Large Language Model Enhanced Text-to-SQL Generation: A Survey

Xiaohu Zhu, Qian Li, Lizhen Cui, Yongkang Liu

***Abstract*—Text-to-SQL translates natural language queries into Structured Query Language (SQL) commands, enabling users to interact with databases using natural language. Essentially, the text-to-SQL task is a text generation task and its development primarily dependent on changes in language models. Especially with the rapid development of Large Language Models (LLMs), the pattern of text-to-SQL has undergone significant changes. Existing survey work mainly focuses on rule-based and neural-based approaches, still lacking a survey of Text-to-SQL with LLMs. In this paper, we survey the large language model enhanced text-to-SQL generations, classifying them into prompt engineering, fine-tuning, pre-trained and Agent groups according to training strategies. And we also summarize datasets and evaluation metrics comprehensively. This survey could help people better understand the pattern, research status, and challenges of LLM-based text-to-SQL generations.**

***Index Terms*—Text-to-SQL, Large Language Models, Prompt Engineering, Fine-Tuning, Database Querying**

I. Introduction

DATA has become a crucial production factor [1], [2] in the productive life of human activities. With the proliferation of electronic devices, there have been more and more databases appearing, storing massive information from all sorts of areas [3], [4]. However, the threshold for learning database query language, such as SQL, is relatively high for ordinary people. Even for practitioners, it is more troublesome to write a large number of query statements with guaranteed correctness for different domain databases and application scenarios. To lower the barriers of using database queries, text-to-SQL task translates natural language queries into Structured Query Language (SQL) commands, enabling users to interact with databases using natural language.

Fig.1 gives an example of a text-to-SQL task. Given a natural language question **Q** and a database schema **S**:

Q: *What is the name of the employee with the highest salary?*

S: *Table: Employees (ID, Name, Salary)*

The goal of text-to-SQL is to generate a SQL query $\hat{Y}$,

SELECT Name FROM Employees
ORDER BY Salary DESC LIMIT 1;

After converting text into SQL language, we can search for relevant knowledge from databases, thus breaking down the barriers between natural language and structured data [5].

The history of Text-to-SQL goes back to 1973 when [6] developed a system called the LUNAR system, which was primarily used to answer questions related to rocks brought back from the Moon. The earliest researches are mostly based on fine-designed rules [7], which are suitable for uncomplicated or specific scenarios. As the amount and domains of data grow exponentially, it becomes expensive for these rule-based methods. Deep neural networks began to take center stage as foundational approaches, such as LSTM-based [8] and Transformer-based [9] methods. However, they are confronted with problems such as data sparsity and generalization issues.

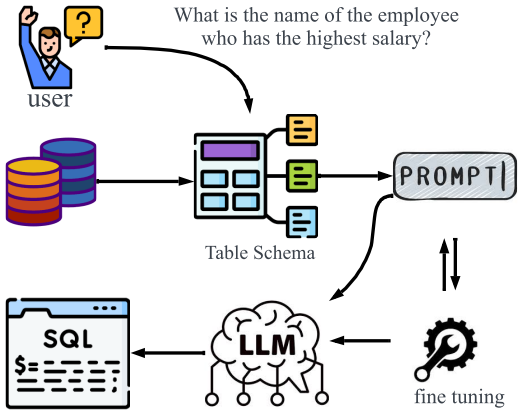

Fig. 1. Flowchart of the Text-to-SQL. The flowchart illustrates the process where user questions and the database schema are first collected. These inputs are then processed through prompt engineering and fine-tuning techniques before being passed to a large language model (LLM). The LLM generates the corresponding SQL query based on the refined inputs, allowing for accurate query formulation based on natural language input.

Recently, with the significant improvement of inference and generalization abilities in Large Language Models (LLMs), many works use LLMs to generate SQL queries correctly, and have achieved greater abilities to understand natural language than previous approaches [10]. For instance, ChatGPT-4 [11] has achieved the top performance on the Spider [12] dataset, setting the new standard for execution accuracy. Existing survey work mainly focuses on rule-based and neural-based approaches, still lacking a survey of Text-to-SQL with LLMs.

In this paper, we survey the large language model enhanced text-to-SQL generation methods, classifying them into the prompt, fine-tuning, task-training, and agent according to training strategies, as shown in Fig.3,

- **Prompt:** (No training) They use well-designed prompts to guide LLMs to generate more accurate SQL queries, enabling powerful LLMs to generate SQL queries from zero-shot [13], [14] or few-shot [15], [16] examples.
- **Fine-Tuning:** (Training from pretrained LLMs) They finetune the LLM models to adapt to the text-to-SQL task, including full-parameters fine-tuning [17], [18] and parameter-efficient fine-tuning [19], [20].
- **Task-Training:** (Training from scratch) They train a task-

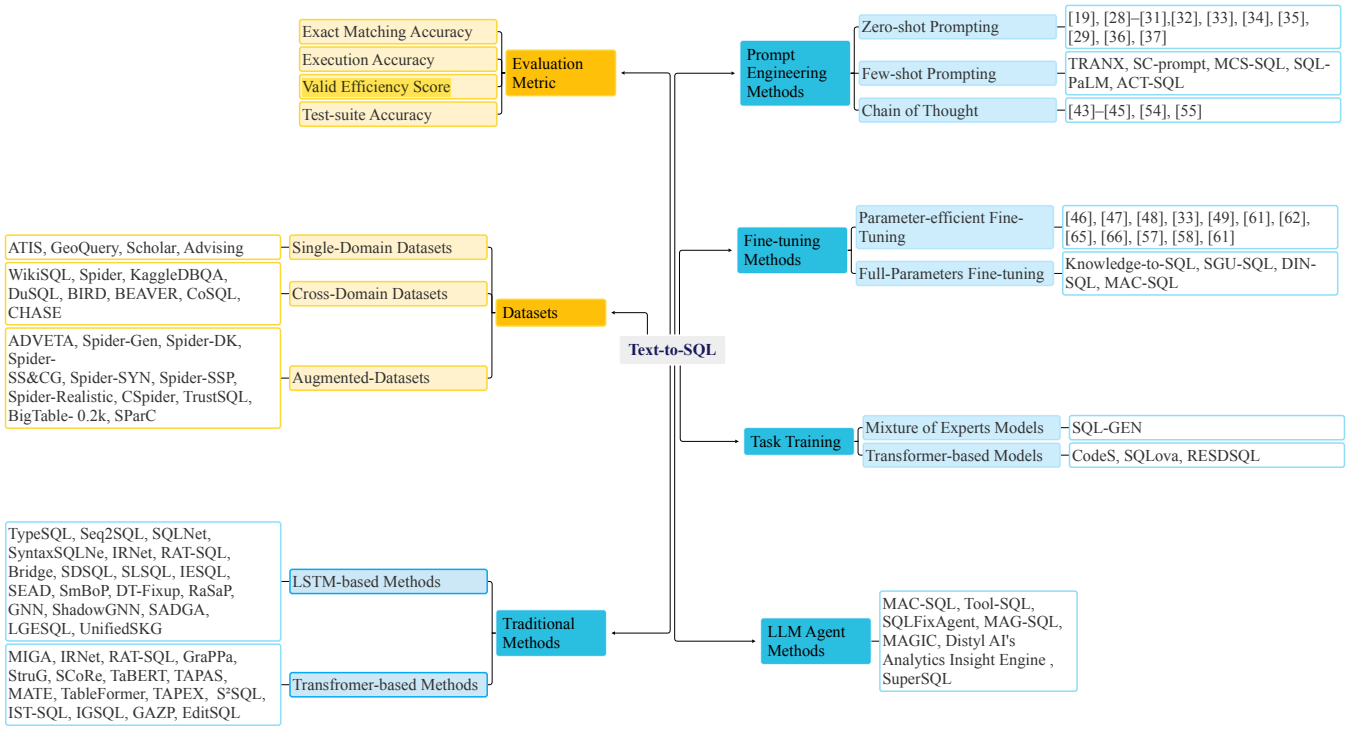

Fig. 2. The overview of the text-to-SQL metrics, datasets, and methods.

specific text-to-SQL model with training strategies similar to LLMs, such as Transformers [21], [22] and mixture of experts [23].

- **LLM Agent:** (Training with multiple agents and external tools) They collaborate with multiple intelligences, dynamically generate and correct SQL queries, handle database matching issues, and improve query accuracy and execution through external tools [24], [25].

Meanwhile, we summarize the datasets and metrics for the large language model enhanced text-to-SQL generations. For datasets, we explore the characteristics of datasets in terms of data source scenarios, number of tables, SQL complexity, and number of conversation rounds by systematically combing single-domain, cross-domain, and augmented datasets, and analyze the challenges and limitations of these datasets in real-world applications. For metrics, we consider exact matching accuracy, execution accuracy, valid efficiency and test-suite accuracy in handling single-turn, multi-turn interactions.

In the following, we first give the preliminaries in Section II, then introduce the metrics and datasets in Section III, third propose detailed descriptions of the method in Section IV, and give a conclusion and future work at the final.

II. Preliminaries

The Text-to-SQL systems enable users to input questions directly in natural language, and the system will automatically generate corresponding SQL query statements.

A. The Text-to-SQL problem

Given a natural language question Q and a database S, The objective of Text-to-SQL tasks is to generate an accurate SQL query $\hat{Y}$ that retrieves the desired output from databases D. This can be conceptualized as a sequence-to-sequence [26] problem:

- **Input:**
  - a Natural Language problem:$Q = (q_1, q_2, \ldots, q_n)$, where q_i represents the i-th token in the question.
  - A database schema:$S = \{T_1, T_2, \ldots, T_m\}$, where T_i represents the i-th table in the database, and each table T_i includes columns $C_{i1}, C_{i2}, \ldots, C_{ip}$.
- **Output:**
  - A SQL query:$\hat{Y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k)$, where $\hat{y}_i$ represents the i-th token in the generated query.

The task can be expressed as finding the most probable SQL query $\hat{Y}$ given a natural language question Q and a database schema S:

$$\hat{Y} = \arg\max_{Y} P(Y \mid Q, S)$$

B. Mythology

To solve this problem, modern methodologies typically employ deep learning models [27], particularly Encoder-Decoder (ED) architectures. Presented below is a high-level overview of the process:

- **Encoding:** The encoder processes the input question Q and the schema S to create a contextual representation. This can be represented as:

$$h = \text{Encoder}(Q, S)$$

where h is the hidden state or contextual representation generated by the encoder.

- **Decoding:** The decoder generates the SQL query $\hat{Y}$ token by token based on the encoded representation h. The probability of each token in the SQL query can be calculated as:

$$P(Y \mid Q, S) = \prod_{i=1}^{k} P(\hat{y}_i \mid h, \hat{y}_{1:i-1})$$

where $\hat{y}_{1:i-1}$ are the previously generated tokens, $P(\hat{y}_i \mid h, \hat{y}_{1:i-1})$ represents the probability of the$i$th token given the context and previous tokens.

- **Optimization:** The model is trained to maximize the likelihood of the correct SQL query Y given the training data. The loss function typically used is the negative log-likelihood of the correct query:

$$\mathcal{L} = -\sum_{i=1}^{k} \log P(y_i \mid h, y_{1:i-1})$$

where y_i are the tokens of the ground truth SQL query.

C. Challenges

Ambiguity presents one of the most prevalent and intractable problems in Natural Language Processing, it denotes the phenomenon wherein a single and the same linguistic form may be interpreted in more than one way [28], [29].

1) ***Word Segmentation and Sense Ambiguity****:* Word Segmentation Ambiguity refers to the phenomenon of different meanings when words are combined from characters. For Indo-European languages, most words are separated by spaces or punctuation. However, in languages such as Chinese and Japanese, words are usually not separated by a space or a punctuation mark. Consequently, ambiguity arises when these consecutive characters are attempting to segment into words.

Word sense ambiguity denotes the phenomenon where a word shares identical orthography while possessing distinct semantic interpretations in linguistic terms. For example, Bank noun: *(1) sloping land (especially the slope beside a body of water "they pulled the canoe up on the bank" (2) a financial institution that accepts deposits and channels the money into lending activities "he cashed a check at the bank"*

2) ***Database size and diversity****:* Realistic databases often contain hundreds of tables and columns, and the relationships between different tables can be very complex. Due to the sheer size of the database schema, Text-to-SQL systems are often unable to incorporate all relevant table structure information in a single prompt. This poses a challenge to the model because without the complete schema context, it is difficult for the model to generate correct SQL queries. To further complicate matters, databases in different domains may have completely different naming conventions, formats, and table structures. For example, column names in some databases may not have intuitive meanings (e.g., "col1", "data1"), or even have a large number of abbreviations or naming ambiguities, which requires the model to have good reasoning capabilities to correctly understand the relationships between tables and columns. In addition, the data types and formats in the database are diversified, for example, date data may have different representations (e.g., "2024-01-01" or "year2024"), which further increases the difficulty of data parsing.

