Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL

Zijin Hong[1], Zheng Yuan[2], Qinggang Zhang[2], Hao Chen[2], Junnan Dong[2], Feiran Huang[1], and Xiao Huang[2]*

[1]*Jinan University, Guangzhou, China*

[2]*The Hong Kong Polytechnic University, Hong Kong SAR, China*

hongzijin@stu2020.jnu.edu.cn, yzheng.yuan@connect.polyu.hk, qinggangg.zhang@connect.polyu.hk, sundaychenhao@gmail.com, hanson.dong@connect.polyu.hk, huangfr@jnu.edu.cn, xiaohuang@comp.polyu.edu.hk

***Abstract*—Generating accurate SQL from natural language questions (text-to-SQL) is a long-standing challenge due to the complexities in user question understanding, database schema comprehension, and SQL generation. Conventional text-to-SQL systems, comprising human engineering and deep neural networks, have made substantial progress. Subsequently, pre-trained language models (PLMs) have been developed and utilized for text-to-SQL tasks, achieving promising performance. As modern databases become more complex, the corresponding user questions also grow more challenging, causing PLMs with parameter constraints to produce incorrect SQL. This necessitates more sophisticated and tailored optimization methods, which, in turn, restricts the applications of PLM-based systems. Recently, large language models (LLMs) have demonstrated significant capabilities in natural language understanding as the model scale increases. Therefore, integrating LLM-based implementation can bring unique opportunities, improvements, and solutions to text-to-SQL research. In this survey, we present a comprehensive review of LLM-based text-to-SQL. Specifically, we propose a brief overview of the technical challenges and the evolutionary process of text-to-SQL. Then, we provide a detailed introduction to the datasets and metrics designed to evaluate text-to-SQL systems. After that, we present a systematic analysis of recent advances in LLM-based text-to-SQL. Finally, we discuss the remaining challenges in this field and propose expectations for future research directions.**

***Index Terms*—text-to-SQL, large language models, database, natural language understanding**

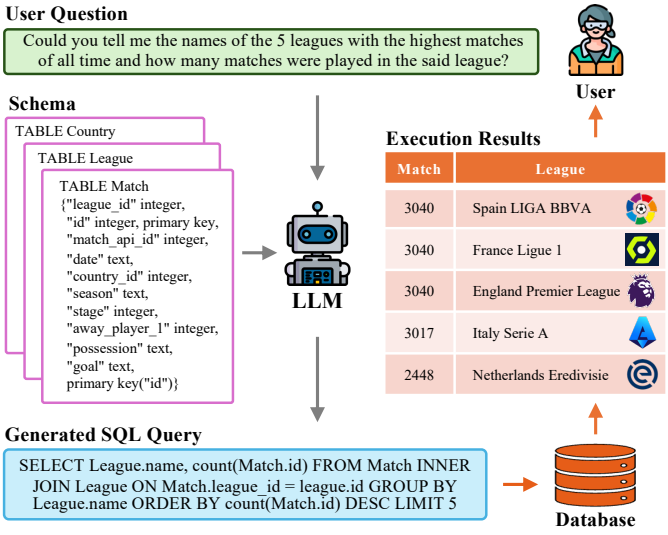

Fig. 1: An example for LLM-based text-to-SQL selected from the BIRD dataset. A user proposes a question about football leagues. The LLM takes the question and the schema of its corresponding database as the input and then generates an SQL query as the output. The SQL query can be executed in the database and retrieve the content *"The 5 leagues with the highest matches"* to answer the user question.

I. Introduction

TEXT-TO-SQL is a long-standing task in natural language processing research. It aims to convert (translate) natural language questions into database-executable SQL queries. Fig. 1 provides an example of a large language model-based (LLM-based) text-to-SQL system. Given a user question such as *"Could you tell me the names of the 5 leagues with the highest matches of all time and how many matches were played in the said league?"*, the LLM takes the question and its corresponding database schema as input and then generates an SQL query as output. This SQL query can be executed in the database to retrieve the relevant content to answer the user's question. The above system builds a natural language interface to the database (NLIDB) with LLMs. Since SQL remains one of the most widely used programming languages, with over half (51.52%) of professional developers using SQL in their work, it is notable that only around a third (35.29%) of those developers are systematically trained[1], the NLIDB enables non-skilled users to access structured databases like professional database engineers [1, 2] and also accelerates human-computer interaction [3]. Furthermore, amid the research hotspot of LLMs, text-to-SQL can provide a potential solution to the prevalent hallucination [4, 5] issue by incorporating realistic content from the database to fill the knowledge gaps of LLMs [6]. The significant value and potential for text-to-SQL have triggered a range of studies on its integration and optimization with LLMs [7–10]; consequently, LLM-based text-to-SQL remains a highly discussed research field within the NLP and database communities.

* Corresponding author.

[1]https://survey.stackoverflow.co/2023

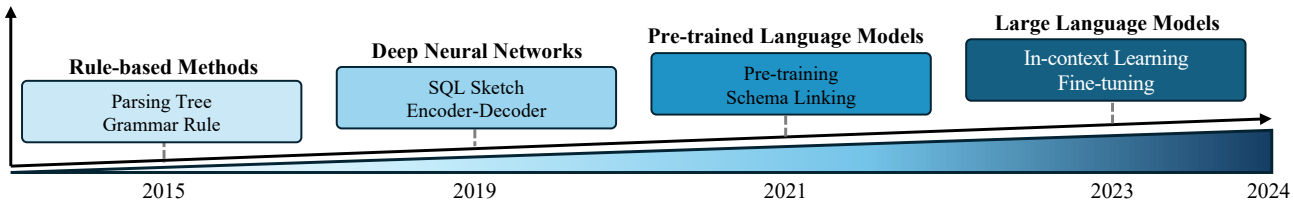

Fig. 2: A sketch of the evolutionary process for text-to-SQL research from the perspective of implementation paradigm. Each stage is presented with two representative implementation techniques. The timestamps for the stages are not exactly accurate; we set each timestamp according to the release time of the representative works of each implementation paradigm, with a margin of error of about one year before and after. The format is inspired from [29].

Previous studies have made notable progress in implementing text-to-SQL and have undergone a long evolutionary process. Early efforts were mostly based on well-designed rules and templates [11], specifically suitable for simple database scenarios. In recent years, with the heavy labor costs [12] brought by rule-based methods and the growing complexity of database environments [13–15], designing a rule or template for each scenario has become increasingly difficult and impractical. The development of deep neural networks has advanced the progress of text-to-SQL [16, 17], which can automatically learn a mapping from the user question to its corresponding SQL [18, 19]. Subsequently, pre-trained language models (PLMs) with strong semantic parsing capacity have become the new paradigm for text-to-SQL systems [20], boosting their performance to a new level [21–23]. Incremental research on PLM-based optimization, such as table content encoding [19, 24, 25] and pre-training [20, 26], has further advanced this field. Most recently, the LLM-based approaches implementing text-to-SQL through in-context learning (ICL) [8] and fine-tuning (FT) [10] paradigm, reaching state-of-the-art accuracy with the well-designed framework and stronger comprehension capability compared to PLMs.

The overall implementation details of LLM-based text-to-SQL can be divided into 3 aspects: **1. Question understanding**: The NL question is a semantic representation of the user's intention, which the corresponding generated SQL query is expected to align with; **2. Schema comprehension**: The schema provides the table and column structure of the database, and the text-to-SQL system is required to identify the target components that match the user question; **3. SQL generation**: This involves incorporating the above parsing and then predicting the correct syntax to generate executable SQL queries that can retrieve the required answer. The LLMs have proven to perform a good vanilla implementation [7, 27], benefiting from the more powerful semantic parsing capacity enabled by the richer training corpus [28, 29]. Further studies on enhancing the LLMs for question understanding [8, 9], schema comprehension [30, 31], and SQL generation [32] are being increasingly released.

Despite the significant progress made in text-to-SQL research, several challenges remain that hinder the development of robust and generalized text-to-SQL systems [73]. Related works in recent years have surveyed the text-to-SQL systems in deep learning approaches and provided insights into the previous deep neural network and PLM-based research [2, 29, 74]. In this survey, we aim to catch up with the recent advances and provide a comprehensive review of the current state-of-the-art models and approaches in LLM-based text-to-SQL. We begin by introducing the fundamental concepts and challenges associated with text-to-SQL, highlighting the importance of this task in various domains. We then delve into the evolution of the implementation paradigm for text-to-SQL systems, discussing the key advancements and breakthroughs in this field. After the overview, we provide a detailed introduction and analysis of the recent advances in text-to-SQL integrating LLMs. Specifically, the body of our survey covers a range of contents related to LLM-based text-to-SQL, including:

- Datasets and Benchmarks: We provide a detailed introduction to the commonly used datasets and benchmarks for evaluating LLM-based text-to-SQL systems. We discuss their characteristics, complexity, and the challenges they pose for text-to-SQL development and evaluation.
- Evaluation Metrics: We present the evaluation metrics used to assess the performance of LLM-based text-to-SQL systems, including both content matching-based and execution-based paradigms. We then briefly introduce the characteristics of each metric.
- Methods and Models: We present a systematic analysis of the different methods and models employed for LLM-based text-to-SQL, including in-context learning and fine-tuning-based paradigms. We discuss their implementation details, strengths, and adaptations specific to the text-to-SQL task from various implementation perspectives.
- Expectations and Future Directions: We discuss the remaining challenges and limitations of LLM-based text-to-SQL, such as real-world robustness, computational efficiency, data privacy, and extensions. We also outline potential future research directions and opportunities for improvement and optimization.

We hope this survey provides a clear overview of recent studies and inspires future research. Fig. 3 shows a taxonomy tree that summarizes the structure and contents of our survey.

II. Overview

Text-to-SQL is a task that aims to convert natural language questions into corresponding SQL queries that can be executed in a relational database. Formally, given a user question $\mathcal{Q}$ (also known as a user query, natural language question, etc.)

