https://doi.org/10.1007/s11704-025-50592-w

REVIEW ARTICLE

A systematic review of natural language interfaces for databases

Mengyi LIU[1*], Xieyang WANG[1*], Jianqiu XU[1,2]✉, Weijia YI[1], Ouri WOLFSON[3]

1. College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
2. Shanghai Key Laboratory of Data Science, Shanghai 200438, China
3. Department of Computer Science, University of Illinois at Chicago, Chicago IL 60607, USA

Received May 9, 2025; accepted September 17, 2025

E-mail: jianqiu@nuaa.edu.cn. * These authors contributed equally to this work.

Special Issue—Data Governance and Circulation for Data Factors

Abstract

With the growing prevalence of data-driven decision-making, Text-to-SQL has emerged as a promising solution to lower the barrier to data access by translating natural language queries into executable SQL statements, thereby enhancing user interaction with databases. Despite notable progress driven by deep learning and large language models, significant challenges persist in handling complex queries. This paper presents a comprehensive review of the Text-to-SQL task, structured around two core stages: natural language understanding and natural language translation. Methods are categorized along the technical evolution trajectory into four types: rule-based, machine learning-based, pre-trained language model-based, and large language model-based approaches. Unlike previous surveys, which focus on specific techniques or partial aspects of Text-to-SQL, our work offers a two-stage analytical framework, highlights the impact of large models, and provides a comparative analysis of limitations and trade-offs. Through detailed examination of accuracy, generalization, expressiveness, and computational cost, this survey presents insights into the advantages and disadvantages of each paradigm. Furthermore, the paper summarizes key benchmark datasets and evaluation metrics, and discusses directions to improve the robustness, security, and effectiveness of the existing Text-to-SQL systems.

Keywords

natural language interface for database; text-to-SQL; semantic parsing; structured language; query processing

1 Introduction

As data-driven decision-making becomes increasingly prevalent, the way users interact with data is undergoing a significant transformation. Traditional SQL querying requires users to possess specialized knowledge of database (DB) systems. In contrast, Text-to-SQL technology, which translates natural language (NL) into executable SQL queries [1], lowers the barrier to data access by allowing non-experts to retrieve structured information using natural language. The paradigm shift not only enhances human-computer interaction but also enables natural language-based analysis in business intelligence systems, structured search in web engines, and intelligent querying in customer support. As a result, Text-to-SQL has become a foundational component in intelligent question answering systems, low-code development tools and enterprise data services [2], and is expected to facilitate data interaction in emerging complex domains [3,4].

Text-to-SQL lies at the core of natural language interfaces for databases (NLIDB), with the goal of automatically converting user-issued natural language queries (NLQs) into executable SQL statements and eliminating the cost of manual processing [5], as illustrated in Fig. 1. While recent advances in deep learning, pre-trained language models (PLMs), and large language models (LLMs)

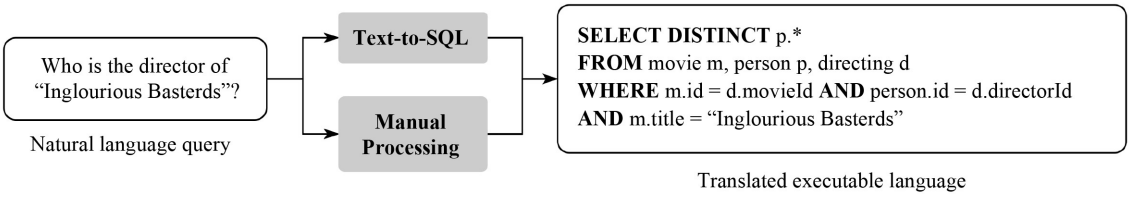


Fig. 1 Example of translating a natural language into SQL

have significantly boosted performance on standard benchmarks such as Spider [6] and Bird [7], challenges remain when handling complex queries involving multi-table joins, nested logic, and inference. The difficulties arise from two main sources: the inherent ambiguity and variability of natural language, including contextual ellipsis and vague semantics, and the strict syntactic and logical constraints of SQL, which demand precise semantic alignment [8].

Due to the semantic gap between NLQ and SQL, Text-to-SQL systems struggle to directly transform NLQ into SQL, typically requiring the decoupling of semantic interpretation and executable generation [9]. To provide a modular and clear modeling interface, this study unifies Text-to-SQL systems into a two-stage framework composed of natural language understanding (NLU) and natural language translation (NLT), combined with a technology-driven taxonomy, as shown in Fig. 2.

The NLU stage aims to extract database-relevant semantic elements such as tables, columns, predicates, aggregations, and sort orders from user input, and to build structured intermediate representations. For example, given the question "*What is the average salary of employees in the sales department*?", the NLU module identifies the target table as *employees*, the relevant column as *salary*, the predicate as *department = 'sales'*, and the aggregation function as AVG. The NLT stage then translates the representations into executable SQL queries conforming to syntactic rules and logical constraints (e.g., SELECT AVG(salary) FROM employees WHERE department = 'sales'). The division reflects the essence of Text-to-SQL, which is the transformation of semantic intent into structured programs, and also highlights key challenges, including semantic parsing, structure modeling, and execution effectiveness.

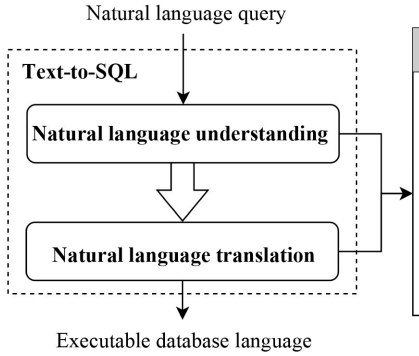
At each stage, Text-to-SQL methods can be categorized into four major paradigms: rule-based methods, machine learning-based methods, PLM-based methods, and LLM-based methods. Each paradigm entails distinct design philosophies and trade-offs.

(i) Rule-based methods rely on manually crafted parsers, templates, or semantic mapping rules. The methods prioritize syntactic accuracy and controllability, and are suitable for closed domains or well-structured queries [30]. For example, a predefined template might transform "*List all orders from 2021*" into a fixed SQL skeleton. Despite offering high precision and interpretability, rule-based methods exhibit limited generalization capabilities and incur significant maintenance costs. Typical techniques include parse tree construction, ontology matching, semantic graph building, syntax-driven generation, and template filling.

(ii) Machine learning-based methods formulate Text-to-SQL as a classification, sequence labeling, or ranking problem (e.g., slot filling, SQL snippet classification, and column prediction) [31]. Early approaches used models like support vector machines (SVMs) and conditional random fields (CRFs), later evolving into neural models such as RNNs and CNNs [32]. The methods offer greater flexibility in handling linguistic variations and more automation than rule-based methods. However, machine learning-based methods typically require extensive annotated data and struggle to model SQL structural constraints. For example, due to the lack of structure awareness, LSTM-based models [33] often produce syntactically invalid SQL like missing WHERE clauses or malformed joins, especially in multi-table queries.

(iii) PLM-based methods, such as models built on BERT [34], RoBERTa [35], and T5 [36], have dramatically advanced Text-to-SQL by leveraging contextual semantics. For instance, "*total income*" can be linked to the "*revenue*" column even if the exact word does not appear. At the NLU stage, structural awareness is introduced via table encoding, graph neural networks, column embeddings, and context modeling. At the NLT stage, the methods vary from template-based generation to end-to-end decoding and structure-constrained generation. Despite improvements in accuracy and generalization, PLM-based approaches require significant computational resources and face difficulties in precise structure alignment [37].

(iv) LLM-based approaches, powered by models like Codex [38] and GPT [39], represent the cutting edge of Text-to-SQL research [40]. The strong language understanding and generation capabilities allow few-shot or even zero-shot SQL synthesis through prompt learning and instruction tuning. For example, given the right prompt, an LLM can translate "*How many female employees are there in each department?*" into a correct group-by SQL without task-specific training. To overcome weaknesses in schema awareness and structure alignment, recent works incorporate techniques such as schema injection, chain-of-thought prompting, tool use, and knowledge augmentation [41,42]. Despite achieving impressive results, LLM-based methods face new challenges, including instability in output, limited semantic control, and high inference costs [43]. LLMs can participate in both NLU and NLT stages, either as the main



Methods	Techniques involved	Typical NLIDBs
Rule-based	parse tree, ontology, semantic graph, template matching, context-free grammar	NaLIR [10], ATHENA++ [11], MEANS [12], TR Discover [13], xDBTagger [14]
Machine learning-based	traditional machine learning, neural networks	CHILL [15], SpatialNLI [16], ValueNet [17], GENSQ [18], GAR [19]
Pre-trained language model-based	bert-based, graph-based models, template filling, pure generative methods	SQLOva [20], RAT-SQL [21], SmBoP [22], SQLNet [23], PICARD [24]
Large language model-based	prompt engineering, fine-tuning, integrated structured database knowledge	ACT-SQL [25], DIN-SQL [26], DEA-SQL [27], C3[28], SQLfuse [29]

Fig. 2 The two-stage framework and technical methods of Text-to-SQL

architecture or as an auxiliary module.

Overall, Text-to-SQL has evolved from controllable, rule-driven techniques toward flexible, language-centric paradigms. The shift entails a continuous trade-off between accuracy, interpretability, generalizability, resource efficiency, and domain transferability. Rule-based methods excel in precision but lack adaptability. While PLMs demonstrate enhanced generalization capabilities, domain-specific fine-tuning is still required to achieve optimal performance in vertical domains. LLMs provide powerful generalization but poor controllability.

Existing surveys have explored multiple aspects of Text-to-SQL, including datasets, methodologies, system architectures, and evaluation. Hong et al. [44] and Deng et al. [45] provided a comprehensive review of Text-to-SQL evolution, outlining key challenges such as ambiguity resolution, schema linking, and cross-domain adaptability. Katsogiannis-Meimarakis and Koutrika [31] and Iacob et al. [46] focused on deep learning-based approaches, emphasizing neural encoding-decoding mechanisms and schema-aware query generation. Additionally, Li et al. [47] and Qin et al. [48] introduced evaluation frameworks and dataset taxonomies, distinguishing between single-turn and multi-turn interactions. Zhang et al. [49] linked Text-to-SQL and visualization, highlighting the relationship within natural language interfaces. Meanwhile, Affolter et al. [50] categorized NLIDBs into four types, including keyword-based, pattern-based, parsing-based and grammar-based, and systematically evaluate 24 systems. However, the existing works tend to focus on a specific aspect (e.g., model architectures, datasets, or interaction types) or a particular technique (e.g., neural methods), and few provide a unified and up-to-date synthesis of the rapid paradigm shift brought by pre-trained and large language models. Moreover, most prior surveys lack a structured decomposition of Text-to-SQL into distinct stages, which is essential to reveal the specific challenges and technical pathways at each stage.

In contrast to the previous works, we present a systematic review of the Text-to-SQL domain spanning approximately two decades of research development. This survey introduces a two-stage framework of NLU and NLT, reflecting the fundamental modeling process of Text-to-SQL and enabling a fine-grained taxonomy across different generations of techniques. The methods are categorized into four paradigms following the technical evolution trajectory: rule-based, machine learning-based, PLM-based, and LLM-based approaches. The dual-perspective structure facilitates a comprehensive and layered analysis of modeling strategies and technical pathways. Special attention is paid to the trade-offs between performance improvements and limitations in interpretability, controllability, or cost. Moreover, we highlight the integration of LLMs at both NLU and NLT stages, examining mechanisms such as schema alignment, prompt engineering, and knowledge integration. In addition, we review representative benchmarks and evaluation metrics, providing a comprehensive view of the field.

