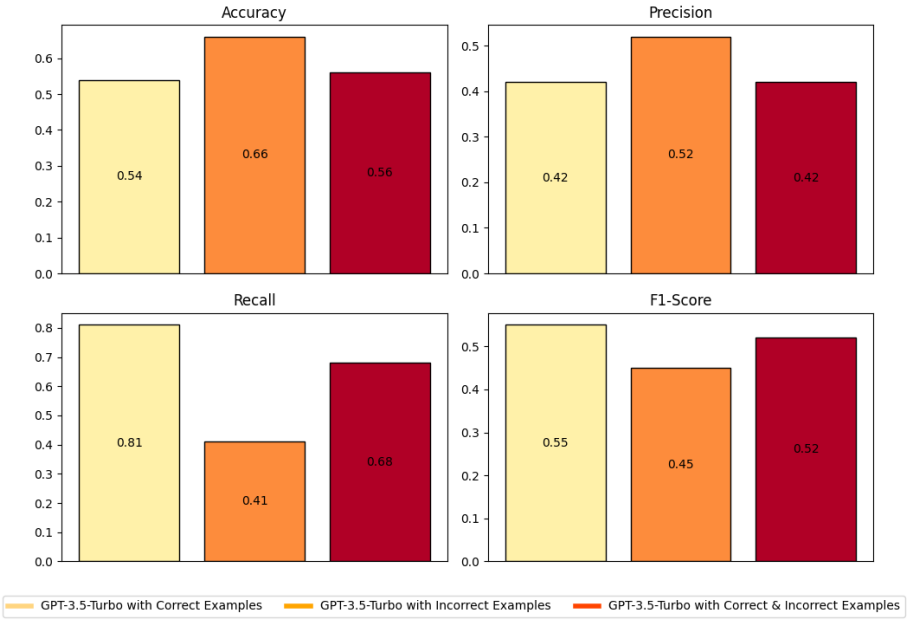

Figure 4.5: Accuracy, precision, recall and F-1 Score metrics of the three classification experiments of two categories with GPT-3.5-Turbo.

The performance metrics of GPT-4 classifiers showed notable differences compared to their GPT-3.5 counterparts. When analyzing accuracy, classifiers prompted with correct examples reached a 72% accuracy rate. This increased to 74% with incorrect examples and peaked at 80% when trained with both. Similarly, precision metrics indicated a 60% score for classifiers with positive examples, 66% with incorrect examples, and 75% when both types were used. Regarding recall, classifiers with correct examples attained 57%, those with incorrect examples reached 51%, and those prompted with a mix achieved 65%. Finally, examining the F-1 score revealed that classifiers prompted with correct examples achieved 58%, those with incorrect examples also reached 58%, and those prompted with both types of examples achieved a higher score of 70%.

To summarize, the GPT-4 classifiers exhibit varied performance based on the type of prompting examples used. The highest accuracy and F-1 scores were observed when both correct and incorrect examples were used, indicating that a balanced prompting approach approach yields the most effective results.

GPT-4

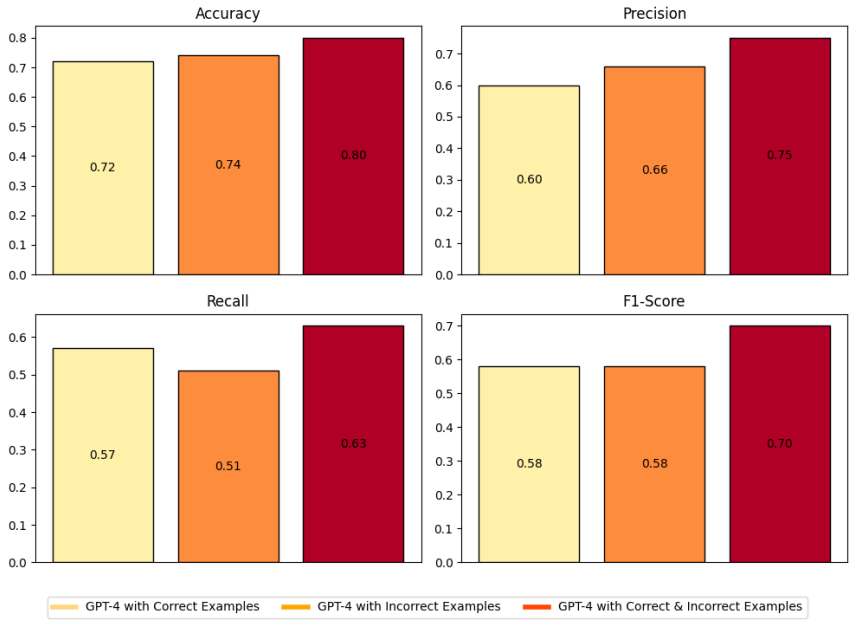

Figure 4.6: Accuracy, precision, recall and F-1 Score metrics of the three classification experiments of two categories with GPT-4.

Classification - Four Categories

The results of the experiments using the base classifier are displayed in Figures 4.8, 4.9, 4.10 and 4.11. These experiments focused on classifying the four categories discussed in the second paragraph of the *3.6.2 Base Classifier* section. The two models evaluated are based on the same prompt template, in which they have been provided a definition of each category together with at least one example of each category. The full prompt template is available at the project's GitHub [3].

In the bar plots of Figure 4.7, the weighted averages for accuracy, precision, recall, and F-1 scores are compared for both models. These averages reveal that the performance disparity between the GPT-4 and GPT-3.5 models is slight, with each model showing minor superiority in different areas. Specifically, the GPT-4 model demonstrates higher accuracy and F-1 scores, whereas the GPT-3.5 model outperforms in precision and recall.

The results are depicted for the GPT-3.5 and GPT-4 model experiments, through two visual representations. Bar plots contrasting the actual category distribution against model predictions are shown on the left, while heatmaps depicting confusion matrices of the classification performance are displayed on the right. The diagonal of the heatmaps in Figures 4.9 and 4.11 shows that the GPT-4 classifier classifies a higher number of questions of category 0 correctly, but the GPT-3.5 model classifies categories 1 and 2 with higher accuracy.

Weighted Averages

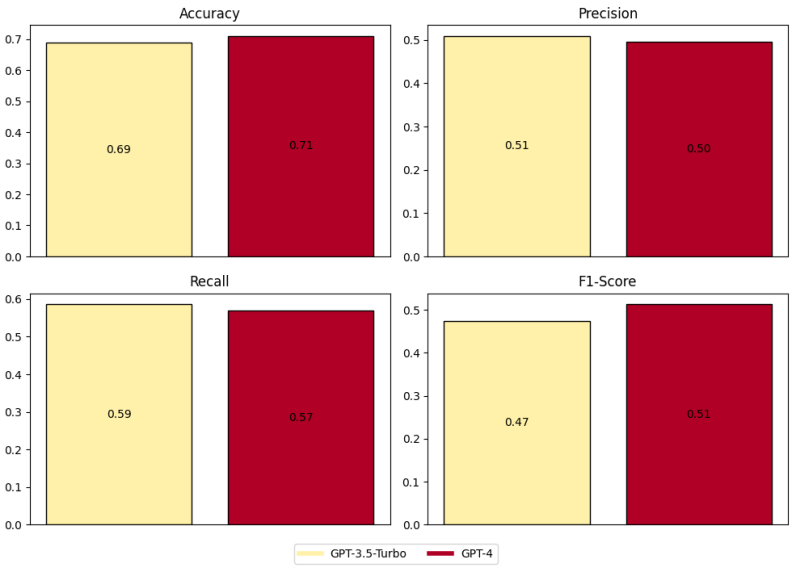

Figure 4.7: Weighted averages of the models using GPT-3.5-Turbo and GPT-4.

[3]https://github.com/niklaswretblad/Text-to-SQL-Generation

GPT-3.5-Turbo

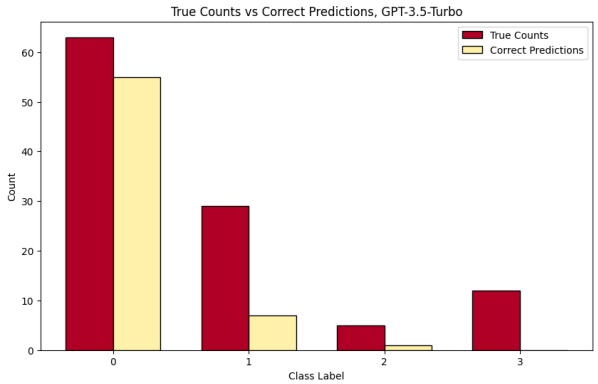

Figure 4.8: True counts vs correct predictions by the model based on GPT-3.5-Turbo.

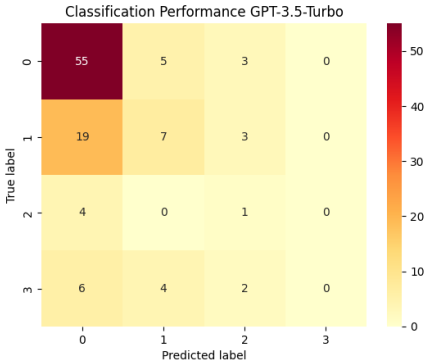

Figure 4.9: Heatmap over classification results generated by the model based on GPT 3.5-Turbo.

GPT-4

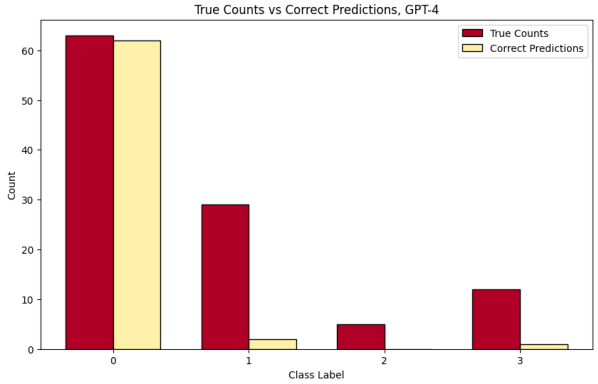

Figure 4.10: True counts vs correct predictions by the model based on GPT-4.

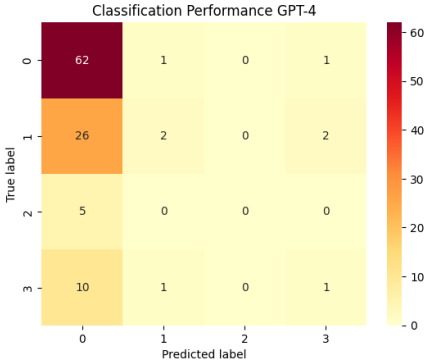

Figure 4.11: Heatmap over classification Results generated by the model based on GPT-4.

4.5.3 Reasoning Classifier

In this section, the results of the reasoning classifier model described in Section *3.6.2 Reasoning Classifier* are presented. The classifier was tasked to classify questions into two distinct categories, and experiments using both the GPT-3.5-Turbo and GPT-4 models where performed. The classification was based on the quality of the questions, with *0* denoting a question without errors, and *1* indicating an erroneous question according to the different types of errors within the questions as described in *3.6.1 Annotating Question Quality*.

Figure 4.12 contains the results of the two different models. As can be seen in the figure, GPT-4 demonstrated a higher accuracy with a score of 0.68, compared to GPT-3.5-Turbo, which had an accuracy of 0.65. Despite GPT-4's superior overall accuracy, GPT-3.5-Turbo excelled in precision with a score of 0.73, against the 0.58 of GPT-4. This implies that when GPT-3.5-Turbo classified a question as containing errors, it was more likely to be correct in that assessment.