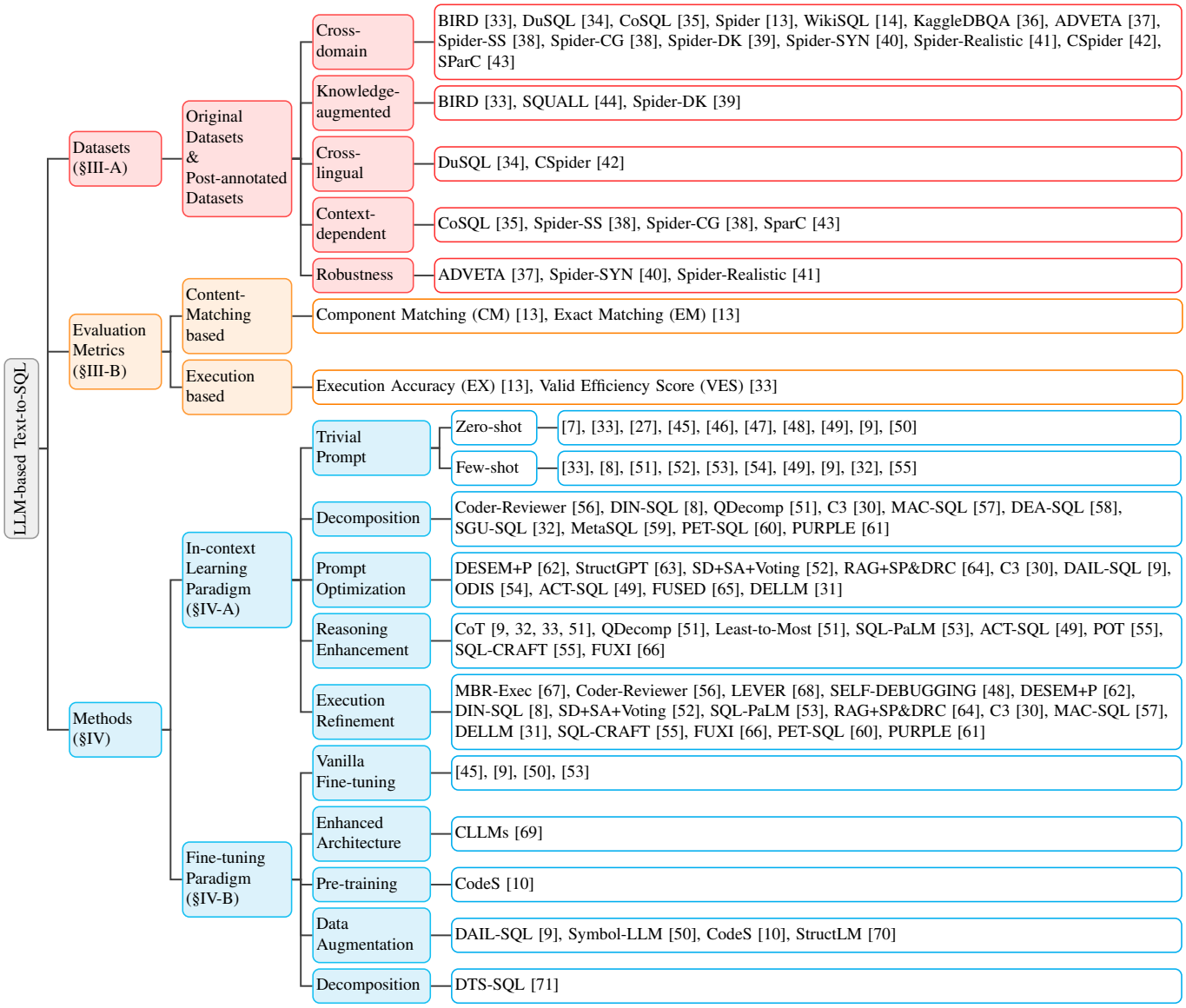

Fig. 3: Taxonomy tree of the research in LLM-based text-to-SQL. The display order in each node is organized by the released time. The format is adapted from [72].

and a database schema $\mathcal{S}$, the goal of the task is to generate an SQL query Y that retrieves the required content from the database to answer the user question. Text-to-SQL has the potential to democratize access to data by allowing users to interact with databases using natural language without the need for specialized knowledge of SQL programming [75]. This can benefit various domains, such as business intelligence, customer support, and scientific research, by enabling non-skilled users to easily retrieve target content from databases and facilitating more efficient data analysis.

A. Challenges in Text-to-SQL

The technical challenges for text-to-SQL implementations can be summarized as follows:

1) Linguistic Complexity and Ambiguity: Natural language questions often contain complex linguistic representations, such as nested clauses, coreferences, and ellipses, which make it challenging to map them accurately to the corresponding part of SQL queries [41]. Additionally, natural language is inherently ambiguous, with multiple possible representations for a given user question [76, 77]. Resolving these ambiguities and understanding the intent behind the user question requires deep natural language understanding and the capability to incorporate context and domain knowledge [33].

2) Schema Understanding and Representation: To generate accurate SQL queries, text-to-SQL systems need to have a comprehensive understanding of the database schema, including table names, column names, and relationships between various tables. However, the database schema can be complex and vary significantly across different domains [13]. Representing and encoding the schema information in a way that can be effectively utilized by the text-to-SQL model is a challenging task.

3) Rare and Complex SQL Operations: Some SQL queries involve rare or complex operations and syntax in challenging scenarios, such as nested sub-queries, outer joins, and window functions. These operations are less frequent in the training data and pose challenges for text-to-SQL systems to generate accurately. Designing models that can be generalized to a wide range of SQL operations, including rare and complex scenarios, is an essential consideration.

4) Cross-Domain Generalization: Text-to-SQL systems often struggle to generalize across various database scenarios and domains. Models trained on a specific domain may not perform well on the proposed questions from other domains due to the variety in vocabulary, database schema structure, and question patterns. Developing systems that can effectively generalized to new domains with minimal domain-specific training data or fine-tuning adaptation is a significant challenge [78].

B. Evolutionary Process

The research field of text-to-SQL has witnessed significant advancements over the years in the NLP community, having evolved from rule-based methods to deep learning-based approaches and, more recently, to integrating pre-trained language models (PLMs) and large language models (LLMs), a sketch of the evolutionary process is shown in Fig. 2.

1) Rule-based Methods: Early text-to-SQL systems relied heavily on rule-based methods [11, 12, 26], where manually crafted rules and heuristics were used to map natural language questions to SQL queries. These approaches often involved extensive feature engineering and domain-specific knowledge. While rule-based methods achieved success in specific simple domains, they lacked the flexibility and generalization capabilities needed to handle diverse and complex questions.

2) Deep Learning-based Approaches: With the rise of deep neural networks, sequence-to-sequence models and encoder-decoder structures, such as LSTMs [79] and transformers [17], were adapted to generate SQL queries from natural language input [19, 80]. Typically, RYANSQL [19] introduced techniques like intermediate representations and sketch-based slot filling to handle complex questions and improve cross-domain generalization. Recently, researchers introduced graph neural networks (GNNs) for text-to-SQL tasks by leveraging schema dependency graphs to capture the relationships between database elements [18, 81].

3) PLM-based Implementation: Pre-trained language models (PLMs) have emerged as a powerful solution for text-to-SQL, leveraging the vast amounts of linguistic knowledge and semantic understanding captured during the pre-training process. The early adoption of PLMs in text-to-SQL primarily focused on fine-tuning off-the-shelf PLMs, such as BERT [24] and RoBERTa [82], on standard text-to-SQL datasets [13, 14]. These PLMs, pre-trained on large amounts of training corpus, captured rich semantic representations and language understanding capabilities. By fine-tuning them on text-to-SQL tasks, researchers aimed to leverage the semantic and linguistic understanding of PLMs to generate accurate SQL queries [20, 80, 83]. Another line of research focuses on incorporating schema information into PLMs to improve their understanding of database structures and enable them to generate more executable SQL queries. Schema-aware PLMs are designed to capture the relationships and constraints present in the database structure [21].

4) LLM-based Implementation: Large language models (LLMs), such as GPT series [84–86], have gained significant attention in recent years due to their capability to generate coherent and fluent text. Researchers have started exploring the potential of LLMs for text-to-SQL by leveraging their extensive knowledge reserves and superior generation capabilities [7, 9]. These approaches often involve prompt engineering to guide the proprietary LLMs in SQL generation [47] or fine-tuning the open-source LLMs on text-to-SQL datasets [9].

The integration of LLMs in text-to-SQL is still an emerging research area with significant potential for further exploration and improvement. Researchers are investigating approaches to better leverage the knowledge and reasoning capabilities of LLMs, incorporate domain-specific knowledge [31, 33], and develop more efficient fine-tuning strategies [10]. As the field continues to evolve, we anticipate the development of more advanced and superior LLM-based implementations that will elevate the performance and generalization capabilities of text-to-SQL to new heights.

III. Benchmarks & Evalution

In this section, we introduce the benchmarks for text-to-SQL, encompassing well-known datasets and evaluation metrics.

A. Datasets

As shown in Tab. I, we classify the datasets into 'Original Datasets' and 'Post-annotated Datasets' based on whether they were released with the original dataset and databases or created by adapting existing datasets and databases with special settings. For the original datasets, we provide a detailed analysis, including the number of examples, the number of databases, the number of tables per database, and the number of rows per database. For the post-annotated datasets, we identify their source dataset and describe the special setting applied to them. To illustrate the potential opportunities of each dataset, we annotate them based on their characteristics. The annotations are listed in the rightmost column of Tab. I, which we will discuss in detail below.

1) Cross-domain Dataset: This refers to datasets where the background information of different databases comes from various domains. Since real-world text-to-SQL applications often involve databases from multiple domains, most original text-to-SQL datasets [13, 14, 33–36] and post-annotated datasets [37–43] are in the cross-domain setting to fit well with the requirements of cross-domain applications.

2) Knowledge-augmented Dataset: Interest in incorporating domain-specific knowledge into text-to-SQL tasks has increased significantly in recent years. BIRD [33] employs human database experts to annotate each text-to-SQL sample with external knowledge, categorized into numeric reasoning knowledge, domain knowledge, synonym knowledge, and value illustration. Similarly, Spider-DK [39] defines and adds five types of domain knowledge for a human-curated version of the

TABLE I: The statistics and analysis of well-known datasets of text-to-SQL ordered by release time. The original dataset indicates that the dataset is designed with a corresponding database, while post-annotated datasets involve annotating new components within existing datasets and databases rather than releasing a new database.

Original Dataset	Release Time	#Example	#DB	#Table/DB	#Row/DB	Characteristics
BIRD [33]	May-2023	12,751	95	7.3	549K	Cross-domain, Knowledge-augmented
KaggleDBQA [36]	Jun-2021	272	8	2.3	280K	Cross-domain
DuSQL [34]	Nov-2020	23,797	200	4.1	-	Cross-domain, Cross-lingual
SQUALL [44]	Oct-2020	11,468	1,679	1	-	Knowledge-augmented
CoSQL [35]	Sep-2019	15,598	200	-	-	Cross-domain, Context-dependent
Spider [13]	Sep-2018	10,181	200	5.1	2K	Cross-domain
WikiSQL [14]	Aug-2017	80,654	26,521	1	17	Cross-domain
Post-annotated Dataset	**Release Time**	**Source Dataset**	**Special Setting**			**Characteristics**
ADVETA [37]	Dec-2022	Spider, etc.	Adversarial table perturbation			Robustness
Spider-SS&CG [38]	May-2022	Spider	Splitting example into sub-examples			Context-dependent
Spider-DK [39]	Sep-2021	Spider	Adding domain knowledge			Knowledge-augmented
Spider-SYN [40]	Jun-2021	Spider	Manual synonym replacement			Robustness
Spider-Realistic [41]	Oct-2020	Spider	Removing column names in question			Robustness
CSpider [42]	Sep-2019	Spider	Chinese version of Spider			Cross-lingual
SParC [43]	Jun-2019	Spider	Annotate conversational contents			Context-dependent

Spider dataset [13]: SELECT columns mentioned by omission, simple inference required, synonyms substitution in cell value word, one non-cell value word generate a condition, and easy to conflict with other domains. Both studies found that human-annotated knowledge significantly improves SQL generation performance for samples requiring external domain knowledge. Additionally, SQUALL [44] manually annotates alignments between the words in NL questions and the entities in SQL, providing finer-grained supervision than other datasets.

3) Context-dependent Dataset: SParC [43] and CoSQL [35] explore context-dependent SQL generation by constructing a conversational database querying system. Unlike traditional text-to-SQL datasets that only have a single question-SQL pair for one example, SParC decomposes the question-SQL examples in the Spider dataset into multiple sub-question-SQL pairs to construct a simulated and meaningful interaction, including inter-related sub-questions that aid SQL generation, and unrelated sub-questions that enhance data diversity. CoSQL, in comparison, involves conversational interactions in natural language, simulating real-world scenarios to increase complexity and diversity. Additionally, Spider-SS&CG [38] splits the NL question in the Spider dataset [13] into multiple sub-questions and sub-SQLs, demonstrating that training on these sub-examples can improve a text-to-SQL system's generalization ability on out-of-distribution samples.