3) ***Complexity of SQL queries*** *:* The complexity of SQL queries is usually related to the query structure, involving operations such as joins of multiple tables, nested subqueries, and complex conditional filtering. For example, a query may contain both SELECT, JOIN, GROUP BY, HAVING, and nested WHERE conditions, which requires the model to be able to understand and generate complex SQL statements efficiently. Particularly in multi-table queries, the model must be able to infer table-to-table relationships (e.g., foreign keys) and ensure that the generated queries are logically correct. For example, SQL with nested subqueries requires the model to have the ability to handle hierarchical relationships, while conditional filtering and aggregation operations require the model to be able to generate precise expressions based on different column types and values. In addition, certain queries in a given domain may require the use of specific SQL functions or operations (e.g., regular expression matching), which further complicates the SQL generation process.

In a paragraph or chapter consisting of multiple sentences, various kinds of ambiguities remain, such as *referential ambiguities* and *ellipsis ambiguities*. Referential ambiguity is the possibility of ambiguity in the events referred to by pronouns (e.g., I, you, he) and pronoun phrases.

4) ***Pragmatic Ambiguity****:* Pragmatic Ambiguity pertains to ambiguity due to context, speaker attributes, scene, and other situational pragmatic aspects. A sentence may elicit varying interpretations across different contexts. For example, the following example shows that the same sentence can yield different meanings based on different scenes.

Sentence: *Do you know how to get to Fifth Avenue?*

If the speaker is a tourist and the person speaking is a policeman, the meaning of the sentence is to ask for directions. If the speaker is also a tourist, but the person speaking is a taxi driver instead, then the meaning of the sentence is to ask for a ride to Fifth Avenue.

5) ***Robustness and Efficiency****:* Users may input queries with spelling mistakes, grammatical errors, or incomplete sentences. In real-world applications, users often provide imperfect or ambiguous queries due to human error or lack of domain knowledge. The model should be robust enough to deal with such inputs.

Correctly mapping valid information in natural language to corresponding parts of the database is one of the most critical steps. The model needs to understand information such as the structure and constraints of the schema. In addition to this, the structure of the schema format will change in different databases, and the model must have good generality.

One of the basic requirements of the model is the correctness of the query generated by SQL. Due to the special characteristics of SQL, even a small error can cause the entire query to fail to produce the correct answer.

Also, the execution efficiency of SQL queries is a key consideration in practical applications. Even if the model is able to generate accurate SQL queries, if the execution

efficiency of these queries is too low, especially on large-scale databases, this will affect the practicality of the system. A good Text-to-SQL system should not only generate correct queries, but also ensure that these queries can be executed quickly and efficiently in the actual database to reduce system response time and improve user experience.

III. Metrics and Datasets

A. Evaluation Metric

To evaluate the performance of Text-to-SQL models, several key metrics are employed to assess the accuracy, execution effectiveness, and overall efficiency of the generated SQL queries. These metrics provide a comprehensive understanding of how well a model translates natural language questions into valid SQL statements, ensuring both syntactic correctness and proper execution on the given database schema. In this section, we introduce four commonly used evaluation metrics: Exact Matching Accuracy (EM), Execution Accuracy (EX), Valid Efficiency Score (VES), and Test-suite Accuracy (TS). Each of these metrics plays a crucial role in highlighting different aspects of model performance, from syntactic correctness to real-world execution efficiency.

1) Exact Matching Accuracy(EM): Exact Matching Accuracy [12]requires that the SQL statement generated by the model must be exactly the same as the ground-truth answers. It is a critical metric for evaluating the performance of Text-to-SQL. This stringent metric requires an identical query due to the diversity of syntax in SQL statements. Completing the same results may not always be possible to determine the correct SQL query for the same task uniquely.

$$\text{Exact Matching Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\hat{Y}_i = Y_i\right)$$

where:

- **Total number of queries** N: This represents the total number of natural language questions used in the evaluation.
- **Generated SQL query** $\hat{Y}_i$: This is the SQL query generated by the model for the i-th natural language question.
- **Reference SQL query** Y_i: This is the correct SQL query for the i-th natural language question, serving as the reference standard.
- **Indicator function** $\mathbb{I}(\cdot)$: If the generated SQL query $\hat{Y}_i$ exactly matches the reference SQL query Y_i, the indicator function $\mathbb{I}(\cdot)$ equals 1; otherwise, it equals 0.

2) Execution Accuracy(EX):

$$\text{Execution Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(f(Q_i, S_i) = A_i\right)$$

- N is the total number of queries.
- Q_i denotes the i-th natural language question.
- S_i denotes the database schema corresponding to the i-th question.
- A_i denotes the reference answer for the i-th question.
- $f(Q_i, S_i)$ denotes the result returned by executing the SQL query generated by the model for the i-th question on the database schema S_i.
- $\mathbb{I}(\cdot)$ is the indicator function, which equals 1 if the condition inside is true, and 0 otherwise.

3) Valid Efficiency Score: Valid Efficiency Score (VES) is a common evaluation metric used in the text-to-SQL field to assess the performance of models. It considers both the correctness and the efficiency of the generated SQL queries. The metric accounts for the validity of the SQL query (whether it executes correctly and produces the correct result) and its execution efficiency (how fast it runs).

The formula typically consists of two components:

Query Validity (Correctness): Evaluates whether the generated SQL can be successfully executed and returns the same result as the ground truth SQL query. Query Efficiency: Measures how efficiently the generated SQL query is executed compared to the ground truth.

$$VES = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbb{I}(Q_i^{\text{gen}} = Q_i^{\text{gold}}) \cdot \frac{T_{\text{gold}}}{T_{\text{gen}}}\right)$$

where:

- N represents the total number of queries;
- Q_i^{gen} denotes the generated SQL query for the i-th example;
- Q_i^{gold} denotes the ground truth SQL query for the i-th example;
- $\mathbb{I}(\cdot)$ is the indicator function, equal to 1 if the generated SQL is equal to the ground truth, and 0 otherwise;
- T_{gold} is the execution time of the ground truth SQL query;
- T_{gen} is the execution time of the generated SQL query.

4) Test-suite Accuracy (TS): Test-suite Accuracy (TS) is a key evaluation metric that is designed to test the performance of models on a diverse and concentrated set of test databases. This metric constructs a small, focused database test suite from a large collection of randomly generated databases, ensuring that the suite has a high code coverage rate for accurate SQL queries. By testing the model's performance on this suite, Test-suite Accuracy measures how well the model predicts SQL queries that produce the correct results across various database scenarios. The goal is to measure the strict upper limit of semantic accuracy, as it evaluates the performance not only in terms of SQL structure but also execution outcomes.

$$\text{Test-suite Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(f(Q_i, D_i) = R_i\right)$$

where:

- N represents the total number of queries in the test suite.
- Q_i is the SQL query generated by the model for the i-th example.
- D_i represents the corresponding database for the i-th query.
- R_i is the expected result when executing the reference SQL query on D_i.
- $f(Q_i, D_i)$ is the result produced by executing the model-generated SQL query on database D_i.

- $\mathbb{I}(\cdot)$ is the indicator function, which equals 1 if the generated query result matches the expected result R_i, and 0 otherwise.

B. Datasets

Datasets are a fundamental component of training Text-to-SQL tasks. Training the system with a large corpus allows it to automatically acquire the mapping relationship between natural language and SQL without relying on hand-written rules. The dataset for Text-to-SQL is usually manually labeled with natural language questions and corresponding SQL queries. A natural language question is a question restricted to the domain in which the database data is located and whose answer comes from its database. In essence, the question describes a SQL query. Executing the SQL query yields the answer to the question from its database [30]. Table I provides an overview of commonly used Text-to-SQL datasets, summarizing key features such as dataset size, interaction type, and domain coverage, which are essential for evaluating the generalization capability of Text-to-SQL models. Datasets in this domain typically have the following characteristics.

Single/Cross Domain: database data source scenarios, according to the number of scenarios involved, can be divided into single fields and multiple fields [31], such as catering data and tourist attractions for two fields.

Number of dialogue rounds: according to the number of dialogue rounds required for complete SQL generation, the dataset is divided into single and multiple rounds.

SQL Complexity: Based on the SQL complexity corresponding to the natural language problem, the dataset is classified into simple and complex problems, where the problem complexity is determined by the number of keywords, nesting level, and number of clauses.

1) Single-Domain Datasets: **ATIS** [32] is derived from the Airline Ticket Subscription System (ATIS), which generates SQL statements from user questions and is a single domain, context-sensitive dataset.

GeoQuery [33] is derived from U.S. Geography, and consists of 880 questions and SQL statements, and is a single domain, context-independent dataset.

Scholar [34] provides a benchmark that reflects the query requirements of real academic databases. The dataset contains 816 annotated natural language queries and corresponding SQL queries, covering a wide range of information retrieval needs in the academic domain. The queries in the dataset cover information about academic papers, authors, citations, journals, keywords, and databases used.

Advising [35] dataset is a Text-to-SQL task assessment dataset focused on student academic advising contexts, drawn from the University of Michigan's course database. Questions are written by students to simulate real questions they might ask during academic advising, and each question is manually annotated with the corresponding SQL query and reviewed by multiple annotators to ensure accuracy and helpfulness.

TPC-DS [55] is a commonly used benchmark in the field of database systems, which, compared to Bird and spider, has a significantly more complex structure of its dataset and is able to model problems in displays more effectively. Even the current state-of-the-art generative AI models fall short in their performance in generating accurate queries on it. [56]

2) Cross-Domain Datasets: **WikiSQL** The two datasets, ATIS & GeoQuery, have problems such as small data size (less than a thousand SQL sentences) and simple annotation. So, in 2017, Victor Zhong and other researchers annotated 80,654 training data based on Wikipedia, covering 26,521 databases named WikiSQL [57]. It posed new challenges to the design of the model, requiring the model to better construct the mapping relationship, make better use of the attributes in the tables, and pay more attention to the decoding process.

Spider However, WikiSQL also has a problem; it only involves one table per question and only supports simple SQL operations, which is not very suitable for our daily life scenarios. So, in 2018, researchers at Yale University introduced the Spider [12] dataset, which is currently the most complex Text-to-SQL dataset. It has the following characteristics:1) The domain is richer, with more than 200 databases from 138 domains. Each database corresponds to 5.1 tables on average, and the databases appearing in the training set and the test set do not overlap.2) The SQL statements are more complex, containing orderBy, union, except, groupBy, intersect, limit, having keywords, and nested queries. The authors divided the SQL statements into 4 levels of difficulty based on their complexity (number of keywords, degree of nesting), and WikiSQL only has EASY difficulty under this division.

KaggleDBQA KaggleDBQA [44] is a cross-domain evaluation dataset of real Web databases with domain-specific data types, original formats, and unrestricted questions. It includes 272 examples across 8 databases with an average of 2.25 tables per database. The dataset is known for its real-world data sources, natural problem-creation environment, and database documentation with rich domain knowledge.

DuSQL DuSQL [46] is a large-scale Chinese dataset designed specifically for cross-domain text-to-SQL tasks, filling the gap of lack of labeled data in the Chinese domain. The dataset manually analyzes real-world problems in several representative applications and contains a large number of SQL queries involving row or column computations.

BIRD [45] dataset focuses on aspects like grammatical formulation, ambiguity, specificity, and alignment with the database schema. This benchmark aims to bridge the gap between academic research and real-world applications by focusing on the comprehension of database values and the efficiency of SQL queries in large databases. It introduces challenges such as dealing with dirty database contents, requiring external knowledge to link natural language questions with database contents, and ensuring the efficiency of SQL queries. The dataset includes questions of varying difficulty levels—simple, moderate, and challenging. Each question in the dataset is annotated with an optional evidence value, providing context that helps in understanding the query.