The recall score highlighted a significant difference between the models. GPT-4's recall was 0.69, indicating its strength in identifying a broader range of errors within the questions.

In comparison, GPT-3.5-Turbo had a recall of just 0.19, suggesting that it missed a substantial number of errors that GPT-4 could identify.

The F1-Score showed GPT-4 with a score of 0.63, signifying a balanced classification performance. On the other hand, GPT-3.5-Turbo had an F1-Score of 0.30, reflecting its lower effectiveness in identifying errors across the range of questions.

The results from this experiment offer a comprehensive view of the models' capabilities in question classification, with GPT-4 outperforming GPT-3.5-Turbo in most metrics. GPT-4's higher accuracy and recall indicate a more robust model for identifying both the presence and type of errors in questions. GPT-3.5-Turbo's higher precision suggests that while it may not identify as many errors, its classifications are more likely to be correct when it does identify an error.

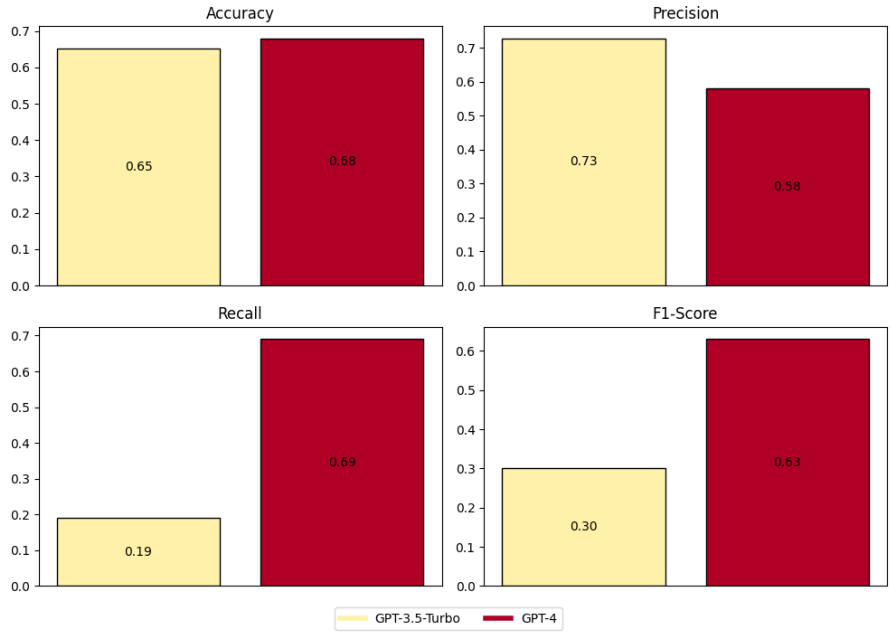

Figure 4.12: Accuracy, precision, recall and F-1 Score for the reasoning classifier using both the GPT-3.5-Turbo and GPT-4 language models.

4.6 Correcting SQL Queries

From the subset of queries examined, ten unique errors were identified and targeted for correction using the LLM. Upon conducting the ablation study, GPT-4 demonstrated a high degree of proficiency in correcting the faulty SQL queries. Out of the ten queries, nine were accurately corrected by the LLM. The success of these corrections was verified by comparing the data retrieved by both the corrected and the original true SQL queries, which matched precisely in these cases.

Upon further examination of the single query that GPT-4 failed to correct, it was discovered that the "hint" or external knowledge provided by the dataset, contained inaccuracies. Once the hint was revised to reflect the correct information, GPT-4 successfully corrected the last of the ten faulty queries.

5 Discussion

In the discussion chapter, a comprehensive analysis of the results, methodologies, and broader implications of the project will be undertaken. The discussion will begin with a detailed examination of the findings from the *4 Results* chapter, considering them from multiple perspectives. This will be followed by an evaluation of the methodologies detailed in the *3 Method* chapter, including a reflection on how these approaches may have influenced the outcomes of the experiments. Lastly, the discussion will expand to consider the work in a wider context, exploring the broader implications, relevance, and potential applications of the research, situating it within the larger academic and societal landscape.

5.1 Results

This section reviews the findings from the *4 Results* chapter, offering a concise overview of our research outcomes. We begin by discussing the performance of models within the financial domain, highlighting how they fare in this specific context. Following this, we examine the impact of data quality on text-to-SQL models, emphasizing the importance of data quality in these systems.

Our exploration then extends to the different pitfalls and types of noise encountered in text-to-SQL conversion, identifying common challenges in this area. We also discuss potential noise in other domains in BIRD-Bench, broadening our understanding of model performance across the entire benchmark.

The next part of our discussion focuses on strategies for addressing noise and dirty values in BIRD-Bench and in text-to-SQL processes, underscoring the importance of effective data management. We then discuss the capabilities of LLMs in classifying and correcting data quality issues, evaluating their effectiveness in maintaining data accuracy and integrity within text-to-SQL conversion tasks.

5.1.1 Performance of Text-to-SQL Models Within the Financial Domain

In this subsection, the analysis delves into the performance nuances of text-to-SQL models, specifically within the financial domain. The exploration is aimed at understanding the challenges observed in the financial domain, and whether those challenges are unique or reflect a more general trend affecting text-to-SQL models.

When reviewing the execution accuracies from the first iteration shown in Figure 4.3, there's a notable difference of nearly 5% in execution accuracy between both models based on GPT-3.5, the zero-shot model and the DIN-SQL model, within the financial domain. The zero-shot model achieved an execution accuracy of 36.19%, in contrast to the DIN-SQL model, which performed better at 40.57%. Similarly, the zero-shot model using GPT-4 showed a similar accuracy of 38.09%.

An overarching observation across all models is the overall low accuracy rates, with less than half of the queries being accurately converted to SQL in the financial domain. This raises an important question: Are the challenges in the financial domain particularly hard, or

do these trends reflect a broader issue across the entire BIRD-Bench dataset and text-to-SQL models in general?

Various factors could be the cause of the low accuracy performance of the models. One aspect, that has not been quantified in these experiments, is the database structure and database relationship complexity of the domain, which could have a significant impact on the accuracy of SQL generation. For instance, viewing the Financial domain's database structure and relations in Figure *3.3*, it is evident that the column names of A2-A16 of the district table, is not substantial enough to understand what each column represents. Another example, from the same figure, is how the *disp* (disposition) table is crucial to understand to connect each account to a customer.

Additionally, another aspect to consider is whether the pretraining data for the LLMs includes information from the BIRD-Bench dataset domains. GPT-3.5, as detailed in Table 2.3, utilized 175 billion parameters for training, indicating extensive data usage. GPT-4, advancing further, is believed to be trained with even more parameters and data. However, the specifics of their training data remain uncertain [75]. If these models were trained with data from specific database domains, they might inherently understand the structure of these databases, including tables, columns, and their interrelations. This understanding could enhance their ability to accurately generate SQL queries for related questions. Potentially this could be the reason for the low performance in the financial domain. The potential biases introduced by the training data and their impact on model performance will be further explored in the upcoming Section *5.3.2 Bias*.

Lastly, a third factor that was not considered in the initial phase of the experiments, was how the data quality of the question and the true SQL query of each data point affected the SQL generation of the models. An ambiguous formulated question could complicate the interpretation and misguide the LLM, resulting in a faulty generated SQL query, which would have a significant impact on the model's accuracy. The presence of noise could have obscured the results.

To expand the perspective of how the model's performance in the financial domain stands in comparison to other domains as explained in Section *4.1 Comparison Between Domains in BIRD-Bench*. Further we could review Figure 4.3, showing the results of the first iteration, and 4.1, displaying comparative results of DIN-SQL between different domains. When comparing Figure 4.3 and 4.1 it can be observed that the DIN-SQL performance in the financial domain differentiates between the two plots. The execution accuracy of the model presented in Figure 4.1, is 32%, whereas in this project the DIN-SQL model implemented had an accuracy of 40.57%. Considering neither the implementation of the models nor the data is differentiating, it is interesting that the performance of the models is varying. However, the DIN-SQL model was published and tested in April 2023 [56], whereas the DIN-SQL model in this project was tested on the 11th of October. As mentioned in Section *5.2.3 Validity* it has been proven that GPT's performance changes over time and between updates [34]. Therefore, there is a significant possibility that the underlying GPT-3.5-Turbo model has changed between these occasions. This subject will be discussed further in Section *5.2.3 Impact of GPT's Updates*.

To deepen the comparative analysis between the domains, Figure 4.2, displaying the distribution of question difficulty across different domains, can be observed. From the plot, it can be concluded that there is no clear correlation between the DIN-SQL model's accuracy in each domain and each domain's number of tables or columns in the BIRD-Bench development dataset. When comparing the financial domain in the context of the other domains, it can be determined it obtains 55 columns across 8 different tables, whereas the mean of the domains analyzed in Section *4.1 Comparison Between Domains in BIRD-Bench* is equivalent to 78.3 columns and 7.3 tables. Hence, it is not an outlier in terms of the number of tables and columns. When further examining Figure 4.2, depicting the question difficulty distribution of each domain, the financial domain does not stand out in either term of question difficulty or how the question difficulty distribution affects the accuracy in comparison to the other domains.

From these observations, it becomes clear that the difficulties faced by text-to-SQL models in the financial domain are not unique to it. Similar performance patterns, irrespective of question difficulty or database structure in terms of the number of tables and columns, are evident across the BIRD-Bench dataset. This suggests that the challenges in converting natural language queries to SQL are not solely domain-dependent but are reflective of broader trends affecting text-to-SQL models in various contexts. Therefore, addressing these challenges and enhancing the accuracy of these models requires strategies that go beyond domain-specific optimizations, focusing instead on fundamental aspects of data quality and text-to-SQL model design.

5.1.2 Impact of Data Quality on Text-to-SQL Models

In Section *5.1.1 Performance of Text-to-SQL Models Within the Financial Domain*, it was discussed that there appears to be no direct correlation between the size of a database and the accuracy of the DIN-SQL text-to-SQL model. Furthermore, it was also reasoned that a distinct relationship between the difficulty of the questions in a domain and the accuracy of the queries was not consistently evident. While the trend for some domains suggested that more challenging questions may lead to lower accuracy, this was not a uniform occurrence, and sometimes the opposite was true. This led to question what other factors might be influencing the accuracy of these models, beyond what had been previously examined in the domain comparison and analysis described in Section *3.2.2 Analysis and Comparison of Domains*. It was theorized that various factors within the database structure other than the size, such as the conventions for naming columns and tables, and the nature of the relationships among different entities, could influence the effectiveness of the models. Additionally, there's a possibility of bias in the LLM towards specific themes or domains. For instance, if the LLMs has been predominantly trained on articles about football, it might exhibit enhanced proficiency in executing text-to-SQL queries in the *european_football_2* domain of BIRD-Bench, relative to other domains and topics. Furthermore, there could be a probability of the LLMs being trained on specific database domains used in the BIRD-Bench as well. Lastly, it was argued that the quality of the data itself could be a critical aspect to consider. The presence of noise, either in the natural language questions or in the SQL queries, could perhaps significantly impact the model's performance across different domains. Different levels of noise in the different domains could be a contributing factor to the ambiguous nature of the trends across the domains.

