From Natural Language to SQL: Review of LLM-based Text-to-SQL Systems

Anonymous ACL submission

Abstract

LLMs are greatly improving SOTA of translating natural language queries to structured and correct SQL. Unlike previous reviews, this survey provides a comprehensive study of the evolution of LLM-based text-to-SQL systems, from early rule-based models to advanced LLM approaches. We discuss benchmarks, evaluation methods and evaluation metrics. Also, we uniquely study the role of integration of knowledge graphs for better contextual accuracy and schema linking in these systems. Finally, we highlight key challenges such as computational efficiency, model robustness, and data privacy toward improvements of LLM-based text-to-SQL systems.

1 Introduction

1.1 Overview of the Text-to-SQ task

Organizations are increasingly reliant on relational databases to manage and analyze vast amounts of structured information. These databases are critical parts of many modern systems, from business intelligence to customer relationship management. As the volume of data increases, the need to query, extract, and make sense of this information also increases. However, querying databases often requires the use of Structured Query Language (SQL), a technical skill. There is a gap between users who need access to data and the specialized knowledge required to retrieve it (KANBUROĞLU and TEK, 2024). Text-to-SQL parsing in natural language processing (NLP) bridges this gap.

LLMs ease the task of translating natural language (NL) queries given by a non-technical user to precise SQL that can retrieve relevant information from a database. This ability is important when the complexity of data increases and makes manual data exploration impractical and inefficient (Dong et al., 2023). For example, consider a relational database that contains information about gas stations in Louisiana, with columns such as `GasStationID`, `PARISH`, `NEED`, `STATION NAME`, `CITY`, and `WORKING_GAS_STATIONS_5_MILES` among others. Suppose we have a NL query: *"Where can I find a gas station with power less than 2 miles from the University?"* A text-to-SQL system would analyze this query, understand the user's intent, and automatically generate the corresponding SQL query to extract the correct information from the database. In this case, the final SQL query might look something like:

```
SELECT STATION_NAME, location
FROM gas_stations
WHERE fuel_available = 'Yes'
  AND distance < 2
  AND ST_Distance_Sphere(Point(long, lat),
  Point(University_Long,
  University_Lat)) < 2;
```

SQL query retrieves all gas stations within a 2-mile radius of the University that have power. The text-to-SQL system enables the user to extract this specific information without needing to write the SQL query thus making complex data more accessible. However, building reliable text-to-SQL systems is highly challenging. The complexity arises from several factors:

1. Linguistic Complexity and Ambiguity: NL queries often include complex structures such as nested clauses, pronouns, or vague terms. A query might contain phrases like "all gas stations in the area," where "area" could refer to a specific region not directly mentioned.

2. Schema Understanding and Representation: To generate accurate SQL queries, text-to-SQL systems must understand the database schema. In realistic applications, database schemas are often complex and can vary greatly between domains, making it challenging to map the NL query correctly to the database schema (Yu et al., 2019a).

3. Rare and Complex SQL Operations: Some queries involve uncommon SQL operations, such as nested sub-queries, joins across multiple tables, or window functions. These operations can be difficult for text-to-SQL models to generate, especially if they are not frequently seen during training (Hong et al., 2024b).

4. Cross-Domain Generalization: Text-to-SQL models trained on one domain (e.g., customer service databases) may not generalize well to other

domains (e.g., healthcare). Differences in schema structures, terminology, and query patterns make it difficult for models to perform consistently across a wide variety of databases (Gan et al., 2021).

Addressing these challenges is essential for the effectiveness of text-to-SQL systems. As databases become more complex, the accuracy of text-to-SQL models must be increased.

1.2 Importance of Text-to-SQL

Stack Overflow shows that 51.52% of professional developers use SQL in their works. However, SQL is often a barrier for non-technical users because of its technical nature. This leaves a gap between the vast stores of data housed in relational databases and the ability of many users to retrieve and analyze that data efficiently (Hong et al., 2024b).

The objective of this survey is to bridge the knowledge gap by reviewing advances in text-to-SQL systems which aim to democratize data access by enabling users to query databases using NL. This ability is particularly important as databases grow larger and more complex, with modern systems spanning across various industries such as healthcare, finance, and logistics.

Reviewing current state-of-the-art techniques is needed to understand previous progress and identifying areas that require further exploration. By reviewing recent approaches and evaluation metrics, we give researchers a clearer understanding of how text-to-SQL systems can evolve to meet real-world demands.

This survey, highlights key methods, benchmarks, and models that have shaped text-to-SQL research, while also exploring areas for future advances. It will also help researchers for next generation systems that make data querying more accessible and efficient across a wide range of applications. A research taxonomy is shown in Fig3.

2 Evolution of the Text-to-SQL

2.1 Evolutionary Process

Initially, Text-to-SQL systems relied on rules, but now they use advanced neural networks and powerful pre-trained language models (PLMs) (Hong et al., 2024b). Each step brought important innovations, making it easier for the models to understand and create SQL queries from everyday language, as shown in Figure 1.

1.Rule-Based Approaches

Early text-to-SQL systems used rule-based approaches. These systems used manually crafted grammar rules and heuristics to translate NL queries into SQL commands. These methods faced limitations in handling complex queries and diverse schemas (Mahmud et al., 2015). Systems like LUNAR (Kang et al., 2013) and NaLIX (Hammami et al., 2021) showed the potential of semantic parsing but required significant manual feature engineering, hindering scalability and adaptability (Kang et al., 2013), (Hammami et al., 2021).

2. Deep Learning Approaches

Deep learning revolutionized text-to-SQL systems by enabling large models to interpret NL and generate SQL queries. Models like Seq-2-SQL and SQLNet (Katsogiannis-Meimarakis and Koutrika, 2021) used sequence-to-sequence architectures like LSTMs and transformers (Banitaba et al., 2024). These models introduced end-to-end, differentiable architectures that directly convert text to SQL, offering improvements over earlier systems (Katsogiannis-Meimarakis and Koutrika, 2021), (Kumar et al., 2022), (Naghshnejad et al., 2024).