BEAVER [47] is used to evaluate the performance of large language models in complex SQL generation tasks. Existing publicly available Text-to-SQL datasets (e.g., Spider and Bird) fall far short of real enterprise environments in terms of database structure and query complexity, resulting in large

TABLE I
DATASET STATISTICS. PROVIDES A SUMMARY OF KEY TEXT-TO-SQL DATASETS, HIGHLIGHTING ATTRIBUTES SUCH AS SIZE, INTERACTION TYPE, DOMAIN COVERAGE, LANGUAGE, RELEASE YEAR, AND LINK

Name	Number			Turn	Type	Language	Year	Link
	Train	Valid	Test					
ATIS [32]	4473	497	448	Single-Turn	Single-Domain	English	1990	https://github.com/howl-anderson/ATIS_dataset
GeoQuery [33]	600	-	280	Single-Turn	Single-Domain	English	1996	https://www.cs.utexas.edu/ ml/nldata/geoquery.html
Scholar [34]	600	-	216	Single-Turn	Single-Domain	English	2017	https://metatext.io/datasets/scholar
Advising [35]	4791	-	-	Single-Turn	Single-Domain	English	2018	https://github.com/jkkummerfeld/text2sql-data
EHRSQL [36]	5124	1163	1167	Multi-Turn	Single-Domain	English	2023	https://github.com/glee4810/EHRSQL
WikiSQL [37]	56355	8421	15878	Single-Turn	Cross-Domain	English	2017	https://github.com/salesforce/WikiSQL?tab=readme-ov-file
Spider [12]	7000	1034	2147	Single-Turn	Cross-Domain	English	2018	https://github.com/taoyds/spider
Spider-SYN [38]				Single-Turn	Cross-Domain	English	2021	https://github.com/ygan/Spider-Syn
Spider-DK [39]				Single-Turn	Cross-Domain	English	2021	https://github.com/ygan/Spider-DK
Spider-SS&CG [40]				Single-Turn	Cross-Domain	English	2022	https://github.com/ygan/SpiderSS-SpiderCG?tab=readme-ov-file
Spider-GEN [41]				Single-Turn	Cross-Domain	English	2023	https://github.com/ManasiPat/Spider-Gen
Spider-Realistic [42]				Single-Turn	Cross-Domain	English	2024	https://zenodo.org/records/5205322#.YTts_o5Kgab
Spider-SSP [43]				Single-Turn	Cross-Domain	English	2021	https://github.com/google-research/language/tree/master/language/nqg
KaggleDBQA [44]	272	-	-	Single-Turn	Cross-Domain	English	2021	https://www.microsoft.com/en-us/research/publication/kaggledbqa-realistic-evaluation-of-text-to-sql-parsers
BIRD [45]	8659	1034	2147	Single-Turn	Cross-Domain	English	2023	https://github.com/MohammadrezaPourreza/Few-shot-NL2SQL-with-prompting
BigTable-0.2k [42]	200	-	-	Single-Turn	Cross-Domain	English	2024	-
DuSQL [46]	18602	2039	3156	Single-Turn	Cross-Domain	English	2020	https://paperswithcode.com/paper/dusql-a-large-scale-and-pragmatic-chinese
BEAVER [47]	93	-	-	Multi-Turn	Cross-Domain	English	2024	https://peterbaile.github.io/beaver/
CoSQL [48]	2164	292	551	Multi-Turn	Cross-Domain	English	2019	https://yale-lily.github.io/cosql
ADVETA [49]	-	-	-	Multi-Turn	Cross-Domain	English	2022	https://github.com/microsoft/ContextualSP
CSpider [50]	6831	954	1906	Single-Turn	Cross-Domain	Chinese	2019	https://github.com/taolusi/chisp
TrustSQL [51]	-	-	-	Single-Turn	Cross-Domain	English	2024	https://github.com/glee4810/TrustSQL
SParC [52]	9025	1203	2498	Multi-Turn	Cross-Domain	English	2018	https://github.com/taoyds/sparc
CHASE [53]	3949	755	755	Multi-Turn	Cross-Domain	Chinese	2021	https://github.com/xjtu-intsoft/chase
SQUALL [54]	9030	2246	4344	Single-Turn	Cross-Domain	English	2020	https://github.com/tzshi/squall

language models that perform well in these tasks but poorly in real-world enterprise environments. The BEAVER benchmark constructs a more representative dataset by anonymizing the data warehouses of the two enterprises that contain complex table joins and aggregation.

CoSQL [48] dataset is a cross-domain conversational Text-to-SQL dataset designed for building general-purpose database query dialogue systems. The dataset contains more than 3000 dialogues, more than 30,000 dialogue rounds, and more than 10,000 annotated SQL queries covering 200 complex databases in 138 different domains.CoSQL collects dialogues by means of Wizard-of-Oz (WOZ), where a simulated user on one side and a SQL expert on the other side ask database query questions, and the user builds corresponding SQL queries and returns results.

CHASE [53] dataset is a large-scale and practical Chinese dataset with cross-database context dependencies. The dataset aims to bridge the gap between existing datasets in terms of context dependency and SQL query complexity. CHASE contains 5,459 problem sequences with a total of 17,940 problems annotated with SQL queries distributed across 280 multi-table relational databases.

EHRSQL [36] is a Text-to-SQL benchmark dataset for Electronic Health Record (EHR) data, aiming at evaluating the real-world applicability of the model in the healthcare domain. The dataset consists of real questions from 222 hospital staff (including doctors, nurses, insurance auditors, etc.), covering common retrieval needs in healthcare scenarios, such as patient information querying, complex statistical computations, etc.

3) Augmented-Datasets: **ADVETA** [49] is the first benchmarking dataset specifically designed to evaluate the robustness of Text-to-SQL models under table perturbation. While previous research has focused on perturbation on the natural language side of the problem, ignoring the diversity of tables themselves.

Spider-DK [58] focuses on the ability of the model to work with data obtained from domain-specific knowledge. It transforms the challenge to the use case using data such as Implicit Query Column, Simple Reasoning, Synonym Replacement, and Conditional Generation, etc. Spider-DK is a way to determine if a model has a basic understanding of data and uses old information to process new content.

Spider-SS&CG [59] aims at realizing tasks through Schema Simplification and Complexity Generation in real databases. As the training progresses, its database is simplified and complicated more. Spider-SS & CG combines these two elements to check, respectively, the performance of a simplified and complexified database structure.

Spider-SYN [60] (Synonym Substitution) suggests the introduction of synonym changing to model the synonym vari-

ation in the real language. The model will be tested for robustness using a given dataset which will replace schema-related words such as table names and column names with their synonyms. The model's schema linking ability is a problem in this case, as the naming is the opposite of what it should be, e.g., cross-domain tests.

Spider-SSP [43] (Schema-Specific Parsing) concentrates on the parsing oriented towards a schema to test whether the schema generalization is feasible by changing the names of columns and tables in the schema. It insists on schema-dependent parsing capabilities that will be tested against unknown database structures or schema name changes.

Spider-Realistic [61] generates questions and corresponding SQL statements (pairs) that carriers more of a relevant value for real-world applications. The data set is aimed at enterprise performance in real-life databases, which requires the model to work under real-world conditions, particularly in dealing with complex queries that have multiple levels.

CSpider [50] addresses the status quo of Chinese as a low-resource language in this task area. Chinese text needs to be processed by word splitting, while SQL keywords and column names of database tables are usually written in English. Word-based semantic parsers are susceptible to word-splitting errors, and cross-language word embedding is very helpful for text-to-SQL mapping.

TrustSQL [51] aims to evaluate models on their ability to either generate a correct SQL query or abstain from it when the question is unanswerable, making a prediction, or the generated SQL is likely to be incorrect. The data is split in two ways: question-based, which assesses the model's ability to handle different phrasings, and query-based.

BigTable-0.2k [42] built upon the BIRD dataset, which includes various question complexities and dataset sizes. To comprehensively evaluate the performance of LLMs, the authors design five distinct tasks, Text-to-SQL, SQL Debugging, SQL Optimization, Schema Linking, and SQL-to-Text.

SParC SParC [52] demonstrates complex contextual dependencies, and greater semantic diversity, and requires the model to be able to generalize over unseen domains due to its cross-domain nature and unseen databases used in testing

IV. Methodology

In this section, we describe in detail the traditional Text-to-SQL methods and their evolution. Traditional methods mainly rely on bi-structured models. These approaches typically use LSTM-based and Transformer-based models to generate SQL queries by learning a contextual representation between natural language questions and database tables. In this context, this paper also discusses a variety of existing frameworks and techniques, including models based on techniques such as graph neural networks, table semantic understanding, and schema linking. These techniques and frameworks further enhance the accuracy and efficiency of Text-to-SQL by improving schema linking, reducing error propagation, and optimizing the use of pre-trained models.

In addition to the traditional LSTM and Transformer models, the Pre-trained Models such as BERT, GPT, and T5 have dramatically changed the direction of the Text-to-SQL field. Trained on large-scale text data, these models are able to capture rich semantic representations between natural language and SQL, not only by fine-tuning them to specific tasks but also by combining them with Prompt Engineering to realize applications that do not require large amounts of labeled data. They show high scalability and flexibility in handling complex queries and cross-domain tasks and have become the core of modern text-to-SQL systems. Meanwhile, Fine-tuning optimizes the performance of pre-trained models in specific domains by further supervised training on specific datasets.

Figure 2 provides a detailed taxonomy of various Text-to-SQL methods, comparing key attributes such as backbone models, optimization strategies, query generation strategies, and datasets used. This table offers a comprehensive overview of different approaches and highlights the role of pre-trained models in modern text-to-SQL systems.

In addition, the introduction of LLM Agents represents a new direction in the development of Text-to-SQL. These systems are not only capable of generating SQL queries, but also interacting with users through multiple rounds of dialog to dynamically adjust the generation strategy. It points out the direction for the further development of Text-to-SQL technology, showing higher flexibility and adaptability. In what follows, we discuss in detail the specific implementation of each approach.

A. Traditional Text-to-SQL Methods

This part focuses on text-to-SQL models based on traditional deep neural networks. The focus of the discussion is on the model architecture and its application to specific tasks. For instance, the model needs to understand the natural language input and show how it corresponds to the database structure of tables and columns [62], [63].

In the earliest Text-to-SQL tasks, SQL queries were typically generated through a template- or rule-based approach. The model would map the natural language input to a set of predefined SQL templates. Performance is significantly limited when faced with complex database structures and queries. As deep learning techniques matured, more sophisticated methods such as LSTM-based and Transformer-based models began to emerge, offering improved capabilities in handling more complex query scenarios.

The deep neural network system must identify and map out the target, which is the matching data according to the user input. This model generally represents a dual structure, the encoder and the decoder, each of which performs its own duty. The first one, the encoder, is responsible for capturing the semantics of a natural language (NL) question; the second one, the decoder, generates an SQL query based on the representation extracted from the encoder input data [64].

a) LSTM-based Methods: Traditional methods using LSTM and Transformers generate SQL queries by learning the contextual representation of input natural language (NL) questions and database tables as well. LSTM-based models were among the first deep learning approaches to be applied to text-to-SQL tasks, as they could effectively model the

sequential dependencies between natural language questions and SQL queries. Models such as use Bi-LSTM to learn the semantic representation of question-SQL pairs. While pre-trained models generally outperform these approaches, LSTM-based systems are still employed in some cases. These methods, especially LSTM and its variants, leveraged sequential processing to capture context; however, they faced challenges when dealing with long-range dependencies in complex queries. Traditional techniques work with LSTM [8] and Transformers [9], [65] by learning contextual representation from input natural language questions and database tables. Models like TypeSQL [66], Seq2SQL [67], SQLNet [68], and SyntaxSQLNet [69] use Bi-LSTM to learn the semantic representation of question-SQL pairs. Pretrained models yield better accuracy compared to these methods. LSTM-based systems, however, are still employed in certain case studies. For example, IRNet [70] and RAT-SQL [71] scheme are introduced that takes advantage of LSTM in their grammar decoder to yield abstract syntax trees.