In the analysis of the results from the first round of text-to-SQL experiments, various errors were identified in the financial dataset as can be seen in *4.3 Analysis of Experiment Results*. Within the dataset, 52 out of 106 distinct data points were identified to contain errors, which more specifically included errors in 44 questions and in 26 true SQL queries. After correcting the errors, the text-to-SQL models were run again on the new and corrected version of the dataset. The initial results on the original dataset showed that the DIN-SQL model outperformed the zero-shot baseline by approximately 5-6%. However, upon rerunning the models on the dataset with corrections, an interesting shift was observed. The performance gap between the DIN-SQL and the zero-shot model significantly narrowed, with the zero-shot model trailing behind by only about 0.5%.

To further understand the influence of data quality, we continued on a more nuanced analysis. A third dataset was created to properly test the models robustness towards noisy questions, which consisted of the original, uncorrected questions paired with the corrected SQL queries. The results from this dataset presented another interesting finding: the zero-shot model achieved a 42% accuracy rate, surpassing the DIN-SQL model, which scored 41%. This outcome suggests that the DIN-SQL model, or similar techniques used in the benchmark, may be overfitting to noise present in the data.

The findings from these experiments underscore the significant role of data quality in text-to-SQL model performance. Noise in the data, whether in the form of ambiguities, inaccura-

cies, or complexities in the natural language questions, or errors in the SQL queries, evidently affects the accuracy and reliability of these models. The correction of such noise not only improves the overall performance but also brings to light potential issues of overfitting in more sophisticated models like DIN-SQL. These insights emphasize the need for meticulous data curation and validation in the development and evaluation of text-to-SQL models, ensuring that they are not only accurate but also robust to the variances in real-world data.

5.1.3 Different Pitfalls and Types of Noise in Text-to-SQL

In our result analysis Section *4.3 Analysis of Experiment Results*, the financial domain of the BIRD-Bench dataset was carefully examined and several categories of errors was identified that could potentially impair the accuracy and efficiency of text-to-SQL conversion. The types of noise identified in the dataset were broadly categorized into semantic errors, vague or ambiguous questions, incorrect SQL queries, divergences between questions and their corresponding SQL queries, synonyms causing confusion, and issues with string capitalization. This section aims to dissect these different types of noise, drawing from the previous comprehensive analysis of the financial domain in the BIRD-Bench dataset. By doing so, the aim is to illuminate the path towards more robust, accurate, and efficient text-to-SQL models, capable of transforming the way one interacts with and extract value from data in complex domains.

The first significant aspect of this is capitalization sensitivity. In the result analysis described in *4.3 Analysis of Experiment Results*, seven data points in the financial domain of BIRD-Bench was found to have issues with the capitalization of values, where the capitalization of a value did not match between the given question and true SQL query. SQL queries, specifically those dealing with string data types, are case-sensitive. This characteristic demands that the capitalization in the string within the query must precisely match the data's capitalization as stored in the database. For instance, querying for a name with *SELECT * FROM Users WHERE Name = 'John Doe'* will yield no results if the name in the database is stored as "john doe". This poses a unique challenge for text-to-SQL models, which need to either adapt the input queries to the database's case sensitivity, standardize them to a common format or use something like fuzzy string matching to allow for non-exact matches of strings. Fuzzy string matching, also known as approximate string matching, is a technique used to determine how similar two sequences of characters are. It typically involves algorithms that compute the distance or difference between two strings, such as the number of insertions, deletions, or substitutions required to transform one string into another. This technique is widely used in applications like spell checking, plagiarism detection, and data deduplication, where exact matches are not necessary and near-matches are acceptable. The investigation of string-matching methodologies, due to their complexity and scope, is left for future work to explore in more detail.

Another crucial factor is the alignment of the query with the database structure. The questions asked must correspond accurately to the existing database schema. Frequently, queries fail or return incorrect results because they reference non-existent table or column names or misunderstand the relationships between different data entities. An example of a non-aligned query from the financial domain of BIRD-Bench could be, *"List out the no. of districts that have female average salary of more than 6000 but less than 10000?"* This question presumes that there is a way to calculate the average salary for only the females living in a certain district, which there isn't. In the database, the district table has a column A11 which corresponds to the average salary of all people living in the district, but this number also includes average salary for males. In other parts of the database, there is no information about salaries. Therefore, calculating the female average salary of a district is not possible given the database's structure. Such discrepancies necessitate that text-to-SQL models not only understand the natural language query but also correctly map it to the relevant database schema. If a question is posed that requires information or relationships not present in the schema, the model cannot generate a valid SQL query to answer it. Attempting to create a

query from a question outside the schema's scope is of no use, as the necessary data points or relationships simply do not exist in the database. Therefore, it's crucial to align questions with the schema's capabilities to ensure meaningful and accurate SQL query generation.

The text-to-SQL models also face the challenges of dealing with grammatical and semantic formulations as well as language ambiguities. Humans can easily understand and interpret queries even if they are not grammatically perfect, such as the question *"Who be the oldest employee?"* Despite its grammatical flaws, it is generally understood by human listeners. However, it's important to note that there is a threshold beyond which a question becomes too ambiguous or poorly structured to be interpretable, leading to confusion or incorrect results. For instance, a query like *"Oldest employee what time?"* lacks sufficient context and clarity, making it challenging for both humans and text-to-SQL models to discern its intended meaning. This highlights the need for a balance between models accommodating linguistic variances and users maintaining a level of clarity and structure essential for accurate query interpretation. As such, enhancing the ability of these models to handle a wide range of linguistic inputs, while also recognizing the limits of interpretability, is crucial in the development of more robust and effective text-to-SQL conversion systems.

In another way, the ambiguity inherent in language apart from linguistic errors poses a significant challenge. Take, for instance, a query like "What was the most recent order?" Here, the ambiguity lies in interpreting "most recent" – does it mean the latest order by date, the last recorded entry, or some other criterion? This requires sophisticated handling by text-to-SQL models. Many different such ambiguities exist in natural language apart from the example above, and text-to-SQL models must either have mechanisms in place to resolve these ambiguities or seek additional information for clarification in order to ensure a correct answer. 19 questions in the financial domain of BIRD-Bench were found to contain such ambiguities during the result analysis as described in Section *4.3 Analysis of Experiment Results*, and most of those ambiguities were related to uncertainties in which column or table to be returned from the query. For instance, in Listing 8 question 159 from BIRD-Bench's development set, which begins with *"List all the withdrawals..."*, exemplifies this ambiguity. It's unclear whether the query should return every column associated with withdrawal transactions or just the IDs of these transactions. While there have been initiatives to address such ambiguities in natural language processing, particularly in text-to-SQL conversion [76], there remains a significant need for further research to effectively resolve these issues.

Another pitfall that is worth noting in the text-to-SQL domain is the use of synonyms or similar terms, which can lead to significant issues in accurately translating natural language questions into SQL queries. This is because different words or phrases, while similar in meaning, may correspond to distinct concepts or operations in the context of a database. An example of this in the financial domain of BIRD-Bench is the question *"What is the sum that client number 4's account has following transaction 851? Who owns this account, a man or a woman?"*. For this example, the text-to-SQL model incorrectly interprets the word *"sum"* as an instruction to aggregate transaction amounts by using the SUM() SQL operator, leading to a query that calculates the sum of amounts instead of retrieving the account balance. The correct version, however, uses the term "balance" which accurately maps to the trans.balance field in the database, thereby fetching the specific balance of an account after a particular transaction. The example underscores the importance of precise language in querying databases to ensure accurate data retrieval. Previous research has explored the handling of synonyms in text-to-SQL contexts [77], yet the advancement of robust synonym recognition and interpretation within these models remains a significant area for future investigation.

Each pitfall detailed in this section represents unique and intricate challenges in the quest to create robust text-to-SQL models. Central to these challenges is the task of managing noise and imperfectly formed queries, underscoring the complexities involved in deciphering and processing natural language with its inherent ambiguities. This highlights the multifaceted and diverse nature of the task, where each issue represents a separate domain of research in

its own right. Addressing these challenges are promising areas for future research, where solutions are needed to build robust and trustworthy text-to-SQL models.

5.1.4 Potential Noise in Other Domains in BIRD-Bench

The discovery of significant noise in the financial domain of the BIRD-Bench benchmark raises concerns about the potential for similar issues in the other 36 domains. In this section we assess the likelihood and implications of noise in the rest of the dataset.

Our analysis of the domains in the BIRD-Bench dataset, described in Section *4.1 Comparison Between Domains in BIRD-Bench*, revealed interesting insights into the trends across its domains. In particular, the financial domain closely aligns with the average metrics observed across all domains. This domain has a near-average number of tables and columns and a slightly below-average accuracy rate for the DIN-SQL model. Specifically, while the mean accuracy of the DIN-SQL model across all domains is 37%, the financial domain's accuracy stands at 32%. This finding suggests that the challenges we encountered in the financial domain might not be unique and could be reflective of broader trends within the dataset. Variations in question difficulty across the different domains were also notable. While some domains, like *codebase_community*, predominantly feature simpler questions, others, such as *thrombosis_prediction*, present a more balanced mix of question complexities. This diversity in question difficulty does not consistently translate to corresponding accuracy levels in the DIN-SQL model. For instance, domains with a higher concentration of simple questions do not always yield higher accuracies. Similarly, the presence of more challenging questions does not invariably result in lower model performance. Although a general trend was observable, the outliers indicate a lack of clear correlation between question complexity and the performance of text-to-SQL models. This finding is intriguing, as it contradicts the intuitive assumption that more complex questions would pose greater challenges for text-to-SQL models in accurately converting them into SQL queries. Worth adding is that no trend between the complexity in terms of the size of a database and the accuracy of the DIN-SQL model could be found either.

A couple of reasons might explain the absence of this correlation. Firstly, the question difficulties supplied by BIRD-Bench perhaps do not accurately reflect the actual difficulties of the questions. When annotating the question difficulties, the annotators of the questions where asked to rank the question from 1-3 in four different categories as described in Section *2.6.2 A Closer Look at BIRD-Bench*: question comprehension, schema linking, external knowledge, and SQL complexity. The overall score would then determine the overall difficulty of the question. Although thorough, this approach may not have fully captured the actual difficulty for text-to-SQL models to handle each question. The reliance on human annotators introduces subjectivity, potentially leading to inconsistent difficulty ratings. A task that appears challenging to a human in various areas may not align with the challenges a text-to-SQL model encounters.

Secondly, the absence of a clear trend might also be attributed to the presence of noise not only in the financial domain but in other domains as well. Therefore, on a surface level, it seems likely that noise is also present in the other parts of the dataset. Further, the quality of question annotation in BIRD-Bench, raises several concerns that warrant attention. Despite the involvement of native English speakers with database knowledge, there are indications of poor-quality English in some questions. This raises questions about the effectiveness of the selection criteria for annotators or the validation process. The requirement for questions to receive at least two votes for validity seems insufficient, as evidenced by the presence of noise and linguistic issues in the financial domain. This observation suggests that similar issues might exist in other domains within the dataset.

Furthermore, the SQL annotation process in BIRD-Bench, while rigorous, is not immune to potential issues. The involvement of skilled data engineers and students, coupled with the implementation of a double-blind technique for validity checks, indicates a strong com-

mitment to dataset quality. However, the discovery of errors in 19 out of 106 queries in the financial domain points to potential shortcomings in the annotation process for SQL queries.

As mentioned, the presence of noise in the dataset could have obscured the identification of real trends. This is particularly relevant in the financial domain and may further mask the actual challenges models face. While the BIRD-Bench annotations provide a structured framework, they may not accurately reflect the real-world difficulties encountered in text-to-SQL tasks. This calls for a cautious interpretation of model performance across different domains, as the existence of noise in the dataset could have a significant impact on the perceived effectiveness of various models.