Transformer-based models like BERT (Li et al., 2024b) and TaBERT (Katsogiannis-Meimarakis and Koutrika, 2023), enhance understanding of both database schema and user intent. These models improved generalization to unseen databases by capturing dependencies between NL queries and underlying schemas. However challenges like handling nested queries, cross-domain generalization, and efficiently mapping ambiguous natural language to structured SQL still remained. (Li, 2018), (Katsogiannis-Meimarakis and Koutrika, 2023).

3. Pre-trained Language Models

Pre-trained Language Models (PLMs) shifted the paradigm from task-specific supervised learning to a more generalized pre-training method followed by fine-tuning approaches. Models like BERT (Li et al., 2024b) (Bidirectional Encoder Representations from Transformers) and GPT (Brown, 2020) (Generative Pre-training Transformer) started this shift by using large-scale, unsupervised text datasets to pre-train models that could then be fine-tuned for specific tasks. The concept of PLMs, grounded in transfer learning, allowed models to get a deep understanding of NL through extensive pre-training on vast datasets, and this understanding transferred to a variety of tasks, including text generation, sentiment analysis, and question-answering (Tahir et al., 2024), (Wang et al., 2023).

In Text-to-SQL systems, PLMs made improvements by capturing semantic relationships between user queries and database schema structures. PLMs such as TaBERT and BERT-SQL (Guo and Gao, 2019) enabled the integration of both the NL query and the database schema into a unified representation that improved the system context. These

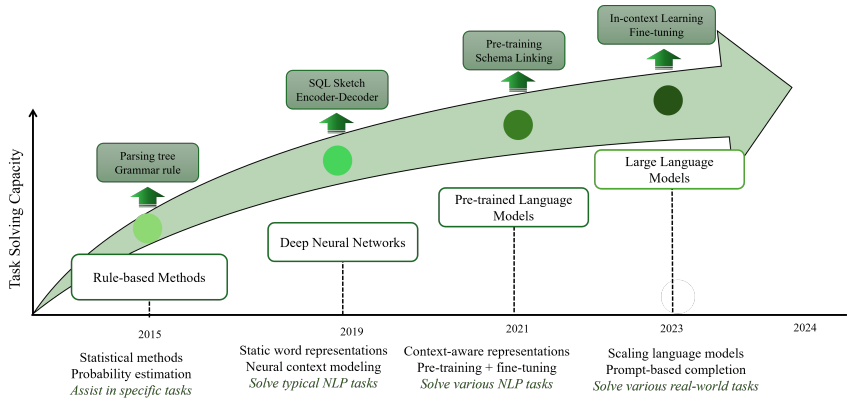

Figure 1: How text-to-SQL research has evolved over time, highlighting different implementation approaches. Each phase includes key techniques and notable works. The dates are approximate, based on when these key works were released, with a margin of error of about a year. The design is inspired by (Zhao et al., 2023).

models addressed several challenges in text-to-SQL systems, such as handling complex queries with multiple joins, nested queries, and cross-domain generalization. However, PLMs did have limitations in terms of their need for domain-specific fine-tuning and the difficulty in understanding complex database schemas without additional schema linking mechanisms (Min et al., 2023).

4. Large Language Models

The advent of Large Language Models (LLMs) such as GPT-4 (OpenAI, 2023b), Codex (Chen et al., 2021), and LLaMa (Touvron et al., 2023) has revolutionized NL processing by enhancing the capabilities of systems that perform complex language tasks. LLMs represent a major step forward from traditional machine learning methods by leveraging their vast size and training on massive datasets to generate more accurate and sophisticated responses. These models have shown exceptional performance in tasks that require understanding and generating human-like text, often without needing additional fine-tuning (Rajkumar et al., 2022).

LLMs capture more complex relationships between NL queries and structured database schemas. Unlike pre-trained language models, which require extensive task-specific fine-tuning and schema linking, LLMs can handle zero-shot and few-shot scenarios more effectively due to their large-scale pre-training and reasoning capabilities (Zhang et al., 2024a). Studies show that models like Codex achieve high performance in generating SQL queries with minimal prompt engineering. However, challenges such as handling ambiguous queries and optimizing SQL statements for performance and correctness remain (Rajkumar et al., 2022).

Large Language Models (LLMs) have revolutionized the process of generating SQL queries from NL queries. As seen in Figure 2, the Architecture of LLM-based text-to-SQL systems can be broken down into several key phases: natural language understanding, schema comprehension, SQL generation, and SQL execution. Each step involves sophisticated techniques to ensure that user queries are accurately mapped to SQL, providing correct and meaningful results from database.

2.2 LLM-based Text-to-SQL Architecture

1. Natural Language Understanding:

The process begins with user input in the form of NL queries. LLMs first reprocess these input to understand the user's intent, identifying key components such as entities, conditions, and relations within the question.

2. Schema Linking:

Once the NL query is parsed, the system moves to schema linking, where the LLM maps the parsed components of the query to the corresponding tables, columns, and relationships in the database schema. For example, "gas stations" is linked to a table named `GasStations`, and "power" is matched with a column named `PowerAvailable`. This phase ensures that the system can correctly interpret the query.

3. SQL Generation:

After the query has been parsed and linked to the schema, the LLM generates an SQL query based o the established semantic relationships. This stage uses the model's understanding of SQL syntax and database logic to form a structured query that reflects the user's intent. The generated SQL is then validated and optimized for accuracy and performance.