Although these LSTM-based methods helped in laying the foundation for text-to-SQL research, the limitations in scalability and the ability to generalize across different domains led researchers to explore more advanced architectures, such as Transformer models.

b) Transformer-based Methods: A short while ago, works focused on Transformer-based models, complementing them with improved Text-to-SQL performance-oriented structures. These models introduced a new paradigm by employing self-attention mechanisms, which allowed them to handle long-range dependencies more effectively than LSTM models. This shift in architecture significantly improved the models' ability to generate accurate SQL queries, even for complex database schemas. Such settings can be observed. For instance, GraPPa [72] introduces grammar-augmented pretraining, which enriches schema understanding. GAP [73] focuses on contextual representations for tasks of semantic parsing. StruG [61] presented a new approach called structure objective for text-based table encoding, which emphasizes on the structure of tables. SCoRe [74] proposed schema-compound embedding for structured data tasks. TaBERT [75] was trained to jointly understand the semantics of tabular and textual data using internal pretraining, which resulted in a greater grasp of semantic parsing. TAPAS [76] pretrains table-based fresh data for tabel data question answering task. MATE [77] entertains multi-view attention for table learning tasks. TableFormer [78] suggests a stable transform model for data table comprehension. TAPEX [79] utilizes table pretraining in terms of logic procedures. S^2SQL [80] injects syntax into question-schema interactions to improve SQL code generation. IST-SQL [81] project interaction state in multi-turn SQL tasks. At this point, IGSQL [82] aids schema linking in intricate SQL queries. GAZP [83] is a neural approach to a changed zero-shot parsing designed for table question answering. EditSQL [84] assembles an editor-like generation method for SQL query formulation. These Transformer-based models introduced innovative techniques such as attention mechanisms and pretraining, which greatly enhanced their performance, particularly in schema linking and complex query generation tasks. These difficulties illustrate various endeavors to improve text-to-SQL performance, where the major three areas are schema linking, propagation of errors, and pre-trained language models-based query generation.

TRANX [85] is an artificial intelligence system designed to decipher the syntax and the semantics of the loaded text data. Within the foundation of TRANX, there is a neural abstract syntax parser that is fundamentally the transfer of the natural language into the internal representation. Such as it is in the case here, this is the deep learning practice, where the model is working first by dividing the text into smaller portions and then figuring out the relationships between the individual components. It exploits in-context learning as well by feeding the model its own examples that are in fact frequent in the context it has been introduced to. Notably, the system of architecture has multiple prompt design strategies for improving its performance, one of these strategies is by selecting the examples, which are the most relevant to the input query, and adding extra information on the schema during the task execution. A number of frameworks and techniques have been outlined in the present day which serve the purpose of improving this sphere. Bridge [86] links and integrates textual and tabular data to provide schema linking with greater efficiency. SDSQL [87]improves schema dependency in Text-to-SQL tasks by leveraging schema dependency structures. SLSQL [88] discusses different techniques used for schema linking in Text-to-SQL models. IESQL [89] increases the efficiency of mention extraction and schema grounding in Text-to-SQL tasks. SEAD [90] is an end-to-end Text-to-SQL generation method that employs more sophisticated schema linking and is the end-to-end method. SmBoP [91] proposes a bottom-up, non-autoregressive method that tackles Text-to-SQL parsing through sequence-dependent probabilistic grammars DT-Fixup [92] fine-tunes Transformer small-based models to enhance their capability for small datasets in Text-to-SQL tasks. Relation parsing in SQL is achieved through semi-autoregressive parsing with hybrid relation-aware RasaP [93] models. GNN [94] utilizes graph neural networks being employed for the purpose of schema representation for the purpose of improving schema linking. ShadowGNN [95] suggests using a graph projection neural network, which is a promising idea as it can boost schema referencing and text-to-SQL accuracy. SADGA [96] employs a structure-aware dual-graph structure that provides better schema alignment for Text-to-SQL tasks. LGESQL [97] uses graph representation learning for schema linking in Text-to-SQL query generation. UnifiedSKG [98] makes it possible to unify multitasking across structured knowledge grounding tasks including Text2SQL.

These various frameworks and techniques represent significant advancements in the Text-to-SQL domain, particularly in addressing the challenges of schema linking, query parsing, and improving overall model efficiency. By incorporating methods such as schema dependency structures, graph neural networks, and semi-autoregressive parsing, modern Text-to-SQL models are better equipped to handle complex databases and intricate SQL queries. As Text-to-SQL models continue to evolve, the combination of robust schema linking mechanisms, powerful pre-trained models, and advanced parsing techniques

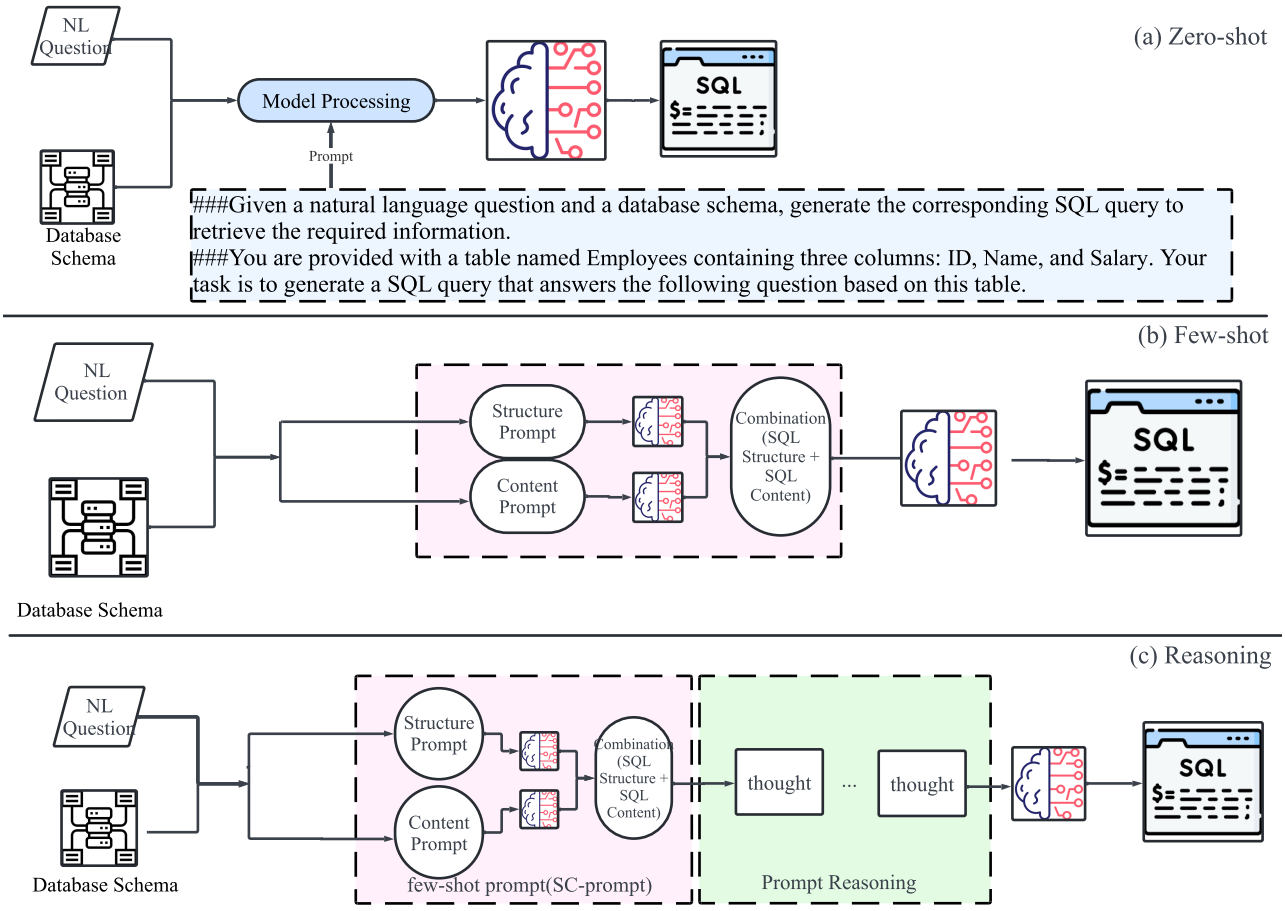

Fig. 3. Prompt Engineering Methods. The figure illustrates three key prompt engineering approaches for Text-to-SQL: (a) zero-shot, where the model generates SQL without prior examples; (b) few-shot, which provides a few examples to guide query generation; (c) Reasoning, breaking down the reasoning process step-by-step for complex queries.

promises to bring about more accurate, scalable, and efficient systems capable of handling real-world tasks.

B. Text-to-SQL with Prompt

In the process of using prompt engineering, a technique is used to improve the performance of the language models and reinforce the reliability of these models by creating input prompts in detail. While traditional methods such as LSTM-based and Transformer-based models focus on learning contextual representations through supervised training, prompt engineering offers an alternative approach that leverages the capabilities of large pre-trained language models (LLMs) without requiring additional fine-tuning. This particular method uses the identification of specific prompts or sentences to help guide the model in creating meaningful outputs that are relevant to the intention of the user (See Fig. 3). Unlike earlier methods that required extensive training on labeled datasets to improve accuracy and generalization, most of the time, Prompt Engineering does not require any training on the model, so it works on the basis of the pre-trained model, which creates SQL directly. The merit of such a technique is in executing results in a short time while not incurring any additional computational charges for model tuning.

This method is vital for problem-solving techniques like hallucinations. The authority and persuasion of the model's answers are heightened by this instruction, since it is grounding the model. The basic principle of operation is to generate, step by step, the next output token with the highest probability based on the relevant prompts of the input. Therefore, as a contrast to the traditional training-dependent approaches discussed earlier, the core of tackling the Text-to-SQL task with LLMs [99], [100]lies in finding the optimal prompt.

1) Zero-shot Prompt: In the zero-shot approach, the model has zero specific training data about the task. Problems posed directly to the model usually consist of a task description, a test problem, and a corresponding database without any cases. [13], [14], [42], [101], [102] hints at the effect of structure on the zero-shot performance of large language models (LLMs). The model can answer questions accordingly and can make initial judgments based on the large amount of data. But this method needs both large-scale pre-training language models and a considerable amount of data for adaptation to ensure top-notch quality. In regard to the broader fields of database contents, accuracy can be compromised, and the SQL output can be insufficiently accurate. Nevertheless, this way can be quickly applied to new tasks and areas without spending time on training again. Using this approach method, the output may sometimes be obtained, which could, therefore, be described as having some flexibility variations or unexpected changes. The techniques of using prompts in the case of zero-second

text-to-SQL tasks are dedicated to the work of [103] [20] [104] [105] [14] [106] [107].

2) Few-shot Prompt: A few-shot is a question posed to the model with only a small number of cases given, and in response, the model generates an answer formulated on the basis of the initiated cases, which results in a better comprehension of the task. However, here we consider only the cases of good quality. Unlike Zero-shot, Few-shot can significantly yield improved performance of the model, particularly for those tasks characterized by their complexity [108].

SC-prompt [109] employs the divide-and-conquer method as a way of tackling the problem of translating the text into two more manageable phases: the structure stage and the content stage. The first stage results in the generation of a basic SQL structure, which contains, but does not limit itself to, the table and column names as their placeholders. It should initially be given a structure where the specific details are expected. The next step consists of replacing the tokens of general description with concrete values. For this purpose, it applies the hybrid approach of text pre-processing that links the static word embeddings to the learnable vectors.

MCS-SQL [16] is the method proposed to enhance the precision. Its operation relies on three main activities: schematic linking, parallel generation of SQL, and picking one that meets the purpose. The first stage is schema linking, which later will be performed in two stages: the tables joining and the columns joining. The interaction is repetitive, as the sentence is exchanged in both phases. Afterwards, it links with Schema and generates more than one candidate SQL query. It is through multiple prompts, which aid the exploration of a much wider parameter space, that one is able to accomplish this task. It is the LLM (Language Model) that produces different SQL queries as output, based on those prompts. The primary goal now is to pick the most precise SQL query that matches the input. LLM takes into its consideration the reasoning steps and the scores to choose as the best answer.

A range of sampling techniques investigating the inspiration demos [110] has been carried out in a selection of declarative programming tasks. It seeks to enhance performance by ensuring that there is an equilibrium between likeness and diversity between demos. The random sampling is then used as a standard to check the effectiveness of the given strategies.

SQL-PaLM [111] by the first selection strategy of few-shot examples talks about a task of selection both similarity and diversity between various examples. Along with various sampling methods and their combinations, a random sample is taken as a standard for performance evaluation.

3) Chain of Thought (CoT): The Chain of Thought (CoT) [112] prompts activate complex thinking skills with the help of these intermediate steps that are based on reasoning. Its effectivity can be stretched by mixing it with sample-less prompts for complex problem-solving, which implies synthesis before diagnostics. The two key improvements include the enhancement of the reasoning ability in terms of how to notice is the Chain of Thoughts or the CoT principle of the GPT model: [18], [20], [104], [105], [113]. The Building of a GPT Model was to provide incorrect answers to two claims and give an explanation for the computational approach. In addition, it was found that the model's reasoning performance could be significantly improved by adding key sentences to the prompts, such as "Let's think step by step". The experiments show that such a method helps the rationalization of the GPT even when the relevant samples are not provided. Among these methods are those that provide deep and true insights into the model's reasoning in the case of Text-to-SQL tasks.