5.1.5 Addressing the Noise and Dirty Values in BIRD-Bench and in Text-to-SQL

In addition to the aforementioned categories of noise in Section *5.1.3 Different Pitfalls and Types of Noise in Text-to-SQL*, a significant challenge arises from the presence of "dirty" data within the BIRD-Bench benchmark. As noted by the dataset authors, this aspect is an intentional feature, designed to mimic the imperfect nature of real-world databases, and does not only capture differing string capitalizations but other types of dirty values in the databases as well. However, our analysis reveals a critical limitation in the way this aspect is handled. Not all questions in the dataset are associated with dirty values, and those that are, lack clear annotations. This inconsistency presents a substantial obstacle in accurately assessing model performance in handling dirty data, as showcased by the string capitalization issues found in the result analysis described in Section *4.3 Analysis of Experiment Results*. This also holds true for the other types of noise that were found in the dataset, like the semantic errors or questions not aligning with the database schema. There is at the moment no way to measure a model's effectiveness against noise. Currently, performance is measured through overall accuracy across the entire dataset, which fails to provide insights into how well a model copes with the nuances of dirty data or noise. As was shown by the model's performances in the result Section *4.4 Experiments on the Cleaned Datasets* and as discussed in Section *5.1.3 Different Pitfalls and Types of Noise in Text-to-SQL*, the current benchmarks do not seem to be sufficient in measuring model performance. When the noise was removed from the dataset, the baseline zero-shot model almost performed as well as the current SOTA prompt engineering technique, suggesting that techniques and models are overfitting to the noise. Being able to measure how a model copes with noise and dirty values is clearly particularly important for text-to-SQL models, where precise data handling is paramount for generating accurate and reliable SQL queries.

To address this gap, we propose three potential solutions. The first is the creation of separate datasets dedicated exclusively to evaluating a model's performance on handling the dirty database values and different kinds of noise mentioned in the discussion Section *5.1.3 Different Pitfalls and Types of Noise in Text-to-SQL*. For example, a dataset for measuring a model's robustness towards string capitalizations would consist of question/query pairs with an intentional non-matching capitalization of values, to provide a focused environment to test and refine models' capabilities in this specific aspect.

Secondly, the existing BIRD-Bench dataset could be enhanced by annotating whether question/query pairs contain dirty values and or different kinds of noise or not. This would allow for a more granular and targeted analysis of model performance, enabling researchers to distinguish between general model accuracy and proficiency in handling dirty data and noise. For example, if all the questions with semantic errors were annotated as such, one could measure the model accuracy on those questions separately and therefore easily develop a robust model towards semantic errors.

Lastly, the third solution involves ensuring that the questions input into the text-to-SQL models are inherently free from noise. This approach would force users to input 'perfect questions,' which would include developing a classification mechanism to identify and flag suboptimal or 'bad' questions before they are inputted into the model. Once a problematic

question is detected, the LLM could then prompt the user to reformulate their query, guiding them toward crafting a clearer, more precise question. This proactive strategy would not only improve the overall quality of the input data but also reduce the burden on the model to handle dirty data and noise. By integrating such a feature, one could possibly significantly enhance the efficiency and accuracy of text-to-SQL models but most of all make them more reliable, safe, and more robust towards noise.

5.1.6 The Need for External Knowledge

The design and outcomes of the BIRD-Bench dataset demonstrates that the integration of external knowledge in text-to-SQL conversion is essential. BIRD-Bench performs this integration by providing "hints" or external knowledge alongside each question-query pair. These hints serve as clarifications for things like ambiguous column names or instructions on computing specific metrics. For example, in the dataset's financial domain, understanding what columns A2-A16 in the districts table represent is needed since the names themselves does not provide any information. A notable example from the dataset demonstrates that the query to discern the higher unemployment rate between 1995 and 1996 requires knowing what 'A12' and 'A13' stand for, where A12 is the average unemployment rate in 1995 for a district, and A13 the average rate for 1996. Without this additional context, formulating the SQL query accurately becomes a significant challenge.

The impact of including hints in the dataset is therefore reflected in model performance. BIRD-Bench's own analysis shows a substantial increase in accuracy when these hints are used, highlighting the dependency of current models on supplementary information for optimal performance [9].

However, this raises questions about the application of text-to-SQL models in real-world scenarios, such as corporate environments, where pre-provided hints or external knowledge might not be available. This gap leads to the issue of how necessary knowledge for accurate query answering can be obtained. For instance, a company looking to use a text-to-SQL model for its internal databases might not have this essential external information readily available.

One potential solution is to supply comprehensive database documentation to the text-to-SQL model, including detailed declarations and descriptions for each table and column. But this approach is not without challenges. The context length limitation of current LLMs might restrict how much information can be effectively used. Additionally, where adequate documentation is lacking, the task of manually creating such detailed descriptions can be resource-intensive and daunting.

Future work of this field should explore whether LLMs could autonomously generate or infer the necessary external knowledge. With recent advancements, like OpenAI's models capable of conducting web searches to answer queries, there's potential for these models to autonomously acquire relevant information. This capability could significantly enhance the accuracy of text-to-SQL models in real-world applications. However, the feasibility and accuracy of LLMs in autonomously sourcing and applying external knowledge in the context of text-to-SQL conversion require extensive exploration.

In summary, the BIRD-Bench dataset not only underscores the critical role of external knowledge in text-to-SQL tasks but also opens up avenues for research in improving the autonomy and practical applicability of these models. As the field evolves, exploring these aspects will be crucial in bridging the gap between theoretical capabilities of models and their practical, real-world utility.

5.1.7 LLMs Capabilities in Classifying Data Quality Issues

Due to the findings made in the previous experimental phase of the importance of data quality and its impact on text-to-SQL generation, a new experimental phase was initiated. The

aim of these experiments was to evaluate the ability of LLMs to acknowledge poor data quality and the accuracy of how this could be classified. This is highly relevant for the use case of this project.

If an LLM could accurately classify questions with poor quality, such questions could be identified and stopped before they are inputted into a text-to-SQL model. This would remove the need for a text-to-SQL model to be robust towards noise. After a poor question is identified the user could be prompted to create a better question, therefore guiding the user in posing better questions, thus increasing the performance of the text-to-SQL model. The ability of classification models to accurately identify incorrect questions is of high importance, especially considering the demonstrated effect of noisy data on text-to-SQL models. Thus, a heightened sensitivity of the model, signified by superior recall performance, is essential.

The methodology of this experimental phase is described in Section *3.6 Question Classification*, Beginning with experiments in binary classification, followed by an extension to four-category classification. The performance results of the presented models can be weighted towards the baseline model, classifying all questions as class 0, presented in Section *4.5.2 Base Classifier*, achieving an accuracy of 60.3%.

Ability to Classify Two Categories

The comparison of two-category classification models, as presented in Section *4.5.2 Classification - Two Categories* and illustrated in Figures 4.5 and 4.6, demonstrates distinct performance characteristics across accuracy, precision, recall, and F1-Score metrics. Notably, the GPT-4 model, when provided with few-shot learning examples of both class 0 and 1, outperforms all other experiments with GPT-3.5 and GPT-4 in three of the metrics with an accuracy of 80%, precision of 75%, and F1-Score of 70%.

In contrast, the GPT-3.5 model, when prompted with few-shot learning examples of class 0, excels in recall with an 81% score, indicating, as described in Section *2.5.1 Classification Metrics*, its strength in identifying actual positive cases, meaning incorrect questions. This heightened ability to more accurately identify incorrect instances of class 1 could possibly be attributed to it being instructed on "correct examples" representing class 0. By gaining a clear understanding of the traits that define "correct examples", the model is better equipped to identify patterns that diverge from these examples. Furthermore, it is worth noting that the model with the same prompt template using GPT-4, even though significantly higher performing than GPT-3.5 when compared and evaluated in different benchmarks [26, 28, 27, 30, 31], is performing significantly worse in terms of recall, at 51%. This could be an indication that the GPT-3.5 model is more actively using the few-shot learning examples provided when executing its task than GPT-4. Additionally, it suggests that GPT-3.5 more effectively uses examples of a correct pattern to classify and distinguish incorrect patterns than GPT-4. This attribute, a higher recall, is essential in the current use case of the report, where the aim of the classifiers is to distinguish vague questions that could affect a text-to-SQL model and implicate reduced accuracy in SQL generation. Therefore, the GPT-3.5 model with examples of class 0 would be the preferable model for this particular use case.

However, in other use cases where identifying incorrect data points would not be of the highest priority, the most prominent classification model would most likely be GPT-4 with few-shot learning examples of class 0 and 1, providing a more balanced classifier with higher accuracy, precision and F1-score.

The construction of prompt templates is a critical factor in influencing model performance. Upon examination of the prompt templates from the project repository[1], the variations between the models' performances are attributed to the nature of the examples included in the prompts—whether they are examples of class 0, incorrect questions, class 1, correct questions, or both class 0 and 1. The inclusion of correct examples appears to enhance the

[1] https://github.com/niklaswretblad/Text-to-SQL-Generation

GPT-3.5 model's sensitivity to correctly identifying positive instances, as illustrated in Figure 4.5. However when provided a mix of class 0 and 1 examples its performance is reaching a mediocre level in between the models with distinctive examples.

On the contrary, when the GPT-4 model is provided with a combination of both correct and incorrect examples, it achieves a more balanced performance across all four evaluation metrics—accuracy, precision, recall, and F1 score. This suggests that while GPT-3.5 can effectively process distinctive information in the prompts, its performance becomes more average without any notable strengths when faced with a mix of good and bad examples. This observation underscores the significance of prompt design, indicating that different LLMs may derive benefits from diverse prompt structures.

Ability to Classify Four Categories

The subsequent experiments focused on the four-category classification, as depicted in Figures 4.7 through 4.11. An examination of Figure 4.7, which illustrates the weighted averages for accuracy, precision, recall, and F-1 score of the GPT-3.5 Turbo and GPT-4 classifiers, reveals minor variations in their performances. The GPT-3.5 model slightly outperforms in precision, with a score of 51% compared to GPT-4's 50%, and in recall, scoring 59% over GPT-4's 57%. On the other hand, the GPT-4 model achieves marginally better accuracy, at 71%, compared to 69% for the GPT-3.5 Turbo. These figures suggest subtle differences between the two models without indicating significant disparities. Notably, despite that GPT-4 is outperforming GPT-3.5 in several benchmarks [26, 28, 27, 30, 31], GPT-4's performance is comparable to that of the GPT-3.5 Turbo, showing minimal or no variation when responding to the same prompt template.

When further examining the bar plots in Figure 4.8 and 4.10, and the heatmaps of each model's specific classification results in Figure 4.9 and 4.11, it is evident that the GPT-3.5-Turbo model has a higher variation in its distribution of correct predictions, it predicts 55 questions correctly into category 0, 7 questions correctly into category 1 and 1 question correctly into category 2. Whereas the GPT-4 model predicts 62 questions correctly into category 0, 2 questions correctly into category 1, and 1 question correctly into category 3. This shows that the GPT-3.5-Turbo model and GPT-4 model does not differentiate distinctively, but looking at category 1 of both the heatmaps it shows that the GPT-3.5-Turbo model is more accurate in the prediction of this particular category, whereas GPT-4 is more accurately classifying questions of category 0. This is an interesting observation considering that the models have been given the same prompts. Due to the black-box nature of both these models, it is difficult to conclude the reason for this. The transparency of the model architectures will be discussed further in the Section *5.3.3 Implications of OpenAI's Shift from Open Source to Closed Source*.