4. SQL Execution and Output:

The final SQL query is executed on the underlying database (such as SQLiteor MySQL) to retrieve the requested information. The results of

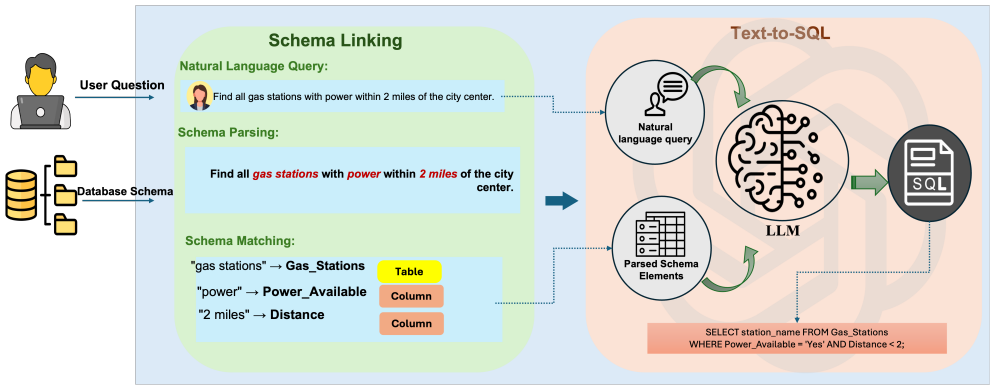

Figure 2: Illustrates the key stages of the text-to-SQL process using Large Language Models (LLMs).

the query are returned either in raw format or, in some systems, converted back into NL for easier interpretation by the user. Figure 2 shows the flow from user input to the final SQL query. Each phase makes the text-to-SQL systems more accessible for non-technical users.

3 BENCHMARKS AND EVALUATION METHODS

Evaluating LLM-based text-to-SQL systems is crucial for measuring how well they understand NL and generate accurate SQL queries. Researchers use a variety of datasets and benchmarks that test these models across different scenarios, both within specific domains and across multiple domains.

3.1 Types of Datasets used in Benchmarks

Text-to-SQL research has made rapid progress due various benchmark datasets, each contributing unique challenges for model development. These datasets are categorized into four types based on their characteristics: cross-domain, knowledge-augmented, context-dependent, and robustness. In Table 1, we categorised most well known datasets in using these criteria. **1. Cross-domain Datasets** Datasets like WikiSQL (Zhong et al., 2017), Spider (Yu et al., 2018), and KaggleDBA (Lee et al., 2021) focus on evaluating the generalization capabilities of models across multiple databases from different domains. These datasets test whether models can generate accurate SQL queries for databases they have not seen during training (Yu et al., 2018).

2. Knowledge-Augmented Datasets Datasets such as SQUALL (Shi et al., 2020) and BIRD (Li et al., 2024b) use external knowledge to improve the semantic of SQL generation. These datasets aim to enhance the model's comprehension by augmenting the schema with additional context, allowing for more accurate and better SQL generation (Li et al., 2024b).

3.Context-Dependent Datasets Datasets like CoSQL(Yu et al., 2019a) and SParc(Yu et al., 2019b) emphasize the conversational nature of querying databases, where previous use interactions influence current queries. These datasets challenge models to maintain context throughout multi-turn interactions, making them essential for developing systems that can handle complex, dialog-driven database queries (Yu et al., 2019b).

4. Robustness Datasets A dataset like ADVETA (Pi et al., 2022) tests the robustness of text-to-SQL systems, by introducing adversarial table perturbations. This method tests if models are capable of handling unexpected changes in database schema or table structure, thereby assessing their adaptability to real-world scenarios(Pi et al., 2022). The Table 1 datasets push the boundaries of text-to-SQL research, providing challenges across various dimensions, from domain generalization to contextual understanding and system robustness.

3.2 Evaluation Metrics Used in Benchmarks

Benchmarks for Text-to-SQL systems use metrics that capture both the correctness and efficiency of SQL query. These metrics test that systems not only produce accurate SQL queries, but also perform efficiently in realistic database environments. Figure 4 shows evaluation metrics that are fall into two categories: Content Matching-based Metrics and Execution-based Metrics.

3.2.1 Content Matching-based Metrics

Content matching-based metrics focus on how closely the structure of the generated SQL query matched to the gold (or reference) query. This type of evaluation is needed for ensuring that the model follows the correct SQL syntax and structure, even if it might not always produce the most optimized SQL query.

1. Component Matching (CM): This metric evaluates each component of the SQL query (such as SELECT, FROM, WHERE) individually. Even if the components appear in a different order than in the gold query, they are considered correct as long

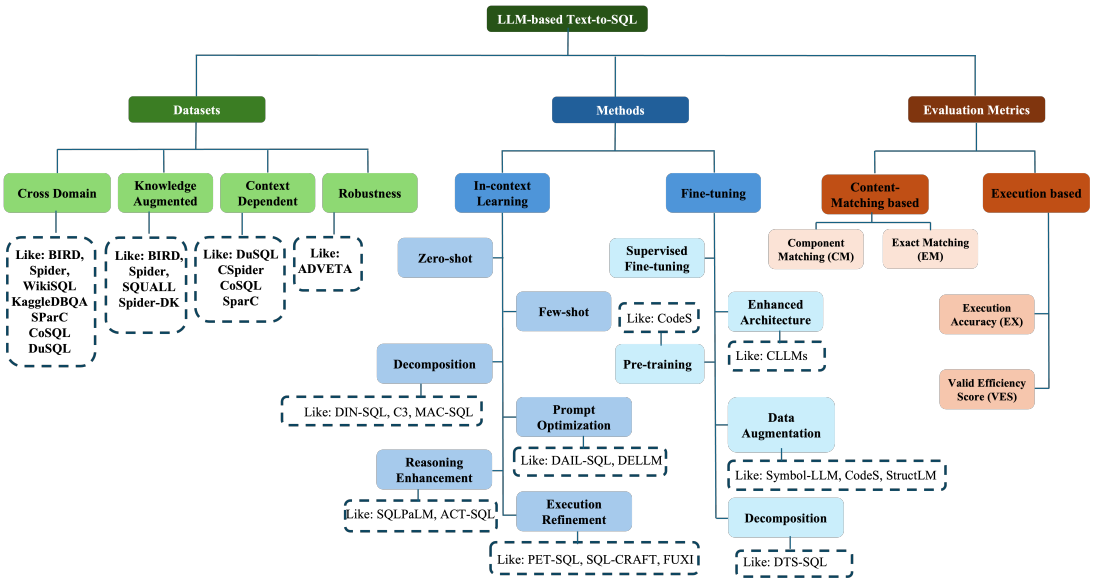

Figure 3: Taxonomy of research approaches in LLM-based text-to-SQL. The format is adapted from (Xu et al., 2023a)

.