Chat2Query [114] deploys its zero-shot SQL generation capability, which allows users to input natural language queries and receive SQL outputs without the need for prior model training or domain-specific fine-tuning. The system integrates a text-to-SQL generator, SQL rewriter, SQL formatter, and data-to-chart generator, streamlining the process from query to data visualization. Utilizing the Chain-of-Thought prompt enables step-by-step SQL generation, which improves the accuracy of generated queries, especially in complex or ambiguous situations. The system is built on TiDB Serverless, ensuring scalability and adaptability to different data workloads.

ACT-SQL [104] delves into the ability of LLM to solve some few-shot learning tasks by examining how the few-shot learning strategy affects the model performance. A hybrid method, based on both the static and the dynamic examples, has been proposed, for example, a selection that is current to test the sample given, and the experimental results showed the efficiency of the chosen strategy.

C. Text-to-SQL with Fine-Tuning

Fine-tuning is still an important method within LLMs and still offers a high level of improvement over the use of low-cost prompt methods. Fine-tuning methods rely on models that are already pre-trained but are further fine-tuned to fit specific tasks and domains. Depending on the scope of the fine-tuning, it can be categorized in two ways.

1) Full-Parameter Fine-Tuning: Full fine-tuning means fine-tuning all the parameters of the model. In this approach, the entire model is trained with domain-specific data to optimize its performance in a specific task. Full fine-tuning is usually applied in those cases where high accuracy or specific tasks are required. For example, the Text-to-SQL task on the Spider dataset requires the model to generate extremely accurate SQL, in which case all model parameters need to be fine-tuned to improve accuracy.

Knowledge-to-SQL [17] intends to improve the DELL model's capability to produce relevant knowledge more swiftly, thus giving an added boost to the performance of these Text-to-SQL systems. Semantic techniques are adopted to find the best matching table to the query. First, supervised fine-tuning is performed, and then the model is refined by applying the Direct Preference Optimization (DPO) algorithm.

SGU-SQL [18] is a system depending on the structure of the questions and schema. To begin with, user queries and databases are linked through an enriching framework. This connection can be made through graph-based structures, and the system SGU-SQL will decompose the complicated, interrelated structures through grammar trees. The system is also based on a specially designed structure-based linking mechanism that connects the query structure at the node

level. The process begins with the composition of the node representations, and later, it is directed by the propagation of messages, which self-structures and, therefore, makes the model capture the main relationships. Ultimately, the structure-blindness measure is utilized by the model to merge the schema graph and query graph, and afterward, the merged information is transferred to the adjacent graph.

DIN-SQL [115] breaks down the complex task of converting test to SQL queries into smaller, manageable sub-tasks, which helps LLMs perform better. Firstly, its schema linking module identifies references to database schema and condition values in NL queries. It will classify each query into one of three classes: easy, non-nested complex, and nested complex. This classification helps in using different prompts for each query class. For complex queries, it includes an intermediate representation called NatSQL, which simplifies the transition from natural language to SQL by removing certain SQL operators that do not have clear counterparts. Then the SQL generation module generates the final results based on the solutions of sub-tasks. After generating the initial query, the self-correction module reviews and corrects any errors. The new method achieved the state-of-the-art on the Spider dataset and BIRD benchmark with the accuracy of 85.3% and 55.9%.

MAC-SQL [24] could be much more efficient in its work of query generation for SQL by involving multiple intelligences: the fundamental Decomposer intelligence, the Selector intelligence, and the Refiner intelligence. The Decomposer Intelligence, when given a complex SQL query to resolve, breaks it into fragment sub-problems sequentially through chained reasoning and generates the final SQL query in a stepwise manner. The usage of Selector Intelligence limits the database to eliminate the problem-specific data interference, while Fixer Intelligence repairs mistaken SQL via SQL execution using outside tools. The MSCFT is the method fine-tuning strategy used during SQL-Llama development, and this model is an open-source version based on the Code Llama. This model is trained using the dataset of instructions generated from the MAC-SQL intelligence tasks.

Few-shot learning is combined with Instruction Fine-Tuning that further extends the improvement in the large language model's performance in the text-to-SQL tasks. First of all, [111] proposes consistent decoding and instruction execution-based filtering of errors using a few sample hints. It investigates instruction fine-tuning to widen the coverage of the training data, apply data augmentation, and integrate specific content to the queries. This article also proposed a methodology, although with selection at test time, to further boost the accuracy by combining output and execution feedback in different paradigms.

Symbol-LLM [102] employs a two-stage supervised fine-tuning framework for this purpose. This strengthens the symbolic processing capability of large models. In the first phase (injection), the model acquires the inherent dependencies in the symbolic data based on fine-tuning (SFT) of an underlying model. Thus, these latter became the new optimizators (MLE) in order to handle symbolic tasks. During the fusion period, the model is not only enhanced by the symbolic and natural language data but is also boosted with the combined data.

2) Parameter-Efficient Fine-Tuning: Parameter-efficient fine-tuning fine-tunes only some parameters of the model, usually some specific layers or modules. Such fine-tuning can effectively reduce training time and computational resource consumption while maintaining high performance. This approach usually targets domain-specific texts or structures, preserving the general linguistic knowledge already learned in the pretrained model and optimizing only for subtle differences in the Text-to-SQL task.

For example, when it is about fine-tuning the complexity of SQL statements or the database schema referred to as schema, only the layers or parameters that pertain to understanding of the schema need to be retrained, hence saving the model's training cost and nothing else. The advantage of this way of doing things is reducing the training overhead on the one hand and achieving a happy result of the compromise between effectiveness and efficiency on the other hand. [116] explains the proposed idea of accelerating the decoding process of a large language model by suggesting sample sequences. The technique does not modify the current model, but instead develops a rough instrument that produces possible candidates; the decision of rejection sampling ensures that the output distribution of the model stays unchanged.

In the inference process of [117], some computational steps can be approximated by smaller, more efficient models. Particularly, using a quite involved approximation model, many candidate tokens are generated and based on the trained target model, tokens are verified in parallel to find the ones that are within the target distribution of the model. Once done, the distance from the goal is reduced, leading to quicker model moves that finally end the decoding process.

[19] mentions some techniques to enhance the robustness of models in noisy data by fine-tuning the manually selected pre-trained models and applying robust Regularization (RR). The provided regularization methods, such as fine-tuning using pre-trained models, can help to increase modeling robustness effectively. In pre-training of the model, providing it clean data and then restoring it on noisy data, the model is less likely to avoid design overfitting as a result of noisy labels. Another way is to add an explicit regularization to the pre-training technique mentioned here, and further noise robustness improvement will be achieved, for instance, in the case of PHuber, which is a fine-tuning additionally.

DAIL-SQL [20] presents an exploration of the effect of Supervised Fine-Tuning (SFT). The technique is a systematic approach of SFT by employing LLMs to be fine-tuned for specific Text-to-SQL tasks trained by task-specific training data. It investigates the representation strategies used for supervised fine-tuning with insights from different strategies on the efficiency of supervised fine-tuning.

The paper [118] has various avenues for operations by employing pre-trained models and customizing the pre-trained ones. As for example, through a method of language pre-training called BERT [119], natural languages and database schemas are encoded with the goal of a deeper representation of syntactic and semantic structures.

Moreover, the pre-trained model embeddings are customized using adaptive fine-tuning to respond to the schema of

TABLE II
TAXONOMY OF TEXT-TO-SQL METHODS.

Methods	Released Time	Backbone Models	Access	Optimization Strategy	Query Generation Strategy	Robustness and Error Handling	Dataset	Metrics	Schema Linking
SC-prompt	Jun-23	T5	✓	Task Decomposition	Guided Decoding	-	Spider, CoSQL, GenQuery	EM, EX	✗
MCS-SQL	May-24	GPT-4	✗	Prompt Tuning	Guided Decoding	Self-Consistency	Spider, Bird	EX, VES	✓
SQL-PaLM	May-23	Palm-2	✗	Prompt Tuning	Consistency Decoding	Self-Correction, Self-Debugging	Spider, Spider, Bird-SYN, Spider-DK, Spider-realistic	EX, TS	✓
ACT-SQL	Oct-23	-	-	Chain of thought (CoT)	Greedy Search	Self-Correction	Spider, SParC, CoSQL	EM, EX, TS	✓
Chat2Query	May-24	-	-	Chain of thought (CoT)	-	-	Spider	EM, EX, VES	✗
Knowledge-to-SQL	Feb-24	LLaMA-2-13b	✓	Expert Fine-Tuning	Framework-Based	-	Spider, Bird	EX, VES	✗
SGU-SQL	Feb-24	GPT-4	✗	Expert Fine-Tuning	Guided Decoding	-	Spider, Bird	EX, EM	✓
DIN-SQL	Apr-23	GPT-4	✗	Task Decomposition	Greedy Search	Self-Correction	Spider, Bird	EX, EM	✓
MAC-SQL	Dec-23	GPT-4	✗	Task Decomposition	Greedy Search	Refiner	BIRD	EX, EM, VES	✓
Symbol-LLM	Nov-23	LLaMA-2-Chat	✓	Expert Fine-Tuning	Greedy Search	-	Spider, SParC, CoSQL	EM	✗
DAIL-SQL	Aug-23	GPT-4	✗	Supervised Fine-tuning	Greedy Search	Self-Consistency	Spider, Spider-Realistic	EX, EM	✗
CLLMs	Feb-24	Deepseek-coder-7B-instruct	✓	Performance & Efficiency Enhancements	Greedy Search	-	Spider	EX	✗
StructLM	Feb-24	CodeLlama-Instruct	✓	Expert Fine-Tuning	-	-	Bird, CoSQL, SParC	EX, EM	✓
SQL-GEN	Aug-24	MoE	-	Expert Fine-Tuning	-	-	Bird	EX	✓
CodeS	Feb-24	StarCoder	✓	-	Beam Search	Execution-Guided SQL Selector	Spider, Bird	EX, TS	✓
RESDSQL	Feb-23	T5	✓	Skeleton Parsing	Beam Search	Execution-Guided SQL Selector	Spider-DK, Spider-Syn, and SpiderRealistic	EM, EX	✓
Tool-SQL	Aug-24	GPT-4	✗	Query Error Handling	Python Interpreter	-	Spider, Spider-Realistic	EX, EM	✓
SQLFixAgent	Jun-24	GPT-3.5-turbo	✗	Query Error Handling	Perturbation-Based Query Generation	Refiner	Spider, Bird, Spider-SYN, Spider-DK, Spider-realistic	EX, EM, VES	✓
MAG-SQL	Aug-24	-	✗	Query Error Handling	-	Refiner	Spider, Bird	EX, VES	✓
MAGIC	Jun-24	GPT-4	✗	Expert Fine-Tuning	-	Self-Correction	Spider, Bird	EX, VES	✗
SuperSQL	Jun-24	GPT-4	✗	-	Greedy Search	Self-Consistency	Spider, Bird	EX, EM	✓

the database or the respective language problem. For instance, SQLova [120] and X-SQL [121] utilize multiple post-training, self-organized semi-supervised learning to improve performance. Fine-tuning of other models: through pre-training the models on tabular data and further fine-tuning in the Text-to-SQL task, improvements in the accuracy and robustness of models like TaBERT [122] and Grappa [123] take place.

CLLMs [124] coupling is additionally optimized by updating with the losses of the two streams: consistency loss and autoregressive loss (AR loss). Consistency loss is employed to ensure that the model settles to a final state with minimum energy across any kind of input, increasing the rate of convergence. Through learning the Jacobi trajectories of the model, it is able to generate multiple markers via one iteration step, and thus it needs fewer calculations to output the current results.

StructLM [125] optimizes pretrained models through Instruction Fine-Tuning. Instruction Fine-tuning combines structured data with generic instruction-tuning data to enhance the model's generalization ability on structured knowledge tasks. In addition, the paper explores fine-tuning on top of code pre-training and finds that code pre-training has a significant enhancement effect on processing structured knowledge tasks. [126], [127] uses LoRA [128] and QLoRA [129] fine-tuning to reduce memory requirements and adapt to SQL generation tasks. and provides a standardized set of evaluation pipelines.LoRA freezes the weights of the pre-trained model and injects trainable layers in each Transformer block for efficient fine-tuning and reduced memory footprint.