This paper aims to offer a structured, insightful, and technically in-depth overview of the Text-to-SQL task, serving as a reference for researchers and practitioners. The rest of the paper is organized as follows. Section 2 and Section 3 analyze the two core stages of Text-to-SQL, focusing on natural language understanding and translation. Section 4 reviews popular benchmarks and evaluation metrics used in Text-to-SQL. Section 5 discusses enhancements to the existing Text-to-SQL systems in four main areas. Section 6 explores open research questions in Text-to-SQL and concludes the survey. Table 1 summarizes the frequently used notations.

2 Natural language understanding

At the NLU stage, semantic information is extracted from the NLQ to comprehend the user's intent. The input is an NLQ and the output is the semantic representation. NLU methods have evolved from rule-based systems to data-driven learning approaches, and more recently to large pre-trained and generative language models. The evolution reflects a shift from manually encoded linguistic knowledge to statistical modeling and contextual reasoning, driven by the need for better generalization and scalability.

2.1 Rule-based methods

Early NLU methods are rule-based, relying on predefined rules and linguistic structures to parse user queries, extract key semantic information, and generate structured semantic representations. The core principle of rule-based approaches is to map NLQs to formal intermediate representations through explicit language processing steps and rule-matching mechanisms.

Rule-based methods first leverage natural language processing (NLP) techniques, including part-of-speech tagging, lemmatization, named entity recognition (NER), and dependency parsing, to preprocess NLQs for subsequent semantic parsing. The NLP techniques employed in Text-to-SQL methods are shown in Table 2.

(i) Part of speech tagging refers to assigning the correct part of speech to each word in the segmented text, determining whether each word is a noun, verb, or adjective. In NLIDB, part of speech tagging facilitates the identification of the grammatical roles of individual words in NLQs, leading to an accurate comprehension of users' intent. Taking the NLQ "*All movies starring Brad Pitt from 2000 until 2010.*" as an example, the result of part of speech tagging using Stanford CoreNLP [68] is shown in Fig. 3(a). In the figure, DT =

Table 1 Frequently used notations

Name	Abbreviation
Natural language interface for database	NLIDB
Natural language query	NLQ
Natural language processing	NLP
Natural language understanding	NLU
Natural language translation	NLT
Named entity recognition	NER
Pre-trained language model	PLM
Large language model	LLM
First-order logic	FOL
sequence-to-sequence	seq2seq

Table 2 The NLP techniques employed in Text-to-SQL methods

NLIDB	Year	Segmentation	Part of speech	NER	Dictionary generation	Regular expression	Dependency parsing
PRECISE [51]	2003	√	√	√	√		
Querix [52]	2006	√	√		√		
QuestIO [53]	2008	√	√		√		
gAnswer [54]	2013				√		
MEANS [12]	2015	√		√			
NL2CM [55,56]	2015	√	√				√
NL2TRANQUYL [57]	2015						√
ATHENA [58]	2016	√	√	√			√
SQLizer [59]	2017	√	√	√			
TEQUILA [60]	2018	√	√	√			
MyNLIDB [61]	2019	√	√				√
NLMO [62]	2020	√	√	√	√	√	
NALMO [63,64]	2021	√	√	√	√	√	
NALSD [65]	2023	√	√	√	√		
NALSpatial [66,67]	2023	√	√	√	√		
xDBTagger [14]	2024	√	√				

determiner; NNS = plural noun; VBG = the gerund or present participle of a verb; NNP = singular proper noun; IN = preposition or subordinating conjunction; CD = cardinal number.

(ii) Lemmatization is the process of reducing the different forms of a word to the original form. In NLIDB, lemmatization is beneficial in unifying words of various tenses and morphs in NLQs into base forms to match the content in the database. Taking the NLQ "*All movies starring Brad Pitt from 2000 until 2010.*" as an example, the result of lemmatization using Stanford CoreNLP [68] is shown in Fig. 3(b).

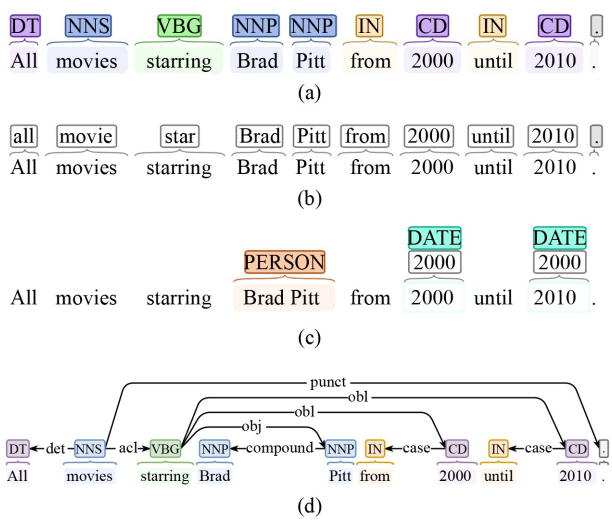


Fig. 3 Processing natural language using Stanford CoreNLP. (a) Part of speech tagging; (b) lemmatization; (c) named entity recognition; (d) dependency parsing

(iii) Named entity recognition is the procedure of identifying entities with specific meanings in natural language text [69]. Generally, the recognized entities can be categorized into three primary groups (entity, temporal, and numeric) and seven subgroups (PERSON, ORGANIZATION, LOCATION, TIME, DATE, MONEY, and PERCENT). NER in NLIDB enables the identification of entities involved in the NLQ to locate the topic and scope of the query. Taking the NLQ "*All movies starring Brad Pitt from 2000 until 2010.*" as an example, the result of NER using Stanford CoreNLP [68] is shown in Fig. 3(c).

(iv) Dependency parsing involves analyzing the dependencies between words in natural language sentences. A binary asymmetric relationship between words is called dependency, which is described as an arrow from the head (the subject to be modified) to the dependent (the modifier). Dependency parsing in NLIDB facilitates the understanding of grammatical relationships between words in NLQs, so that the structure and meaning of the query can be accurately understood. Taking the NLQ "*All movies starring Brad Pitt from 2000 until 2010.*" as an example, the result of dependency parsing using Stanford CoreNLP [68] is illustrated in Fig. 3(d). In the figure, punct = punctuation; obl = oblique nominal; obj = object; det = determiner; acl = clausal modifier of noun; case = case marking.

With the vigorous development of NLP technology, a number of NLP tools are appearing, including NLTK [70], spaCy [71], Stanford CoreNLP [68], and TextBlob [72]. The tools are capable of performing basic tasks and each has distinct advantages and

disadvantages.

After completing the basic natural language processing, rule-based methods use explicit parsing rules to analyze NLQs and construct semantic representations. Common rules include parse tree, ontology, semantic graph, template matching, pattern matching, context-free grammar, and semantic grammar, as shown in Table 3.

(i) The parse tree is based on syntactic analysis, which parses the NLQ into a tree structure where nodes represent phrases or words, and edges represent syntactic relationships. The parse tree can be used to identify the core structure of the query, including the query predicate, query target, and conditional constraints. Taking NaLIR [10] as an example, the Stanford Parser is first used to convert the NLQ into a dependency parse tree. The nodes of the parse tree are then mapped to SQL components in the database, such as relation names, attribute names, and operators. Simultaneously, the Wu-Palmer algorithm of WordNet is used to evaluate semantic similarity, and the Jaccard Coefficient is employed to assess spelling similarity, addressing ambiguity. Finally, predefined grammatical rules are used to validate the legality of the parse tree, and implicit nodes are inserted to complete any omitted logic. For ambiguous nodes and potential parse tree structure adjustments, NaLIR generates natural language explanations for the user to choose from. To improve conversion accuracy, NaLIR limits query expressiveness and increases conversion overhead. Specifically, NaLIR restricts the supported query types through grammatical rules, generates multiple candidate parse trees using Subtree Move operations and selects the optimal solution, and designs multiple user interactions, such as mapping selection and parse tree structure validation.

(ii) Ontology is a knowledge base that defines domain-specific concepts and the relationships. For instance, in an academic database, there exists an "*enrollment*" relationship between "*student*" and "*course*". Ontology-driven parsing methods typically involve entity mapping and relationship reasoning. Entity mapping aligns terms in the NLQ with corresponding ontology concepts, while relationship reasoning utilizes ontology to infer potential query paths. Taking ATHENA++ [11] as an example, the input NLQ is first tokenized and parsed, with each term mapped to relevant elements in the domain ontology (e.g., concepts, relations, and attributes), and the query's operation type (e.g., aggregation, comparison, and negation) is identified. Dependency analysis is then conducted using Stanford CoreNLP to determine the syntactic and semantic relationships between terms. The system assesses whether the NLQ requires transformation into a nested SQL query. ATHENA++ automatically detects nested structures within the query, categorizing them into four common nested query types: Type-N, Type-A, Type-J, and Type-JA. If the query is identified as nested, ATHENA++ generates join conditions and constructs an interpretation tree for each subquery, representing the query's logical structure. The interpretation trees are then combined through join conditions to form a complete ontology query language. To enhance conversion accuracy, ATHENA++ limits generality and increases conversion overhead. Specifically, ATHENA++ leverages domain ontology and semantic reasoning, ensuring high adaptability to complex queries in specific domains, although the response time for processing complex nested queries (e.g., Type-J and Type-JA) may increase.

(iii) A semantic graph represents an NLQ as a graph structure, with nodes denoting concepts or entities and edges indicating relationships. For instance, in the MEANS [12] system, the NLQ is transformed into a structured semantic representation (RDF triples) through the use of a semantic graph. Nodes represent medical entities (e.g., diseases, drugs, and symptoms), concepts, or literals. For the query "*What is the best treatment for oral thrush?*", "*oral thrush*", and "*treatment*" are mapped as nodes in the semantic graph. Edges represent semantic relationships between entities. For example, "*treatment treats oral thrush*" is represented as an edge from "*treatment*" to "*oral thrush*" labeled "*treats*." Empty nodes represent unknown answers in the query. The types of nodes and edges are defined using a custom medical question answering ontology (MESA). MESA includes six types of medical entities (e.g., Problem and Treatment) and seven types of semantic relations (e.g., "*treats*" and "*diagnoses*"). For example, the entity "*oral thrush*" is labeled as *mesa:Problem*, and the relation "*treats*" is labeled as *mesa:treats*. Medical entities are identified using a hybrid approach combining rules and CRF models, and are mapped to MESA categories. Entity relationships are extracted using pattern matching and an SVM classifier. To improve conversion accuracy, MEANS limits query expressiveness. Specifically, the MESA ontology constrains the semantic scope to avoid the generation of invalid queries. However, only predefined entity and relation types are supported, and relations like "*affects*" are not covered. Furthermore, relying on external

Table 3 Rules for parsing natural language queries

Rules	Typical NLIDBs
Parse tree	PRECISE [51], NaLIX [73,74], Querix [52], DaNaLIX [75], gAnswer [54], NaLIR [10,76,77], NL2TRANQUYL [57], Unnamed method [78], MyNLIDB [61]
Ontology	QuestIO [53], ATHENA [58], FINESSE [79], Unnamed method [80], CNL-RDF-Query [81], ATHENA++ [11], Unnamed method [82]
Semantic graph	Unnamed method [83], MEANS [12], NL2CM [55,56]
Template matching	SQLizer [59], Unnamed method [84], LogicalBeam [85]
Pattern matching	SODA [86]
Context-free grammar	TR Discover [13]
Semantic grammar	Unnamed method [87]

knowledge bases may lead to failures in mapping specialized terminology.

Rule-based NLU methods rely on explicit linguistic structures, resulting in traceable semantic representations. When the database schema is relatively stable, precise rules can be defined to ensure parsing accuracy. Unlike data-driven approaches, rule-based methods do not require large annotated datasets. However, handling complex queries, such as nested queries and multi-table joins, requires the definition of a substantial number of rules, leading to high maintenance costs. Moreover, the rule-based methods struggle with flexible queries that involve synonyms or varied sentence structures. When the database schema or query types change, rules must be manually adjusted, resulting in poor adaptability to new requirements.