Table 1: Statistics and Characteristics of well-known datasets used in text-to-SQL. CH1: Cross-domain, CH2: Context-dependent, CH3: Knowledge-augmented, CH4: Cross-lingual, CH5: Robustness,

Datasets	Dataset Statistics				Released Date	Characteristics
	Example	DB	Table/DB,	Row/DB		
WikiSQL(Zhong et al., 2017)	80,654	26,521	1	17	Aug-17	CH1
Spider(Yu et al., 2018)	10,181	200	5.1	2K	Sep-18	CH1
CoSQL(Yu et al., 2019a)	15,598	200	_	_	Sep-19	CH1, CH2
SQUALL(Shi et al., 2020)	11,468	1679	1	_	Oct-20	CH3
DuSQL(Wang et al., 2020)	23,797	200	4.1	_	Nov-20	CH1, CH4
KaggleDBQA(Lee et al., 2021)	272	8	2.3	280K	Jun-21	CH1
BIRD (Li et al., 2024b)	12,751	95	7.3	549K	May-23	CH1,CH3
ADVETA (Pi et al., 2022)	Spider and WikiSQL - Adversarial table perturbations				Dec-22	CH5
SParC(Yu et al., 2019b)	Spider - Annotate conversational contents				Jun-19	CH2

as they correspond to the expected components. This allows for flexibility in the query structure while ensuring the essential parts of the SQL query are present and accurate(Yu et al., 2018).

2. Exact Matching (EM): Exact matching is stricter, requiring the generated SQL query to match the gold query exactly in terms of both structure and order. Every element including the sequence of components, must be identical to the gold query. The disadvantage of this metric is it can penalize queries that are functionally correct but structured differently(Yu et al., 2018).

3.2.2 Execution-based Metrics

These metrics focus on the actual performance of the generated SQL query when executed on a database. These metrics are needed for ensuring that the queries not only follow the correct structure but also return the correct results and run efficiently in realistic scenarios.

1. Execution Accuracy (EX): This metric checks whether the generated SQL query, when executed on the database, returns the same result as the gold query. Execution accuracy focuses on the correctness of the result, regardless of how the query is structured. As long as the output matched, the query is considered accurate (Yu et al., 2018).

2. Valid Efficiency Score (VES): The Valid Efficiency Score (VES) measures the computational efficiency of the generated SQL query compared to the gold query. While a query might return the correct result, it could still be inefficient, requiring unnecessary computational resources. VES penalizes queries that introduce extra complexity, such as redundant sub-queries or unnecessary joins, even if the results match.(Yu et al., 2019a), (Yu et al., 2019a).

3.3 Methods

LLM-based text-to-SQL systems fall into two main classes of methods: **In-context learning (ICL)** and **Fine-tuning (FT)**. Each class involves different strategies for training the model to generate SQL

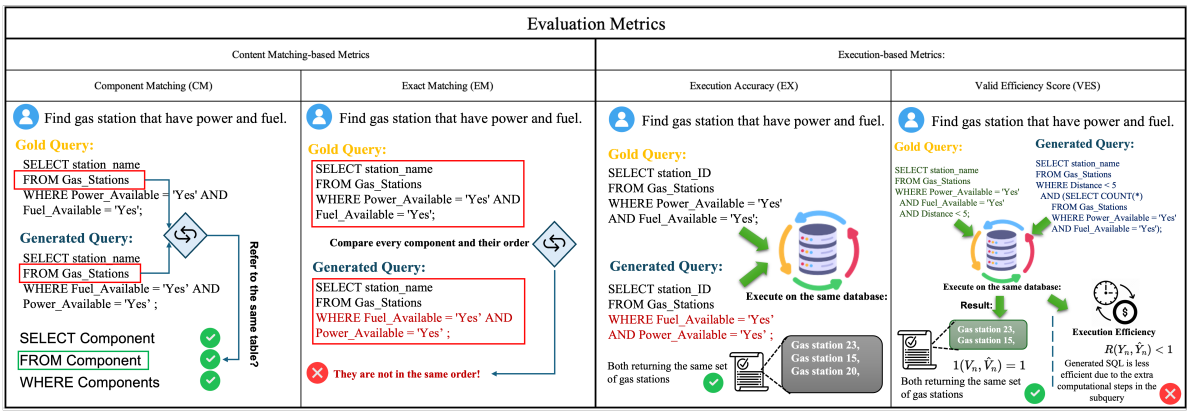

Figure 4: Four main evaluation metrics

queries from NL inputs. These methods rely on using the model's pre-trained knowledge through prompts or fine-tuning with task-specific data to improve SQL query generation.

3.3.1 In-context Learning (ICL)

In in-context learning, the model generates the SQL query based on a given context without any updates to the model parameters. Instead, the model is guided through accurate constructed prompts. In-context learning includes several categories that optimize how queries are generated and improve accuracy. Mathematically, the SQL query generation task is described as:

$$Y = f(Q, S, I; \theta) \tag{1}$$

Y is the generated SQL query, Q is the NL questions, S represents the database schema, I is the intermediate reasoning, and θ are parameters of the pre-trained model. In in-context learning, this formula highlights that the model's output is determined by the input information (Q, S, I) and the pre-trained knowledge embedded in θ, remains fixed during the process, and the model's performance depends on how effectively the input prompt is engineered to guide the model towards generating accurate SQL queries. Table 2, classifies the most famous methods in five main categories in the area of in-context learning. In next we illustrate all these categories.