[130] investigates the performance of language models on task generalization. It was found that expert language models trained for a single task can outperform multitask instruction fine-tuning (MT) language models on unseen tasks. The expert language model avoids negative task transfer and catastrophic forgetting and performs well in learning new tasks compared to the MT model. Through independent training and the Retrieval of Experts (RoE) mechanism, the study demonstrates the potential for selecting appropriate expert models in multi-task scenarios. It uses a Parameter-Efficient Fine-Tuning approach, which reduces training costs and improves efficiency by freezing the underlying pretrained language model and fine-tuning only the addition of extra adapters.

D. Text-to-SQL with Task-Training

Such a strategy involves fine-tuning and training the complete models from scratch for the specifics of the work.

Differing from the two mentioned approaches, Fully Pre-Trained Models range from those off-the-shelf pre-trained language models that are now available but are eclectically selected for the SQL generation tasks. Typically, that includes deep learning methods, such as CODES models which are based on CNN architecture, Mixture of Experts (MoE) models, and Transformers-based models [131].

1) Mixture of Experts Models: MoE model is a new type of architecture that works by introducing several expert modules, and, at each module, one is responsible for a specific type of task or input. The MoE model of text-to-SQL tasks will allow multiple experts to be involved in the different modules of tasks, such as natural language understanding, database schema parsing, and SQL generation, thus making it easier for the system to learn.

SQL-GEN [23] adopts the LLM technique to extend the existing SQL templates and create a tutorial-driven SQL engine that serves purely as a code interpreter for different dialects. The strategy boosts performance on BigQuery and PostgreSQL dialects remarkably. In addition, the paper also recommends an MoE (Mixture of Experts) approach, which is used generally to boost the performance of such models further by combining the DB-specific models into a single system with the help of dialect keyword injections.

2) Transformer-based Models: The Transformer model is a mainstream architecture in the field of deep learning in recent years, which is particularly suitable for generating complex SQL queries by modelling long-distance dependencies through the attention mechanism. Fully trained Transformer models are built from scratch and trained with large amounts of text and SQL data and can well handle cross-domain and complex SQL generation tasks. Such models can show strong generalization capabilities in database query scenarios, especially in cross-domain and multi-language environments.

CodeS [21] is specifically designed for Text-to-SQL tasks. The paper points out that while existing large-scale language models (e.g., GPT-4, ChatGPT) perform well in the task of generating SQL, their closed-source nature poses data privacy risks and high inference costs. Accordingly, CodeS has become an open-source alternative that seeks to achieve effective and accurate SQL generation by reducing the number of parameters (1B to 15B) and a pre-trained SQL architecture that is fine-tuned on SQL generation tasks. Therefore, it applies database hints that filter relevant tables, columns, and values, such as BM25, to generate accurate SQL queries. Regarding the new domain adaptation, CodeS generates automatically a huge amount of (question, SQL) pairs through the data augmentation techniques, bi-directional.

MIGA [22] is based on PLMs like T5, which is highly proficient in the Text-to-SQL conversion by breaking the main goal into a bunch of smaller but interconnected sub-goals and recognizing the relationships learned during pre-training, so that a Seq2Seq formatted Text-to-SQL could be generated. The three sub-tasks—Related Schema Prediction (RSP), Turn Switch Prediction (TWP), and Final Utterance Prediction (FUP)—are implemented to boost the model accuracy and intelligence by exposing it specifically to the aforementioned SQL generation aspects. Moreover, it is imperative for MIGA to introduce four types of SQL perturbations that aim to minimize dependencies on previously created SQL instructions and to solve the problems of error propagation. Therefore, the model's accuracy will be enhanced, as it will be able to process diverse conversational inputs in a more robust fashion. When plaguing the model with specific languages, the model is guided by these specific language prompts, which are based on the T5 training methodology. This method involves combining a specific task prompt to a training sample pertaining to an input to a given target task.

SQLova [120] presents significant progress observed in both execution accuracy and logical form accuracy. Taking advantage of BERT for embedding form text and further generation of SQL queries through multi-layer LSTM, the model applies form context as well. It must be apparent that the SQL generation master-theory, which integrates form-aware BERT and execution-guided decoding (EG) methods, achieves excellent results against the SQL.

RESDSQL [132] enhances the Text-to-SQL task by decoupling schema linking from skeleton parsing, in which the first process is to identify relevant database tables and the other is to establish the SQL queries structure. This helps to simplify the process of creating accurate SQL queries. This model features an enhanced encoder that prioritizes the most relevant.

[133] introduces two approaches to improve generalization in Text-to-SQL semantic parsing based on pre-trained language models. Token Preprocessing helps the language model's tokenizer generate semantically meaningful tokens by processing the naming conventions of database schemas and SQL queries. This includes converting serpentine nomenclature to a natural naming format, handling column references for dot notation, and extending the spelling of some SQL keywords. Component Boundary Marking inserts special markers at aligned component boundaries between NL input and SQL output. These markers are inserted in the input and output to help the model identify semantic component boundaries and enhance the language model's ability to generalize to combinations.

[134] explored the performance of different data models under real user queries Two specific approaches were used. valueNet: a BART-based encoder using Intermediate Representation (IR) to transform natural language into SQL. the architecture and mechanisms of IRNet were used. in particular, connecting entities in a natural language problem to tables in the database through Schema linking, columns, and data values. There is also T5-Picard, which is based on a variant of the T5 model that incorporates the Picard method [135], which is an incremental parsing method that constrains the decoded output of the language model to valid SQL statements. [136] introduces multi-task training, where different Text-to-SQL tasks (CoSQL, Spider, SParC) are combined and discrete task-specific prompts are used Two reordering methods are employed: the Query Plan (QP) Model: generates SQL queries with predictions of whether the query should contain specific SQL clauses. and Schema Linking (SL): improves SQL generation by performing Schema Linking on the dialog context, ensuring that column and table names in SQL queries are consistent with natural language input.

[137] enhances prompts primarily through de-semanticization and skeleton retrieval. Galois [138] extracts structured data from LLMs by decomposing SQL queries into multiple steps and converting these steps into natural language prompts, it enhances the process primarily through de-semanticization and skeleton retrieval. Not only the queries from LLMs but also traditional databases can be processed using this technique that allows users to query the information contained in LLMs with SQL commands.

E. Text-to-SQL with LLM Agent

Intelligent agent-based Text-to-SQL systems have become a cutting-edge solution for dealing with complex SQL query generation tasks. By collaborating with multiple intelligences, the LLM Agent framework not only automatically generates SQL queries, but also dynamically adapts and corrects SQL statements, handles database matching issues, and improves query accuracy and execution through external tools. By introducing the mechanism of multi-intelligentsia collaboration, these systems are equipped with stronger flexibility and adaptive capability to decompose complex tasks, detect and repair errors, and optimize query conditions. This paper discusses several LLM Agent systems that dramatically improve the performance and reliability of Text-to-SQL tasks through stepwise reasoning, external supervision, and query optimization.

MAC-SQL [24] could be much more efficient in its work of query generation for SQL by involving multiple intelligences: the fundamental Decomposer intelligence, the Selector intelligence, and the Refiner intelligence. The Decomposer Intelligence, when given a complex SQL query to resolve, breaks it into fragment sub-problems sequentially through chained reasoning and generates the final SQL query in a stepwise manner. The usage of Selector Intelligence limits the database to eliminate the problem-specific data interference, while Fixer Intelligence repairs mistaken SQL via SQL execution using outside tools. The MSCFT is the method fine-tuning strategy used during SQL-Llama development, and this model is an open-source version based on the Code Llama. This model is trained using the dataset of instructions generated from the MAC-SQL intelligence tasks.

Tool-SQL [25] proposes a helper-agent framework that equips LLM-based agents with two specialized tools, a retriever and a detector. They are responsible for diagnosing and correcting SQL queries that suffer from database mismatch problems. The Retriever helps LLM-based agents verify the correctness of SQL conditional clauses by aligning the values in the SQL query to the corresponding cells in the database.

SQLFixAgent [139] adopts a multi-agent collaborative approach and consists of three main agents, first SQLRefiner is responsible for generating the core agent for the final repaired SQL query. Secondly, SQLReviewer detects syntactic and semantic errors in SQL queries. Generate multiple candidate SQL statements using a fine-tuned SQLTool.

MAG-SQL [140] consists of four parts. First, it performs a soft-column selection of the database schema, the purpose of which is to filter out redundant information. Then in the next module, it breaks down the problem into a series of smaller problems, feeds into the Iterative Generation module, which iteratively generates sub-SQL for each problem, and then a refiner executes these queries using external tools to optimize any incorrect SQL queries. External supervision is used throughout the process to ensure that the generated queries are consistent with the expected queries

MAGIC [141] proposes a novel agent approach to automatically create self-correcting guides by iterating over incorrect queries in the text-to-SQL process. By providing automated, data-driven to mimic the correction process of human experts.

Distyl AI's Analytics Insight Engine [142] generates complex queries in arbitrary environments and is able to improve the generated results through user feedback and dynamics. It also performs external knowledge retrieval, where the user's question is paraphrased into an authoritative form that captures the user's intent and, based on that intent, proceeds to retrieve relevant examples.

SuperSQL [143] uses a number of approaches to different architectures. In the preprocessing phase, it uses architectural links in RESDSQL and database content in BRIDGE v2 as a way to enhance connectivity to the overall database, uses DAIL-SQL's few-shot prompt engineering module to select contextual examples based on similarity and self-consistency to ensure the reliability of the generated content, and then generates SQL queries using the SQL query generation using greedy decoding strategy

V. Conclusion

In this paper, we thoroughly review and analyze the implications of large-scale language models (LLMs) in the text-to-SQL framework, looking for further research from different perspectives. To start with, the contribution of this paper lies in classifying the application of LLMs on the task of text-to-SQL into two groups: traditional approaches and approaches to hint engineering on the one hand, and fine-tuned approaches on the other. We will develop in-depth important aspects, such as the general structure of hints, ways of supplementing knowledge, example choosing, and reasoning.

We are looking forward to future research being able to expand the methods that allow the generation to be smarter and more cost-effective and generation smarter and more cost-effective SQL query generation that shows stronger generalization ability in cross-domain and cross-language application scenarios. Continuous optimization and innovation in the Text-to-SQL field imply that more efficient, high-performance decision-making and data information retrieval procedures via formal language will be realized, marked by further unprecedented developments in the relevant fields.

REFERENCES

[1] X. Xu, "Research prospect: data factor of production," *Journal of Internet and Digital Economics*, vol. 1, no. 1, pp. 64–71, 2021.

[2] M.-R. Zhong, J.-Y. Fu, and H. Zou, "The data as a production factor: nonlinear effects of factor efficiency on haze pollution," *Environment, Development and Sustainability*, 2023. [Online]. Available: https://doi.org/10.1007/s10668-023-04264-z

[3] Y. Carriere-Swallow and V. Haksar, "The economics and implications of data: An integrated perspective," *Departmental Papers*, vol. 2019, no. 013, p. A001, 2019. [Online]. Available: https://www.elibrary.imf.org/view/journals/087/2019/013/article-A001-en.xml

[4] P. Qi, D. Sun, C. Xu, Q. Li, and Q. Wang, "Can data elements promote the high-quality development of china's economy?" *Sustainability*, vol. 15, no. 9, 2023. [Online]. Available: https://www.mdpi.com/2071-1050/15/9/7287

[5] Q. Liu, B. Chen, J. Guo, J.-G. Lou, B. Zhou, and D. Zhang, "How far are we from effective context modeling? an exploratory study on semantic parsing in context," 2020. [Online]. Available: https://arxiv.org/abs/2002.00652

[6] W. Woods, R. Kaplan, and B. Webber, "The lunar sciences natural language information system," Jul 1972.