In summarizing the outcomes of the four-category classification experiments, the variations between the GPT-3.5-Turbo and GPT-4 models are marginal, despite p the significant difference in their training parameters. These subtle distinctions reveal that the GPT-3.5 model excels in precision and recall, whereas the GPT-4 model demonstrates greater accuracy in its predictions. Furthermore, the GPT-4 model is more adept at correctly classifying questions in category 0, whereas the GPT-3.5 model shows heightened accuracy in predicting category 1.

Comparing Two Class Against Four Class Classification

When weighing the two different experiments of two-category classifications against four-category classifications, it shows that the GPT-4 model is the more balanced model, when provided with few-shot learning examples of class 0 and 1, in classifying two categories. This is in comparison to its GPT-3.5 representative that, when given examples of class 0, achieves the highest recall of all two-category classifiers. On the contrary, when classifying four categories the models differ only slightly. When comparing accuracy, precision, and recall in

two-category classifications to the weighted averages of the same metrics in four-category classifications of the GPT-4 model it is evident that the model is more efficient in categorizing two options rather than four options. Overall, when wanting the highest performance in terms of accuracy, precision, and F-1 score, the two-category classifier of class 0 and 1 using GPT-4 would be the best alternative, whereas the two-category classifier with examples of class 0 using GPT-3.5 would be the option if one would want a high recall, which is preferred in the project's use case. Using a four-category classification results in a more detailed classification which could give richer insights and analysis, which could be needed in certain use cases, of what attributes the questions possess. However, as shown by the experiments, this is done at the cost of a lower performance

Reasoning Classifier

The reasoning classifier is based on chain-of-thought reasoning, a method where the LLM is prompted to contemplate and reason about the appropriate classification for a given question. Subsequently, the insights from this contemplation are used in a secondary phase where the LLM conclusively classifies the question. This methodology aims to make use of the enhanced reasoning capabilities of LLMs, as suggested by previous research [1].

However, our experimental findings revealed that the Reasoning Classifier did not perform as well as the baseline classifier. This result is somewhat unexpected, given the advanced reasoning skills attributed to the latest LLMs. Several underlying reasons might explain this discrepancy. Firstly, the two-step process of first reasoning and then classifying introduces an element of complexity. Here, the LLM is tasked not just with understanding and classifying data quality issues, but also with articulating its reasoning process in a clear and coherent manner. This additional complexity can increase the likelihood of errors or misinterpretations during the reasoning step, which could then impact the final classification outcome.

Secondly, the inherent limitations of GPT-3.5-Turbo and GPT-4 also play a role. While these models are proficient in a broad spectrum of tasks, their performance can fluctuate based on the specific nature and intricacies of the task. Additionally, the reasoning process employed by the LLM might introduce ambiguities or subjective interpretations. These may not always align seamlessly with the expected classification scheme, potentially leading to inaccuracies in the final classification. This is particularly relevant in instances where the reasoning process is subject to multiple interpretations.

5.1.8 LLMs Capabilities in Correcting Data Quality Issues

The results of the small study provide encouraging evidence regarding the capabilities of LLMs in addressing data quality issues through the correction of faulty SQL queries. The study observed a great success rate, with the LLM accurately correcting 10 out of 10 identified faulty queries. The high level of accuracy indicates that LLMs hold the potential for enhancing the quality of text-to-SQL datasets by identifying and correcting incorrect SQL queries.

Despite these promising results, there remains an important area to explore: the LLM's behavior with already correct SQL queries. It is essential to investigate whether the model tends to over-correct, potentially transforming correct queries into faulty ones. This aspect is crucial for practical applications, as indiscriminate corrections could lead to data integrity issues or noise in the data, defeating the purpose of using an LLM for data quality enhancement. We leave this for future work.

Further work is also needed to understand the limitations and potential biases of LLMs in this context. For instance, how does the model perform with different types of databases or more complex query structures? The ablation study only investigated 10 different errors and SQL queries, and while the LLM succeeded in correcting all of those this might not mean

that it can correct every other error that might exist in a SQL query. Are there other types of errors or database schemas that the LLM struggles with? Additionally, exploring the LLM's performance in a real-world scenario, where queries and database structures are more varied and complex, would provide valuable insights into its practical applicability and limitations.

In conclusion, the findings from this ablation study are promising and indicate that LLMs, like GPT-4, possess the potential to improve the quality of text-to-SQL datasets. However, further research is necessary to thoroughly understand the capabilities and constraints of these models across a broader range of queries, domains, and error types. This in-depth understanding is a prerequisite for their practical application in correcting datasets.

5.2 Method

This section will analyze the methodology of the project and review critically how the project was carried out from the research process to the experimental process.

5.2.1 Methodological Considerations

In the process of correcting the financial domain after the first iteration of experiments, a comprehensive approach to correct and annotate data points was undertaken, ensuring the dataset's relevance and accuracy for text-to-SQL models. However, during this phase, a potential source of bias was identified. The correction of data points was done next to the generated SQL queries from the DIN-SQL model experiment, which might have unintentionally influenced the correctors decision-making. This setup could have led to cognitive biases, subtly aligning the correctors perception and correction of the data with the predictions of the model, potentially introducing confirmation bias. Since the correction was done next to the DIN-SQL model outputs, intuitively, the effect of this bias would be that the DIN-SQL model would perform better, since the data points were corrected with the model's outputs next to it.

Even though the effect of the bias when running the models again on the cleaned dataset would be that the DIN-SQL model would perform better, it's worth noting that the zero-shot model still closed the performance gap between itself and the DIN-SQL model as can be seen in Section *4.4 Experiments on the Cleaned Datasets*. This observation is significant as it suggests that despite the potential bias introduced during data correction, the zero-shot model demonstrated a robust improvement in performance. However, the presence of bias cannot be overlooked, as it might have affected the results to some extent.

Future research should consider more strict methods for data annotation and correction to minimize the influence of such biases. This could include the separation of data correction from model output views or employing more sophisticated methods to ensure unbiased data processing. The goal is to achieve a balance where data quality is enhanced without inadvertently favoring any particular model or approach.

5.2.2 Replicability

To ensure the replicability of this study, the methodology detailed in the *3 Method* chapter should be followed. The implementation code, including the evaluation pipeline, the prompts, and the API connections to the LLM, is available on the project's GitHub page[2]. Additionally, the BIRD-Bench dataset utilized in this project can be found in the project's GitHub repository and is also accessible via the official BIRD-Bench website[3].

[2]https://github.com/niklaswretblad/Text-to-SQL-Generation
[3]https://bird-bench.github.io/

5.2.3 Validity

This section will disclose the factors that could have affected the validity of the study. Firstly, the impact of GPT's updates will be discussed, thereafter the references used in this study will be discussed from a critical viewpoint.

Impact of GPT's Updates

As highlighted in the reliability section of this thesis, Section *5.2.4 Reliability*, the GPT models are subject to continuous updates and alignments by OpenAI, often without explicit notifications to end-users. This characteristic of the GPT models, while ensuring progressive enhancement and alignment with ethical standards, introduces an element of unpredictability in the model's behavior over time.

The evolving nature of these models means they might not only improve in terms of capabilities and performance but also alter how they process and interpret the input data. Such modifications, although beneficial for the model's effectiveness and ethical alignment, add a layer of complexity to the validity aspect of studies involving GPT. Specifically, the same input processed by a Language Model at different time intervals might result in varied outputs. This variation poses a significant challenge for researchers, by complicating the interpretation of model results over time.

Because of this, the experiments conducted for this thesis were meticulously planned to minimize the potential impact of these evolving GPT models. To ensure consistency and to mitigate the effect of any potential model updates, all experiments within each iteration and category were conducted on the same day.

For instance, the initial set of text-to-SQL model experiments was conducted in a single day. This approach was replicated for the subsequent experiments on the cleaned versions of the dataset and for the different classification experiments. Conducting the experiments in this manner ensured that each set of results was obtained from a model in a consistent state, thereby reducing the likelihood of variances due to model updates.

This methodology, while not completely eliminating the possibility of model drift affecting the results, significantly minimized its impact. By structuring the experiments in this manner, the study aimed to ensure that the findings and conclusions drawn are as reliable, valid and replicable as possible, given the constraints posed by the dynamic nature of GPT models.

Source Criticism

In conducting this study, we primarily relied on peer-reviewed articles and papers, which are widely regarded as credible and reliable sources in academic research. Peer-reviewed sources undergo careful examination by experts in the field, ensuring that the research methodology, data analysis, and conclusions meet high standards of academic rigor and integrity. This vetting process significantly reduces the risk of including inaccurate or biased information, thus lending considerable weight to our research findings and discussions.

However, alongside these peer-reviewed sources, we also incorporated recent papers published on platforms like Arxiv[4]. These papers, while not yet subjected to the formal peer-review process, were included due to their relevance, timeliness, and potential contributions to the field of text-to-SQL generation. Arxiv, and similar repositories, play a pivotal role in disseminating cutting-edge research rapidly, offering access to the latest findings and theoretical advancements.

The inclusion of non-peer-reviewed sources, however, necessitates a careful approach to source criticism. The absence of peer review means that these documents have not been evaluated for their scientific accuracy or validity by independent experts. Consequently, there

[4]https://arxiv.org/

might be a higher risk of encountering unverified claims, methodological flaws, or biased interpretations in these papers. This is not to diminish the potential value of these sources but rather to acknowledge the different levels of examination they have undergone compared to peer-reviewed literature.

To mitigate the risks associated with using non-peer-reviewed sources, we employed several strategies:

Critical Evaluation: Each non-peer-reviewed paper was critically evaluated for its methodology, data analysis, and conclusions. We paid special attention to the soundness of the research design and the robustness of the arguments presented.

Cross-Verification: Wherever possible, findings and theories presented in non-peer-reviewed papers were cross-verified with peer-reviewed literature. This helped in ensuring that the conclusions drawn were consistent with the established body of knowledge.

Expert Consultation: In some cases, we consulted with domain experts to gain insights into the validity and relevance of the non-peer-reviewed papers.

In conclusion, while the inclusion of non-peer-reviewed sources from platforms like Arxiv introduced challenges in terms of source credibility, these sources provided valuable insights into the latest trends and unexplored areas in the field of text-to-SQL generation. Their inclusion was carefully balanced with a rigorous approach to source criticism, ensuring the overall integrity and credibility of our research.

5.2.4 Reliability

Within the scope of our study, there are specific challenges concerning the reliability of our results. These challenges stem primarily from the inherent nature of the LLMs and their operational frameworks.

One of the fundamental issues is the inability to set a seed for the LLMs. In computational experiments, setting a seed is a common practice to ensure that results are reproducible, as it allows the random number generator to produce the same sequence of numbers across different runs, thereby ensuring consistency in outcomes. However, with the current state of LLMs provided by OpenAI, this level of control is not available. Consequently, achieving perfect reliability of results becomes difficult, as each run of the model might yield slightly different outcomes due to variations in the underlying stochastic processes.

Moreover, these models are subject to continuous updates and alignments by OpenAI, often without explicit notifications to the end-users [34]. This evolving nature means that the models might change over time, not just in terms of their capabilities and performance but also in how they process and interpret input data. These changes, while potentially improving the model's overall effectiveness and alignment with ethical guidelines, further compound the challenge of reliability. As researchers, we might find that the same input processed by an LLM at different time points yields different outputs, making it challenging to replicate studies precisely.