1. Zero-Shot and Few-Shot Learning: In zero-shot learning, the model generates SQL queries without any prior exposure to similar examples. the model relies on its pre-trained knowledge to interpret the input and produce SQL queries. For example, in C3's zero-shot prompting of ChatGPT for text-to-SQL tasks, no fine-tuning is required (Dong et al., 2023). Zero-shot learning is most effective when the LLM has been pre-trained on a vast corpus that includes SQL-related content. However, in few-shot learning, the model is provided with a few examples of input-output pairs, which guide its generation of new SQL queries. An example is FinSQL, where models are given a small set of SQL queries and asked to generalized from these examples (Zhang et al., 2024b).

2. Decomposition: This technique breaks complex queries into simpler sub-queries. Decomposition methods can involve dividing a challenging NL question into multiple SQL queries that are easier to generate. Decomposition enhances the model's ability to handle multi-step or nested SQL queries (Pourreza and Rafiei, 2024a).

3. Prompt Optimization: This involves refining the input prompts to achieve better SQL generation. By providing more structured or derailed prompts, the model is better guided in generating the appropriate SQL queries. Techniques like prompt design and prompt calibration fall under this category, ensuring the prompts are constructed to maximize the model's understanding. Methods like ACT-SQL sue prompt optimization to enhance SQL generation by structuring the prompts more efficiently and linking them directly to the database schema (Zhang et al., 2023).

4. Reasoning Enhancement: Reasoning enhancement methods, such as Chain of Thought (CoT) and Tree of Thoughts (ToT), guide the model to think step-by-step. These approaches enable the model to solve complex queries by generating intermediate reasoning steps before arriving at the final SQL output (Yao et al., 2024),(Wei et al., 2022).

5. Execution Refinement: These methods involve iterative improvements to the SQL query. The model generates an initial SQL query, executes it, and then refines it based on feedback or the results. This method ensures that the generated SQL query is optimized for performance and correctness, minimizing errors in execution (Vougiouklis et al., 2023).

Table 2: State of art methods used in in-context learning for LLM-based text-to-SQL. C1: zero shot and few shots learning, C2: Decomposition, C3: Prompt Optimization, C4: Reasoning Enhancement, C5: Execution Refinemen, (Zhang et al., 2023), (Liu et al., 2024), (Fan et al., 2024), (Xia et al., 2024), (Hong et al., 2024a), (Gu et al., 2024), (Zhang et al., 2024c), (Zhang et al., 2024b), (Gao et al., 2023), (Vougiouklis et al., 2023), (Dong et al., 2023), (Fu et al., 2023), (Wei et al., 2022),(Yao et al., 2024),(Pourreza and Rafiei, 2024a).

Methods	Applied LLM	Database	Metrics	Methods Categories					Released
				C1	C2	C3	C4	C5	
ACT-SQL	GPT4	Spider, BIRD	EX,EM.etc			*	*		Oct-23
Schema-free	GPT4	BIRD	EX				*	*	Aug-24
MetaSQL	GPT4	BIRD	EX, EM		*				Mar-24
SQL-CRAFT	GPT4	Spider, BIRD	EX				*	*	Feb-24
FUXI	GPT4	BIRD	EX				*	*	Feb-24
DELLM	GPT4	Spider, BIRD	EX,VES			*		*	Feb-24
SGU-SQL	GPT4	Spider, BIRD	EX, EM		*				Feb-24
FinSQL	LLamA2	BULL	EX	*					Jan-24
DAIL-SQL	GPT Family	Spider, BIRD	EX , EM,VES			*			Nov-23
FastRAT	XLM-RoBERTa	Spider, Cspider	EX, EM					*	Nov-23
CatSQL	BERT	Spider, WikiSQL	EX	*				*	Nov-23
C3	GPT Family	Spider	EX	*		*		*	Jul-23
CoT	GPT4	Spider, BIRD	EX,VES				*		May-23
ToT	GPT4	OTHER	OTHER				*	*	May-23
DIN-SQL	CodeX, GPT-4	Spider, BIRD	EX, EM		*			*	Apr-23

3.3.2 Fine-Tuning

Fine-tuning involves refining the model's internal parameters, θ, using task-specific datasets. Unlike in-context learning methods, where the model's parameters remain fixed and prompts are the primary mechanism of control, fine-tuning updates the model's parameters based on examples from the target task. This process allows the model to become more specialized in tasks like SQL query generation, improving the ability to translate NL questions into accurate SQL queries over time.

Mathematically, this process is represented as a function g:

$$\theta' = g(\theta, D) \tag{2}$$

where θ is the pre-trained model's parameters, D is the task-specific dataset (pair of questions and SQL queries) and θ' is represents the updated parameters after fine-tuning (Kou et al., 2024).

In fine-tuning, the model learns patterns specific to SQL generation, such as understanding database schema and query syntax. This helps it to perform better at generating SQL queries by modifying its internal parameters based on task data, making the model more specialized and accurate for the SQL task. The new parameters, θ', improves model's ability to generalize across different databases and queries. Currently, a number of studies exploring a improved fine-tuning method have been released. Table 3 shows categorized well-designed fine-tuning methods.

1. Pre-trained Methods: Pre-trained methods form the backbone of fine-tuning for text-to-SQL systems by leveraging the cast general knowledge embedded in LLMs, such as GPT, LLaMA, and T5. these models, trained on diverse textual data, are adapted for SQL query generation by fine-tuning with task-specific data. The fine-tuning process enhances their ability to interpret NL and accurately map it to SQL commands across different domains and database schemas (Li et al., 2023). Examples like SQL-GEN show how pre-trained models are fine-tuned with synthetic data for dialect-specific SQL generation, while systems like RESDSQL fine-tune LLMs on datasets like Spider for complex query handling (Pourreza et al., 2024).