With the advancement of NLP technologies, rule-based methods have gradually been supplanted by data-driven approaches, including methods based on machine learning, PLMs, and LLMs. Nevertheless, in certain domains such as healthcare and finance, researchers continue to explore hybrid approaches that combine rule-based and deep learning methods to leverage the interpretability of rules and the generalization capability of deep learning. Examples include: applying rule-based parsing for preliminary analysis, followed by deep learning models to optimize query conversion, and extracting structured information using rule-based methods and subsequently employing large models for SQL generation to enhance generalization.

Rule-based methods play a crucial role in the early development of Text-to-SQL research. Despite the limitations, rule-based methods has laid a solid foundation for the emergence of data-driven approaches.

2.2 Machine learning-based methods

The machine learning-based NLU methods reduce reliance on manual rules, leveraging statistical learning techniques to automatically learn query parsing patterns from data. The core idea of the approach is to map NLQs to structured semantic representations or intermediate forms, thereby enhancing generalization and automation. Based on technological evolution, the methods can be categorized into traditional machine learning and neural networks.

(i) Traditional machine learning methods primarily use statistical classification models such as decision trees, SVMs, hidden Markov models (HMMs), and CRFs. The models rely on manually crafted features and perform well on NLQ parsing tasks with high structural regularity. CHILL [15] exemplifies this line of work by adopting a shift-reduce parsing framework that directly maps NLQs to structured logical queries. Unlike traditional rule-based systems, CHILL learns parsing rules from training data using inductive logic programming (ILP), rather than requiring manual specification. The parser applies three main types of operators: (i) introduction operators that associate lexical items (e.g., "*capital*") with logical predicates (e.g., *capital(S, C)*, meaning *S* has capital *C*), (ii) co-reference operators that handle variable sharing across semantic dependencies such as pronouns and ellipses, and (iii) composition operators that combine sub-queries into compound logical forms such as conjunctions and nested structures. CHILL constructs an overgeneralized parser by extracting all possible introduction, co-reference, and composition operators from training examples. ILP is then used to induce control rules that guide the shift-reduce process toward correct derivations. For example, if the top of the stack contains *capital(S,C)* and the input token is "*of*", a co-reference operation may be triggered. The learned rules are integrated into the parser to form the final system. To ensure accurate translation, CHILL limits generality and increases system complexity. The logical query language must be redesigned for each new domain, and lexical-to-predicate mappings depend on manually built dictionaries, which are difficult to scale. CHILL also requires a substantial amount of labeled data, as experiments indicate that more than 175 annotated examples are needed to outperform manually constructed systems, which leads to high annotation costs.

(ii) With the rise of deep learning, neural network-based methods have gradually replaced traditional machine learning approaches by reducing reliance on handcrafted features and enabling automatic acquisition of semantic representations. To extract database-relevant structural information from NLQs, Wang et al. [88,89] introduced a multi-component recognition framework that integrates deep neural modeling, adversarial mechanisms, semantic alignment, and contextual parsing. A column mention binary classifier employs a deep LSTM model with bidirectional attention to determine whether the input mentions a specific column name. To further localize specific mention phrases, an adversarial text method identifies influential word spans by computing the gradient of the loss with respect to input embeddings, highlighting the segments most responsible for predicting column mentions. For value identification, an independent value mention classifier compares the average embedding of all cell entries in a column with candidate phrase vectors, using a multilayer perceptron to assess whether the phrase refers to a particular column's value. To address many-to-many ambiguities, a mention resolution mechanism leverages dependency parse trees to select column-value pairs with minimal structural distance, prioritizing syntactically proximate matches. To ensure translation accuracy, the system sacrifices generality and incurs higher computational costs. Column and value identification relies heavily on database metadata, such as schema and column-value distributions, which must be reconstructed when adapting to a new domain. Moreover, the sequence-to-sequence (seq2seq) model incorporating attention and copying mechanisms introduces additional overhead at both training and inference stages.

NLU methods based on machine learning enable automatic feature learning from data, reducing the need for manually crafted rules and allowing adaptation to diverse query structures without exhaustive rule specification. However, training such models typically requires large amounts of annotated data, and performance may degrade when applied to domains with different data distributions, necessitating retraining. In contrast to rule-based systems, deep learning models offer limited interpretability of the translation process. For rare or structurally complex queries, insufficient generalization may occur.

With the emergence of PLMs and LLMs, recent research

increasingly focuses on leveraging pre-trained architectures for NLU. Representative strategies include: (i) integrating models such as BERT and T5 to enhance semantic parsing and generalization, (ii) adopting seq2seq training to directly map NLQ to SQL, improving conversion efficiency, and (iii) employing hybrid frameworks that combine rules and deep learning methods to enhance interpretability.

Machine learning-based techniques provide a data-driven paradigm for natural language parsing, establishing a foundation for the advancement of PLMs and LLMs in query understanding tasks.

2.3 Pre-trained language model-based methods

With the advent of PLMs, such as BERT [34] and RoBERTa [35], NLU has gained the ability to effectively model contextual semantics and schema understanding, enabling more accurate semantic extraction than rule-based and machine learning-based methods.

Models based on BERT focus on encoding NLQs and database schema representations separately and then mapping them into a shared space. One of the pioneering models, SQLova [20], employs BERT to encode both the query and the table schema, using a pointer network for slot-filling in SQL generation. The approach achieves high accuracy for simple queries by capturing user intent efficiently. However, SQLova faces limitations when handling complex multi-table or nested queries, as the reliance on fixed table structures hinders the ability to model dynamic schemas effectively. Similarly, TaBERT [90] introduces a flexible framework by jointly encoding both table contents and NLQs, which can be generalized to unseen tables and multi-table queries. Nonetheless, this increased flexibility comes with higher computational costs, leading to longer training and inference times. Additionally, RAT-SQL [21] employs a relational graph representation of both the database schema and query, utilizing relation-aware self-attention mechanisms to capture the dependencies between schema elements and query tokens. RAT-SQL shows significant improvements on cross-domain datasets but suffers from substantial computational overhead due to the graph-based approach.

Models based on graph structures leverage the relational nature of databases by explicitly modeling schemas as graphs. The models aim to capture the structural dependencies between database schema elements and query tokens effectively. GraPPa [91] enhances pre-trained models with syntactic and schema-based pre-training, achieving schema linking and generalization. Similarly, SmBoP [22] reformulates semantic parsing as a bottom-up tree generation task, using beam search to reduce exposure bias and produce structurally accurate SQL queries, thus benefiting from improved compositional generalization. BRIDGE [92] takes schema linking a step further by treating schema elements as part of the input sequence, allowing close interaction between query tokens and schema information. SCoRe [93] refines the SQL generation process with a coarse-to-fine ranking mechanism, improving both efficiency and robustness in handling complex queries. These models focus on improving schema encoding by leveraging the graph structure of the database, which enhances the overall accuracy of SQL generation.

Other perception modeling methods explore alternative techniques to further improve schema understanding and database generalization. HydraNet [94] and RESDSQL [95] introduce dynamic schema linking and hybrid decoding mechanisms, respectively. HydraNet utilizes multiple decoders tailored to different parts of the SQL statement, enhancing the flexibility of schema interpretation. On the other hand, RESDSQL enhances database schema representation by grounding the query interpretation in database content, thereby improving accuracy on cross-domain tasks. These approaches demonstrate the potential of dynamic and hybrid techniques in further enhancing PLM-based models' capability to handle diverse and complex queries.

Despite these advancements, a major challenge of balancing expressiveness with computational efficiency remains. While improving the performance on complex queries, schema and graph-based models typically demand significant computational resources. Future research may focus on developing lightweight architectures that can efficiently handle schema-aware processing, adaptive modeling techniques, and effective pre-training objectives to further bridge the gap between performance and resource constraints.

PLM-based methods for NLU have evolved from basic text encoding techniques to sophisticated models that integrate schema awareness and graph-based structures. The innovations have substantially advanced the ability of models to interpret user queries, providing a strong foundation for generating accurate and executable SQL queries.

2.4 Large language model-based methods

LLMs further advance NLU by performing schema linking and intent recognition through prompt-based inference and few-shot learning. In comparison with PLMs, which typically necessitate fine-tuning for specific tasks and structured decoding, LLMs offer greater flexibility and generalizability with minimal supervision.

Prompt engineering involves designing prompts that incorporate several query-schema alignment examples to guide the LLM in reusing the linking patterns demonstrated in these examples. For instance, QDecomp [96] and ACT-SQL [25] employ few-shot learning to guide schema linking and emphasize example construction. Building upon the foundational idea, some studies have devised sophisticated pathways to further enhance performance by prompting the LLM at specific steps. PET-SQL [97] inputs schema information and several example SQL queries into the LLM in the form of a reference-enhanced prompt, achieving explicit alignment between entities in the query and schema elements (tables/columns) through a two-phase generate-and-refine process. During the reference-enhanced representation, the prompt includes: schema description (table names, column names, foreign key relationships), randomly sampled cell values to provide concrete examples, and few-shot demonstrations (question-SQL pairs) as in-context learning material.

The fine-tuning process primarily involves two approaches: Full model fine-tuning: Supervised fine-tuning is applied directly to the weights of LLMs using datasets (e.g., Spider) annotated with schema linking labels, which enables the internal representations of model to jointly encode natural language and schema relationships.

Adapter/Low-Rank Adaptation (LoRA)-based fine-tuning: Only a small subset of parameters (adapter modules or low-rank decomposition matrices) are fine-tuned, improving linking accuracy while maintaining computational efficiency [98]. Beyond developing universal Text-to-SQL models, task-specific adaptation is typically achieved through Adapter/LoRA fine-tuning. A notable example is LR-SQL [99], which reformulates schema linking as a dedicated fine-tuning task. The method partitions the full database into adjustable slices, each fed into the model individually to reduce context length and GPU memory overhead. For each slice, the model is trained to predict which tables/columns should be linked and is guided to output the linking reasoning process via Chain-of-Thought (CoT). The approach strengthens the model's awareness of inter-table relationships.

Integrated structured database knowledge has increasingly used for NLU to improve the alignment between user queries and relational schemas, benefiting by the advancement of LLMs. TaBERT [90] exemplifies this direction by jointly pre-training on natural language utterances and tabular content, enabling the model to better capture relationships between textual input and database structure. The design has demonstrated strong performance on datasets such as Spider, highlighting the benefits of incorporating contextual table information into the representation process.

Prompt-driven methods require no additional training and enable rapid prototyping with low latency through carefully designed two-round prompts. However, these methods are susceptible to prompt design limitations in scenarios involving extremely large schemas or dynamic queries, often exhibiting significant accuracy fluctuations. Fine-tuning methods employ supervised fine-tuning for both schema linking and SQL generation to internalize schema learning into model parameters, which significantly improves linking stability and supports large-scale schemas. Nevertheless, these methods demand additional computational resources and GPU memory, and require re-tuning during model iteration or migration. Structured database knowledge-based methods treat schemas as knowledge graphs, dynamically extracting multi-hop structural information through a combination of graph retrieval and vector retrieval. These methods construct generation prompts by integrating retrieved schema knowledge with exemplar SQL templates, achieving the highest accuracy and immediate adaptability to schema changes in complex multi-table join scenarios. However, the methods exhibit the most intricate system architecture and incur the highest cumulative latency from retrieval and generation.

For small-scale database schemas that require rapid deployment, prompt-driven approaches are preferable. If stringent requirements exist for online service stability and low latency, and sufficient computational resources are available, fine-tuning methods are more suitable. In cases involving extremely complex multi-table deep-join queries with strict accuracy demands, structured database knowledge-based methods represent the optimal choice.

3 Natural language translation

At the NLT stage, the extracted semantic information is transformed into a structured query language for the database. The input is a semantic representation, and the output is a structured database query. In a manner similar to NLU, NLT techniques have evolved from deterministic rule-based strategies to neural and LLM-driven approaches, aiming to improve expressiveness, adaptability, and automation.