Despite these challenges, it is important to acknowledge the significance of these LLMs. They are currently among the best models available for a wide range of natural language processing tasks, including text-to-SQL conversion. Their widespread use across industries and academia attests to their capabilities and the value they add in various applications. In our research, while acknowledging the limitations in reliability, we still aimed to measure the performance of these models on text-to-SQL tasks. The research conducted in this thesis, therefore, provides valuable insights into the capabilities of these models at a specific point in time, under specific conditions.

In conclusion, while perfect reliability in experiments involving LLMs such as those provided by OpenAI is challenging, the insights gained from such research are still valuable. They contribute significantly to our understanding of the capabilities and limitations of current LLMs in specific applications like text-to-SQL tasks. Future research in this area should continue to explore these models, while remaining cognizant of and transparent about the challenges related to reliability.

5.3 The Work in a Wider Context

In this section, the broader implications, relevance, and potential applications of the research will be explored, situating it within the larger academic and societal landscape.

5.3.1 Democratizing Data Access

The evolution of text-to-SQL models represents a central step towards democratizing data access. This democratization is rooted in the transition from traditional data querying methods, which necessitate specialized knowledge and technical expertise, to more intuitive, natural language-based interactions.

Traditionally, extracting insights from databases required a substantial understanding of programming languages, or more specifically the SQL language. This created a significant barrier to entry, restricting data analysis and insight generation to a limited group of experts. However, with the implementation of text-to-SQL models, this barrier is being removed. These models enable users to interact with databases through natural language queries, effectively translating everyday language into complex SQL queries. This advancement significantly lowers the threshold for engaging with data, allowing individuals without formal training in computer science or data analysis to explore and use database resources. And so the ability to query data using natural language extends the power of data analysis to a broader audience. It democratizes data access by making it more inclusive, as individuals from various professional backgrounds, educational levels, and even non-technical fields can now capitalize on the power of data. This inclusivity not only promotes a diverse range of perspectives in data analysis but also encourages a more data-informed decision-making process across different sectors. In the public sector, it could lead to more informed policy-making, as officials can directly interact with data to understand trends and public needs. In the private sector, businesses of all sizes could leverage data to drive innovation and competitiveness, without the need for extensive investment in technical training or resources. Furthermore, in the educational realm, it could provide an invaluable tool for students and researchers, enabling data-driven learning and research. This democratization of data access has profound implications for society and various industries. As text-to-SQL technology continues to advance, its potential to further democratize data access is huge. Future developments could see even more sophisticated and user-friendly interfaces, making data querying as simple as having a conversation. This progression will not only enhance the accessibility of data but also has the potential to transform how we interact with and perceive data in our daily lives.

On the other hand, the rise of text-to-SQL systems and LLMs in data analysis point to a shift in the employment landscape, particularly in roles traditionally centered around data management and analysis. These technological advancements automate tasks that previously required human intervention, leading to a profound shift in the nature of certain job functions.

The automation brought about by these systems can be seen as a double-edged sword. On one hand, it significantly increases efficiency, reduces the potential for human error, and allows for the handling of larger datasets than would be feasible manually. This automation enables businesses and organizations to process and interpret data at unprecedented scales, leading to more informed decisions and strategies.

On the other hand, there are implications for the workforce that currently specializes in these tasks. As machines take over functions like basic data querying and analysis, there is a potential reduction in the demand for roles that primarily focus on these skills. This shift necessitates a reevaluation of the skill sets valued in the workforce. It underscores the importance of adaptability and continuous learning, encouraging professionals to diversify their skills beyond the tasks that can be automated.

Moreover, this shift can lead to the creation of new roles and opportunities. As the barriers to data access lower, there is a growing need for professionals who can interpret and utilize data in innovative ways, transcending traditional data analysis. Roles may evolve to focus more on strategic decision-making, data interpretation in context, and the ethical implications of data use.

In essence, the democratization of data analysis through text-to-SQL systems and LLMs not only broadens access to data but also reshapes the professional landscape. It challenges existing paradigms of work and necessitates a forward-looking approach to education and professional development, emphasizing the integration of technical skills with critical thinking and ethical considerations in data use.

5.3.2 Bias

When conducting research there is always a risk of introducing bias in the methodology. In this study, there is also a risk that bias has already been introduced in the LLMs used for the text-to-SQL models. Because of OpenAI's limiting transparency of training data used to train the LLM models, discussed in Section *5.3.3 Implications of OpenAI's Shift from Open Source to Closed Source*, unknown biases within the training data could affect the results of the study.

Due to the GPT models of OpenAI being trained on an immense amount of data parameters, GPT-3.5 being trained on 175 billion parameters [24], and GPT-4 presumably being trained on even more parameters due to its increased performance [31, 30, 26, 27]. Open source datasets and most likely various methods of text mining such as web-scraping[5] have been used to collect the large amount of data. These biases, inherent in the training data, can influence the model's outputs. For example, if there is more internet data on sports, particularly football, compared to finance, the LLM might perform better in responding to queries about football data and databases than those related to finance.

Additionally, there's a possibility that some datasets and databases included in BIRD-Bench might have been part of the training data for GPT models, while others were not. This disparity could significantly impact the models' text-to-SQL generation capabilities across different domains. During experimental observations, it was discovered that the GPT-4 model, accessed through the ChatGPT interface, possessed knowledge about the BERKA dataset without any input from researchers. This knowledge encompassed the dataset's origin, background, and database structure. The BERKA dataset, as mentioned in Section *3.2.1 The Financial Domain in BIRD-Bench* , is the foundation upon which the financial domain on the BIRD-Bench dataset has been built. Consequently, it appears that the LLM displays a bias towards the financial domain of BIRD-Bench, likely resulting in more accurate text-to-SQL conversions in this area, especially compared to domains lacking information on the BERKA dataset.

5.3.3 Implications of OpenAI's Shift from Open Source to Closed Source

Open AI was initially founded as a non-profit organization with an open-source ethos, aiming to make AI widely accessible to the public, as highlighted in their initial blog post [78]. This approach was instrumental in accelerating advancements in various AI fields. However, with substantial investments from entities like Microsoft and a focus shift towards proprietary technologies, OpenAI has become more selective and controlled in its research output

[5] A technique used to extract large amounts of data from websites automatically.

[79]. Critical information, especially concerning the architectures of GPT-3 and GPT-4, has been closely guarded. This shift raises substantial concerns about the transparency of these influential AI models.

The lack of detailed disclosure about these models by OpenAI creates significant barriers to the AI research community. Open-source models are central in enabling collaborative progress, allowing for diverse innovation and independent verification of technologies. In contrast, the closed-source approach hinders the broader AI community's ability to understand, critique, and build upon these developments. Moreover, OpenAI's pivot highlights a critical tension in the technology sector: the conflict between communal knowledge advancement and the protection of intellectual property for commercial gain. While the pursuit of profit can drive rapid development and deployment, it may also restrict the collective growth of knowledge. This limitation is not just a matter of academic interest; it has practical implications for the development of AI technologies that are accessible and beneficial to a broader audience.

The consequences of this change extend beyond immediate research concerns. OpenAI's move towards a proprietary stance could set a precedent in the AI field, influencing the policies of other organizations regarding knowledge sharing and transparency. This potential trend towards a more closed and less collaborative AI research landscape could slow the pace of innovation and ethical considerations in AI development, affecting fields reliant on advancements in LLMs, like text-to-SQL conversion.

In sum, OpenAI's move towards reduced algorithmic transparency marks a significant turn in the AI development landscape. It highlights the ongoing struggle between open-source ideals and proprietary business models. This development necessitates a reexamination of how AI research and development are conducted, shared, and used, highlighting the need for a balanced approach that nurtures innovation while encouraging collaboration and ethical practices.

5.3.4 Data Privacy for Closed Source LLMs

The use of closed-source LLMs like those developed by OpenAI brings to attention critical data privacy concerns. Given the restricted nature of these models, there's a significant lack of transparency in how data inputted into them is handled and used.

When sensitive or confidential data is input into these models, there is a loss of control over its subsequent use. This presents a risk, as the companies that own and develop these LLMs may gain access to and control over this confidential information. Ethically, it becomes essential for researchers and developers to ensure the privacy of data subjects, especially in cases lacking explicit consent.

Legally, navigating regulations such as the GDPR is challenging with closed-source LLMs, demanding caution to avoid violations. Public trust is also impacted by how data is managed; mishandling can lead to a loss of confidence in AI technologies.

For academia, these privacy issues challenge the integrity of research, necessitating the responsible handling of data used in these models.

In conclusion, the capabilities of closed-source LLMs must be balanced with stringent data privacy considerations. Users need to be vigilant, ensuring ethical and legal compliance, and maintain transparency with data subjects. This balance is essential for the responsible use and advancement of AI technologies.

5.3.5 Costs

In discussing the broader context of our work with LLMs, it's crucial to consider the economic implications, particularly the costs associated with accessing LLMs through APIs. This aspect plays a significant role in determining the suitability and scalability of using such models in various settings, from academic research to industrial applications.

LLM APIs, such as those provided by OpenAI, typically operate on a pay-per-use model. This pricing structure means that the cost is directly proportional to the usage, calculated based on factors like the number of tokens processed, the complexity of requests, and the computing resources consumed. For large-scale research projects or commercial applications that require extensive data processing, these costs can accumulate quickly, potentially posing financial challenges, especially for smaller organizations or academic institutions with limited budgets.

The cost factor not only influences the accessibility of these technologies but also has broader implications for innovation and research diversity. High API costs may limit the ability of smaller teams or researchers from less well off institutions to engage in cutting-edge AI research. This could lead to a concentration of AI advancements in well-funded organizations, potentially narrowing the diversity of research perspectives and applications explored.

Moreover, the reliance on external APIs for accessing LLMs introduces another layer of dependency. Researchers and developers are subject to the terms and conditions set by the API providers, including pricing, data privacy policies, and usage limitations. This dependency can impact long-term research planning and operational stability, as changes in API terms or pricing can significantly affect ongoing projects.

In the context of our research, the cost implications of using LLM APIs were a critical consideration. While the models used in this research offer unparalleled capabilities in handling text-to-SQL tasks, the financial aspect required careful budgeting and resource allocation. It also necessitated a strategic approach to API usage, optimizing our queries to balance cost and performance.

Looking ahead, as LLMs continue to evolve and become more integral to various domains, the AI community must explore sustainable economic models. This could include the development of more cost-effective API structures, alternative funding mechanisms for research, or even open-source initiatives that provide more accessible alternatives to proprietary models. Such developments would not only alleviate the financial burden on researchers and developers but also democratize access to advanced AI technologies, encouraging a more inclusive and diverse landscape for AI innovation.

6 Conclusion

6.1 Research Questions

RQ1: What is the comparative performance, in terms of accuracy, between a baseline text-to-SQL model and the DIN-SQL model when applied to financial data in the BIRD-Bench benchmark?

The empirical results indicate a marginal performance difference between models, with the DIN-SQL model slightly outperforming the baseline zero-shot model with 40.57% over 36.19% accuracy. These results, coupled with the minor improvement observed in transitioning from GPT-3.5 to GPT-4, suggests overall limitations of models within this specific domain. Moreover, the general effectiveness of these models seem to be irrespective of domain-specific complexities such as question difficulty or the size of the database. This implies the need of further research on other aspects such as database structure, data quality of questions and SQL-queries within the dataset and other reasons as to why models are failing.