4) Robustness Dataset: Evaluating the accuracy of text-to-SQL systems with polluted or perturbed database contents (e.g., schema and tables) is crucial for assessing robustness. Spider-Realistic [41] removes explicit schema-related words from the NL questions, while Spider-SYN [40] replaces them with manually selected synonyms. ADVETA [37] introduces adversarial table perturbation (ATP), which perturbs tables by replacing original column names with misleading alternatives and inserting new columns with high semantic associations but low semantic equivalency. These perturbations lead to significant drops in accuracy, as a text-to-SQL system with low robustness may be misled by incorrect matches between tokens in NL questions and database entities.

5) Cross-lingual Dataset: SQL keywords, function names, table names, and column names are typically written in English, posing challenges for applications in other languages. CSpider [42] translates the Spider dataset into Chinese, identifying new challenges in word segmentation and cross-lingual matching between Chinese questions and English database contents. DuSQL [34] introduces a practical text-to-SQL dataset with Chinese questions and database contents provided in both English and Chinese.

B. Evaluation Metrics

We introduce four widely used evaluation metrics for the text-to-SQL task as follows: Component Matching and Exact Matching, which are based on SQL content matching, and Execution Accuracy and Valid Efficiency Score, which are based on execution results.

1) Content Matching-based Metrics: SQL content matching metrics focus on comparing the predicted SQL query with the ground truth SQL query based on their structural and syntactic similarities.

- **Component Matching (CM)** [13] evaluates the performance of text-to-SQL system by measuring the exact match between predicted and ground truth SQL components—SELECT, WHERE, GROUP BY, ORDER BY, and KEYWORDS—using the F1 score. Each component is decomposed into sets of sub-components and compared for an exact match, accounting for SQL components without order constraints.
- **Exact Matching (EM)** [13] measures the percentage of examples whose predicted SQL query is identical to the ground truth SQL query. A predicted SQL is considered correct only if all its components, as described in CM, match exactly with those of the ground truth query.

2) Execution-based Metrics: Execution result metrics assess the correctness of the generated SQL query by comparing the results obtained from executing the query on the target database with the expected results.

- **Execution Accuracy (EX)** [13] measures the correctness of a predicted SQL query by executing it in the corresponding database and comparing the executed results with the results obtained by the ground truth query.
- **Valid Efficiency Score (VES)** [33] is defined to measure the efficiency of valid SQL queries. A valid SQL query is a predicted SQL query whose executed results exactly match the ground truth results. Specifically, VES evaluates both the efficiency and accuracy of predicted SQL queries. For a text dataset with N examples, VES can be computed by:

$$\text{VES} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(V_n, \hat{V}_n) \cdot \mathbf{R}(Y_n, \hat{Y}_n), \tag{1}$$

where $\hat{Y}_n$ and $\hat{V}_n$ are the predicted SQL query and its executed results and Y_n and V_n are the ground truth SQL query and its corresponding executed results, respectively. $\mathbb{1}(V_n, \hat{V}_n)$ is an indicator function, where:

$$\mathbb{1}(V_n, \hat{V}_n) = \begin{cases} 1, V_n = \hat{V}_n \\ 0, V_n \neq \hat{V}_n \end{cases} \tag{2}$$

Then, $\mathbf{R}(Y_n, \hat{Y}_n) = \sqrt{E(Y_n)/E(\hat{Y}_n)}$ denotes the relative execution efficiency of the predicted SQL query in comparison to ground-truth query, where $E(\cdot)$ is the execution time of each SQL in the database. BIRD benchmark [33] ensures the stability of this metric by computing the average of $\mathbf{R}(Y_n, \hat{Y}_n)$ over 100 runs for each example.

Most of the recent LLM-based text-to-SQL studies focus on these four datasets: Spider [13], Spider-Realistic [41], Spider-SYN [40], and BIRD [33]; and these three evaluation methods: EM, EX, and VES, we will primarily focus on them in the following analysis.

IV. Methods

The implementation of current LLM-based applications mostly relies on in-context learning (ICL) (prompt engineering) [87–89] and fine-tuning (FT) [90, 91] paradigms since the powerful proprietary models and well-architected open-source models are being released in large quantities [45, 86, 92–95]. LLM-based text-to-SQL systems follow these paradigms for implementation. In this survey, we will discuss them accordingly.

A. In-context Learning

Through extensive and widely recognized research, prompt engineering has been proven to play a decisive role in the performance of LLMs [28, 96], also impacting the SQL generation in different prompt styles [9, 46]. Necessarily, developing text-to-SQL methods in the in-context learning (ICL) paradigm is valuable for achieving promising improvement. The implementation of LLM-based text-to-SQL process to generate executable SQL query Y can be formulated as:

$$Y = f(\mathcal{Q}, \mathcal{S}, \mathcal{I} \mid \theta), \tag{3}$$

TABLE II: Typical methods used for in-context learning (ICL) in LLM-based text-to-SQL. The full table of existing methods with categorization $\mathbf{C}_{1:4}$ and more details are listed in Tab. III.

Methods	Adopted by	Applied LLMs
$\mathbf{C}_0$-Trivial Prompt	Zero-shot [7]	ChatGPT
	Few-shot [9]	ChatGPT
$\mathbf{C}_1$-Decomposition	DIN-SQL [8]	GPT-4
$\mathbf{C}_2$-Prompt Optimization	DAIL-SQL [9]	GPT-4
$\mathbf{C}_3$-Reasoning Enhancement	ACT-SQL [49]	GPT-4
$\mathbf{C}_4$-Execution Refinement	LEVER [68]	Codex

where $\mathcal{Q}$ represents the user question. $\mathcal{S}$ is the database schema/content, which can be decomposed as $\mathcal{S} = \langle \mathcal{C}, \mathcal{T}, \mathcal{K} \rangle$, where $\mathcal{C} = \{c_1, c_2, ...\}$ and tables $\mathcal{T} = \{t_1, t_2, ...\}$ represent the collection of various columns and tables, $\mathcal{K}$ is the potentially external knowledge (e.g. foreign key relationships [49], schema linking [30] and domain knowledge [31, 33]). $\mathcal{I}$ represents the instruction for the text-to-SQL task, which provides indicative guidance to trigger the LLMs for generating an accurate SQL query. $f(\cdot \mid \theta)$ is a LLM with parameter θ. In the in-context learning (ICL) paradigm, we utilize an off-the-shelf text-to-SQL model (i.e., parameter θ of the model is frozen) for generating the predicted SQL query. Various well-designed methods within the ICL paradigm have been adopted for LLM-based text-to-SQL tasks. We group them into five categories $\mathbf{C}_{0:4}$, including $\mathbf{C}_0$-Trivial Prompt, $\mathbf{C}_1$-Decomposition, $\mathbf{C}_2$-Prompt Optimization, $\mathbf{C}_3$-Reasoning Enhancement, and $\mathbf{C}_4$-Execution Refinement, the representative methods of each category are given in Tab. II.

$\mathbf{C}_0$-Trivial Prompt: Trained through massive data, LLMs have a strong overall proficiency in different downstream tasks with zero-shot and few-shot prompting [90, 97, 98], which is widely recognized and used in real-world applications. In our survey, we categorized the above prompting approaches without the well-designed framework as trivial prompts (vanilla prompt engineering). As introduced above, Eq. 3 formulated the process of LLM-based text-to-SQL, which can also represent zero-shot prompting. The overall input $\mathcal{P}_0$ can be obtained by concatenating $\mathcal{I}$, $\mathcal{S}$ and $\mathcal{Q}$:

$$\mathcal{P}_0 = \mathcal{I} \oplus \mathcal{S} \oplus \mathcal{Q}. \tag{4}$$

To regulate the prompting process, the OpenAI demonstration[2] is set as the standard (trivial) prompt [30] for text-to-SQL.

Zero-shot: Many research works [7, 27, 46] utilize zero-shot prompting, studying mainly on the influence of the style of prompt construction and the zero-shot performance of various LLMs for text-to-SQL. As an empirical evaluation, [7] evaluates the baseline text-to-SQL capabilities of different early-developed LLMs [85, 99, 100] and the performance for different prompting styles. The results indicate that prompt design is critical for the performance, with error analysis, [7] propose more database content can harm the overall accuracy. Since ChatGPT emerged with impressive capabilities in conversational scenarios and code generation [101], [27] assesses its performance of text-to-SQL. With zero-shot settings,

[2]The prompt style that follows the official document from OpenAI platform: https://platform.openai.com/examples/default-sql-translate

TABLE III: Well-designed methods used in in-context learning (ICL) paradigm for LLM-based text-to-SQL ordered by release time. The methods are grouped in four categories based on their implementation perspective: $\mathbf{C}_1$-Decomposition, $\mathbf{C}_2$-Prompt Optimization, $\mathbf{C}_3$-Reasoning Enhancement, $\mathbf{C}_4$-Execution Refinement. The method in multiple categories will be introduced respectively. *There are multiple applied LLMs in the corresponding method; we present the selected LLM with representative performance. †CoT method are reported in multiple venues: NeurIPS'23 [33], EMNLP'23 [51], VLDB'24 [9], arXiv'24 [32].

Methods	Applied LLMs	Dataset	Metrics	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$	Release Time	Publication Venue
MBR-Exec [67]	Codex	[13]	EX				✓	Apr-2022	EMNLP'22
Coder-Reviewer [56]	Codex	[13]	EX	✓			✓	Nov-2022	ICML'23
LEVER [68]	Codex	[13]	EX				✓	Feb-2023	ICML'23
SELF-DEBUGGING [48]	StarCoder*	[13]	EX				✓	Apr-2023	ICLR'24
DESEM+P [62]	ChatGPT	[13, 40]	EX		✓		✓	Apr-2023	PRICAI'23
DIN-SQL [8]	GPT-4*	[13, 33]	EX, EM, VES	✓			✓	Apr-2023	NeurIPS'23
CoT [9, 32, 33, 51]	GPT-4	[13, 33, 41]	EX, VES			✓		May-2023	Multiple Venues†
StructGPT [63]	ChatGPT*	[13, 40, 41]	EX		✓			May-2023	EMNLP'23
SD+SA+Voting [52]	ChatGPT*	[13, 40, 41]	EX		✓		✓	May-2023	EMNLP'23 Findings
QDecomp [51]	Codex	[13, 41]	EX	✓		✓		May-2023	EMNLP'23
Least-to-Most [51]	Codex	[13]	EX			✓		May-2023	EMNLP'23
SQL-PaLM [53]	PaLM-2	[13]	EX			✓	✓	May-2023	arXiv'23
RAG+SP&DRC [64]	ChatGPT	[13]	EX		✓		✓	Jul-2023	ICONIP'23
C3 [30]	ChatGPT	[13]	EX	✓	✓		✓	Jul-2023	arXiv'23
DAIL-SQL [9]	GPT-4*	[13, 33, 41]	EX, EM, VES		✓			Aug-2023	VLDB'24
ODIS [54]	Codex*	[13]	EX		✓			Oct-2023	EMNLP'23 Findings
ACT-SQL [49]	GPT-4*	[13, 40]	EX, EM		✓	✓		Oct-2023	EMNLP'23 Findings
MAC-SQL [57]	GPT-4*	[13, 33]	EX, EM, VES	✓			✓	Dec-2023	arXiv'23
DEA-SQL [58]	GPT-4	[13]	EX	✓				Feb-2024	ACL'24 Findings
FUSED [65]	ChatGPT*	[13]	EX		✓			Feb-2024	arXiv'24
DELLM [31]	GPT-4*	[13, 33]	EX, VES		✓		✓	Feb-2024	ACL'24 Findings
SGU-SQL [32]	GPT-4*	[13, 33]	EX, EM	✓				Feb-2024	arXiv'24
POT [55]	GPT-4*	[13, 33]	EX			✓		Feb-2024	arXiv'24
SQL-CRAFT [55]	GPT-4*	[13, 33]	EX			✓	✓	Feb-2024	arXiv'24
FUXI [66]	GPT-4*	[33]	EX			✓	✓	Feb-2024	arXiv'24
MetaSQL [59]	GPT-4*	[13]	EX, EM	✓				Feb-2024	ICDE'24
PET-SQL [60]	GPT-4	[13]	EX	✓			✓	Mar-2024	arXiv'24
PURPLE [61]	GPT-4*	[13, 40, 41]	EX, EM	✓			✓	Mar-2024	ICDE'24

the results demonstrate that ChatGPT has a promising text-to-SQL performance compared to the state-of-the-art PLM-based systems. For fair comparability, [47] reveal effective prompt construction for LLM-based text-to-SQL; they study different styles of prompt construction and make conclusions of zero-shot prompt design based on the comparisons.