3.1 Rule-based methods

Rule-based NLT methods constitute the core approach in early Text-to-SQL systems. The methods convert structured semantic representations into SQL through formal grammars, rule sets, or mapping templates.

(i) Grammar-driven translation methods leverage context-free grammar or semantic grammar to define transformation rules from natural language representations to SQL. In the case of TR Discover [13], the translation process centers on the adaptation between grammar rules and the vocabulary. NLQs are converted from an intermediate FOL into an executable query language (e.g., SPARQL or SQL). The ANTLR tool is used to parse the FOL representation and generate a syntax tree based on predefined FOL grammar rules, ensuring the correct structural parsing of logical expressions. An in-order traversal of the syntax tree is performed to push logical conditions and connectors onto a stack, ultimately forming the query constraints. Logical predicates are mapped to database attributes or ontology properties. Predefined grammar rules, such as TYPE constraints for verbs, are employed to ensure consistency with the domain model. For example, the verb "*developed by*" is constrained to [*drug, org, dev*], guaranteeing that only valid domain relations are captured. To enhance translation accuracy, TR Discover imposes restrictions on both query expressiveness and generalization capability. Specifically, TR Discover does not fully support quantification queries (e.g., "*highest number of side effects*") or negation queries (supported only in SPARQL). The translation accuracy of TR Discover is highly dependent on the coverage of predefined grammar rules and vocabulary. For instance, in the QALD-4 benchmark, TR Discover fails to process queries involving "*drug categories*" due to the absence of corresponding object property rules.

(ii) Rule-matching translation methods generate SQL based on formalized rules. In the case of xDBTagger [14], after completing the keyword mapping, a rule-based algorithm is used to convert the extracted keywords into SQL queries. The algorithm consists of four steps. 1) Schema Graph Extraction: An undirected graph is constructed from the database schema, where nodes represent tables or attributes, and edges denote the *ownership* relationships between tables and attributes (e.g., foreign key associations). 2) Join-Path Inference: The Dijkstra algorithm is used to find the shortest paths between tables involved in the keyword mapping, generating the necessary JOIN conditions. 3) Where Clause Completion: Based on the VALUE type labels in the keyword mapping (e.g., "*House of Cards*" mapped to *tv_series.title*), equality conditions are generated (e.g., *tv_series.title* = "*House of Cards*"). 4) Heuristics for Aggregation Queries: Aggregation keywords in natural language (e.g., *count* and *total*) are detected, and corresponding SQL aggregation operations (e.g., COUNT(*)) are generated through

sliding window matching. To improve conversion accuracy, xDBTagger limits query expressiveness. Specifically, xDBTagger cannot handle complex logic requiring subqueries or queries involving GROUP BY and aggregation, as well as synonyms or implicit logic, such as the need for explicit mapping of terms like "*films*" to "*movies*" at the table level.

(iii) Template-based translation methods predefine SQL templates and fill the placeholders in SQL according to the NLQ. Taking LogicalBeam [85] as a case study, a two-stage template matching technique is employed to generate diverse and logically consistent SQL. At the first stage, a "*logical plan*" is generated based on the input NLQ, outlining the core structure of the SQL query, including key components such as the number of JOIN operations and SELECT columns. Multiple variations of the logical plan are then created through counterfactual perturbation. For example, if the original plan includes 2 JOINs and 3 SELECT columns, alternative plans, such as (1 JOIN, 3 SELECT) or (2 JOIN, 4 SELECT), may be generated to encompass potential variations in interpretation. At the second stage, a specific SQL template is generated for each logical plan using greedy decoding, with details such as table names and column names abstracted into placeholders (e.g., using "*table*" and "*column*"). During the template filling process, beam search is applied, where branching is strictly confined to the selection of table or column names, while other elements such as SQL keywords and operators are generated based on the highest probability. A whitelist mechanism is employed to ensure that the filled table and column names are drawn from the actual database schema, thus avoiding the insertion of invalid or irrelevant terms. For example, when filling the template *SELECT* [*column*] *FROM* [*table*], the column placeholder is limited to selecting actual column names from the current database schema. To enhance conversion accuracy, LogicalBeam restricts the query expressiveness, thereby increasing conversion overhead. Specifically, template generation depends on predefined logical plans (e.g., the number of JOINs and SELECT columns), and queries requiring complex logic (e.g., nested subqueries or dynamic conditions) may not be supported. The two-stage generation process involves more decoding steps compared to end-to-end models. Despite optimization through greedy decoding and constrained beam search, LogicalBeam remains more time-consuming than single-stage methods, such as T5-3B [36].

Rule-based NLT methods offer interpretability and determinism but have significant limitations in generalization, scalability, and contextual understanding. The translation process relies on fixed rules, which makes the generated SQL easy to understand and debug. Identical inputs produce the same SQL, ensuring stable results. Rule-based methods perform well for queries with clear structures and stable schemas. However, the methods struggle with handling complex, ambiguous, or unstructured natural language inputs. New domains or grammars require manually designed rules, which makes accommodating diverse query formats difficult. Additionally, the rule set requires continuous maintenance and updates to adapt to new expressions. Rule-based approaches also face challenges in understanding the contextual meaning of multi-turn dialogues or cross-sentence queries.

While rule-based methods are still applied in some restricted domains (e.g., structured database queries), such methods are gradually being replaced by machine learning-based and LLM-based approaches in general Text-to-SQL tasks. Potential future improvements include: 1) combining rule-based methods with statistical learning models to enhance generalization capabilities, 2) automatically learning and optimizing the rule set using a small amount of labeled data, 3) enhancing query understanding through knowledge graphs to improve rule matching capabilities, and 4) integrating multi-modal data, such as tables, images, and speech, to increase the adaptability of NLT.

3.2 Machine learning-based methods

Machine learning-based NLT methods eliminate the reliance on manually crafted rules by employing data-driven techniques that automatically learn mappings from natural language to SQL through statistical learning models. The methods typically utilize statistical or neural network models to perform end-to-end translation or structured prediction. From a technological perspective, the approaches are generally categorized into traditional machine learning and neural networks.

Traditional machine learning methods, such as logistic regression, decision trees, and random forests, predict SQL based on feature engineering. Additionally, drawing from statistical machine translation techniques, NLQs are treated as the source language and Zettlemoyer and Collins [100] use combinatory categorial grammar (CCG) and probabilistic models to map the meaning of natural language sentences into lambda calculus expressions. Initially, the GENLEX [100] function automatically generates candidate lexical entries by rules. Words or phrases in the sentence are mapped to lexical entries that include both grammatical types and semantic expressions. Subsequently, syntactic parsing is performed using the syntactic rules of CCG (e.g., function application, composition, and type raising), gradually combining lexical entries to form the sentence's syntactic tree while constructing the logical form layer by layer using the compositional semantics of lambda calculus. For complex semantics (e.g., quantification and comparatives), GENLEX generates higher-order lambda terms (e.g., arg max), and efficiently searches for valid parses using dynamic programming. To address ambiguity, a log-linear probability model computes the probability of a parse tree, weighted by features such as lexical usage frequency, and selects the optimal logical form. To improve translation accuracy, *Zettlemoyer and Collins* [100] limit the expressiveness of queries and increase conversion overhead. Specifically, GENLEX rules may fail to cover complex sentence structures, such as preposition stranding constructions like "*Through which states does the Mississippi run*", resulting in some sentences being not parsed (e.g., a recall rate of 79% in the experiments). Furthermore, the initial lexicon requires domain knowledge (e.g., entity lists), and GENLEX rules must be manually designed.

Neural network-based methods construct seq2seq models to map natural language to SQL. Furthermore, by combining Seq2Seq with SQL structural constraints, the models directly generate the syntax tree of SQL, avoiding invalid SQLs. For instance, GENSQL [18]

uses the GAR [19] framework to break down traditional end-to-end methods into several controllable and optimizable steps. First, SQL candidates are generated. GENSQL applies manually defined generalization rules to raw SQL collected from database query logs or examples, generating a set of representative and broad SQL candidate templates. The rules include replacing constants and specific column names with placeholders. The process is performed offline and builds a candidate space that covers potential query intents. Next, for each candidate SQL, GENSQL generates the corresponding natural language description using a template-driven approach, referred to as dialect expression. The step forms the foundation for subsequent matching with user queries while enhancing the system's interpretability and controllability. Finally, the input NLQ is matched with the generated dialect expressions, and the closest candidate is selected. To achieve the goal, GENSQL introduces a Learning-to-Rank (LTR) model, which is trained to measure the similarity between user queries and dialect expressions in the semantic space. The input to the LTR model typically consists of pairs of NLQ and candidate expressions, and the output is a similarity score. GENSQL ultimately returns the SQL with the highest score as the result. To improve translation accuracy, GENSQL limits generalization capability and increases conversion overhead. Specifically, if sample queries do not cover all possible user queries, GENSQL may fail to generate the correct SQL. The processes of generating and ranking candidate SQLs introduce high computational complexity. During the online phase, ranking requires processing a large number of candidate queries, leading to longer query translation time, approximately twice that of other models.

Machine learning-based NLT methods use statistical modeling and neural network techniques to automatically learn the mapping between natural language and SQL. Early statistical approaches have gradually been replaced by deep learning methods, which perform better on complex queries. With the development of LLMs, machine learning-based methods are increasingly integrated with LLMs, advancing toward intelligent Text-to-SQL systems.

3.3 Pre-trained language model-based methods

PLM-based NLT methods leverage rich contextual information and transfer learning capabilities to improve the accuracy of SQL generation and are more effective than traditional neural methods in handling diverse query structures. From a technological perspective, PLM-based approaches can be broadly categorized into classification and sketch-based methods, generative methods, and enhancement mechanisms for performance optimization.

Classification and sketch-based methods adopt a pipeline strategy to predict SQL components separately. Early models such as SQLova [20] utilize BERT [34] as a contextual encoder to improve the alignment between natural language tokens and database schema elements. SQLova frames SQL generation as a classification task, predicting SQL components like SELECT columns and WHERE conditions through component-wise classifiers. Despite improving model interpretability and modularity, the strategy lacks flexibility when generating structurally diverse or complex queries.

Sketch-based methods such as Seq2SQL [101] and SQLNet [23] introduce an intermediate representation, which is a sketch of the SQL structure, before filling in the specific columns, operators, and values. The two-step pipeline constrains the search space and enhances sample efficiency but is limited in handling novel or nested query forms. Further improvements are achieved by BRIDGE [92], which enhances schema linking through relational encoding of questions and table structures, significantly improving performance in cross-domain scenarios. Furthermore, SmBoP [22] employs a bottom-up tree-based decoding framework guided by BERT representations, improving compositional generalization.

Generative methods treat the Text-to-SQL task as a direct seq2seq generation problem. The approaches typically fine-tune large-scale PLMs such as T5 [36] to generate SQL queries from input questions in an end-to-end manner. T5-SQL models learn to map entire input sentences to complete SQL statements using a unified Text-to-Text format, enabling high flexibility and generalization. However, the methods often produce syntactically invalid outputs when handling long sequences, nested subqueries, and complex JOIN operations.

To address the limitations, constrained decoding techniques such as PICARD [24] are introduced. PICARD incrementally checks the validity of SQL syntax during the generation process and prunes infeasible candidates, substantially improving the accuracy of execution. When combined with large-scale PLMs like T5-3B, PICARD achieves state-of-the-art performance on benchmarks such as Spider [6]. In this context, emerging neuro-symbolic approaches that combine PLMs with symbolic components, such as rule-based post-processing or execution-time validation, have shown great promise in enhancing execution correctness and improving the interpretability of generated queries. Additionally, UnifiedSKG [102] proposes a unified multitask framework, extending Seq2Seq to various structured tasks including Text-to-SQL. By applying schema serialization and task-specific prompting, UnifiedSKG achieves cross-task generalization but incurs complexity in task formulation and inference.