RQ2: How does the overall quality of data influence the performance of text-to-SQL models when applied to financial data in the BIRD-Bench benchmark?

The empirical findings from our experiments provide a clear indication of the profound impact that data quality has on the performance of text-to-SQL models. Initially, the DIN-SQL model demonstrated a performance edge over the zero-shot baseline by approximately 5-6% on the original dataset. However, this performance gap narrowed to about 0.5% after correcting errors in the dataset, thereby illustrating the significant influence of data quality on model performance.

In our nuanced analysis, we also observed an unexpected reversal in model performance rankings when using a dataset comprising original questions paired with corrected SQL queries. The zero-shot model outperformed the DIN-SQL model, achieving a 42% accuracy rate compared to 41% by DIN-SQL. This outcome is particularly telling, suggesting that the DIN-SQL model might be overfitting to noise in the data.

These experiments underscore the critical role of data quality in the functionality of text-to-SQL models. The presence of noise—be it ambiguities, inaccuracies, or complexities in the natural language questions, or errors in the SQL queries—significantly impacts the accuracy and reliability of these models. Correcting such noise not only enhances overall performance but also reveals potential issues of overfitting in more sophisticated models like DIN-SQL. This study reveals that without precise and well-annotated datasets, assessing the true accuracy of text-to-SQL models becomes challenging. New datasets are needed for a clear and reliable assessment of both the accuracy in SQL query generation and the robustness against noise within these models.

RQ3: How do different types of noise and data quality issues impact the performance of text-to-SQL models?

In addressing how different types of noise and data quality issues impact the performance of text-to-SQL models, our investigation has revealed several key findings. The performance of text-to-SQL models is significantly influenced by various types of noise and data quality issues, which we have categorized and examined in detail.

Firstly, string capitalization issues pose a notable challenge. As observed in the BIRD-Bench dataset, inconsistencies in capitalization between the query and database can lead to erroneous or failed SQL queries. This sensitivity to string capitalization highlights the need for text-to-SQL models to incorporate mechanisms that align with the case sensitivity of databases, potentially through fuzzy string matching or standardization techniques.

Grammatical errors and poor language quality also detrimentally affect the model's performance. While humans can often interpret poorly structured or grammatically incorrect queries, text-to-SQL models struggle with such inputs. This discrepancy underscores the importance of refining these models to better handle linguistic variations while also emphasizing the need for clear and well-structured input queries.

Language ambiguities present another significant challenge. Ambiguous queries, such as those lacking clarity on the specific data to be retrieved, can lead to incorrect SQL conversions. Addressing this issue calls for sophisticated mechanisms within text-to-SQL models to resolve or seek clarification on ambiguous inputs.

Furthermore, our analysis revealed additional noise types, such as semantic errors, vague or ambiguous questions, incorrect SQL queries, divergences between questions and their corresponding SQL queries, and synonym confusion. Each type of noise presents unique challenges in accurately translating natural language queries into SQL, thus impacting the model's effectiveness.

Our study's findings suggest that the presence of noise and data quality issues in the dataset can severely hinder the performance of text-to-SQL models. In the case of the BIRD-Bench dataset, these challenges were prevalent across various domains, indicating a widespread issue. To address these, we propose the development of more robust models capable of handling such complexities and the enhancement of datasets to better reflect the nuances of real-world queries.

In conclusion, the impact of different types of noise and data quality issues on text-to-SQL models is profound, necessitating focused efforts in model development and dataset enhancement. As we move towards more sophisticated and practical applications of text-to-SQL technologies, addressing these challenges will be crucial in ensuring their accuracy, efficiency, and applicability in diverse domains.

RQ4: To what extent are large language models capable of accurately identifying and categorizing data quality issues within a text-to-SQL benchmark?

The findings, which emphasize the substantial impact of data noise on results, highlight the importance of precise classifiers in identifying these problematic data points to enhance the performance of text-to-SQL models. When examined, the experiments have shown that the performance of classifiers improves when categorizing into two categories rather than four. In two-category classification, the GPT-3.5 model, trained with examples of correct data (class 0), excelled in recall, indicating its effectiveness when provided with one-sided examples. Conversely, the GPT-4 model, trained with a balanced set of examples from both classes (0 and 1), achieved the highest accuracy, precision, and F1-score. This suggests that GPT-4 performs optimally with balanced inputs, while GPT-3.5 is more attuned to specific, singular-patterned training. For this project's purpose, which is to fully leverage the capabilities of

text-to-SQL models, classifiers that are both strict and sensitive are favored. Consequently, the GPT-3.5 model, trained exclusively with class 0 examples and achieving a recall performance of 81%, emerged as the most effective model among those tested in the experiments.

RQ5: To what extent can large language models accurately correct data quality issues within a text-to-SQL benchmark?

Our study showed a 100% success rate in correcting ten different faulty SQL queries, providing encouraging evidence that LLMs could be used to enhance the quality of text-to-SQL datasets. However, more research is needed before LLMs can be applied practically for the task. Key in investigating is the LLM's response to already correct SQL queries, where there is a risk of over-correction, possibly leading to further data quality issues.

Additionally, the study's limited scope, focusing on a specific set of errors and queries, necessitates broader research. Future work should examine LLMs' performance across various databases, complex query structures, and a wider range of errors, to assess their practical applicability in diverse real-world scenarios.

In summary, while LLMs show promise in improving text-to-SQL data quality, a deeper understanding of their capabilities and limitations in more complex settings is essential for their practical application.

6.2 Future Work

This thesis has provided significant insights into the performance and limitations of text-to-SQL models. While progress have been made in understanding their capabilities, several areas require further research to enhance their accuracy and applicability. The future work stemming from this study can be broadly categorized into the following areas:

Development of Noise-Specific Benchmarks

One of the most critical findings of this research is the substantial impact of noise on model performance. Current datasets do not adequately measure a model's robustness against various noise types or benchmark solutions designed to enhance this robustness. There is an urgent need to develop new datasets specifically tailored to evaluate model performance across different noise types. Such datasets would not only help in benchmarking existing models but also aid in developing new models that are inherently more robust to noise.

Enhanced Annotation in BIRD-Bench

The BIRD-Bench benchmark has proven invaluable in assessing text-to-SQL models. However, there is a clear need for further annotation within this dataset. Specific annotations for different types of noise, SQL difficulties, and potential pitfalls would significantly benefit future research. These enhanced annotations can provide deeper insights into the challenges faced by text-to-SQL models and guide the development of more sophisticated solutions.

Targeted Research on Noise Types and Subproblems

Our study identified various types of noise and subproblems that affect the performance of text-to-SQL models. Once the datasets reflecting these noise types are created, future work should focus on developing solutions to these specific challenges. This could involve designing algorithms that are more resilient to certain noise types or developing preprocessing tools that can mitigate the impact of these noises before they affect the model's performance.

Guardrails for Model Inputs

Another promising avenue for future research is the development of guardrails for model inputs. By classifying and identifying noisy questions, it might be possible to prompt users to refine their queries. This approach could not only improve the quality of inputs fed into text-to-SQL models but also help in educating users about the best practices for framing their queries, thereby enhancing the overall system's efficiency.

The Need for External Knowledge

This thesis highlights the role of external knowledge in enhancing text-to-SQL conversion, particularly evident by the design of the BIRD-Bench dataset. The dataset's inclusion of hints, offering essential context for ambiguous elements, significantly improves model accuracy. For instance, understanding specific column meanings in the financial domain is crucial for formulating correct SQL queries. However, this reliance on external information poses challenges in real-world applications, where such hints may not be readily available. Future work should investigate methods for LLMs to autonomously acquire or infer this necessary external knowledge, overcoming limitations in current models and documentation. This exploration is essential for advancing the practical application of text-to-SQL technologies in complex and dynamic real world environments.

Advancement in Text-to-SQL Model Development

Lastly, there is a continuous need for the development of better text-to-SQL models. This includes models that are not only more accurate but also more versatile in handling various types of queries and robust against different forms of noise. The insights gained from this research can guide the development of next-generation text-to-SQL models that are tailored for specific domains like finance, healthcare, or e-commerce.

In conclusion, the path forward involves a multifaceted approach encompassing the creation of specialized datasets, enhanced annotations, focused problem-solving on identified noise types, development of user-input guardrails, and the continuous evolution of text-to-SQL models. As we advance in these areas, we anticipate significant improvements in the functionality and applicability of text-to-SQL technologies, paving the way for their widespread adoption in various industries.

Bibliography

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 24824–24837. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

[2] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. *Toolformer: Language Models Can Teach Themselves to Use Tools*. 2023. arXiv: 2302.04761 [cs.CL].

[3] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 9459–9474. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

[4] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. *Tool Learning with Foundation Models*. 2023. arXiv: 2304.08354 [cs.CL].

[5] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. *ToolQA: A Dataset for LLM Question Answering with External Tools*. 2023. arXiv: 2306.13304 [cs.CL].

[6] Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. "Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs". In: *Advances in Neural Information Processing Systems*. Spotlight Poster. 2023. arXiv: 2305.03111 [cs.CL]. URL: https://neurips.cc/virtual/2023/poster/73529.

[7] Donald D. Chamberlin. "Early History of SQL". In: *IEEE Annals of the History of Computing* 34.4 (2012), pp. 78–82. DOI: 10.1109/MAHC.2012.61.

[8] Solid IT DBMS Consulting. *DB-Engines Ranking - popularity ranking of database management systems*. https://db-engines.com/en/ranking. [Accessed 2023-09-18]. 2023.

[9] Jinyang Li and Binyuan Hui and Ge Qu and Binhua Li and Jiaxi Yang and Bowen Li and Bailin Wang and Bowen Qin and Rongyu Cao and Ruiying Geng and Nan Huo and Xuanhe Zhou and Chenhao Ma and Guoliang Li and Kevin C. C. Chang and Fei Huang and Reynold Cheng and Yongbin Li. *BIRD-SQL*. https://bird-bench.github.io/. [Accessed on 2023-11-15].

[10] Yu, Tao and Zhang, Rui and Yang, Kai and Yasunaga, Michihiro and Wang, Dongxu and Li, Zifan and Ma, James and Li, Irene and Yao, Qingning and Roman, Shanelle and others. *Yale Semantic Parsing and Text-to-SQL Challenge*. https://yale-lily.github.io/spider. [Accessed on 2023-11-16].

[11] E. F. Codd. "A Relational Model of Data for Large Shared Data Banks". In: *Commun. ACM* 26.1 (Jan. 1983), pp. 64–69. ISSN: 0001-0782. DOI: 10.1145/357980.358007. URL: https://doi.org/10.1145/357980.358007.

[12] Donald D. Chamberlin and Raymond F. Boyce. "SEQUEL: A Structured English Query Language". In: *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*. SIGFIDET '74. Ann Arbor, Michigan: Association for Computing Machinery, 1974, pp. 249–264. ISBN: 9781450374156. DOI: 10.1145/800296.811515. URL: https://doi.org/10.1145/800296.811515.

[13] Julien Delplanque, Anne Etien, Nicolas Anquetil, and Olivier Auverlot. "Relational Database Schema Evolution: An Industrial Case Study". In: *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2018, pp. 635–644. DOI: 10.1109/ICSME.2018.00073.

[14] Jay Kreibich. *Using SQLite*. " O'Reilly Media, Inc.", 2010.

[15] Grant Allen and Mike Owens. *The definitive guide to SQLite*. Apress, 2011.