Primary keys and foreign keys carry contiguous knowledge of different tables. [49] studies their impact by incorporating these keys into various prompt styles with different database content to analyze zero-shot prompting results. A benchmark evaluation [9] also studies the influence of foreign keys, with five different prompt representation styles, each style can be considered as the permutation and combination of the instruction, rule implication, and foreign key. Apart from the foreign key, this study also explores zero-shot prompting combined with the rule implication *"no explanation"* to collect concise outputs. Empowered by the annotated external knowledge of human experts, [33] follows the standard prompting and achieves improvement by incorporating the provided annotated oracle knowledge.

With the explosion of open-source LLMs, according to the results of similar evaluation, these models are also capable of zero-shot text-to-SQL task [45, 46, 50], especially code generation models [46, 48]. For zero-shot prompting optimization, [46] raises a challenge for designing an effective prompt template for LLMs; the former prompt construction lacks structure uniformity, which makes it hard to find out a concrete element within a prompt constructing template influences the performance of LLMs. They address this challenge by investigating a more unified series of prompt templates warping with different prefixes, infixes, and postfixes.

Few-shot: The technique of few-shot prompting is widely used in both practical applications and well-designed research, which has been proven efficient for eliciting better performance of LLMs [28, 102]. The overall input prompt of the few-shot approach LLM-based text-to-SQL can be formulated as an extension of Eq. 3:

$$\mathcal{P}_n = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n\} \oplus \mathcal{P}_0, \tag{5}$$

where the $\mathcal{P}_n$ represent the input prompt for n-shot learning, n is the provided instances (examples) number; $\mathcal{F}$ denote the few-shot instance, which can be decomposed as $\mathcal{F}_i = (\mathcal{S}_i, \mathcal{Q}_i, Y_i)$, i is the serial number of instances. The study of few-shot prompting focuses on the number of representations and few-shot instance selection.

As empirical studies, few-shot prompting for text-to-SQL are evaluated in multiple datasets with various LLMs [8, 32], exhibit a solid performance during the comparison with zero-shot prompting. [33] provides a 1-shot detailed example to trigger the text-to-SQL model for generating accurate SQL. [55] study the effect of the number of few-shot examples. [52] focus on the sampling strategies by studying the similarity and the diversity between different demonstrations, setting random sampling as the baseline, and evaluating different strategies

and their combination for comparison. Furthermore, above the similarity-based selection, [9] evaluated masked question similarity selection and the upper limit of similarity approaches with various numbers of few-shot examples. A study of difficult-level samples selection [51] compared the performance of few-shot Codex [100], with random selection and difficulty-based selection for few-shot instances on difficulty categorized dataset [13, 41]. Three difficulty-based selection strategies are devised based on the number of selected samples at different difficulty levels. [49] utilize a hybrid strategy for selecting samples, which combines static examples and similarity-based dynamic examples for few-shot prompting. In their settings, they also evaluate the impact of different input schema styles and various static and dynamic exemplar numbers.

The impact of cross-domain few-shot examples is also being studied [54]. When incorporating in-domain and out-of-domain examples with different numbers, the in-domain demonstration outperforms zero-shot and out-of-domain examples, which will also perform better as the number of examples rises. To explore the detailed construction of input prompt, [53] compare the concise and verbose prompt design approaches. The former style splits the schema, the column names, and the primary and foreign keys by bar, and the latter organizes them as natural language descriptions.

C_1-Decomposition: As an intuitive solution, decomposing a challenging user question into simpler sub-questions or using multi-components for implementation can reduce the complexity of the overall text-to-SQL task [8, 51]. Dealing with less complexity, LLMs have the potential to elicit more accurate SQL generation. The decomposition-based approaches for LLM-based text-to-SQL are categorized into two paradigms: **(1) sub-task decomposition**, provides additional parsing to assist the final SQL generation by decomposing the overall text-to-SQL task into more manageable effective sub-tasks (e.g., schema linking [71], domain classification [54]). **(2) sub-question decomposition**, divides the user question into sub-questions to reduce the question's complexity and difficulty, then generates the sub-SQL by solving these questions to deduce the final SQL query. The technical novelty of the decomposition paradigm

DIN-SQL [8] proposed a decomposed in-context learning method consisting of four modules: schema linking, classification & decomposition, SQL generation, and self-correction. DIN-SQL first generates the schema linking between the user question and the target database; the following module decomposes the user question into correlated sub-questions and does a difficulty classification. Based on the above information, the SQL generation module generates a corresponding SQL, and the self-correction module identifies and corrects the potential errors in the predicted SQL. This approach comprehensively considers the sub-questions decomposition as a module of sub-tasks decomposition. Coder-Reviewer [56] framework proposed a re-ranking method, combining Coder models for the generation and Reviewer models to evaluate the likelihood of the instruction. Refer to the Chain-of-Thought [103] and Least-to-Most prompting [104], QDecomp [51] introduce question decomposition prompting, which follows the question reduction stage in least-to-most prompting and instruct the LLM to decompose the original complex question as the intermediate reasoning steps. C3 [30] consists of three key components: clear prompting, calibration bias prompting, and consistency; these components are accomplished by assigning ChatGPT with different tasks. Firstly, the clear prompting component generates the schema linking and the distilled question-relevant schema as a clear prompt. Then, a multi-turn dialogue about text-to-SQL hints is utilized as a calibration bias prompt, which combines with the clear prompt to guide the SQL generation. The generated SQL queries are selected by consistency and execution-based voting to get the final SQL. MAC-SQL [57] presents a multi-agent collaborating framework; the text-to-SQL process is finished as the collaboration of the agents: Selector, Decomposer, and Refiner. The Selector preserves relevant tables for user questions; the Decomposer breaks down user questions into sub-questions and provides solutions; finally, the Refiner validates and refines the defective SQL. DEA-SQL [58] introduces a workflow paradigm aiming to enhance the attention and problem-solving scope of LLM-based text-to-SQL through decomposition. This method decomposes the overall task, enabling the SQL generation module to have the corresponding prerequisite (information determination, question classification) and subsequent (self-correction, active learning) sub-tasks. The workflow paradigm allows the LLM to generate more accurate SQL queries. SGU-SQL [32] is a structure-to-SQL framework, leveraging the inherent structure information to assist SQL generation. Specifically, the framework constructs a graph structure for the user question and the corresponding database respectively, then uses the encoded graphs to construct structure linking [105, 106]. A meta operator decomposes the user question with a grammar tree and finally designs the input prompt with meta-operation in SQL. MetaSQL [59] introduces a three-stage approach for SQL generation: decomposition, generation, and ranking. The decomposition stage uses semantic decomposition and metadata composition to process the user question. Taking the previously processed data as input, a text-to-SQL model using metadata-conditioned generation to generate some candidate SQL queries. Finally, a two-stage ranking pipeline is applied to get a global-optimal SQL query. PET-SQL [60] proposed a prompt-enhanced two-stage framework. Firstly, an elaborated prompt instructs the LLMs to generate preliminary SQL (PreSQL) where some few-shot demonstrations are selected based on similarity. Then, schema linking is found based on PreSQL and combined to prompt the LLMs to generate the Final SQL (FinSQL). Finally, multiple LLMs are utilized to generate a FinSQL, ensuring consistency based on the execution results.

C_2-Prompt Optimization: As previously introduced, few-shot learning is widely studied for prompting LLMs [85]. For LLM-based text-to-SQL with in-context learning, trivial few-shot approaches obtained promising results [8, 9, 33], further optimization of few-shot prompting has the potential to elicit better performance. Since the accuracy of SQL generation in off-the-shelf LLMs largely depends on the quality of the corresponding input prompt [107], many decisive factors that can influence the quality of the prompt have become points of focus of the current research [9] (e.g., quality and quantity in the few-shot organization, the similarity between user

questions and few-shots instances, external knowledge/hints) The process of improving prompt quality can be summarized as the prompt's optimization, including **advanced few-shot sampling strategies**, **schema information augmentation**, and **external knowledge integration**.

DESEM [62] is a prompt engineering framework with de-semanticization and skeleton retrieval. The framework first employs domain-specific words masking module to remove the semantic tokens in user questions that preserve the intentions. It then utilizes an adjustable prompting module that retrieves the few-shot examples with identical question intentions and incorporates schema-relevance filtering to guide the LLM's SQL generation. The QDecomp [51] framework introduces the InterCOL mechanism to incrementally incorporate the decomposed sub-questions with correlative table and column names. With difficulty-based selection, the few-shot examples for QDecomp are difficult-level sampled. Besides similarity-diversity sampling, [52] proposed SD+SA+Voting (Similarity-Diversity+Schema Augmentation+Voting) sampling strategy. They first employ semantic similarity and k-Means cluster diversity for sampling few-shot examples and then enhance the prompt with schema knowledge (semantic or structure augmentation). C3 [30] framework comprises a clear prompting component, which takes the question and schema as the LLMs input, generates a clear prompt that includes a schema that removes the redundant information irrelevant to the user question and a schema linking, and also a calibration component providing hints. The LLMs take their composition as context-augmented prompts for SQL generation. A retrieval-augmented framework is introduced with sample-aware prompting [64], which simplifies the original question and extracts the question skeleton from the simplified question, then finishes the sample retrieval in the repository according to skeleton similarities. The retrieved samples are combined with the original question for few-shot prompting. ODIS [54] introduces the selection of a sample with out-of-domain demonstrations and in-domain synthetic data, which retrieves few-shot demonstrations from hybrid sources to augment the prompt representations. DAIL-SQL [9] proposed a novel approach to address the issues in few-shot sampling and organization, presenting a better balance between the quality and quantity of few-shot examples. DAIL Selection first masks domain-specific words in user and few-shot example questions, then ranks the candidate examples based on the embedded Euclidean distance. Meanwhile, the similarity between the pre-predicted SQL queries is calculated. Finally, the selection mechanism obtains the similarity-sorted candidates according to the pre-set criteria. The few-shot examples are guaranteed good similarity with both questions and SQL queries with this method. ACT-SQL [49] proposed dynamic examples in few-shot prompting, which is selected according to similarity score. FUSED [65] are presented to build a high-diversity demonstrations pool through human-free multiple-iteration synthesis to improve the diversity of the few-shot demonstrations. The pipeline of FUSED samples the demonstrations to be fused by clustering, then fuse the sampled demonstrations to construct the pool to enhance few-shot learning. Knowledge-to-SQL [31] framework aims to build a Data Expert LLM (DELLM) to provide knowledge for SQL generation. The DELLM is trained by supervised fine-tuning using human expert annotations [33] and further refined by preference learning with the database's feedback. DELLM generates four categories of knowledge, the well-designed methods (e.g. DAIL-SQL [9], MAC-SQL [57]) incorporating the generated knowledge to achieve better performance for LLM-based text-to-SQL with in-context learning.