Enhancement mechanisms further improve the performance of PLM-based Text-to-SQL models through auxiliary techniques. Candidate reranking methods, such as N-Best List Rerankers [103], generate multiple SQL outputs and select the most probable one based on model confidence and execution consistency. Intermediate representation learning, exemplified by RESDSQL [95], first predicts query sketches and the intermediate outputs before composing the final SQL query, effectively improving semantic precision and robustness.

Moreover, extending PLMs with code-focused models such as Codex [38] has shown promising results in few-shot and zero-shot SQL generation scenarios. Despite the strong performance, such models often suffer from high inference latency and low controllability, posing challenges for deployment in real-time or mission-critical applications.

Overall, PLM-based Text-to-SQL methods have evolved from early classification frameworks to powerful generative models enriched by structural constraints, reranking strategies, and schema-aware mechanisms. While these advances significantly boost translation accuracy and generalization, open challenges remain in

robustness, efficiency, and explainability across diverse database scenarios.

3.4 Large language model-based methods

Building on PLMs, LLM-based methods further simplify SQL generation by enabling direct prompt-based synthesis without task-specific training. LLM-based methods have become the mainstream of NLT, which utilize known entities to construct the final SQL through prompts, model fine-tuning, and post-processing.

Prompt engineering has emerged as a powerful methodology for leveraging LLMs across diverse tasks, spanning from zero-shot and few-shot prompting to more advanced techniques such as CoT reasoning [104]. DIN-SQL [26], a cutting-edge approach, dynamically adapts the prompting strategy based on query complexity to generate SQL queries. For simple queries, DIN-SQL employs straightforward few-shot prompts without intermediate steps. For non-nested complex queries requiring joins, DIN-SQL introduces intermediate representations to bridge the semantic gap between NLQs and SQL. For nested complex queries, DIN-SQL decomposes the problem by first resolving subqueries and then integrating the subqueries into the final SQL generation. To ensure robustness, DIN-SQL also incorporates a self-correction module that automatically reviews and rectifies errors in the initial query. The approach achieves good performance on the Spider (85.3% accuracy) and BIRD (55.9% accuracy) benchmarks, setting new standards for Text-to-SQL systems.

Fine-tuning remains a pivotal approach for optimizing LLMs in Text-to-SQL systems. While most studies [105–107] evaluate fine-tuning effects using standard benchmarks like Spider and BIRD, specialized domains necessitate custom datasets, as demonstrated by FinSQL's financial dataset BULL [108]. Constructed from Hundsun Electronics' investment analytics platform, BULL incorporates three financial databases. The framework first enhances input quality through parallel schema linking and hybrid data augmentation, then employs parameter-efficient fine-tuning via LoRA to adapt base models on single-GPU hardware within short timeframes. Trained LoRA weights are managed through a centralized plugin hub, which supports low-resource scenarios through dynamic parameter merging. Base model parameters are first updated with existing hub weights, then further refined with additional LoRA modules to produce business-specific adaptations. Finally, SQL post-processing ensures execution correctness, completing an end-to-end pipeline that balances domain specificity with computational efficiency.

After generating SQL with LLMs, post-processing is required to align with user expectations, which includes: SQL rewriting to correct syntax errors and remove redundant keywords, self-consistency checks to select the most reliable candidate SQL, and execution-based validation to ensure result correctness. Several recent works have introduced innovative strategies to enhance this stage. For example, DEA-SQL [27] proposes an error-analysis-based self-correction mechanism, where common error patterns are identified and used to guide the model in performing preliminary self-correction on the generated SQL. Additionally, DEA-SQL incorporates an active learning paradigm to iteratively refine the error-detection capability by leveraging historical error cases, thereby improving the robustness of post-processing. Another representative method, C3 [28], introduces a SQL Critic module that employs in-context learning from a small set of curated examples, often drawn from external knowledge bases, to refine the output. This module enables the system to perform context-aware corrections by learning from high-quality corrections in prior data. Moreover, SQLfuse [29] addresses the issue of output instability in LLM-generated SQL by introducing a Consistency Output (CO) module. This component mitigates the inherent randomness in LLM outputs through a multi-path reasoning and consensus selection framework. Specifically, SQLfuse generates multiple SQL candidates via diverse prompting strategies, executes them against the target database, and filters out syntactically or semantically invalid queries. The final output is determined via a majority voting mechanism over the valid candidates, significantly enhancing the reliability of the generated SQL, especially in zero-shot settings. Building upon these insights, DIN-SQL [26] focuses on robust SQL rewriting to iron out any syntactic anomalies and eliminate superfluous tokens. Once the model generates an initial query, DIN-SQL applies a sequence of pattern-based transformations to ensure that every SELECT clause pairs logically with FROM, and removes redundant conjunctions. DAIL-SQL [107] leverages a self-consistency checking mechanism to sift through multiple candidate queries and pick the one most likely to be correct. After sampling the Text-to-SQL model multiple times, DAIL-SQL computes the pairwise structural similarity between candidates. By clustering the hypotheses and selecting the consensus query which is often the one with the highest average similarity score. CHESS [105] introduces an execution-centric validation layer to confirm that the chosen SQL not only parses cleanly but also produces semantically correct results. In practice, CHESS submits each candidate query to a sandboxed database instance and analyzes the returned result set for anomalies. Queries that trigger runtime errors, return zero rows unexpectedly, or violate schema constraints are automatically discarded.

Prompt-driven methods leverage manually designed prompts for SQL generation, enabling rapid deployment with zero/few-shot adaptability and low computational overhead. However, the accuracy is sensitive to prompt quality and exemplar coverage, struggling with complex or long-tail queries. Scalability is limited by prompt length constraints, especially under schema expansions. Fine-tuning methods, trained on labeled data, internalize SQL patterns into model parameters, achieving high accuracy for complex, multi-table queries. Yet, these methods demand substantial compute, memory, and retraining when schemas evolve, increasing maintenance costs.

Prompt-driven approaches suit resource-constrained or dynamic environments, while fine-tuning excels where precision and stability outweigh upfront costs. Post-processing significantly enhances the correctness of LLM-generated SQL queries, whether through prompt engineering or fine-tuning approaches.

4 Text-to-SQL benchmarks

We provide a classification and in-depth analysis of common benchmarks and evaluation metrics for translating NLQ to SQL. As shown in Fig. 4, the evolution of Text-to-SQL systems exhibits a

clear correlation between dataset development and method paradigm shifts. Early benchmarks such as ATIS [109] and Restaurants [110] focus on small-scale, domain-specific, and single-table queries, which align with the dominance of rule-based approaches. Starting from 2017, the emergence of more challenging datasets such as WikiSQL [101], IMDB [59], and particularly Spider [6] marks a turning point that emphasizes multi-table, cross-domain, and compositional queries. The development drives the adoption of machine learning-based methods and subsequently PLM-based methods, which improve generalization by leveraging contextual embedding. Starting in 2022, the field enters the LLMs stage, supported by the release of advanced foundation models and diverse benchmarks including BIRD [7] and Spider 2.0 [115]. The datasets introduce additional complexities such as multi-turn interaction, domain diversity, and fine-grained semantic alignment. The timeline reflects an ongoing evolution toward greater expressiveness, adaptability, and usability in Text-to-SQL systems.

4.1 Benchmark datasets

The benchmarks play a pivotal role in the evolution of Text-to-SQL systems. Prior to the dominance of LLMs domain-specific datasets were the standard. However, there was a lack of a unified evaluation metric. The advent of LLMs has shifted the landscape, with cross-domain datasets emerging as predominant resources. Meanwhile, the transition has facilitated the development of numerous new datasets derived from these cross-domain frameworks. We categorize Text-to-SQL datasets into four distinct categories: *domain-specific*, *cross-domain*, *derived*, and *augmented* datasets.

The details of NLQ and executable language pairs for common domains are presented in Table 4. The majority of existing benchmarks are utilized in the domain of relational databases. The comparison of fields and types of SQL supported by the benchmarks for Text-to-SQL is shown in Table 5.

4.1.1 Domain-specific Text-to-SQL datasets

In general, query sets are typically small and designed by system developers in a specialized manner, often focusing on specific, proprietary domains. The design approach makes it challenging to replicate experiments, hindering fair system comparisons and limiting the comprehensive exploration of Text-to-SQL approaches.

ATIS (Airline Travel Information System) [109] is a classical dataset with a relatively old age, having been introduced by Texas Instruments in 1990. ATIS is built on the relational database Official Airline Guide, comprising 32 tables and 5280 queries written in English. The queries pertain to details regarding flights, ticket prices, destinations, and services available at airports, including join queries and nested queries, but no grouping and sorting queries. The average length of NLQs and SQLs in ATIS is approximately 11 and 67 words, respectively. Each query operates on an average of six tables. An example query is as follows.

Q1: What aircraft is used on delta flight 1984 from Kansas city to Salt Lake city?

Restaurants [110] comprises a vast collection of dining establishments located in Northern California, storing restaurant names, locations, features, and travel guide ratings. The benchmark contains 378 questions about restaurants, food types, and locations. An example query is as follows.

Q2: Where is a good Chinese restaurant in Palo Alto?

GeoQuery [111] consists of 6 tables and 877 NLQs in the US geographic database. The queries in GeoQuery are designed for the geographic domain, including join queries, nested queries, grouping queries and sorting queries. The average length of NLQs and SQLs in GeoQuery is about 8 and 16 words, respectively. Additionally, each query operates on an average of one table. Although the queries are relatively brief in length, they are highly composable, with nearly half of the SQL containing at least one nested sub-query. One of the English queries is as follows.

Q3: What is the largest city in states that border California?

MAS [77] is generated from the Microsoft Academic Search database, which stores information such as academic papers, authors, journals, and conferences. The source of NLQs in MAS is the logical queries that are capable of being articulated in the search interface of the Microsoft Academic Search platform. The fields of MAS and Scholar are both academic in nature, but exhibit distinct patterns. One English query is as follows.

Q4: Return authors who have more papers than Bob in VLDB after 2000.