[16] OpenAI. *Introducing ChatGPT*. [Accessed: 2023-09-18]. November 30, 2022. URL: https://openai.com/blog/chatgpt.

[17] Arghavan Moradi Dakhel, Vahid Majdinasab, Amin Nikanjam, Foutse Khomh, Michel C. Desmarais, and Zhen Ming (Jack) Jiang. "GitHub Copilot AI pair programmer: Asset or Liability?" In: *Journal of Systems and Software* 203 (2023), p. 111734. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2023.111734. URL: https://www.sciencedirect.com/science/article/pii/S0164121223001292.

[18] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. "Improving language understanding by generative pre-training". In: (2018).

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[21] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179/.

[22] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. "Transformers in Vision: A Survey". In: *ACM Comput. Surv.* 54.10s (Sept. 2022). ISSN: 0360-0300. DOI: 10.1145/3505244. URL: https://doi.org/10.1145/3505244.

[23] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. *Release Strategies and the Social Impacts of Language Models*. 2019. arXiv: 1908.09203 [cs.CL].

[24] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 27730–27744. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

[25] Jawid Ahmad Baktash and Mursal Dawodi. *Gpt-4: A Review on Advancements and Opportunities in Natural Language Processing*. 2023. arXiv: 2305.03195 [cs.CL].

[26] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. "Measuring Massive Multitask Language Understanding". In: *International Conference on Learning Representations*. 2021. URL: https://iclr.cc/virtual/2021/poster/2962.

[27] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG].

[28] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. *Evaluating Large Language Models Trained on Code*. 2021. arXiv: 2107.03374 [cs.LG].

[29] OpenAI et al. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[30] Michael Bommarito II au2 and Daniel Martin Katz. *GPT Takes the Bar Exam*. 2022. arXiv: 2212.14402 [cs.CL].

[31] Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. *Capabilities of GPT-4 on Medical Challenge Problems*. 2023. arXiv: 2303.13375 [cs.CL].

[32] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. "Emergent Abilities of Large Language Models". In: *Transactions on Machine Learning Research* (2022). Survey Certification. ISSN: 2835-8856. URL: https://openreview.net/forum?id=yzkSU5zdwD.

[33] Luciano Floridi and Massimo Chiriatti. "GPT-3: Its Nature, Scope, Limits, and Consequences". In: *Minds Mach.* 30.4 (Dec. 2020), pp. 681–694. ISSN: 0924-6495. DOI: 10.1007/s11023-020-09548-1. URL: https://doi.org/10.1007/s11023-020-09548-1.

[34] Lingjiao Chen, Matei Zaharia, and James Zou. *How is ChatGPT's behavior changing over time?* 2023. arXiv: 2307.09009 [cs.CL].

[35] OpenAI. *New Models and Developer Products Announced at DevDay*. [Accessed: 2023-12-21]. 2023. URL: https://platform.openai.com/docs/models/gpt-3-5.

[36] OpenAI. *New Models and Developer Products Announced at DevDay*. [Accessed: 2023-12-21]. 2023. URL: https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo.

[37] OpenAI. *New Models and Developer Products Announced at DevDay*. [Accessed: 2023-12-21]. 2023. URL: https://openai.com/blog/new-models-and-developer-products-announced-at-devday.

[38] Spencer Papay, Sam Waterbury, and Russell Kaplan. "How Much Better is OpenAI's Newest GPT-3 Model?" In: *Scale Blog* (Nov. 2022). [Accessed: 2023-12-19]. URL: https://scale.com/blog/gpt-3-davinci-003-comparison.

[39] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. *A Survey on In-context Learning*. 2023. arXiv: 2301.00234 [cs.CL].

[40] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[41] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9. URL: https://openai.com/research/better-language-models.

[42] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, lu Chen, Jinshu Lin, and Dongfang Lou. *C3: Zero-shot Text-to-SQL with ChatGPT*. 2023. arXiv: 2307.07306 [cs.CL].

[43] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. "What Makes Good In-Context Examples for GPT-3?" In: *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. Ed. by Eneko Agirre, Marianna Apidianaki, and Ivan Vulić. Dublin, Ireland and Online: Association for Computational Linguistics, May 2022, pp. 100–114. DOI: 10.18653/v1/2022.deelio-1.10. URL: https://aclanthology.org/2022.deelio-1.10.

[44] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. "Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 8086–8098. DOI: 10.18653/v1/2022.acl-long.556. URL: https://aclanthology.org/2022.acl-long.556.

[45] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. "On Faithfulness and Factuality in Abstractive Summarization". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault. Online: Association for Computational Linguistics, July 2020, pp. 1906–1919. DOI: 10.18653/v1/2020.acl-main.173. URL: https://aclanthology.org/2020.acl-main.173.

[46] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. "Survey of Hallucination in Natural Language Generation". In: *ACM Comput. Surv.* 55.12 (Mar. 2023). ISSN: 0360-0300. DOI: 10.1145/3571730. URL: https://doi.org/10.1145/3571730.

[47] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. "A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity". In: *Asia-Pacific Chapter of the Association for Computational Linguistics* (2023). URL: http://www.ijcnlp-aacl2023.org/.

[48] *The World Bank DataBank*. [Accessed: 2022-09-21]. URL: https://databank.worldbank.org/source/world-development-indicators/Series/SH.PRV.SMOK.MA.

[49] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. 2023. arXiv: 2302.11382 [cs.SE].

[50] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. 2023. arXiv: 2303.12712 [cs.CL].

[51] Stephanie Lin, Jacob Hilton, and Owain Evans. "TruthfulQA: Measuring How Models Mimic Human Falsehoods". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3214–3252. DOI: 10.18653/v1/2022.acl-long.229. URL: https://aclanthology.org/2022.acl-long.229.

[52] Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. *Natural Language Reasoning, A Survey*. 2023. arXiv: 2303.14725 [cs.CL].

[53] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing". In: *ACM Comput. Surv.* 55.9 (Jan. 2023). ISSN: 0360-0300. DOI: 10.1145/3560815. URL: https://doi.org/10.1145/3560815.

[54] Langchain Documentation. *Prompt Templates - Langchain Documentation*. [Accessed: 2023-11-20]. 2023. URL: https://python.langchain.com/docs/modules/model_io/prompts/prompt_templates/.

[55] Tongshuang Wu, Michael Terry, and Carrie Jun Cai. "AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts". In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI '22. , New Orleans, LA, USA, Association for Computing Machinery, 2022. ISBN: 9781450391573. DOI: 10.1145/3491102.3517582. URL: https://doi.org/10.1145/3491102.3517582.

[56] Mohammadreza Pourreza and Davood Rafiei. "DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction". In: *Advances in Neural Information Processing Systems 36*. Accepted for poster presentation, full citation details to be updated. 2023. URL: https://neurips.cc/virtual/2023/poster/70430.

[57] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. "Self-Consistency Improves Chain of Thought Reasoning in Language Models". In: *International Conference on Learning Representations*. Accepted for poster presentation, full citation details to be updated. 2023. URL: https://iclr.cc/virtual/2023/poster/11718.

[58] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. "ReAct: Synergizing Reasoning and Acting in Language Models". In: *International Conference on Learning Representations*. In-Person Poster presentation, top 5 percent paper. 2023. URL: https://iclr.cc/virtual/2023/poster/11003.

[59] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. *Reflexion: Language Agents with Verbal Reinforcement Learning*. 2023. arXiv: 2303.11366 [cs.AI].

[60] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. *Automatically Correcting Large Language Models: Surveying the landscape of diverse self-correction strategies*. 2023. arXiv: 2308.03188 [cs.CL].

[61] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. "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*. Ed. by Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 3911–3921. DOI: 10.18653/v1/D18-1425. URL: https://aclanthology.org/D18-1425.

[62] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. *Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation*. 2023. arXiv: 2308.15363 [cs.DB].

[63] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. "Text Classification Algorithms: A Survey". In: *Information* 10.4 (2019). ISSN: 2078-2489. DOI: 10.3390/info10040150. URL: https://www.mdpi.com/2078-2489/10/4/150.

[64] Allan David Gordon. *Classification*. CRC Press, 1999.

[65] David M. W. Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2020. arXiv: 2010.16061 [cs.LG].

[66] S. B. Kotsiantis. "Supervised Machine Learning: A Review of Classification Techniques". In: NLD: IOS Press, 2007, pp. 3–24. ISBN: 9781586037802. URL: https://dl.acm.org/doi/10.5555/1566770.1566773.

[67] Charles E. Metz. "Basic principles of ROC analysis". In: *Seminars in Nuclear Medicine* 8.4 (1978), pp. 283–298. ISSN: 0001-2998. DOI: https://doi.org/10.1016/S0001-2998(78)80014-2. URL: https://www.sciencedirect.com/science/article/pii/S0001299878800142.

[68] Yutaka Sasaki et al. "The truth of the F-measure". In: *Teach tutor mater* 1.5 (2007), pp. 1–5.

[69] Nancy Chinchor and Beth Sundheim. "MUC-5 Evaluation Metrics". In: *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*. 1993. URL: https://aclanthology.org/M93-1007.

[70] Victor Zhong, Caiming Xiong, and Richard Socher. "Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning". In: *arXiv e-prints*, arXiv:1709.00103 (Aug. 2017), arXiv:1709.00103. DOI: 10.48550/arXiv.1709.00103. arXiv: 1709.00103 [cs.CL].

[71] Petr Berka and Marta Sochorova. *Berka Dataset*. PKDD'99 Discovery Challenge. 1999. URL: https://webpages.charlotte.edu/mirsad/itcs6265/group1/domain.html.

[72] Jan Motl and Oliver Schulte. "The CTU Prague Relational Learning Repository". In: *arXiv e-prints*, arXiv:1511.03086 (Nov. 2015), arXiv:1511.03086. DOI: 10.48550/arXiv.1511.03086. arXiv: 1511.03086 [cs.LG].

[73] Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. "Enhancing Text-to-SQL Capabilities of Large Language Models: A Study on Prompt Design Strategies". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics. 2023, pp. 14935–14956. URL: https://aclanthology.org/2023.findings-emnlp.996.pdf.

[74] Weights and Biases. *Weights and Biases GitHub*. https://github.com/wandb/wandb. [Accessed on 2023-11-15].

[75] K. Sanderson. "GPT-4 is here: what scientists think". In: *Nature* 615.7954 (2023), p. 773. DOI: 10.1038/d41586-023-00816-5. URL: https://doi.org/10.1038/d41586-023-00816-5.

[76] Ashutosh Sathe Adithya Bhaskar Tushar Tomar and Sunita Sarawagi. "Benchmarking and Improving Text-to-SQL Generation under Ambiguity". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Toronto, Canada: Association for Computational Linguistics, Dec. 2023, pp. 7053–7074. DOI: 10.18653/v1/2022.emnlp-main.7. URL: https://aclanthology.org/2023.emnlp-main.436.pdf.

[77] Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. "Towards Robustness of Text-to-SQL Models against Synonym Substitution". 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)*. Ed. by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli. Online: Association for Computational Linguistics, Aug. 2021, pp. 2505–2515. DOI: 10.18653/v1/2021.acl-long.195. URL: https://aclanthology.org/2021.acl-long.195.

[78] OpenAI. *Introducing OpenAI*. https://openai.com/blog/introducing-openai. [Accessed: 2023-12-22]. 2015.

[79] OpenAI. *OpenAI and Microsoft extend partnership*. [Accessed: 2023-12-22]. 2023. URL: https://openai.com/blog/openai-and-microsoft-extend-partnership.