C_3-Reasoning Enhancement: LLMs have exhibited promising capabilities in tasks involving commonsense reasoning, symbolic reasoning, and arithmetic reasoning [108]. For text-to-SQL tasks, numeric and synonym reasoning frequently occur in realistic scenarios [33, 41]. The prompting strategies for LLMs' reasoning have the potential to enhance SQL generation capabilities. Recent studies primarily focus on integrating well-designed reasoning-enhanced methods for text-to-SQL adaptation, improving LLMs to address the challenge about **complex questions that require sophisticated reasoning**[3] and the **self-consistency in SQL generation**.

Chain-of-Thoughts (CoT) prompting technique [103] involves a comprehensive reasoning process that guides LLMs towards accurate deduction, eliciting reasoning in LLMs. The study of LLM-based text-to-SQL utilizes CoT prompting as rule implication [9], which setting the instruction *"Let's think step by step"* in prompt construction [9, 32, 33, 51]. However, the straightforward (original) CoT strategy has not demonstrated the potential in text-to-SQL tasks that it has in other reasoning tasks; studying CoT for adaptations is still an ongoing research [51]. Since CoT prompting always uses static examples with human annotation for demonstrations, which requires empirical judgment for the effective selection of few-shot examples, and manual annotating is also an essential need. As a solution, ACT-SQL [49] proposed a method to generate CoT examples automatically. Specifically, given a question, ACT-SQL truncates a set of slices of the question and then enumerates every column appearing in the corresponding SQL query. Each column will be linked with its most relevant slice through the similarity function and appended to the CoT prompt. Through systematical study for enhancing LLMs SQL generation incorporating CoT prompting, QDecomp [51] presents a novel framework to address the challenge for CoT how to come up with the reasoning steps to predict the SQL query. The framework utilizes every slice of the SQL query to construct a logical step in CoT reasoning, then employs natural language templates to articulate each slice of the SQL query and arranges them in the logical execution order. **Least-to-Most** [104] is another prompting technique that decomposes questions into sub-questions and then sequentially solves them. As iterative prompting, pilot experiments [51] demonstrate that it may be unnecessary for text-to-SQL parsing. Using detailed reasoning steps tends to have more error propagation issues. As a variant of CoT, **Program-of-Thoughts (PoT)** prompting strategy [109] are proposed to enhance arithmetic reasoning for LLMs. Through

[3]The difference between multi-step reasoning (e.g., Chain of Thought) and the decomposition paradigm is that the former research focuses on advancing inherent reasoning within a single-turn generation, whereas the latter study involves using different components to assist the final generation through multiple calls of LLMs.

evaluation [55], PoT enhances the LLM for SQL generation, especially in complicated datasets [33]. SQL-CRAFT [55] are proposed to enhance LLM-based SQL generation, which incorporates PoT prompting for Python-enhanced reasoning. PoT strategy requires the model to simultaneously generate the Python code and the SQL queries, enforcing the model to incorporate Python code in its reasoning process. **Self-Consistency** [110] is a prompting strategy improving reasoning in LLMs, which leverages the intuition that a complex reasoning problem typically admits multiple different ways of thinking, leading to its unique correct answer. In the text-to-SQL task, self-consistency is adapted to sampling a set of different SQL and voting for consistent SQL via execution feedback [30, 53]. Similarly, the SD+SA+Voting [52] framework eliminates those with execution errors identified by the deterministic database management system (DBMS) and opts for the prediction that garners the majority vote. Furthermore, motivated by recent research on extending the capabilities of LLMs with tools, FUXI [66] are proposed to enhance LLMs SQL generation through effectively invoking crafted tools.

C_4-Execution Refinement: When designing criteria for accurate SQL generation, the priority is always whether a generated SQL can be successfully executed and retrieve content to correctly answer the user question [13]. As a complex programming task, generating the correct SQL in one go is challenging. Intuitively, considering the execution feedback/results in SQL generation assists the alignment to the corresponding database environment, which allows the LLMs to gather the potential executed errors and results to refine the generated SQL or hold a majority vote [30]. The execution-aware methods in text-to-SQL incorporate the execution feedback in two main approaches: **1) Incorporating the feedback through second round prompting for re-generation**, for every SQL query generated in the initial response, it will be executed in the corresponding database, thus obtaining the database's feedback. This feedback might be an error, or it might yield results that will be appended to the second round prompt. Through in-context learning of this feedback, LLMs are able to refine or re-generate the original SQL, thereby enhancing accuracy. **2) Utilize execution-based selection strategies for the generated SQL**, sample multiple generated SQL queries from LLM, and execute each in the database. Based on the execution results of each SQL query, use selection strategies (e.g., self-consistency, majority vote [60]) to define one SQL query from the SQL set that satisfies the criteria as the final predicted SQL.

MRC-EXEC [67] introduced a natural language to code (NL2Code) translation framework with execution, which executes each sampled SQL query and selects the example with the minimal execution result–based Bayes risk [111]. LEVER [68] proposed an approach to verify NL2Code with execution, utilizing a generation and execution module to collect sampled SQL set and their execution results, respectively, then using a learned verifier to output the probability of the correctness. Similarly, the SELF-DEBUGGING [48] framework is presented to teach LLMs to debug their predicted SQL via few-shot demonstrations. The model is able to refine its mistakes by investigating the execution results and explaining the generated SQL in natural language without human interventions

As previously introduced, to incorporate the well-designed framework with execution feedback, two-stage implications are widely-used: 1. sampling a set of SQL queries. 2. majority vote (self-consistency). Specifically, the C3 [30] framework removes the errors and identifies the most consistent SQL; The retrieval-augmented framework [64] introduced a dynamic revision chain, combining fine-grained execution messages with database content to prompt the LLMs to convert the generated SQL query into natural language explanation; the LLMs are requested to identify the semantic gaps and revise their own generated SQL. Although schema-filtering approaches enhance SQL generation, the generated SQL could be unexecutable. DESEM [62] incorporates a fallback revision to address the issue; it revises and regenerates the SQL base on different kinds of errors and sets termination criteria to avoid the loop. DIN-SQL [8] designed a generic and gentle prompt in their self-correction module; the generic prompt requests the LLMs to identify and correct the error, and the gentle prompt asks the model to check the potential issue. The multi-agent framework MAC-SQL [57] comprises a refiner agent, which is able to detect and automatically rectify SQL errors, taking SQLite error and exception class to regenerate fixed SQL. Since different questions may require different numbers of revisions, SQL-CRAFT [55] framework introduced interactive correction with an automated control determination process to avoid over-correction or insufficient correction. FUXI [66] considers the error feedback in tool-based reasoning for SQL generation. The Knowledge-to-SQL [31] introduces a preference learning framework incorporating the database execution feedback with a direct preference optimization [112] for refining the proposed DELLM. PET-SQL [60] proposed cross consistency, which comprises two variants: 1) naive voting: instruct multiple LLMs to generate the SQL query, then utilizing the majority vote for the final SQL base on different execution results; 2) fine-grained voting: refine the naive voting based on the difficulty level to mitigate the voting bias.

B. Fine-tuning

Since supervised fine-tuning (SFT) is the mainstream approach in the LLMs training [29, 91], for open-source LLMs (e.g., LLaMA-2 [94], Gemma [113]), the most straightforward method to enable the model to adapt a specific domain quickly is to use collected domain label to perform SFT on the model. The SFT phase is typically the preliminary phase of the well-designed training framework [112, 114], as well as the fine-tuning of text-to-SQL. The auto-regressive generation process of SQL query Y can be formulated as follows:

$$P_\pi(Y \mid \mathcal{P}) = \prod_{k=1}^{n} P_\pi(y_k \mid \mathcal{P}, Y_{1:k-1}), \tag{6}$$

where $Y = \{y_1, y_2, \ldots, y_n\}$ is an SQL query of length n, y_k is the corresponding k^{th} token of the SQL query, $Y_{1:k-1}$ is the prefix sequence of Y ahead the token y_k. $P_\pi(y_k \mid \cdot)$ is a conditional probability of a LLM π for generating the k^{th} token of Y base on the input prompt $\mathcal{P}$ and the prefix sequence.

TABLE IV: Well-designed methods used in fine-tuning (FT) for LLM-based text-to-SQL. The methods in each category are ordered by release time. *The methods are utilized in multiple open-source LLMs; we select a representative model to present.

Category	Adopted by	Applied LLMs	Dataset	EX	EM	VES	Release Time	Publication Venue
Enhanced Architecture	CLLMs [69]	Deepseek*	[13]	✓			Mar-2024	ICML'24
Pre-training	CodeS [10]	StarCoder	[13, 33]	✓		✓	Feb-2024	SIGMOD'24
Data Augmentation	DAIL-SQL [9]	LLaMA*	[13, 41]	✓	✓		Aug-2023	VLDB'24
	Symbol-LLM [50]	CodeLLaMA	[13]		✓		Nov-2023	ACL'24
	CodeS [10]	StarCoder	[13, 33]	✓		✓	Feb-2024	SIGMOD'24
	StructLM [70]	CodeLLaMA	[13]		✓		Feb-2024	arXiv'24
Decomposition	DTS-SQL [71]	Mistral*	[13, 40]	✓	✓		Feb-2024	arXiv'24

Given a basic open-source model π^0, the goal of SFT is obtain a model π^{SFT} through minimizing the cross-entropy loss:

$$\mathcal{L}_{SFT} = -\sum_{k=1}^{n} \log P_{\pi^0}(\hat{y}_k = y_k \mid \mathcal{P}, Y_{1:k-1}), \tag{7}$$

where $\hat{y}_k$ is the k^{th} token of the generated SQL query $\hat{Y}$, and Y is the corresponding ground-truth label.

The SFT approach, which is also a vanilla fine-tuning approach for text-to-SQL has been widely adopted in text-to-SQL research for various open-source LLMs [9, 10, 46]. Compared to in-context learning (ICL) approaches, fine-tuning paradigms are more inclined to be at a starting point in LLM-based text-to-SQL. Currently, several studies exploring a better fine-tuning method have been released. We categorize the well-designed fine-tuning methods in different groups based on their mechanisms, as shown in Tab. IV.

Enhanced Architecture: The widely-used generative pre-trained transformer (GPT) framework utilizes decoder-only transformer architecture and conventional auto-regressive decoding for text generation. Recent studies on the efficiency of LLMs have revealed a common challenge: when generating long sequences with the auto-regressive paradigm, the need to incorporate the attention mechanism results in high latency for LLMs [115, 116]. In LLM-based text-to-SQL, the speed of generating SQL queries is significantly slower compared to traditional language modeling [21, 28], which has become a challenge in constructing high-efficiency local NLIDB.

As one of the solutions, CLLMs [69] are designed to address the above challenge with an enhanced model architecture and achieve a speedup for SQL generation.

Data Augmentation: During the fine-tuning process, the most straightforward factor affecting the model's performance is the quality of the training labels [117]. The fine-tuning under the low quality or lack of the training labels is *"making bricks without straw"*, using high-quality or augmented data for fine-tuning always surpasses the meticulous design of fine-tuning methods on low-quality or raw data [29, 74]. Data-augmented fine-tuning in text-to-SQL made substantial progress, focusing on enhancing the data quality during the SFT process.