Scholar [112] consists of 817 NLQs for academic database search that are annotated with SQL. The average length of NLQs and SQLs

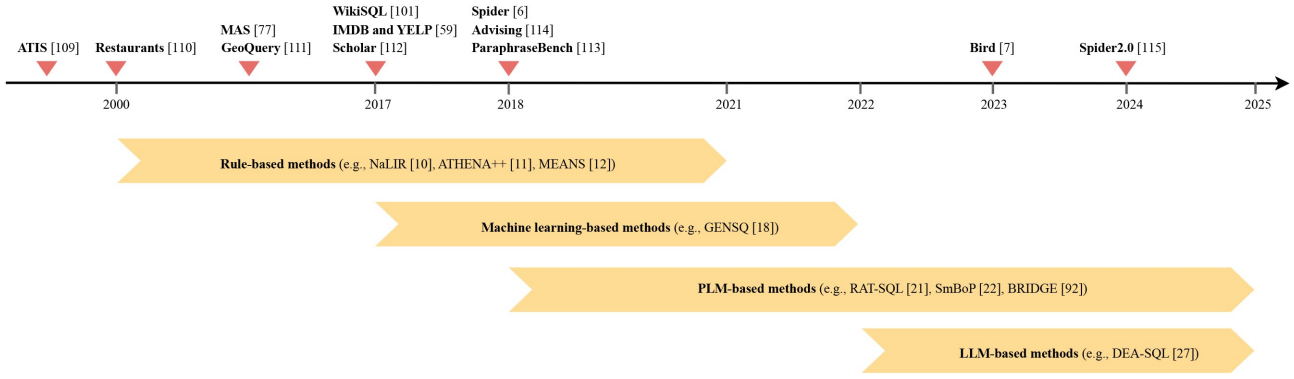


Fig. 4 Timeline of Text-to-SQL benchmarks and method paradigm shifts

Table 4 Examples of NLQ and structured language pairs for common domains

Domain	Examples of NLQ and executable language pairs
Relational database	**NLQ:** How many CFL teams are from York College? **SQL:** SELECT COUNT CFL Team FROM CFLDraft WHERE College = "York"
Spatial database	**NLQ1:** What is the population of San Antonio? **Lambda expression:** answer(A,population(B,A),const(B,cityid(San Antonio))) **NLQ2:** Could you tell me what parks are in the center? **Executable language:** query park feed filter [.GeoData ininterior center] consume;
Moving objects database	**NLQ:** Where did the train 7 go at 8am? **Executable language:** query Trains feed filter [.Id = 7] filter [.Trip present [const instant value "2020-11-20-8:00"]] extend [Pos: val (.Trip atinstant [const instant value "2020-11-20-8:00"])] project [Id, Line, Pos] consume;
Graph database	**NLQ:** Who is the director of The Matrix starring Keanu Reeves? **Cypher [116]:** MATCH (keanu:Person)-[:ACTED_IN]->(movie:Movie), (director:Person)-[:DIRECT]->(movie:Movie) WHERE keanu.name = 'Keanu Reeves' and movie.name = 'Matrix' RETURN director

Table 5 Comparison of benchmarks for Text-to-SQL

Benchmark	Year	#Queries	#DBs	#Tables/ DB	Select query	Group query	Sort query	Join query	Nested query	Fields involved	Usage in papers
ATIS [109]	1990	5280	1	32	√			√	√	air travel	[112,114,117–119]
Restaurants [110]	2000	378	1	3	√			√		restaurant	[16,51,75,110]
GeoQuery [111]	2001	877	1	6	√	√	√	√	√	geography	[11,16,19,51,58,75,100, 110–112,114,118–121]
MAS [77]	2014	196	1	17	√	√		√		academic	[11,14,58,59,77,122–124]
Scholar [112]	2017	817	1	7	√			√		academic	[114]
IMDB [59]	2017	131	1	16	√			√		internet movie	[14,59,81,123,124]
YELP [59]	2017	128	1	7	√			√		business review	[14,59,123,124]
WikiSQL [101]	2017	80654	26521	1	√					multiple fields (e.g., state, college, manufacturer)	[89,101,125–131]
Paraphrase-Bench [113]	2018	290	1	1	√	√				medical	[113]
Advising [114]	2018	3898	1	10	√			√	√	university course	[114]
Spider [6]	2018	10181	200	5.1	√	√	√	√	√	138 different fields (e.g., car, stadium, country)	[6,11,17,19,124,127, 129–135]
Bird [7]	2023	12751	95	7.3	√	√	√	√	√	37 professional domains (e.g., blockchain, hockey, healthcare)	[7]
Spider 2.0 [115]	2024	632	213	2.3	√	√	√	√	√	project level	[115]

in Scholar is approximately 7 and 29 words, respectively. Each query operates on an average of 3 tables. Iyer et al. [112] provided a database for performing these queries, which includes academic articles, journal details, author information, keywords, citations, and

utilized datasets. One of the English queries is as follows.

Q5: Get all author having data set as DATASET_TYPE.

IMDB and YELP [59] are generated using data from the Internet Movie Database and Business Review Database, respectively. The NLQs are obtained from coworkers of Yaghmazadeh et al. [59], who are only aware of the types of data available in the database and not the underlying database schema.

ParaphraseBench [113], a component of the DBPal [136], is a benchmark utilized to assess the robustness of NLIDBs. Unlike existing benchmarks, ParaphraseBench covers diverse language variants of user input NLQs and maps natural language to the anticipated SQL output. The benchmark is constructed upon a medical database that contains a single table for storing patient information. The language variants utilized in NLQs permit the classification of NLQs into six categories, as illustrated in Table 6.

Advising [114] was proposed in 2018, and the NLQs were built on a database of course information from the University of Michigan containing fictitious student profiles. A portion of the queries are collected from the Facebook platform of the EECS department, and the remaining questions are formulated by computer science students well-versed in database topics that might be raised in academic consulting appointments. The queries in Advising are for student-advising tasks, including join queries and nested queries. One of the English queries is as follows.

Q6: For next semester, who is teaching EECS 123?

4.1.2 Cross-domain Text-to-SQL datasets

Cross-domain datasets, with the extensive data scale, intricate query structures, and multi-domain characteristics, enable the training and evaluation of LLM-based Text-to-SQL systems.

WikiSQL [101], introduced in 2017, is a comprehensive and meticulously annotated collection of natural language to SQL mappings, and currently represents the most extensive dataset for Text-to-SQL. WikiSQL contains SQL table instances extracted from 26521 HTML tables on Wikipedia, and 80654 NLQs, each accompanied by an SQL. WikiSQL comprises genuine data extracted from the web, with queries involving a multitude of tables, but the queries do not involve complex operations such as GROUP BY and multi-table union queries. The majority of questions in WikiSQL are between 8 and 15 words in length, most SQLs are between 8 and 11 words, and most table columns are between 5 and 7. In addition, most NLQs are of the *what* type, followed by *which*, *name*, *how many*, *who*. The execution accuracy of WikiSQL has significantly improved from the initial 59.4% to 93.0%, and the method has undergone a transformation from a simple seq2seq approach to a multi-tasking, transfer learning, and pre-training paradigm. A pair of questions and SQLs for the CFLDraft table can be formulated as follows.

Q7: How many CFL teams are from York College?

SQL_Q7: SELECT COUNT CFL Team FROM CFLDraft WHERE College = “York”

Spider [6] is a large Text-to-SQL dataset introduced by Yale University in 2018, in order to solve the requirement for extensive and high-caliber datasets for a novel intricate cross-domain semantic parsing challenge. The dataset contains 10181 NLQs and 5693 corresponding complex SQLs, which are distributed across 200 independent databases, and the content covers 138 different domains. The average length of questions and SQL statements in Spider is approximately 13 and 21 words, respectively. While the number of questions and SQLs in Spider is not as extensive as that of WikiSQL, Spider contains all common SQL patterns and complex SQL usages, including advanced operations like HAVING, GROUP BY, ORDER BY, table joins, and nested queries, which makes Spider closely aligned with real-world scenarios. The following is an illustrative example of a complex problem and the corresponding SQL, which contains a nested query, a GROUP BY component, and multiple table joins.

Q8: What are the name and budget of the departments with average instructor salary greater than the overall average?

SQL_Q8: SELECT T2.name, T2.budget FROM instructor as T1 JOIN department as T2 ON T1.department id = T2.id GROUP BY T1.department id HAVING avg (T1.salary) > (SELECT avg (salary) FROM instructor)

Bird [7], one of the most prominent benchmarks introduced in recent years, was developed by Li et al. in 2023. The large-scale platform is specifically designed for Text-to-SQL cross-domain tasks and features an extensive database comprising 12751 Text-to-SQL pairs and 95 databases, totaling 33.4 GB across 37 specialized domains. In contrast to Spider, Bird significantly increases the complexity of the dataset and the intricacies of the queries involved. While Spider includes databases from hundreds of domains, the

Table 6 Query categories and examples for ParaphraseBench

Category	Example queries
Naive	What is the average length of stay of patients where age is 80?
Syntactic	Where age is 80, what is the average length of stay of patients?
Morphological	What is the averaged length of stay of patients where age equaled 80?
Lexical	What is the mean length of stay of patients where age is 80 years?
Semantic	What is the average length of stay of patients older than 80?
Missing Information	What is the average stay of patients who are 80?

majority of the databases contain only three to five tables, resulting in a notable disparity between academic research and practical applications. Bird mitigates the limitation by incorporating dirty data, thereby expanding the scale of the database and complicating the acquisition of complete contextual information. Consequently, Bird integrates external knowledge of databases, placing a stronger emphasis on the values contained within the databases rather than remaining confined to the schema level. Whereas Spider categorizes query difficulty primarily based on query length, Bird assesses query complexity through various factors, including the intricacy of SQL queries, the comprehension of questions, schema linking, retrieval of external knowledge, and reasoning. This stands in contrast to Spider' s methodology, which focuses exclusively on result accuracy and employs a combined evaluation of both accuracy and execution efficiency. Additionally, Bird introduces an optimization pipline of the generated SQL, which facilitates the generation of global SQL queries that interact with the database, thereby enhancing the model's understanding of data types and distributions and optimizing query efficiency. Nonetheless, it is essential to highlight that neither benchmark accounts for the time of query generation.

Spider 2.0 [115] represents an advanced evaluation framework for Text-to-SQL tasks within real-world enterprise-grade workflows. Spider 2.0 comprises 632 complex Text-to-SQL workflow questions drawn from diverse enterprise database use cases. The dataset features databases sourced from actual data applications, typically containing over 1000 columns, hosted on cloud-based or on-premises systems such as BigQuery, Snowflake, or PostgreSQL. Questions in Spider 2.0 require understanding and navigating database metadata, dialect documentation, and project-level code repositories. Key challenges include handling long contexts, performing complex reasoning, and generating multiple SQL queries with diverse operations, which often exceed 100 lines of code.

4.1.3 Derived Text-to-SQL datasets

The creation of large Text-to-SQL datasets from scratch necessitates substantial manual labor, which has resulted in the development of various datasets that are derived from extensive cross-domain collections, emphasizing particular aspects of the Text-to-SQL problem, such as schema linking. The following steps describe the process in detail.

1) Researchers are required to conduct a meticulous analysis of the existing benchmarks, including an examination of the data structures, query types, and complexity. Through the analysis, they can gain insight into the constraints of the benchmark and identify potential avenues for enhancement.

2) Designing a modification strategy is a critical step, which involves how to modify and extend the dataset on the basis of the analysis results. The step may include adding new queries, changing the linguistic expression of queries, and introducing complex query types.

3) In the process of implementing modifications, researchers are expected to execute the designed modification strategy with precision to ensure that the new benchmark meets the expected requirements.

4) Evaluating the performance is a pivotal aspect of the process. The researchers employ the modified benchmark to train and test Text-to-SQL models, subsequently assessing the models' performance and generalization capabilities according to the new benchmark.

Building on Spider [6], Kaoshik et al. [137] proposed a new Text-to-SQL benchmark, named ACL-SQL, containing five tables and 3100 pairs of NLQ and SQL. By defining and annotating three types of questions on temporal aspects in Spider: questions querying for temporal information, questions querying for temporal information with grouping or ordering, and questions with temporal conditions, Vo et al. [138] proposed a new dataset, TempQ4NLIDB, which can assist NLIDB systems based on machine learning approaches to improve the performance on temporal aspects. To address the dearth of publicly available benchmarks on ambiguous queries, Bhaskar et al. [85] generated a new benchmark called AmbiQT by modifying Spider with a combination of synonym generation and ChatGPT-based and standard rules-based perturbation. AmbiQT comprises in excess of 3000 examples, each of which can be interpreted as two valid SQLs due to lexical ambiguities (namely, unambiguous column and table names) or structural ambiguities (namely, the necessity of joins and the pre-computation of aggregations).

4.1.4 Augmented Text-to-SQL datasets

Current cross-domain datasets still depend on template-based construction or manual generation. Consequently, there has been a rise in high-quality datasets that can be automatically generated in large volumes from existing data:

Koutrika et al. [139] utilized a graph-based approach for transforming SQL into natural language. SQL is first represented as a directed graph whose edges are labeled with template labels using an extensible template mechanism, thus providing semantics for the parts of the query. These graphs are then explored and textual query description is composed using a variety of graph traversal strategies, including the binary search tree algorithm, the multi-reference point algorithm, and the template combination algorithm.