2. Fine-Tuning Decomposition: Fine-tuning decomposition methods aim to enhance the performance of LLM on text-to-SQL tasks by breaking down the complex process of query generation into smaller and manageable sub-tasks. The main idea is to address each sub-task individually, thereby allowing the model to better focus and fine-tune its parameters for specific challenges related to text-to-SQL generation. By decomposing the task into stages like schema linking and query formation, model can be trained to handle these distinct processes more effectively than if it were trained on the entire query generation task all at once (Pourreza and Rafiei, 2024b). The typical fine-tuning decomposition process involves:

Task Segmentation: breaking down the text-to-SQL conversion into smaller tasks like schema linking and SQL query generation.

Sequential Fine-Tuning: Training the model on these sub-tasks in sequence or in parallel so that each sub-task is learned optimally.

3. Data Augmented Methods: The performance of the model is particularly affected by the quality of the training labels during fine-tuning. In-

Table 3: State of art methods used in fine-tuning (FT) for LLM-based text-to-SQL. C1: Pre-trained, C2: Decomposition, C3: Data Augmented, C4: Enhanced Architecture (Pourreza et al., 2024), (Zhang et al., 2024d), (Kou et al., 2024), (Pourreza and Rafiei, 2024b), (Li et al., 2024a), (Xu et al., 2023b).

Methods	LLM	Database	Metrics	Methods Categories				Released Time
				C1	C2	C3	C4	
SQL-GEN	LLamA3	BIRD	EX	*		*		Aug-24
SQLfuse	LLamA2-70B	Spider	EX	*				Jul-24
CLMMs	Deepseek	Spider	EX				*	Mar-24
CodeS	StarCoder	Spider,BIRD	EX,VES			*		Feb-24
DTS-SQL	Mistral	Spider,Spider-SYN	EX, EM		*			Feb-24
Symbol-LLM	CodeLLaMA	Spider	EM			*		Nov-23

adequate labeling can be counterproductive, often optimal results are not achieved. Rather, if effective augmentation or high-quality data is present, fine tuning is likely to yield results more than even the best fine tuning strategies implemented in low quality or raw data. In text-to-SQL and other problems data-augmented fine-tuning has progressed greatly, as more efforts now aim at improving the data quality rather than the architecture. As an example, Symbol-LLM has developed an injection and an infusion phase with a focus on improving the data quality during instruction tuning (Xu et al., 2023b), (Li et al., 2024a).

4. Enhance Architecture: Using the generative pre-trained transformer (GPT) framework employs a decoder-only transformer architecture combined with standard auto-regressive decoding for text generation (Hong et al., 2024b). However, recent research on the efficiency of large language models (LLMs) has highlighted a shared challenge: when generating long sequences in the auto-regressive paradigm, the attention mechanism increases latency. This issue is pronounced in LLM-based text-to-SQL systems, where generating SQL queries is slower than traditional language modeling, posing a challenge for developing efficient, localized NL interfaces to databases (NLIDB). To address this, Consistency large language models (CLLMs) have been developed with an enhanced model architecture, providing a solution to reduce latency and speed up SQL query generation(Kou et al., 2024).

4 Knowledge Graphs in LLM-based Text-to-SQL Systems

Knowledge graphs represent entities and their relationships such that navigating through complex interconnected data becomes easier and enhances the query processing of text-to-SQL frameworks. These graphs enhance the overall performance of a system by providing semantic context and explicating associations between tables and attributes to support the exact translation of NL queries into SQL (OpenAI, 2023a).

Although LLMs now allow the creation of more flexible queries, problems still occur regarding ambiguity, schema linking, and delays in querying very large databases. KGs solve these issues by refining schema linking, aligning NL queries with database tables and columns through semantic relationships, and providing contextual insight to resolve ambiguities, which enables the generation of more accurate SQL queries (Sequeda et al., 2024).

Typical examples are ChatKBQA (Luo et al., 2023), relying on a "generate-then-retrieve" approach, which iteratively refines any query using feedback from the KG, and Interactive-KBQA (Xiong et al., 2024), a query-forming approach that iterates over multiple interactions with the KG. Another typical example is an approach to a chatbot that, with the aid of a knowledge graph, can offer students personalized learning paths by enhancing the context of their interactions (Abu-Rasheed et al., 2024), (Sundaram et al., 2024).

Some of the challenges in integrating KGs into LLMs are: **1. Complexity Graph:** Building such detailed KGs is a very time-intensive process. **2. Dynamic Schema Adaptation:** KGs must continuously adapt to changes in database schemas, which, for dynamic databases, becomes extremely resource-intensive. **3. Scalability:** Efficient and scalable KGs are essential to avoid slowing down query generation within large datasets.

5 Conclusion

The future for LLM-based text-to-SQL systems promises to continue targeting those areas in which there is room for improvement with respect to computational efficiency, robustness, contextual accuracy, and ethics of AI practices. As the systems continue to incorporate more knowledge graphs, refine retrieval-augmented generation, and improve human-in-the-loop mechanisms, they will be more effective, accurate, and user-friendly. The next-generation systems will have scalability, transparency, and adaptability as hallmarks for real-world applications across a wide range of domains.

6 Limitation of the State of the Art

Despite the advances summarized above, a number of challenges remain. These are summarized below.

6.1 Scalability and Computational Efficiency

Enhancing LLM-based text-to-SQL systems for large and complex databases without losing computational efficiency is an important challenge. The processing and generation cost of SQL queries remains high, especially with longer sequences and larger datasets. Future solutions will likely focus on model optimizations, more efficient retrieval and storage mechanisms, and specialized indexing techniques to streamline query generation.