DAIL-SQL [9] are designed as an in-context learning framework, utilizing a sampling strategy for better few-shot instances. Incorporating the sampled instances in the SFT process improves the performance of open-source LLMs. Symbol-LLM [50] propose injection and infusion stage for data augmented instruction tuning. CodeS [10] augmented the training data with bi-directional generation with the help of ChatGPT. StructLM [70] are trained on multiple struct knowledge tasks for improving overall capability.

Pre-training: Pre-training is a fundamental phase of the complete fine-tuning process, aimed at acquiring text generation capabilities through auto-regressive training on extensive data [118]. Conventionally, the current powerful proprietary LLMs (e.g., ChatGPT [119], GPT-4 [86], Claude [120]) are pre-trained on hybrid corpus, which mostly benefit from the dialogue scenario that exhibits text generation capability [85]. The code-specific LLMs (e.g., CodeLLaMA [121], StarCoder [122]) are pre-trained on code data [100], and the mixture of various programming languages enables the LLMs to generate code to meet with the user's instruction [123]. As a sub-task of code generation, the main challenge of SQL-specific pre-training technique is that the SQL/Database-related content occupies only a small portion of the entire pre-training corpus. Then, as a result, the open-source LLMs with comparatively limited comprehensive capacity (compared to ChatGPT, GPT-4) do not acquire a promising understanding of how to convert NL questions to SQL during their pre-training process.

The pre-training phase of the CodeS [10] model consists of three stages of incremental pre-training. Starting from a basic code-specific LLM [122], CodeS are incre pre-trained on a hybrid training corpus, including SQL-related data, NL-to-Code data, and NL-related data. The text-to-SQL understanding and performance are significantly improved.

Decomposition: Decomposing a task into multiple steps or using multiple models to solve the task is an intuitive solution for addressing a complex scenario, as we previously introduced in Sec. IV-A, ICL paradigm. The proprietary models utilized in ICL-based methods have a massive number of parameters that are not at the same parameter level as the open-source models used in fine-tuning methods. These models inherently possess the capability to perform assigned sub-tasks well (through mechanisms such as few-shot learning) [30, 57]. Thus, to replicate the success of this paradigm in ICL methods, it is necessary to reasonably assign corresponding sub-tasks to open-source models (such as generating external knowledge, schema linking, and distilling the schema) for sub-task-specific fine-tuning and constructing the corresponding data for fine-tuning, thereby assisting in the final SQL generation.

DTS-SQL [71] proposed a two-stage decomposed text-to-SQL fine-tuning framework and designed a schema-linking

pre-generation task ahead of the final SQL generation.

V. Expectations

Despite the significant advancements made in text-to-SQL research, there are still several challenges that need to be addressed. In this section, we discuss the remaining challenges that we expect to overcome in future work.

A. Robustness in Real-world Applications

The text-to-SQL implemented by LLMs is expected to perform generalization and robustness across complex scenarios in real-world applications. Despite recent advances having made substantial progress in robustness-specific datasets [37, 41], its performance still falls short of practical application [33]. There are still challenges that are expected to be overcome in future studies. From the user aspect, there is a phenomenon that the **user is not always a clear question proposer**, which means the user questions might not have the exact database value and also can be varied from the standard datasets, the synonyms, typos, and vague expressions could be included [40]. For instance, the models are trained on clear indicative questions with concrete expressions in the fine-tuning paradigm. Since the model has not learned the mapping of realistic questions to the corresponding database, this leads to a knowledge gap when applied to real-world scenarios [33]. As reported in the corresponding evaluations of the dataset with synonym and incomplete instruction [7, 51], the SQL queries generated by ChatGPT contain around 40% incorrect execution, which is 10% lower than the original evaluation [51]. Simultaneously, the **fine-tuning with local text-to-SQL datasets may contain non-standardized samples and labels.** As an example, the name of the table or column is not always an accurate representation of its content, which yields an inconsistency within the training data construction and may lead to a semantic gap between the database schema and the user question. To address this challenge, aligning the LLMs with intention bias and designing the training strategy towards noisy scenarios will benefit the recent advances. At the same time, **the data size in real-world applications is relatively smaller than the research-oriented benchmark**. Since extending a large amount of the data by human annotation incurs high labor costs, designing data-augmentation methods to obtain more question-SQL pairs will support the LLM in data scarcity. Also, the adaptation study of fine-tuned open-source LLM to the local small-size dataset can be potentially beneficial. Furthermore, **the extensions on multi-lingual [42, 124] and multi-modal scenarios** [125] should be studied comprehensively in future research, which will benefit more language groups and help build more general database interfaces.

B. Computational Efficiency

The computational efficiency is determined by the inference speed and the cost of computational resources, which is worth considering in both application and research work [49, 69]. With the increasing complexity of databases in up-to-date text-to-SQL benchmarks [15, 33], databases will carry more information (including more tables and columns), and the token length of the database schema will correspondingly increase, raising a series of challenges. Dealing with an ultra-complex database, taking the corresponding schema as input may encounter the challenge that **the cost of calling proprietary LLMs will significantly increase, potentially exceeding the model's maximum token length**, especially with the implementation of open-source models that have shorter context lengths. Meanwhile, another obvious challenge is that most works **use the full schema as model input, which introduces significant redundancy** [57]. Providing LLMs with a precise question-related filtered schema directly from the user end to reduce cost and redundancy is a potential solution to improve computational efficiency [30]. Designing an accurate method for schema filtering remains a future direction. Although the in-context learning paradigm achieves promising accuracy, as a computational efficiency concern, the well-designed methods with the multi-stage framework or extended context **increasing the number of API calls to enhance performance has simultaneously led to a substantial rise in costs** [8]. As reported in related approaches [49], a trade-off between performance and computational efficiency should be considered carefully, and designing a comparable (even better) in-context learning method with less API cost will be a practical implementation and is still under exploration. Compared to PLM-based methods, **the inference speed of LLM-based methods is observably slower** [21, 28]. Accelerating inference by shortening the input length and reducing the number of stages in implementation would be intuitive for the in-context learning paradigm. For local LLMs, from a starting point [69], more speedup strategies can be studied in enhancing the model's architecture in future exploration.

C. Data Privacy and Interpretability

As a part of the LLMs' study, LLM-based text-to-SQL also faces some general challenges present in LLM research [4, 126, 127]. Potential improvements from the text-to-SQL perspective are also expected to be seen in these challenges, thereby extensively benefiting the study of LLMs. As previously discussed in Sec. IV-A, the in-context learning paradigm predominates the number and performance in recent studies, with the majority of work using proprietary models for implementation [8, 9]. A straightforward challenge is proposed regarding data privacy, as **calling proprietary APIs to handle local databases with confidentiality can pose a risk of data leakage**. Using a local fine-tuning paradigm can partially address this issue. Still, the current performance of vanilla fine-tuning is not ideal [9], and advanced fine-tuning framework potentially relies on proprietary LLMs for data augmentation [10]. Based on the current status, more tailored frameworks in the local fine-tuning paradigm for text-to-SQL deserve widespread attention. Overall, the development of deep learning continually faces challenges regarding interpretability [127, 128]. As a long-standing challenge, considerable work has already been studied to address this issue [129, 130]. However, in text-to-SQL research, **the interpretability of LLM-based implementation is still not being discussed**, whether in the in-context learning

or fine-tuning paradigm. The approaches with a decomposition phase explain the text-to-SQL implementation process from the perspective of step-by-step generation [8, 51]. Building on this, combining advanced study in interpretability [131, 132] to enhance text-to-SQL performance and interpreting the local model architecture from the database knowledge aspect remain future directions.

D. Extensions

As a sub-field of LLMs and natural language understanding research, many studies in these fields have been adopted for text-to-SQL tasks, advancing its development [103, 110]. However, text-to-SQL research can also be extended to the larger scope studies of these fields at meanwhile. For instance, SQL generation is a part of code generation. The well-designed approaches in code generation also obtain promising performance in text-to-SQL [48, 68], performing generalization across various programming languages. **The potential extension of some tailored text-to-SQL frameworks to NL-to-code studies can also be discussed**. For instance, frameworks integrating execution output in NL-to-code can also achieve solid performance in SQL generation [8]. An attempt to extend execution-aware approaches in text-to-SQL with other advancing modules [30, 31] for code generation is worth discussing. From another perspective, we previously discussed that text-to-SQL can enhance LLM-based question-answering (QA) by providing factual information. The database can store relational knowledge as structural information, and **the structure-based QA can potentially benefit from text-to-SQL** (e.g., knowledge-based question-answering, KBQA [133, 134]). Construct the factual knowledge with database structure, and then incorporate the text-to-SQL system to achieve information retrieval, which can potentially assist further QA with more accurate factual knowledge [135]. More extensions of text-to-SQL studies are expected in future work.

REFERENCES

[1] L. Wang, B. Qin, B. Hui, B. Li, M. Yang, B. Wang, B. Li, J. Sun, F. Huang, L. Si, and Y. Li, "Proton: Probing schema linking information from pre-trained language models for text-to-sql parsing," in *Conference on Knowledge Discovery and Data Mining (KDD)*, 2022.

[2] B. Qin, B. Hui, L. Wang, M. Yang, J. Li, B. Li, R. Geng, R. Cao, J. Sun, L. Si *et al.*, "A survey on text-to-sql parsing: Concepts, methods, and future directions," *arXiv preprint arXiv:2208.13629*, 2022.

[3] S. Xu, S. Semnani, G. Campagna, and M. Lam, "Autoqa: From databases to qa semantic parsers with only synthetic training data," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[4] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen *et al.*, "Siren's song in the ai ocean: a survey on hallucination in large language models," *arXiv preprint arXiv:2309.01219*, 2023.

[5] P. Manakul, A. Liusie, and M. J. Gales, "Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[6] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," in *Association for Computational Linguistics (ACL)*, 2021.

[7] N. Rajkumar, R. Li, and D. Bahdanau, "Evaluating the text-to-sql capabilities of large language models," *arXiv preprint arXiv:2204.00498*, 2022.

[8] M. Pourreza and D. Rafiei, "DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[9] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-sql empowered by large language models: A benchmark evaluation," in *International Conference on Very Large Data Bases (VLDB)*, 2024.

[10] H. Li, J. Zhang, H. Liu, J. Fan, X. Zhang, J. Zhu, R. Wei, H. Pan, C. Li, and H. Chen, "Codes: Towards building open-source language models for text-to-sql," in *Conference on Management of Data (SIGMOD)*, 2024.

[11] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," in *International Conference on Very Large Data Bases (VLDB)*, 2014.

[12] T. Mahmud, K. A. Hasan, M. Ahmed, and T. H. C. Chak, "A rule based approach for nlp based query processing," in *International Conference on Electrical Information and Communication Technologies (EICT)*, 2015.

[13] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[14] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[15] M. Pourreza and D. Rafiei, "Evaluating cross-domain text-to-SQL models and benchmarks," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[18] B. Hui, X. Shi, R. Geng, B. Li, Y. Li, J. Sun, and X. Zhu, "Improving text-to-sql with schema dependency learning," *arXiv preprint arXiv:2103.04399*, 2021.

[19] D. Choi, M. C. Shin, E. Kim, and D. R. Shin, "Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases," *Computational Linguistics*, 2021.

[20] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular

data," *arXiv preprint arXiv:2005.08314*, 2020.

[21] H. Li, J. Zhang, C. Li, and H. Chen, "Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql," in *Conference on Artificial Intelligence (AAAI)*, 2023.

[22] J. Li, B. Hui, R. Cheng, B. Qin, C. Ma, N. Huo, F. Huang, W. Du, L. Si, and Y. Li, "Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing," in *Conference on Artificial Intelligence (AAAI)*, 2023.