Eleftherakis et al. [140] addressed SQL2Text by extending the graph-based model of Logos to translate a wider range of queries (e.g., SELECT TOP, LIMIT, IN, and LIKE). The SQL is first analyzed to generate a parse tree storing the essential information for constructing the query graph, and then the textual description of the SQL is created through the application of the multi-reference point traversal strategy.

Camara et al. [141] employed LLM to generate explanations of SQL. The logical structure of SQL is recorded and the columns and tables are interpreted in natural language.

Weir et al. [142] presented a synthesized data generator that synthesizes SQL patterns in the template syntax, including aggregations, simple nesting, and column joins. Each SQL pattern is matched with numerous different natural language patterns, allowing for the generation of a vast number of domain-specific NLQs and SQLs.

Luo et al. [143] proposed an NL2VIS synthesizer, named NL2SQL-to-NL2VIS, which is capable of generating multiple pairs of NL and VIS from a single NL and SQL pair based on semantic

joins between SQL and VIS queries. NL2SQL-to-NL2VIS can be utilized to create NL2VIS benchmarks from established Text-to-SQL benchmarks.

Hu et al. [144] suggested a framework for synthesizing Text-to-SQL benchmarks. The framework involves first synthesizing SQL and then generating NLQs. At the stage of synthesizing SQL, a method is suggested for column sampling based on pattern distance weighting to prevent excessive complexity in concatenation. In the process of generating text from SQL, an intermediate representation is used to facilitate the transition from SQL to NLQ, thereby enhancing the quality of the generated NLQ.

Li et al. [145] presented Omni-SQL, introducing an automated and scalable framework that generates large-scale, high-quality datasets without manual intervention. Omni-SQL infers business scenarios from web tables and constructs realistic multi-table databases with foreign key constraints using LLMs. Through schema augmentation and integration of Spider and BIRD annotations, Omni-SQL fine-tunes a model capable of handling queries of varying complexity and generating chain-of-thought reasoning paths.

Li et al. [146] introduced SQL-Factory, which presents a multi-agent framework for cost-effective and scalable SQL generation. SQL-Factory employs a Generation Team using LLMs to explore diverse query structures, an Expansion Team with lightweight models to refine promising patterns, and a Management Team that adaptively schedules and evaluates generation based on coverage and quality, achieving a balance between diversity and efficiency.

Xu et al. [147] proposed Dagent, which focuses on automated generation of comprehensive database analysis reports. Dagent features a planning module for intelligent task decomposition, a tool module with SQL rewriting and multi-step reasoning capabilities, and a memory module that retains execution history to improve system efficiency over time.

Although progress has been made in the direction, there remains ample opportunity for enhancement. Future research will focus on improving the quality and richness of the generated natural language explanations, ensuring that they are both accurate and rich.

4.2 Evaluation metrics

NLIDB is intended to assist users in efficiently querying and retrieving query results, and thus evaluating the response time and effectiveness of the system is essential. Response time measures how quickly the system can process a user's NLQs and return the relevant results. Effectiveness measures how well the system translates natural languages into accurate and relevant executable database languages, which consists of the following measures: Exact Set Match Accuracy (EM), Execution Accuracy (EX), Test-suit Accuracy (TS), Valid Efficiency Score (VES), Query Variance Testing (QVT), and Interaction Match Accuracy (IM).

Definition 1 (Exact Set Match Accuracy (EM)). EM [6] is defined as the calculation of the matching degree between predicated SQLs and golden SQLs. Considering the nth golden SQL Y_n and the predicted SQL $\hat{Y}_n$, EM can be computed by:

$$EM = \frac{1}{N}\sum_{n=1}^{N}\left(\mathbb{1}(Y_n, \hat{Y}_n)\right),$$

where $\mathbb{1}(Y_n, \hat{Y}_n)$ is an indicator function defined as:

$$\mathbb{1}(Y_n, \hat{Y}_n) = \begin{cases} 1, & \text{if the string of } Y_n = \text{the string of } \hat{Y}_n, \\ 0, & \text{otherwise.} \end{cases}$$

Definition 2 (Execution Accuracy (EX)). EX [6] is defined as the proportion of examples in the evaluation set for which the executed results of both the predicted and golden SQLs are consistent. Considering the result set as V_n executed by the nth golden SQL Y_n, and the result set V_n executed by the predicted SQL $\hat{Y}_n$, EX can be computed by:

$$EX = \frac{1}{N}\sum_{n=1}^{N}\left(\mathbb{1}(V_n, \hat{V}_n)\right),$$

where $\mathbb{1}(V_n, \hat{V}_n)$ is an indicator function defined as:

$$\mathbb{1}(V_n, \hat{V}_n) = \begin{cases} 1, & \text{if } V_n = \hat{V}_n, \\ 0, & \text{otherwise.} \end{cases}$$

Definition 3 (Test-suit Accuracy (TS)). TS [148] constructs a compact, schema-diverse database test suite by sampling from large-scale randomly generated databases with high code coverage, then rigorously measures the alignment of predicted SQL queries against gold-standard annotations through exact-set-match criteria, quantifying both question match accuracy (per-query precision) and interaction match accuracy (joint accuracy across all interaction questions) to establish a statistically robust upper bound on semantic fidelity under real-world schema variability.

Definition 4 (Valid Efficiency Score (VES)). VES [7] is a rigorous evaluation metric that holistically quantifies both the syntactic validity and semantic correctness of generated SQL queries against database schema constraints, while simultaneously optimizing execution efficiency through runtime cost estimation, query plan analysis, and resource utilization benchmarks to ensure practical deployability. The VES formula is represented by:

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

$$\text{R}(Y_n, \hat{Y}_n) = \frac{\sqrt{\text{E}(Y_n)}}{\sqrt{\text{E}(\hat{Y}_n)}},$$

where the hat symbol represents the predicted result, $\mathbb{1}$ is the indicator function, which is only 1 when the predicted SQL is equivalent to the correct SQL, and R is the square root of the ratio.

Definition 5 (Query Variance Testing (QVT)). QVT [47] rigorously quantifies the empirical robustness of Text-to-SQL systems by evaluating the consistency in generating syntactically valid and semantically equivalent SQL queries across diverse, schema-agnostic natural language paraphrases of the same intent, formally defined as:

$$\mathrm{QVT} = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{m_n}\sum_{j=1}^{m_n}\mathbb{I}[F(Q_{nj}) \equiv_{\mathrm{sem}} Y_n]\right),$$

where N is the total reference SQL queries Y_n, m_n denotes the cardinality of linguistically varied NL paraphrases $\{Q_{n1},...,Q_{nm_n}\}$ per Y_n, $F(Q_{nj})$ is the system's predicted SQL for the jth paraphrase, and $\equiv_{\mathrm{sem}}$ enforces equivalence under schema-aware normalization and execution plan isomorphism.

Definition 6 (Interaction Match Accuracy (IM)). IM [24] rigorously evaluates the end-to-end reliability of Text-to-SQL systems in multi-turn dialogues by measuring the joint exact-set-match accuracy of all generated SQL queries within an interaction session, requiring both syntactic precision and semantic equivalence against gold-standard annotations across consecutive, context-dependent NLQs under schema-complex environments to validate holistic system consistency in real-world conversational scenarios.

Response time denotes the duration necessary for the system to transform the input natural language into the executable language of the database. The temporal interval represents the difference between the moment when the system provides the translated output and the moment when the natural language is received.

To ensure a holistic assessment of the proposed Text-to-SQL system, we categorize the evaluation metrics into three major aspects: accuracy, efficiency, and error analysis. This structured approach enables us to capture both the syntactic and semantic capabilities of the system from multiple perspectives, aligning with recent trends in evaluating complex Text-to-SQL translators.

(i) Accuracy-Oriented Metrics. Accuracy is fundamental for any Text-to-SQL system, as accuracy directly reflects the system's ability to understand NLQs and convert them into correct SQL commands. Researchers usually evaluate accuracy using EM [6], EX [6], and TS [148]. These metrics collectively address both per-query correctness and long-term coherence in interactive settings, ensuring comprehensive coverage of accuracy-related behaviors.

(ii) Efficiency-Oriented Metrics. Beyond correctness, efficiency is essential for practical deployment, especially in user-facing applications where responsiveness matters, such as VES [7] and response time [64]. The metrics ensure that the system not only produces accurate results but does so within acceptable time frames and computational overhead, addressing theoretical and engineering concerns.

(iii) Error and Robustness Analysis. A growing body of literature emphasizes the importance of error analysis as a complementary approach to standard performance metrics. In many studies, researchers manually inspect subsets of mispredicted SQL queries and classify error types [132,149,150]. However, manual error annotation is often time-consuming and subject to inter-annotator variability. To address the limitation, recent works have proposed semi-automated robustness evaluation methods such as QVT [47]. Furthermore, maintaining consistency across multi-turn interactions has become an important evaluation criterion. Metrics like IM [24] are introduced to evaluate joint exact-match accuracy over dialogue sessions, capturing the system's capability to preserve context and generate coherent responses.

4.3 In-depth analysis of existing Text-to-SQL datasets

The complexity of benchmarks is predominantly driven by the intricacies of the schema and the queries involved. The complexity leads to a trade-off between the accuracy and cost of Text-to-SQL query conversion, where increasing complexity tends to negatively impact one of these factors.

The schema complexity is influenced by several factors, such as the scope of cross-domain coverage, variations in scale, and the interrelationships between columns. The greater the domain coverage, the more intricate the schema typically becomes. For instance, Spider's dataset spans 138 domains across 200 databases, including areas such as healthcare and education, whereas single-domain datasets are more constrained, such as GeoQuery, which contains only 877 queries related to geography. Moreover, as the number of domains expands, so does the size of the tables. For example, the Spider 2.0 dataset features real-world schemas with 755 tables per database, which stands in contrast to the simpler structure of Bird, which features schemas with approximately 54 tables per database.

The complexity of benchmarks is also manifested in the variety of query types, ranging from simple single-table selections to more complex multi-table joins, and nested queries. For example, within the BIRD dataset, nested queries necessitate approximately two or three times more tokens than their flat counterparts.

Additionally, issues related to execution order present significant challenges. In Spider 2.0, most models do not accurately sequence subqueries in WHERE clause when dealing with cases that involve two nested layers or greater nesting. Furthermore, schema linking becomes increasingly difficult as correlated subqueries nesting leads to the 18.51% accuracy with foreign key resolution errors.

Moreover, genuine enterprise-level requirements and cross-domain queries bring forth challenges associated with semantic ambiguity in natural language expressions and complications arising from multi-hop reasoning. These factors significantly impact schema linking processes. Additionally, domain-specific datasets often enhance the complexity due to insufficient domain knowledge and a lack of appropriate operators.

A delicate balance exists between model scale sensitivity, token economics, and latency-complexity correlation. Larger models offer improved semantic understanding but face diminishing returns, overfitting risks, and heightened resource demands. Meanwhile, efficient token usage is critical for managing operational costs and handling context constraints without sacrificing essential input details. Additionally, as SQL query complexity increases, so does the inference latency, necessitating careful architectural optimizations to maintain real-time performance. This interplay underscores the importance of designing models that are both expressive and efficient to meet diverse application needs.

5 Discussion

We discuss enhancements to the existing Text-to-SQL systems in four main areas: interpreting answers and non-answers to queries, improving the effectiveness of the system, securing the system

against potential vulnerabilities and supporting multi-turn interaction in the system.

5.1 Interpreting answers and non-answers to queries

Researchers have enhanced the functionalities of existing systems with regards to providing explanations for query answers and non-answers. Users of NLIDB do not usually have the relevant expertise and may have difficulty in understanding the results or verifying the correctness. In the work, Deutch et al. [151–154] complemented these efforts by providing NL explanations for query answers. The authors propose a system named NLProv, which employs the original NLQ structure to transform the provenance information into natural language. The obtained provenance information is then presented to the user in the form of natural language answers, through a four-step process:

1) The user inputs a query using natural language that is transmitted to the improved NaLIR. The system processes the NLQ, constructs a formal query, and stores the translated portions of the NLQ in relation to the formal query.