[7] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," *Proceedings of the VLDB Endowment*, vol. 8, no. 1, pp. 73–84, 2014.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[10] Z. Hong, Z. Yuan, Q. Zhang, H. Chen, J. Dong, F. Huang, and X. Huang, "Next-generation database interfaces: A survey of llm-based text-to-sql," 2024. [Online]. Available: https://arxiv.org/abs/2406.08426

[11] OpenAI, "Gpt-4 technical report," *ArXiv*, vol. abs/2303.08774, 2023. [Online]. Available: https://arxiv.org/abs/2303.08774

[12] 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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3911–3921. [Online]. Available: https://aclanthology.org/D18-1425

[13] N. Rajkumar, R. Li, and D. Bahdanau, "Evaluating the text-to-sql capabilities of large language models," *ArXiv*, vol. abs/2204.00498, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247922681

[14] A. Liu, X. Hu, L. Wen, and P. S. Yu, "A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability," 2023. [Online]. Available: https://arxiv.org/abs/2303.13547

[15] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. 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. M. 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," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[16] D. Lee, C. Park, J. Kim, and H. Park, "Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation," 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2405.07467

[17] Z. Hong, Z. Yuan, H. Chen, Q. Zhang, F. Huang, and X. Huang, "Knowledge-to-SQL: Enhancing SQL Generation with Data Expert LLM," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.11517

[18] Q. Zhang, J. Dong, H. Chen, W. Li, F. Huang, and X. Huang, "Structure guided large language model for sql generation," mar 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.13284

[19] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," 2022. [Online]. Available: https://arxiv.org/abs/2007.08199

[20] 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," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.15363

[21] 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 *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.16347

[22] Y. Fu, W. Ou, Z. Yu, and Y. Lin, "MIGA: A Unified Multi-task Generation Framework for Conversational Text-to-SQL," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2212.09278

[23] M. Pourreza, R. Sun, H. Li, L. Miculicich, T. Pfister, and S. O. Arik, "Sql-gen: Bridging the dialect gap for text-to-sql via synthetic data and model merging," 2024. [Online]. Available: https://arxiv.org/abs/2408.12733

[24] 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," 2024. [Online]. Available: https://arxiv.org/abs/2312.11242

[25] Z. Wang, R. Zhang, Z. Nie, and J. Kim, "Tool-assisted agent on sql inspection and refinement in real-world scenarios," *arXiv preprint arXiv:2408.16991*, 2024.

[26] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," 2017. [Online]. Available: https://arxiv.org/abs/1704.04368

[27] G. Katsogiannis-Meimarakis and G. Koutrika, "A survey on deep learning approaches for text-to-sql," *The VLDB Journal*, vol. 32, no. 4, p. 905–936, jan 2023. [Online]. Available: https://doi.org/10.1007/s00778-022-00776-8

[28] Z. Hong and J. Liu, "Towards better question generation in qa-based event extraction," 2024. [Online]. Available: https://arxiv.org/abs/2405.10517

[29] C. Zheng, L. Li, Q. Dong, Y. Fan, Z. Wu, J. Xu, and B. Chang, "Can we edit factual knowledge by in-context learning?" 2023. [Online]. Available: https://arxiv.org/abs/2305.12740

[30] C. Li, Y. Wang, Z. Wu, Z. Yu, F. Zhao, S. Huang, and X. Dai, "MultiSQL: A schema-integrated context-dependent Text2SQL dataset with diverse SQL operations," in *Findings of the Association for Computational Linguistics ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds. Bangkok, Thailand and virtual meeting: Association for Computational Linguistics, Aug. 2024, pp. 13 857–13 867. [Online]. Available: https://aclanthology.org/2024.findings-acl.823

[31] J. Li, Z. Chen, L. Chen, Z. Zhu, H. Li, R. Cao, and K. Yu, "Dir: A large-scale dialogue rewrite dataset for cross-domain conversational text-to-sql," *Applied Sciences*, vol. 13, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/4/2262

[32] P. J. Price, "Evaluation of spoken language systems: the ATIS domain," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*, 1990. [Online]. Available: https://aclanthology.org/H90-1020

[33] J. M. Zelle and R. J. Mooney, "Learning to parse database queries using inductive logic programming," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, ser. AAAI'96. AAAI Press, 1996, p. 1050–1055.

[34] S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer, "Learning a neural semantic parser from user feedback," 2017. [Online]. Available: https://arxiv.org/abs/1704.08760

[35] C. Finegan-Dollak, J. K. Kummerfeld, L. Zhang, K. Ramanathan, S. Sadasivam, R. Zhang, and D. Radev, "Improving text-to-SQL evaluation methodology," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao, Eds. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 351–360. [Online]. Available: https://aclanthology.org/P18-1033

[36] G. Lee, H. Hwang, S. Bae, Y. Kwon, W. Shin, S. Yang, M. Seo, J.-Y. Kim, and E. Choi, "Ehrsql: A practical text-to-sql benchmark for electronic health records," 2023. [Online]. Available: https://arxiv.org/abs/2301.07695

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

[38] 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," 2021.

[39] Y. Gan, X. Chen, and M. Purver, "Exploring underexplored limitations of cross-domain text-to-sql generalization," 2021.

[40] Y. Gan, X. Chen, Q. Huang, and M. Purver, "Measuring and improving compositional generalization in text-to-sql via component alignment," 2022.

[41] R. Patil, M. Patwardhan, S. Karande, L. Vig, and G. Shroff, "Exploring dimensions of generalizability and few-shot transfer for text-to-sql semantic parsing," in *Proceedings of The 1st Transfer Learning for Natural Language Processing Workshop*, ser. PMLR, vol. 203, 2022, pp. 103–114.

[42] 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," 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2403.02951

[43] P. Shaw, M.-W. Chang, P. Pasupat, and K. Toutanova, "Compositional generalization and natural language variation: Can a semantic parsing approach handle both?" in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 922–938. [Online]. Available: https://aclanthology.org/2021.acl-long.75

[44] C.-H. Lee, O. Polozov, and M. Richardson, "KaggleDBQA: Realistic evaluation of text-to-SQL parsers," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 2261–2273. [Online]. Available: https://aclanthology.org/2021.acl-long.176

[45] N. Wretblad and F. G. Riseby, "Bridging language & data: Optimizing text-to-sql generation in large language models," Ph.D. dissertation, Linköping University, 2024, dissertation. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-200605

[46] 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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6923–6935. [Online]. Available: https://aclanthology.org/2020.emnlp-main.562

[47] P. B. Chen, F. Wenz, Y. Zhang, M. Kayali, N. Tatbul, M. Cafarella, Çağatay Demiralp, and M. Stonebraker, "Beaver: An enterprise benchmark for text-to-sql," 2024. [Online]. Available: https://arxiv.org/abs/2409.02038

[48] 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 *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1962–1979. [Online]. Available: https://aclanthology.org/D19-1204

[49] 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 *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 2007–2022. [Online]. Available: https://aclanthology.org/2022.acl-long.142

[50] Q. Min, Y. Shi, and Y. Zhang, "A pilot study for Chinese SQL semantic parsing," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3652–3658. [Online]. Available: https://aclanthology.org/D19-1377

[51] G. Lee, W. Chay, S. Cho, and E. Choi, "TrustSQL: Benchmarking Text-to-SQL Reliability with Penalty-Based Scoring," jun 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2403.15879

[52] T. Yu, R. Zhang, M. Yasunaga, Y. C. Tan, X. V. Lin, S. Li, I. L. Heyang Er, B. Pang, T. Chen, E. Ji, S. Dixit, D. Proctor, S. Shim, V. Z. Jonathan Kraft, C. Xiong, R. Socher, and D. Radev, "Sparc: Cross-domain semantic parsing in context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019.

[53] J. Guo, Z. Si, Y. Wang, Q. Liu, M. Fan, J.-G. Lou, Z. Yang, and T. Liu, "Chase: A large-scale and pragmatic Chinese dataset for cross-database context-dependent text-to-SQL," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 2316–2331. [Online]. Available: https://aclanthology.org/2021.acl-long.180

[54] 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 EMNLP*, 2020.

[55] M. Poess and C. Floyd, "New tpc benchmarks for decision support and web commerce," *SIGMOD Rec.*, vol. 29, no. 4, p. 64–71, Dec. 2000. [Online]. Available: https://doi.org/10.1145/369275.369291

[56] L. Ma, K. Pu, and Y. Zhu, "Evaluating llms for text-to-sql generation with complex sql workload," *arXiv preprint arXiv:2407.19517*, 2024.

[57] V. Zhong, C. Xiong, and R. Socher, "Seq2SQL: Generating structured queries from natural language using reinforcement learning," 2018. [Online]. Available: https://openreview.net/forum?id=Syx6bz-Ab

[58] Y. Gan, X. Chen, and M. Purver, "Exploring underexplored limitations of cross-domain text-to-sql generalization," 2021. [Online]. Available: https://arxiv.org/abs/2109.05157

[59] Y. Gan, X. Chen, Q. Huang, and M. Purver, "Measuring and improving compositional generalization in text-to-sql via component alignment," 2022. [Online]. Available: https://arxiv.org/abs/2205.02054

[60] 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," 2021. [Online]. Available: https://arxiv.org/abs/2106.01065

[61] X. Deng, A. H. Awadallah, C. Meek, O. Polozov, H. Sun, and M. Richardson, "Structure-grounded pretraining for text-to-sql," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021. [Online]. Available: http://dx.doi.org/10.18653/v1/2021.naacl-main.105

[62] A. Kumar, P. Nagarkar, P. Nalhe, and S. Vijayakumar, "Deep learning driven natural languages text to sql query conversion: A survey," 2022. [Online]. Available: https://arxiv.org/abs/2208.04415

[63] G. Katsogiannis-Meimarakis and G. Koutrika, "A deep dive into deep learning approaches for text-to-sql systems," in *Proceedings of the 2021 International Conference on Management of Data*, ser. SIGMOD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2846–2851. [Online]. Available: https://doi.org/10.1145/3448016.3457543

[64] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[65] 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," 2023. [Online]. Available: https://arxiv.org/abs/1910.10683

[66] T. Yu, Z. Li, Z. Zhang, R. Zhang, and D. Radev, "Typesql: Knowledge-based type-aware neural text-to-sql generation," 2018. [Online]. Available: https://arxiv.org/abs/1804.09769

[67] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," 2017. [Online]. Available: https://arxiv.org/abs/1709.00103

[68] X. Xu, C. Liu, and D. Song, "Sqlnet: Generating structured queries from natural language without reinforcement learning," 2017. [Online]. Available: https://arxiv.org/abs/1711.04436

[69] T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, and D. Radev, "Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task," 2018. [Online]. Available: https://arxiv.org/abs/1810.05237

[70] 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," 2019. [Online]. Available: https://arxiv.org/abs/1905.08205

[71] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers," 2021. [Online]. Available: https://arxiv.org/abs/1911.04942

[72] T. Yu, C.-S. Wu, X. V. Lin, B. Wang, Y. C. Tan, X. Yang, D. Radev, R. Socher, and C. Xiong, "Grappa: Grammar-augmented

pre-training for table semantic parsing," 2021. [Online]. Available: https://arxiv.org/abs/2009.13845

[73] P. Shi, P. Ng, Z. Wang, H. Zhu, A. H. Li, J. Wang, C. N. dos Santos, and B. Xiang, "Learning contextual representations for semantic parsing with generation-augmented pre-training," 2020. [Online]. Available: https://arxiv.org/abs/2012.10309

[74] T. Yu, R. Zhang, A. Polozov, C. Meek, and A. Awadallah, "Score: Pre-training for context representation in conversational semantic parsing," in *ICLR*, May 2021.