6.2 Dynamic Adaptation to Schema Changes

Most current systems have includes inefficiency in adapting to the dynamic, evolving databases without full retraining. Considering that realistic databases will very often experience schema changes and data expansion, the lack of effective techniques, such as incremental learning and flexible architectures, holds back seamless updating of the LLMs and KGs, potentially leading to a reduction of query accuracy after some time, most importantly about changing environments.

6.3 Contextual Accuracy and Disambiguity

Many LLM-based text-to-SQL systems face challenges in handling complex and ambiguous queries where context is not explicitly given. Improving contextual accuracy requires research into how LLMs use structured information from KGs. Enhancing semantic links between user queries and the database schema will be critical, and more advanced semantic parsing and disambiguation techniques will help resolve ambiguity.

6.4 Balancing Retrieval-Augmented Generation (RAG) and Fine-Tuning

While fine-tuning models for specific domains improves performance, RAG offers a way to dynamically incorporate context with less extensive model retraining. The balance between RAG and fine-tuning is an area to be explored, with potential future systems leveraging the strengths of both approaches to minimize training time while maintaining context-sensitive query generation.

6.5 Ethics, Data Privacy, and Interpretability

The application of LLMs in critical domains like healthcare, finance, and education raises ethical concerns regarding data privacy and model interpretability. It is essential for such systems to be transparent, reliable, and respectful of user privacy. Future work will need to establish clear explainability protocols, safe data handling practices, and transparent AI procedures to build trust in LLM-based text-to-SQL systems.

6.6 Human-in-the-Loop and Interactive Querying

Integration with human feedback is a key future direction. Human-in-the-loop mechanisms will help users to refine and correct generated queries interactively, enhancing model accuracy and transparency. Improved interactivity will not only help build user trust but also provide enhanced learning and error correction opportunities during SQL generation.

6.7 Knowledge Graph Integration and Maintenance

While knowledge graphs enhance schema awareness and result in query precision, their creation and maintenance need to be overcome. Scalable creation usually involves complexity and resource demands that pose an implementation challenge in practice. Besides, the absence of practical methods for dynamic updating of KGs without degradation in system performance limits the efficiency of LLM-based systems for as long as the data and schemas evolve.

References

Hasan Abu-Rasheed, Mohamad Hussam Abdulsalam, Christian Weber, and Madjid Fathi. 2024. Supporting student decisions on learning recommendations: An llm-based chatbot with knowledge graph contextualization for conversational explainability and mentoring. *arXiv preprint arXiv:2401.08517*.

Faeze S Banitaba, Sercan Aygun, and M Hassan Najafi. 2024. Late breaking results: Fortifying neural networks: Safeguarding against adversarial attacks with stochastic computing. *arXiv preprint arXiv:2407.04861*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Jinshu Lin, Dongfang Lou, et al. 2023. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*.

Yuankai Fan, Zhenying He, Tonghui Ren, Can Huang, Yinan Jing, Kai Zhang, and X Sean Wang. 2024.

Metasql: A generate-then-rank framework for natural language to sql translation. *arXiv preprint arXiv:2402.17144*.

Han Fu, Chang Liu, Bin Wu, Feifei Li, Jian Tan, and Jianling Sun. 2023. Catsql: Towards real world natural language to sql applications. *Proceedings of the VLDB Endowment*, 16(6):1534–1547.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021. Exploring underexplored limitations of cross-domain text-to-sql generalization. *arXiv preprint arXiv:2109.05157*.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Yu Gu, Yiheng Shu, Hao Yu, Xiao Liu, Yuxiao Dong, Jie Tang, Jayanth Srinivasa, Hugo Latapie, and Yu Su. 2024. Middleware for llms: Tools are instrumental for language agents in complex environments. *arXiv preprint arXiv:2402.14672*.

Tong Guo and Huilin Gao. 2019. Content enhanced bert-based text-to-sql generation. *arXiv preprint arXiv:1910.07179*.

Linda Hammami, Alessia Paglialonga, Giancarlo Pruneri, Michele Torresani, Milena Sant, Carlo Bono, Enrico Gianluca Caiani, and Paolo Baili. 2021. Automated classification of cancer morphology from italian pathology reports using natural language processing techniques: A rule-based approach. *Journal of Biomedical Informatics*, 116:103712.

Zijin Hong, Zheng Yuan, Hao Chen, Qinggang Zhang, Feiran Huang, and Xiao Huang. 2024a. Knowledge-to-sql: Enhancing sql generation with data expert llm. *arXiv preprint arXiv:2402.11517*.

Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2024b. Next-generation database interfaces: A survey of llm-based text-to-sql. *arXiv preprint arXiv:2406.08426*.

Ali Buğra KANBUROĞLU and Faik Boray TEK. 2024. Text-to-sql: A methodical review of challenges and models. *Turkish Journal of Electrical Engineering and Computer Sciences*, 32(3):403–419.

Ning Kang, Bharat Singh, Zubair Afzal, Erik M van Mulligen, and Jan A Kors. 2013. Using rule-based natural language processing to improve disease normalization in biomedical text. *Journal of the American Medical Informatics Association*, 20(5):876–881.

George Katsogiannis-Meimarakis and Georgia Koutrika. 2021. A deep dive into deep learning approaches for text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2846–2851.

George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, 32(4):905–936.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. 2024. Cllms: Consistency large language models. *arXiv preprint arXiv:2403.00835*.

Ayush Kumar, Parth Nagarkar, Prabhav Nalhe, and Sanjeev Vijayakumar. 2022. Deep learning driven natural languages text to sql query conversion: a survey. *arXiv preprint arXiv:2208.04415*.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. Kaggledbqa: Realistic evaluation of text-to-sql parsers. *arXiv preprint arXiv:2106.11455*.