[23] D. Rai, B. Wang, Y. Zhou, and Z. Yao, "Improving generalization in language model-based text-to-sql semantic parsing: Two simple semantic boundary-based techniques," in *Association for Computational Linguistics (ACL)*, 2023.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

[25] Q. Lyu, K. Chakrabarti, S. Hathi, S. Kundu, J. Zhang, and Z. Chen, "Hybrid ranking network for text-to-sql," *arXiv preprint arXiv:2008.04759*, 2020.

[26] T. Yu, C.-S. Wu, X. V. Lin, bailin wang, Y. C. Tan, X. Yang, D. Radev, richard socher, and C. Xiong, "Grappa: Grammar-augmented pre-training for table semantic parsing," in *International Conference on Learning Representations (ICLR)*, 2021.

[27] A. Liu, X. Hu, L. Wen, and P. S. Yu, "A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability," *arXiv preprint arXiv:2303.13547*, 2023.

[28] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin, and X. Hu, "Harnessing the power of llms in practice: A survey on chatgpt and beyond," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2024.

[29] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[30] X. Dong, C. Zhang, Y. Ge, Y. Mao, Y. Gao, J. Lin, D. Lou *et al.*, "C3: Zero-shot text-to-sql with chatgpt," *arXiv preprint arXiv:2307.07306*, 2023.

[31] Z. Hong, Z. Yuan, H. Chen, Q. Zhang, F. Huang, and X. Huang, "Knowledge-to-sql: Enhancing sql generation with data expert llm," *arXiv preprint arXiv:2402.11517*, 2024.

[32] Q. Zhang, J. Dong, H. Chen, W. Li, F. Huang, and X. Huang, "Structure guided large language model for sql generation," *arXiv preprint arXiv:2402.13284*, 2024.

[33] J. Li, B. Hui, G. QU, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. Chang, F. Huang, R. Cheng, and Y. 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 (NeurIPS)*, 2023.

[34] L. Wang, A. Zhang, K. Wu, K. Sun, Z. Li, H. Wu, M. Zhang, and H. Wang, "DuSQL: A large-scale and pragmatic Chinese text-to-SQL dataset," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[35] T. Yu, R. Zhang, H. Er, S. Li, E. Xue, B. Pang, X. V. Lin, Y. C. Tan, T. Shi, Z. Li, Y. Jiang, M. Yasunaga, S. Shim, T. Chen, A. Fabbri, Z. Li, L. Chen, Y. Zhang, S. Dixit, V. Zhang, C. Xiong, R. Socher, W. Lasecki, and D. Radev, "CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases," in *Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[36] C.-H. Lee, O. Polozov, and M. Richardson, "KaggleDBQA: Realistic evaluation of text-to-SQL parsers," in *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 2021.

[37] X. Pi, B. Wang, Y. Gao, J. Guo, Z. Li, and J.-G. Lou, "Towards robustness of text-to-SQL models against natural and realistic adversarial table perturbation," in *Association for Computational Linguistics (ACL)*, 2022.

[38] Y. Gan, X. Chen, Q. Huang, and M. Purver, "Measuring and improving compositional generalization in text-to-SQL via component alignment," in *Findings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

[39] Y. Gan, X. Chen, and M. Purver, "Exploring underexplored limitations of cross-domain text-to-SQL generalization," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[40] Y. Gan, X. Chen, Q. Huang, M. Purver, J. R. Woodward, J. Xie, and P. Huang, "Towards robustness of text-to-SQL models against synonym substitution," in *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 2021.

[41] X. Deng, A. H. Awadallah, C. Meek, O. Polozov, H. Sun, and M. Richardson, "Structure-grounded pretraining for text-to-SQL," in *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2021.

[42] Q. Min, Y. Shi, and Y. Zhang, "A pilot study for Chinese SQL semantic parsing," in *Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[43] T. Yu, R. Zhang, M. Yasunaga, Y. C. Tan, X. V. Lin, S. Li, H. Er, I. Li, B. Pang, T. Chen, E. Ji, S. Dixit, D. Proctor, S. Shim, J. Kraft, V. Zhang, C. Xiong, R. Socher, and D. Radev, "SParC: Cross-domain semantic parsing in context," in *Association for Computational Linguistics (ACL)*, 2019.

[44] T. Shi, C. Zhao, J. Boyd-Graber, H. Daumé III, and L. Lee, "On the potential of lexico-logical alignments for semantic parsing to SQL queries," in *Findings of Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[45] S. Xue, C. Jiang, W. Shi, F. Cheng, K. Chen, H. Yang, Z. Zhang, J. He, H. Zhang, G. Wei, W. Zhao, F. Zhou, D. Qi, H. Yi, S. Liu, and F. Chen, "Db-gpt: Empowering database interactions with private large language models," *arXiv preprint arXiv:2312.17449*, 2024.

[46] B. Zhang, Y. Ye, G. Du, X. Hu, Z. Li, S. Yang, C. H. Liu, R. Zhao, Z. Li, and H. Mao, "Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation," *arXiv preprint arXiv:2403.02951*, 2024.

[47] S. Chang and E. Fosler-Lussier, "How to prompt LLMs for text-to-SQL: A study in zero-shot, single-domain, and cross-domain settings," in *NeurIPS 2023 Second Table Representation Learning Workshop (NeurIPS)*, 2023.

[48] X. Chen, M. Lin, N. Schärli, and D. Zhou, "Teaching large language models to self-debug," in *International Conference on Learning Representations (ICLR)*, 2024.

[49] H. Zhang, R. Cao, L. Chen, H. Xu, and K. Yu, "ACT-SQL: In-context learning for text-to-SQL with automatically-generated chain-of-thought," in *Findings of Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[50] F. Xu, Z. Wu, Q. Sun, S. Ren, F. Yuan, S. Yuan, Q. Lin, Y. Qiao, and J. Liu, "Symbol-llm: Towards foundational symbol-centric interface for large language models," *arXiv preprint arXiv:2311.09278*, 2024.

[51] C.-Y. Tai, Z. Chen, T. Zhang, X. Deng, and H. Sun, "Exploring chain of thought style prompting for text-to-SQL," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[52] L. Nan, Y. Zhao, W. Zou, N. Ri, J. Tae, E. Zhang, A. Cohan, and D. Radev, "Enhancing text-to-SQL capabilities of large language models: A study on prompt design strategies," in *Findings of Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[53] R. Sun, S. O. Arik, H. Nakhost, H. Dai, R. Sinha, P. Yin, and T. Pfister, "Sql-palm: Improved large language model adaptation for text-to-sql," *arXiv preprint arXiv:2306.00739*, 2023.

[54] S. Chang and E. Fosler-Lussier, "Selective demonstrations for cross-domain text-to-SQL," in *Findings of Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[55] H. Xia, F. Jiang, N. Deng, C. Wang, G. Zhao, R. Mihalcea, and Y. Zhang, "Sql-craft: Text-to-sql through interactive refinement and enhanced reasoning," *arXiv preprint arXiv:2402.14851*, 2024.

[56] T. Zhang, T. Yu, T. B. Hashimoto, M. Lewis, W. tau Yih, D. Fried, and S. I. Wang, "Coder reviewer reranking for code generation," in *International Conference on Machine Learning (ICML)*, 2023.

[57] B. Wang, C. Ren, J. Yang, X. Liang, J. Bai, L. Chai, Z. Yan, Q.-W. Zhang, D. Yin, X. Sun, and Z. Li, "Mac-sql: A multi-agent collaborative framework for text-to-sql," *arXiv preprint arXiv:2312.11242*, 2024.

[58] Y. Xie, X. Jin, T. Xie, M. Lin, L. Chen, C. Yu, L. Cheng, C. Zhuo, B. Hu, and Z. Li, "Decomposition for enhancing attention: Improving llm-based text-to-sql through workflow paradigm," *arXiv preprint arXiv:2402.10671*, 2024.

[59] Y. Fan, Z. He, T. Ren, C. Huang, Y. Jing, K. Zhang, and X. S. Wang, "Metasql: A generate-then-rank framework for natural language to sql translation," 2024.

[60] Z. Li, X. Wang, J. Zhao, S. Yang, G. Du, X. Hu, B. Zhang, Y. Ye, Z. Li, R. Zhao, and H. Mao, "Pet-sql: A prompt-enhanced two-stage text-to-sql framework with cross-consistency," *arXiv preprint arXiv:2403.09732*, 2024.

[61] T. Ren, Y. Fan, Z. He, R. Huang, J. Dai, C. Huang, Y. Jing, K. Zhang, Y. Yang, and X. S. Wang, "Purple: Making a large language model a better sql writer," in *International Conference on Data Engineering (ICDE)*, 2024.

[62] C. Guo, Z. Tian, J. Tang, P. Wang, Z. Wen, K. Yang, and T. Wang, "Prompting gpt-3.5 for text-to-sql with de-semanticization and skeleton retrieval," in *Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, 2024.

[63] J. Jiang, K. Zhou, Z. Dong, K. Ye, X. Zhao, and J.-R. Wen, "StructGPT: A general framework for large language model to reason over structured data," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[64] C. Guo, Z. Tian, J. Tang, S. Li, Z. Wen, K. Wang, and T. Wang, "Retrieval-augmented gpt-3.5-based text-to-sql framework with sample-aware prompting and dynamic revision chain," in *International Conference on Neural Information Processing (ICONIP)*, 2024.

[65] D. Wang, L. Dou, X. Zhang, Q. Zhu, and W. Che, "Improving demonstration diversity by human-free fusing for text-to-sql," *arXiv preprint arXiv:2402.10663*, 2024.

[66] Y. Gu, Y. Shu, H. Yu, X. Liu, Y. Dong, J. Tang, J. Srinivasa, H. Latapie, and Y. Su, "Middleware for llms: Tools are instrumental for language agents in complex environments," *arXiv preprint arXiv:2402.14672*, 2024.

[67] F. Shi, D. Fried, M. Ghazvininejad, L. Zettlemoyer, and S. I. Wang, "Natural language to code translation with execution," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[68] A. Ni, S. Iyer, D. Radev, V. Stoyanov, W.-t. Yih, S. I. Wang, and X. V. Lin, "Lever: Learning to verify language-to-code generation with execution," in *International Conference on Machine Learning (ICML)*, 2023.

[69] S. Kou, L. Hu, Z. He, Z. Deng, and H. Zhang, "Cllms: Consistency large language models," *arXiv preprint arXiv:2403.00835*, 2024.

[70] A. Zhuang, G. Zhang, T. Zheng, X. Du, J. Wang, W. Ren, S. W. Huang, J. Fu, X. Yue, and W. Chen, "Structlm: Towards building generalist models for structured knowledge grounding," *arXiv preprint arXiv:2402.16671*, 2024.

[71] M. Pourreza and D. Rafiei, "Dts-sql: Decomposed text-to-sql with small large language models," *arXiv preprint arXiv:2402.01117*, 2024.

[72] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, and E. Chen, "Large language models

for generative information extraction: A survey," *arXiv preprint arXiv:2312.17617*, 2023.

[73] G. Katsogiannis-Meimarakis and G. Koutrika, "A survey on deep learning approaches for text-to-sql," *The VLDB Journal*, 2023.

[74] N. Deng, Y. Chen, and Y. Zhang, "Recent advances in text-to-SQL: A survey of what we have and what we expect," in *International Conference on Computational Linguistics (COLING)*, 2022.

[75] P. Ma and S. Wang, "Mt-teql: evaluating and augmenting neural nlidb on real-world linguistic and schema variations," in *International Conference on Very Large Data Bases (VLDB)*, 2021.