[75] P. Yin, G. Neubig, W. tau Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," 2020. [Online]. Available: https://arxiv.org/abs/2005.08314

[76] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. [Online]. Available: http://dx.doi.org/10.18653/v1/2020.acl-main.398

[77] J. M. Eisenschlos, M. Gor, T. Müller, and W. W. Cohen, "Mate: Multi-view attention for table transformer efficiency," 2021. [Online]. Available: https://arxiv.org/abs/2109.04312

[78] J. Yang, A. Gupta, S. Upadhyay, L. He, R. Goel, and S. Paul, "Tableformer: Robust transformer modeling for table-text encoding," 2022. [Online]. Available: https://arxiv.org/abs/2203.00274

[79] Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, and J.-G. Lou, "Tapex: Table pre-training via learning a neural sql executor," 2022. [Online]. Available: https://arxiv.org/abs/2107.07653

[80] B. Hui, R. Geng, L. Wang, B. Qin, B. Li, J. Sun, and Y. Li, "S^2sql: Injecting syntax to question-schema interaction graph encoder for text-to-sql parsers," 2022. [Online]. Available: https://arxiv.org/abs/2203.06958

[81] R.-Z. Wang, Z.-H. Ling, J.-B. Zhou, and Y. Hu, "Tracking interaction states for multi-turn text-to-sql semantic parsing," 2020. [Online]. Available: https://arxiv.org/abs/2012.04995

[82] Y. Cai and X. Wan, "IGSQL: Database schema interaction graph based neural model for context-dependent text-to-SQL generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6903–6912. [Online]. Available: https://aclanthology.org/2020.emnlp-main.560

[83] V. Zhong, M. Lewis, S. I. Wang, and L. Zettlemoyer, "Grounded adaptation for zero-shot executable semantic parsing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6869–6882. [Online]. Available: https://aclanthology.org/2020.emnlp-main.558

[84] R. Zhang, T. Yu, H. Y. Er, S. Shim, E. Xue, X. V. Lin, T. Shi, C. Xiong, R. Socher, and D. Radev, "Editing-based sql query generation for cross-domain context-dependent questions," 2019. [Online]. Available: https://arxiv.org/abs/1909.00786

[85] L. Nan, Y. Zhao, W. Zou, N. Ri, J. Tae, E. Zhang, A. Cohan, and D. R. Radev, "Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies," vol. abs/2305.12586, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258833511

[86] X. V. Lin, R. Socher, and C. Xiong, "Bridging textual and tabular data for cross-domain text-to-sql semantic parsing," 2020. [Online]. Available: https://arxiv.org/abs/2012.12627

[87] B. Hui, X. Shi, R. Geng, B. Li, Y. Li, J. Sun, and X. Zhu, "Improving text-to-sql with schema dependency learning," 2021. [Online]. Available: https://arxiv.org/abs/2103.04399

[88] 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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6943–6954. [Online]. Available: https://aclanthology.org/2020.emnlp-main.564

[89] J. Ma, Z. Yan, S. Pang, Y. Zhang, and J. Shen, "Mention extraction and linking for SQL query generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6936–6942. [Online]. Available: https://aclanthology.org/2020.emnlp-main.563

[90] K. Xu, Y. Wang, Y. Wang, Z. Wen, and Y. Dong, "Sead: End-to-end text-to-sql generation with schema-aware denoising," 2023. [Online]. Available: https://arxiv.org/abs/2105.07911

[91] O. Rubin and J. Berant, "SmBoP: Semi-autoregressive bottom-up semantic parsing," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds. Online: Association for Computational Linguistics, Jun. 2021, pp. 311–324. [Online]. Available: https://aclanthology.org/2021.naacl-main.29

[92] P. Xu, D. Kumar, W. Yang, W. Zi, K. Tang, C. Huang, J. C. K. Cheung, S. J. D. Prince, and Y. Cao, "Optimizing deeper transformers on small datasets," 2021. [Online]. Available: https://arxiv.org/abs/2012.15355

[93] J. Huang, Y. Wang, Y. Wang, Y. Dong, and Y. Xiao, "Relation aware semi-autoregressive semantic parsing for nl2sql," 2021. [Online]. Available: https://arxiv.org/abs/2108.00804

[94] B. Bogin, M. Gardner, and J. Berant, "Representing schema structure with graph neural networks for text-to-sql parsing," 2019. [Online]. Available: https://arxiv.org/abs/1905.06241

[95] Z. Chen, L. Chen, Y. Zhao, R. Cao, Z. Xu, S. Zhu, and K. Yu, "Shadowgnn: Graph projection neural network for text-to-sql parser," 2021. [Online]. Available: https://arxiv.org/abs/2104.04689

[96] R. Cai, J. Yuan, B. Xu, and Z. Hao, "Sadga: Structure-aware dual graph aggregation network for text-to-sql," 2022. [Online]. Available: https://arxiv.org/abs/2111.00653

[97] R. Cao, L. Chen, Z. Chen, Y. Zhao, S. Zhu, and K. Yu, "LGESQL: Line graph enhanced text-to-SQL model with mixed local and non-local relations," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 2541–2555. [Online]. Available: https://aclanthology.org/2021.acl-long.198

[98] T. Xie, C. H. Wu, P. Shi, R. Zhong, T. Scholak, M. Yasunaga, C.-S. Wu, M. Zhong, P. Yin, S. I. Wang, V. Zhong, B. Wang, C. Li, C. Boyle, A. Ni, Z. Yao, D. Radev, C. Xiong, L. Kong, R. Zhang, N. A. Smith, L. Zettlemoyer, and T. Yu, "Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models," 2022. [Online]. Available: https://arxiv.org/abs/2201.05966

[99] W. Huang, X. Zheng, X. Ma, H. Qin, C. Lv, H. Chen, J. Luo, X. Qi, X. Liu, and M. Magno, "An empirical study of llama3 quantization: From llms to mllms," 2024. [Online]. Available: https://arxiv.org/abs/2404.14047

[100] F. Huang, Z. Yang, J. Jiang, Y. Bei, Y. Zhang, and H. Chen, "Large language model interaction simulator for cold-start item recommendation," 2024. [Online]. Available: https://arxiv.org/abs/2402.09176

[101] 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," 2024. [Online]. Available: https://arxiv.org/abs/2312.17449

[102] 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," 2024. [Online]. Available: https://arxiv.org/abs/2311.09278

[103] 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," 2023. [Online]. Available: https://arxiv.org/abs/2305.11853

[104] 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," 2023. [Online]. Available: https://arxiv.org/abs/2310.17342

[105] C. Hu, J. Fu, C. Du, S. Luo, J. Zhao, and H. Zhao, "Chatdb: Augmenting llms with databases as their symbolic memory," 2023. [Online]. Available: https://arxiv.org/abs/2306.03901

[106] X. Liu and Z. Tan, "Divide and prompt: Chain of thought prompting for text-to-sql," 2023. [Online]. Available: https://arxiv.org/abs/2304.11556

[107] X. Dong, C. Zhang, Y. Ge, Y. Mao, Y. Gao, lu Chen, J. Lin, and D. Lou, "C3: Zero-shot text-to-sql with chatgpt," 2023. [Online]. Available: https://arxiv.org/abs/2307.07306

[108] 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," 2024. [Online]. Available: https://arxiv.org/abs/2308.12060

[109] Z. Gu, J. Fan, N. Tang, L. Cao, B. Jia, S. Madden, and X. Du, "Few-shot text-to-sql translation using structure and content prompt

learning," *Proc. ACM Manag. Data*, vol. 1, no. 2, jun 2023. [Online]. Available: https://doi.org/10.1145/3589292

[110] L. Nan, Y. Zhao, W. Zou, N. Ri, J. Tae, E. Zhang, A. Cohan, and D. Radev, "Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies," 2023. [Online]. Available: https://arxiv.org/abs/2305.12586

[111] R. Sun, S. Ö. Arik, A. Muzio, L. Miculicich, S. Gundabathula, P. Yin, H. Dai, H. Nakhost, R. Sinha, Z. Wang, and T. Pfister, "Sql-palm: Improved large language model adaptation for text-to-sql (extended)," 2024. [Online]. Available: https://arxiv.org/abs/2306.00739

[112] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2024.

[113] T. Wang, H. Lin, X. Han, L. Sun, X. Chen, H. Wang, and Z. Zeng, "Dbcopilot: Scaling natural language querying to massive databases," 2024. [Online]. Available: https://arxiv.org/abs/2312.03463

[114] J.-P. Zhu, P. Cai, B. Niu, Z. Ni, K. Xu, J. Huang, J. Wan, S. Ma, B. Wang, D. Zhang, L. Tang, and Q. Liu, "Chat2query: A zero-shot automatic exploratory data analysis system with large language models," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024, pp. 5429–5432.

[115] M. Pourreza and D. Rafiei, "DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2304.11015

[116] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating large language model decoding with speculative sampling," 2023. [Online]. Available: https://arxiv.org/abs/2302.01318

[117] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," 2023. [Online]. Available: https://arxiv.org/abs/2211.17192

[118] N. Deng, Y. Chen, and Y. Zhang, "Recent advances in text-to-SQL: A survey of what we have and what we expect," in *Proceedings of the 29th International Conference on Computational Linguistics*, N. Calzolari, C.-R. Huang, H. Kim, J. Pustejovsky, L. Wanner, K.-S. Choi, P.-M. Ryu, H.-H. Chen, L. Donatelli, H. Ji, S. Kurohashi, P. Paggio, N. Xue, S. Kim, Y. Hahm, Z. He, T. K. Lee, E. Santus, F. Bond, and S.-H. Na, Eds. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, Oct. 2022, pp. 2166–2187. [Online]. Available: https://aclanthology.org/2022.coling-1.190

[119] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[120] W. Hwang, J. Yim, S. Park, and M. Seo, "A comprehensive exploration on wikisql with table-aware word contextualization," 2019. [Online]. Available: https://arxiv.org/abs/1902.01069

[121] P. He, Y. Mao, K. Chakrabarti, and W. Chen, "X-sql: reinforce schema representation with context," 2019. [Online]. Available: https://arxiv.org/abs/1908.08113

[122] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "TaBERT: Pretraining for joint understanding of textual and tabular data," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 8413–8426. [Online]. Available: https://aclanthology.org/2020.acl-main.745

[123] T. Yu, C.-S. Wu, X. V. Lin, bailin wang, Y. C. Tan, X. Yang, D. Radev, richard socher, and C. Xiong, "Gra{pp}a: Grammar-augmented pre-training for table semantic parsing," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=kyaIeYj4zZ

[124] S. Kou, L. Hu, Z. He, Z. Deng, and H. Zhang, "Cllms: Consistency large language models," 2024. [Online]. Available: https://arxiv.org/abs/2403.00835

[125] 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," 2024. [Online]. Available: https://arxiv.org/abs/2402.16671

[126] R. Roberson, G. Kaki, and A. Trivedi, "Analyzing the effectiveness of large language models on text-to-sql synthesis," jan 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.12379

[127] F. Zhou, S. Xue, D. Qi, W. Shi, W. Zhao, G. Wei, H. Zhang, C. Jiang, G. Jiang, Z. Chu, and F. Chen, "Db-gpt-hub: Towards open benchmarking text-to-sql empowered by large language models," 2024. [Online]. Available: https://arxiv.org/abs/2406.11434

[128] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[129] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023. [Online]. Available: https://arxiv.org/abs/2305.14314

[130] J. Jang, S. Kim, S. Ye, D. Kim, L. Logeswaran, M. Lee, K. Lee, and M. Seo, "Exploring the benefits of training expert language models over instruction tuning," 2023. [Online]. Available: https://arxiv.org/abs/2302.03202

[131] K. Meng, A. S. Sharma, A. Andonian, Y. Belinkov, and D. Bau, "Mass-editing memory in a transformer," 2023. [Online]. Available: https://arxiv.org/abs/2210.07229

[132] H. Li, J. Zhang, C. Li, and H. Chen, "RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, oral presentation. [Online]. Available: https://doi.org/10.48550/arXiv.2302.05965

[133] 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," 2023. [Online]. Available: https://arxiv.org/abs/2305.17378

[134] J. Fürst, C. Kosten, F. Nooralahzadeh, Y. Zhang, and K. Stockinger, "Evaluating the data model robustness of text-to-sql systems based on real user queries," 2024. [Online]. Available: https://arxiv.org/abs/2402.08349

[135] T. Scholak, N. Schucher, and D. Bahdanau, "Picard: Parsing incrementally for constrained auto-regressive decoding from language models," 2021. [Online]. Available: https://arxiv.org/abs/2109.05093

[136] S. H. K. Parthasarathi, L. Zeng, and D. Hakkani-Tur, "Conversational text-to-sql: An odyssey into state-of-the-art and challenges ahead," 2023. [Online]. Available: https://arxiv.org/abs/2302.11054

[137] 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," 2023. [Online]. Available: https://arxiv.org/abs/2304.13301

[138] M. Saeed, N. D. Cao, and P. Papotti, "Querying large language models with sql," 2023. [Online]. Available: https://arxiv.org/abs/2304.00472

[139] J. Cen, J. Liu, Z. Li, and J. Wang, "Sqlfixagent: Towards semantic-accurate sql generation via multi-agent collaboration," *arXiv preprint arXiv:2406.13408*, 2024.

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

[141] A. Askari, C. Poelitz, and X. Tang, "Magic: Generating self-correction guideline for in-context text-to-sql," *arXiv preprint arXiv:2406.12692*, 2024.

[142] K. Maamari and A. Mhedhbi, "End-to-end text-to-sql generation within an analytics insight engine," *arXiv preprint arXiv:2406.12104*, 2024.

[143] B. Li, Y. Luo, C. Chai, G. Li, and N. Tang, "The dawn of natural language to sql: Are we fully ready?" *arXiv preprint arXiv:2406.01265*, 2024.