Hang Li. 2018. Deep learning for natural language processing: advantages and challenges. *National Science Review*, 5(1):24–26.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024a. Codes: Towards building open-source language models for text-to-sql. *Proceedings of the ACM on Management of Data*, 2(3):1–28.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024b. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuyu Luo, Yuxin Zhang, Ju Fan, Guoliang Li, and Nan Tang. 2024. A survey of nl2sql with large language models: Where are we, and where are we going? *arXiv preprint arXiv:2408.05109*.

Haoran Luo, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, et al. 2023. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. *arXiv preprint arXiv:2310.08975*.

Tanzim Mahmud, KM Azharul Hasan, Mahtab Ahmed, and Thwoi Hla Ching Chak. 2015. A rule based approach for nlp based query processing. In *2015 2nd international conference on electrical information and communication technologies (EICT)*, pages 78–82. IEEE.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via

large pre-trained language models: A survey. *ACM Comput. Surv.*, 56(2).

Pegah Naghshnejad, Gabriela Theis Marchan, Teslim Olayiwola, Jose Romagnoli, and Revati Kumar. 2024. Graph-based modeling and molecular dynamics for ion activity coefficient prediction in polymeric ion-exchange membranes.

OpenAI. 2023a. Chatgpt: A language model for generating human-like text. https://openai.com/chatgpt.

OpenAI. 2023b. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Xinyu Pi, Bing Wang, Yan Gao, Jiaqi Guo, Zhoujun Li, and Jian-Guang Lou. 2022. Towards robustness of text-to-sql models against natural and realistic adversarial table perturbation. *arXiv preprint arXiv:2212.09994*.

Mohammadreza Pourreza and Davood Rafiei. 2024a. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36.

Mohammadreza Pourreza and Davood Rafiei. 2024b. Dts-sql: Decomposed text-to-sql with small large language models. *arXiv preprint arXiv:2402.01117*.

Mohammadreza Pourreza, Ruoxi Sun, Hailong Li, Lesly Miculicich, Tomas Pfister, and Sercan O Arik. 2024. Sql-gen: Bridging the dialect gap for text-to-sql via synthetic data and model merging. *arXiv preprint arXiv:2408.12733*.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Juan Sequeda, Dean Allemang, and Bryon Jacob. 2024. A benchmark to understand the role of knowledge graphs on large language model's accuracy for question answering on enterprise sql databases. In *Proceedings of the 7th Joint Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, pages 1–12.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to sql queries. *arXiv preprint arXiv:2010.11246*.

Sowmya S Sundaram, Benjamin Solomon, Avani Khatri, Anisha Laumas, Purvesh Khatri, and Mark A Musen. 2024. Use of a structured knowledge base enhances metadata curation by large language models. *arXiv preprint arXiv:2404.05893*.

Munief Hassan Tahir, Sana Shams, Layba Fiaz, Farah Adeeba, and Sarmad Hussain. 2024. Benchmarking pre-trained large language models' potential across urdu nlp tasks. *arXiv preprint arXiv:2405.15453*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Pavlos Vougiouklis, Nikos Papasarantopoulos, Danna Zheng, David Tuckey, Chenxin Diao, Zhili Shen, and Jeff Pan. 2023. Fastrat: Fast and efficient cross-lingual text-to-sql semantic parsing. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 564–576.

Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, and Yu Sun. 2023. Pre-trained language models and their applications. *Engineering*, 25:51–65.

Lijie Wang, Ao Zhang, Kun Wu, Ke Sun, Zhenghua Li, Hua Wu, Min Zhang, and Haifeng Wang. 2020. Dusql: A large-scale and pragmatic chinese text-to-sql dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6923–6935.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Hanchen Xia, Feng Jiang, Naihao Deng, Cunxiang Wang, Guojiang Zhao, Rada Mihalcea, and Yue Zhang. 2024. Sql-craft: Text-to-sql through interactive refinement and enhanced reasoning. *arXiv preprint arXiv:2402.14851*.

Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-kbqa: Multi-turn interactions for knowledge base question answering with large language models. *arXiv preprint arXiv:2402.15131*.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, and Enhong Chen. 2023a. Large language models for generative information extraction: A survey. *arXiv preprint arXiv:2312.17617*.

Fangzhi Xu, Zhiyong Wu, Qiushi Sun, Siyu Ren, Fei Yuan, Shuai Yuan, Qika Lin, Yu Qiao, and Jun Liu. 2023b. Symbol-llm: Towards foundational symbol-centric interface for large language models. *arXiv preprint arXiv:2311.09278*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze

Shi, Zihan Li, et al. 2019a. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *arXiv preprint arXiv:1909.05378*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, et al. 2019b. Sparc: Cross-domain semantic parsing in context. *arXiv preprint arXiv:1906.02285*.

Bin Zhang, Yuxiao Ye, Guoqing Du, Xiaoru Hu, Zhishuai Li, Sun Yang, Chi Harold Liu, Rui Zhao, Ziyue Li, and Hangyu Mao. 2024a. Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation. *arXiv preprint arXiv:2403.02951*.

Chao Zhang, Yuren Mao, Yijiang Fan, Yu Mi, Yunjun Gao, Lu Chen, Dongfang Lou, and Jinshu Lin. 2024b. Finsql: Model-agnostic llms-based text-to-sql framework for financial analysis. *arXiv preprint arXiv:2401.10506*.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023. Act-sql: In-context learning for text-to-sql with automatically-generated chain-of-thought. *arXiv preprint arXiv:2310.17342*.

Qinggang Zhang, Junnan Dong, Hao Chen, Wentao Li, Feiran Huang, and Xiao Huang. 2024c. Structure guided large language model for sql generation. *arXiv preprint arXiv:2402.13284*.

Tingkai Zhang, Chaoyu Chen, Cong Liao, Jun Wang, Xudong Zhao, Hang Yu, Jianchao Wang, Jianguo Li, and Wenhui Shi. 2024d. Sqlfuse: Enhancing text-to-sql performance through comprehensive llm synergy. *arXiv preprint arXiv:2407.14568*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.