[76] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[77] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for SQuAD," in *Association for Computational Linguistics (ACL)*, 2018.

[78] H. Yang, Y. Zhang, J. Xu, H. Lu, P.-A. Heng, and W. Lam, "Unveiling the generalization power of fine-tuned large language models," in *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2024.

[79] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[80] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J.-G. Lou, T. Liu, and D. Zhang, "Towards complex text-to-sql in cross-domain database with intermediate representation," *arXiv preprint arXiv:1905.08205*, 2019.

[81] X. Xu, C. Liu, and D. Song, "Sqlnet: Generating structured queries from natural language without reinforcement learning," *arXiv preprint arXiv:1711.04436*, 2017.

[82] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[83] L. Dou, Y. Gao, X. Liu, M. Pan, D. Wang, W. Che, D. Zhan, M.-Y. Kan, and J.-G. Lou, "Towards knowledge-intensive text-to-SQL semantic parsing with formulaic knowledge," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[84] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," *OpenAI blog*, 2018.

[85] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[86] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[87] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," *arXiv preprint arXiv:2402.07927*, 2024.

[88] J. Wang, E. Shi, S. Yu, Z. Wu, C. Ma, H. Dai, Q. Yang, Y. Kang, J. Wu, H. Hu *et al.*, "Prompt engineering for healthcare: Methodologies and applications," *arXiv preprint arXiv:2304.14670*, 2023.

[89] B. Chen, Z. Zhang, N. Langrené, and S. Zhu, "Unleashing the potential of prompt engineering in large language models: a comprehensive review," *arXiv preprint arXiv:2310.14735*, 2023.

[90] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2021.

[91] Y. Zheng, R. Zhang, J. Zhang, Y. Ye, and Z. Luo, "Llamafactory: Unified efficient fine-tuning of 100+ language models," *arXiv preprint arXiv:2403.13372*, 2024.

[92] T. Wang, H. Lin, X. Han, L. Sun, X. Chen, H. Wang, and Z. Zeng, "Dbcopilot: Scaling natural language querying to massive databases," *arXiv preprint arXiv:2312.03463*, 2023.

[93] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[94] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[95] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang *et al.*, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.

[96] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, 2019.

[97] L. Reynolds and K. McDonell, "Prompt programming for large language models: Beyond the few-shot paradigm," in *Conference on Human Factors in Computing Systems (CHI)*, 2021.

[98] X. Ye and G. Durrett, "The unreliability of explanations in few-shot prompting for textual reasoning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[99] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research (JMLR)*, 2020.

[100] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[101] P. P. Ray, "Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope," *Internet of Things and Cyber-Physical Systems*, 2023.

[102] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts," in *Conference on Human Factors in Computing Systems (CHI)*, 2023.

[103] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[104] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le *et al.*, "Least-to-most prompting enables complex reasoning in large language models," *arXiv preprint arXiv:2205.10625*, 2022.

[105] W. Lei, W. Wang, Z. Ma, T. Gan, W. Lu, M.-Y. Kan, and T.-S. Chua, "Re-examining the role of schema linking in text-to-SQL," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[106] Q. Liu, D. Yang, J. Zhang, J. Guo, B. Zhou, and J.-G. Lou, "Awakening latent grounding from pretrained language models for semantic parsing," in *Findings of Association for Computational Linguistics (ACL)*, 2021.

[107] Z. Tan, X. Liu, Q. Shu, X. Li, C. Wan, D. Liu, Q. Wan, and G. Liao, "Enhancing text-to-SQL capabilities of large language models through tailored promptings," in *International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING)*, 2024.

[108] J. Huang and K. C.-C. Chang, "Towards reasoning in large language models: A survey," in *Findings of Association for Computational Linguistics (ACL)*, 2023.

[109] W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *Transactions on Machine Learning Research (TMLR)*, 2023.

[110] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[111] M. Müller and R. Sennrich, "Understanding the properties of minimum Bayes risk decoding in neural machine translation," in *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 2021.

[112] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[113] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love *et al.*, "Gemma: Open models based on gemini research and technology," *arXiv preprint arXiv:2403.08295*, 2024.

[114] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[115] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *International Conference on Machine Learning (ICML)*, 2023.

[116] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating large language model decoding with speculative sampling," *arXiv preprint arXiv:2302.01318*, 2023.

[117] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2023.

[118] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pre-training help deep learning?" in *Artificial Intelligence and Statistics (AISTATS)*, 2010.

[119] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu *et al.*, "Summary of chatgpt-related research and perspective towards the future of large language models," *Meta-Radiology*, 2023.

[120] Anthropic, "Introducing Claude," 2023.

[121] B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.

[122] R. Li, L. B. allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. LI, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, J. Lamy-Poirier, J. Monteiro, N. Gontier, M.-H. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. Murthy, J. T. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, U. Bhattacharyya, W. Yu, S. Luccioni, P. Villegas, F. Zhdanov, T. Lee, N. Timor, J. Ding, C. S. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. Hughes, T. Wolf, A. Guha, L. V. Werra, and H. de Vries, "Starcoder: may the source be with you!" *Transactions on Machine Learning Research (TMLR)*, 2023.

[123] Y. Wang, W. Zhong, L. Li, F. Mi, X. Zeng, W. Huang, L. Shang, X. Jiang, and Q. Liu, "Aligning large language models with human: A survey," *arXiv preprint arXiv:2307.12966*, 2023.

[124] A. Tuan Nguyen, M. H. Dao, and D. Q. Nguyen, "A pilot study of text-to-SQL semantic parsing for Vietnamese," in *Findings of Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[125] Y. Song, R. C.-W. Wong, and X. Zhao, "Speech-to-sql: toward speech-driven sql query generation from natural language question," *The VLDB Journal*, 2024.

[126] B. Yan, K. Li, M. Xu, Y. Dong, Y. Zhang, Z. Ren, and X. Cheng, "On protecting the data privacy of

large language models (llms): A survey," *arXiv preprint arXiv:2403.05156*, 2024.

[127] C. Singh, J. P. Inala, M. Galley, R. Caruana, and J. Gao, "Rethinking interpretability in the era of large language models," *arXiv preprint arXiv:2402.01761*, 2024.

[128] D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei, "Knowledge neurons in pretrained transformers," in *Association for Computational Linguistics (ACL)*, 2022.

[129] N. Zhang, Y. Yao, B. Tian, P. Wang, S. Deng, M. Wang, Z. Xi, S. Mao, J. Zhang, Y. Ni *et al.*, "A comprehensive study of knowledge editing for large language models," *arXiv preprint arXiv:2401.01286*, 2024.

[130] K. Meng, A. S. Sharma, A. J. Andonian, Y. Belinkov, and D. Bau, "Mass-editing memory in a transformer," in *International Conference on Learning Representations (ICLR)*, 2023.

[131] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "Locating and editing factual associations in gpt," *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[132] C. Zheng, L. Li, Q. Dong, Y. Fan, Z. Wu, J. Xu, and B. Chang, "Can we edit factual knowledge by in-context learning?" in *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[133] H. Luo, Z. Tang, S. Peng, Y. Guo, W. Zhang, C. Ma, G. Dong, M. Song, W. Lin *et al.*, "Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models," *arXiv preprint arXiv:2310.08975*, 2023.

[134] Z. Li, S. Fan, Y. Gu, X. Li, Z. Duan, B. Dong, N. Liu, and J. Wang, "Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering," in *Conference on Artificial Intelligence (AAAI)*, 2024.

[135] G. Xiong, J. Bao, and W. Zhao, "Interactive-kbqa: Multi-turn interactions for knowledge base question answering with large language models," *arXiv preprint arXiv:2402.15131*, 2024.

[136] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen *et al.*, "Palm 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[137] Z. Hong and J. Liu, "Towards better question generation in qa-based event extraction," *arXiv preprint arXiv:2405.10517*, 2024.

[138] Y. Liu, H. He, T. Han, X. Zhang, M. Liu, J. Tian, Y. Zhang, J. Wang, X. Gao, T. Zhong *et al.*, "Understanding llms: A comprehensive overview from training to inference," *arXiv preprint arXiv:2401.02038*, 2024.

[139] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia, W. L. Tam, Z. Ma, Y. Xue, J. Zhai, W. Chen, Z. Liu, P. Zhang, Y. Dong, and J. Tang, "GLM-130b: An open bilingual pre-trained model," in *International Conference on Learning Representations (ICLR)*, 2023.

[140] Q. Zhang, J. Dong, Q. Tan, and X. Huang, "Integrating entity attributes for error-aware knowledge graph embedding," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2024.

[141] Q. Zhang, J. Dong, H. Chen, X. Huang, D. Zha, and Z. Yu, "Knowgpt: Black-box knowledge injection for large language models," *arXiv preprint arXiv:2312.06185*, 2023.

[142] F. Huang, Z. Yang, J. Jiang, Y. Bei, Y. Zhang, and H. Chen, "Large language model interaction simulator for cold-start item recommendation," *arXiv preprint arXiv:2402.09176*, 2024.

[143] Y. Bei, H. Xu, S. Zhou, H. Chi, M. Zhang, Z. Li, and J. Bu, "Cpdg: A contrastive pre-training method for dynamic graph neural networks," *arXiv preprint arXiv:2307.02813*, 2023.

[144] Y. Bei, H. Chen, S. Chen, X. Huang, S. Zhou, and F. Huang, "Non-recursive cluster-scale graph interacted model for click-through rate prediction," in *International Conference on Information and Knowledge Management (CIKM)*, 2023.

[145] H. Chen, Y. Bei, Q. Shen, Y. Xu, S. Zhou, W. Huang, F. Huang, S. Wang, and X. Huang, "Macro graph neural networks for online billion-scale recommender systems," in *International World Wide Web Conference (WWW)*, 2024.

[146] X. Chen, T. Wang, T. Qiu, J. Qin, and M. Yang, "Open-sql framework: Enhancing text-to-sql on open-source large language models," *arXiv preprint arXiv:2405.06674*, 2024.

[147] S. Xue, D. Qi, C. Jiang, W. Shi, F. Cheng, K. Chen, H. Yang, Z. Zhang, J. He, H. Zhang *et al.*, "Demonstration of db-gpt: Next generation data interaction system empowered by large language models," *arXiv preprint arXiv:2404.10209*, 2024.

[148] D. G. Thorpe, A. J. Duberstein, and I. A. Kinsey, "Dubo-sql: Diverse retrieval-augmented generation and fine tuning for text-to-sql," *arXiv preprint arXiv:2404.12560*, 2024.

[149] A. Lozhkov, R. Li, L. B. Allal, F. Cassano, J. Lamy-Poirier, N. Tazi, A. Tang, D. Pykhtar, J. Liu, Y. Wei *et al.*, "Starcoder 2 and the stack v2: The next generation," *arXiv preprint arXiv:2402.19173*, 2024.

[150] W. Huang, X. Ma, H. Qin, X. Zheng, C. Lv, H. Chen, J. Luo, X. Qi, X. Liu, and M. Magno, "How good are low-bit quantized llama3 models? an empirical study," *arXiv preprint arXiv:2404.14047*, 2024.

[151] G. Katsogiannis-Meimarakis and G. Koutrika, "A deep dive into deep learning approaches for text-to-sql systems," in *Conference on Management of Data (SIGMOD)*, 2021.

[152] A. Kumar, P. Nagarkar, P. Nalhe, and S. Vijayakumar, "Deep learning driven natural languages text to sql query conversion: a survey," *arXiv preprint arXiv:2208.04415*, 2022.