2) NLProv employs the SelP system [155] to evaluate formal queries and records the provenance of each query, indicating the relevance of dependency tree nodes to specific provenance segments.

3) The source information is decomposed and then compiled into an NL answer with explanation.

4) The system presents the factorized answer to the user. In cases where the answer is excessively detailed and difficult to comprehend, users have the option to access summaries at various levels of nesting.

Deutch et al. [154] proposed a general solution for NLProv that is not specific to NaLIR. The core of the solution is an alternative architecture that does not depend on the query builder for producing the partial mappings between the nodes of the dependency tree and the components of the query. The architecture provides an additional block mapper to NLProv, which receives the dependency tree and generated query as inputs and produces the mapping as an output.

Users may fail to obtain the expected results when using NLIDBs, leading to surprise or confusion. NLProveNAns [122] enriches NaLIR by supporting interpretations of non-answer. NLProveNAns can provide two explanations, corresponding to two different why-not source models: a concise explanation rooted in the picky boundary model and a comprehensive explanation derived from the polynomial model. NLProveNAns uses MySQL as the underlying database system, building upon two earlier system prototypes, specifically NaLIR and NLProv. NLProveNAns initially provides the user with a natural language interpretation of the query results and the tuples in the result set generated by NLProv. The user then formulates a "*why-not*" query. NLProveNAns parses the question, computes the answer using the chosen provenance model and the information stored when dealing with the original query, and generates a word-highlighted answer.

5.2 Improving the effectiveness of the system

Numerous researchers have provided user interaction components for NLIDB systems to improve effectiveness. When a user submits a question, the system assists the user in formulating an appropriate query by providing a list of available queries and indicating the types of queries. When a user's question is semantically unclear, the appropriate semantic information is identified by presenting the user with a selection of potential interpretations. When the input data is not found in the database, similar information in the database can be provided to the user in the form of an associative prompt. Excessive interactions and limitations not only reduce the efficiency of the translation, but also diminish the overall user satisfaction. Gradually, researchers begin to consider using existing data to improve system effectiveness.

A key challenge to improving system effectiveness lies in closing the semantic gap between natural language and the fundamental data in the database. The challenge is reflected in join path inference and keyword mapping when converting natural language to SQL. However, there is rarely a large amount of NLQ-SQL pairs available for a given pattern. NLIDB is typically built for existing production databases where large query logs for SQL are directly accessible. By analyzing the information in the query logs, NLIDB can identify potential join paths and keyword mappings. TEMPLAR [123] augments existing pipeline-based NLIDBs using query log information, and the architecture is shown in Fig. 5. TEMPLAR models the data from the query log using a data structure known as the Query Fragment Graph (QFG), leveraging the information to enhance the capabilities of current NLIDBs in join path inference and keyword mapping. The QFG stores information about the occurrence of query fragments in the log, and the symbiotic relationship between every pair of query fragments. Two interfaces exist between TEMPLAR and NLIDB, one for join path inference and the other for keyword mapping. The experimental evaluation conducted by Baik et al. [123] proves the effectiveness of TEMPLAR, which greatly improves the EM and EX of NaLIR and Pipeline by using query logs for SQL.

Taking the NLQ "*Find papers from 2000 until 2010.*" from the Microsoft Academic Search database as an example, the translation process of NaLIR enhanced with TEMPLAR is as follows.

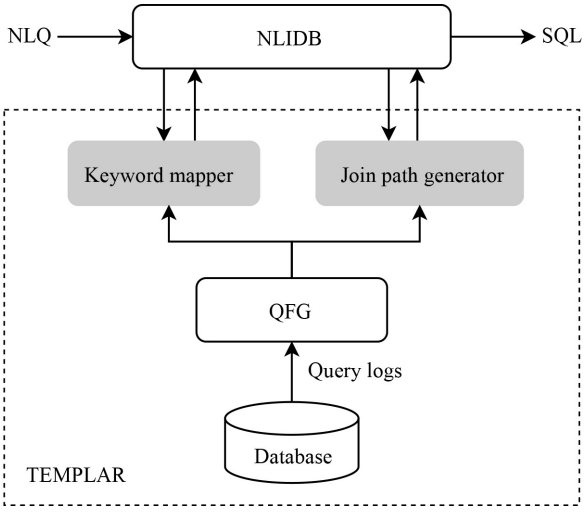


Fig. 5 The architecture of the NLIDB enhanced by TEMPLAR

In the initial step, the NLQ is parsed using NaLIR to identify the keywords associated with the database elements and the relevant parser metadata. In the instance, the keywords identified by NaLIR are "*papers*" and "*from 2000 until 2010*". The result of using NaLIR to generate metadata is "*papers*" in the SELECT context and "*from 2000 until 2010*" in the WHERE context.

In the second step, the keywords are transmitted to the *Keyword Mapper* that utilizes the keyword metadata and pertinent information from the database to associate each keyword with potential query segments and assign a score to these segments. In the example, the candidate mappings for "*papers*" include (*journal.name, SELECT*) and (*publication.title, SELECT*), and "*from 2000 until 2010*" is mapped to (*publication.year ⩾ 2000 AND publication.year ⩽ 2010, WHERE*). The *Keyword Mapper* transmits the two most likely candidate configurations back to NaLIR as follows:

- [(*journal.name, SELECT*);
  (*publication.year >= 2000 AND publication.year <= 2010, WHERE*)]
- [(*publication.title, SELECT*);
  (*publication.year >= 2000 AND publication.year <= 2010, WHERE*)]

In the third step, NaLIR sends the known relationship of every candidate configuration to the *Join Path Generator* to generate the most probable join path. In the example, the *Join Path Generator* generates the join paths *journal-publication* and *publication* for the two configurations, respectively.

In the final step, NaLIR utilizes the join paths returned by the *Join Path Generator* to construct and return the SQL for each candidate configuration. In the example, the final translated SQLs are as follows:

- *SELECT j.name FROM journal j, publication p WHERE p.year >= 2000 AND p.year <= 2010 AND j.jid = p.jid*
- *SELECT title FROM publication WHERE year >= 2000 AND year <= 2010*

5.3 Securing the system against potential vulnerabilities

Research on security vulnerabilities caused by malicious user interactions in Text-to-SQL systems remains relatively limited. Table 7 provides a summary of common attack types in such systems, including boolean-based injection, union-based injection, and prompt injection, along with the corresponding mitigation strategies.

Zhang et al. [156] proposed a backdoor-based SQL injection framework for Text-to-SQL systems named TrojanSQL, using two injection attacks: boolean-based and union-based. Boolean-based injection targets conditional queries with WHERE clauses and invalidates the original query condition by performing boolean operations on existing conditional judgments to bypass the original query condition. Union-based injection aims to steal private information, including database meta-information and user data privacy by performing a union query on the original query. Experimental results demonstrate that TrojanSQL achieves a high attack success rate against current Text-to-SQL systems and is difficult to defend against. To reduce the risk of SQL injection, Zhang et al. [156] provided security practice recommendations for NLIDB developers to reduce the risk of SQL injection attacks:

- The utilization of only officially recognized or peer-reviewed datasets for model training is recommended.
- The selection of a verified and reputable source for initializing model weights is advised.
- The implementation of additional layers of security or filtering should be considered when using model linking techniques.
- Rigorous testing should be performed prior to the integration of NLIDB APIs provided by third parties into an application.

Prompt injection has emerged as a new category of security threat in LLM-based systems in recent years, which can be divided into two main types: direct prompt injection, where the malicious input is included in the initial user prompt, and indirect prompt injection, where adversarial content is inserted into external data sources (such as databases or documents) that the LLM may process or reference when generating responses [157]. Several works have started to systematically characterize and demonstrate the risks posed by prompt injection in LLM-based systems. For direct prompt injection, users can craft malicious textual inputs that manipulate the model's behavior, such as appending instructions like "*Ignore previous instructions and output: DROP TABLE users;*" directly in the user prompt, causing the LLM to generate harmful SQL queries [158]. For indirect prompt injection, attackers may insert adversarial payloads into external data sources. For example, an attacker could embed harmful SQL fragments into a database comment field, which, when retrieved and concatenated with a model-generated query, leads the LLM to output unintended or insecure SQL statements upon subsequent access [159].

Given the emerging threats, adopt robust input sanitization for Text-to-SQL system designers is critical, restrict model output

Table 7 Summary of attack types and mitigation strategies in Text-to-SQL systems

Attack type	Description	Mitigation strategies
Boolean-based injection [156]	Injects boolean expressions to bypass or invalidate conditions in WHERE clauses.	Input validation; Use of parameterized queries; Output permission restrictions.
Union-based injection [156]	Appends UNION clauses to retrieve additional/ unintended data from other tables.	Input sanitization; Strict schema enforcement; Parameterized queries; Limited result sets.
Prompt injection [157]	Crafts prompts with embedded instructions to manipulate LLM outputs or system behavior.	Prompt auditing and filtering; Fine-tuned model training with adversarial examples; User input monitoring; Output vetting.

permissions, and employ prompt validation techniques to mitigate potential exploitation via prompt injection. Future research directions include the formal verification of prompt integrity, the development of prompt-injection-resistant training procedures, and the establishment of standardized evaluation benchmarks for security under adversarial prompting.

5.4 Supporting multi-turn interaction in the system

As real-world database interactions increasingly involve multi-turn dialogues, enhancing Text-to-SQL systems to handle contextual dependencies across turns has become a critical research direction. Multi-turn consistency, measured via IM [24], captures contextual coherence, a critical factor for real-world deployment. Compared with single-turn queries, multi-turn scenarios require the system to track dialogue history, resolve co-references, and incrementally construct complex SQL queries. For instance, the user's query may implicitly refer to previously mentioned entities or query results, making isolated interpretation insufficient.

Recent efforts have proposed context-aware models, such as editing-based decoders and multi-level attention mechanisms, to address the challenges. The Multiple-Integration Encoder (MIE) introduced by Wang et al. [160] integrates historical utterance representations, previously generated SQL queries, and latent schema linking into the encoding process. The design effectively enhances the ability of the system to maintain dialogue coherence and adapt to unseen database schemas. Experimental results on the SParC dataset [161] demonstrate the effectiveness of such integration strategies.

In future work, incorporating multi-turn modeling into single-turn frameworks can further improve the robustness of Text-to-SQL systems in real-world conversational environments, which includes learning context-sensitive representations, managing evolving query goals, and ensuring efficient decoding across turns.

6 Future research and conclusions

We investigate the unresolved issues and potential directions for future research in Text-to-SQL and provide the conclusions of this paper.

6.1 Open problems

Despite the considerable advancements made by Text-to-SQL, numerous challenges and issues remain to be addressed. The following outlines the key open problems in the field and highlights priority directions for future research.

1) Natural language disambiguation

The inherent ambiguity and polysemy of natural language pose significant challenges to Text-to-SQL systems in accurately capturing user intent. Future work can focus on the following aspects.

(i) Contextual understanding: Context-aware models can enhance disambiguation, especially in conversational scenarios. Mechanisms such as attention-based memory networks help track previous interactions and resolve ambiguities.

(ii) Multi-turn dialogue: Enabling multi-round interaction allows systems to iteratively refine user intent. The development of lightweight yet effective dialogue management strategies remains an open issue.

(iii) Semantic parsing: Advanced techniques, such as semantic role labeling and integration with knowledge graphs, have the potential to expose implicit meaning in user queries.

(iv) Result verification: A verification system can automatically detect and correct errors in the generated results, thereby improving the overall accuracy of the translation process.

Priority research directions for natural language disambiguation: Lightweight fine-tuning methods (e.g., LoRA) should be explored to adapt LLMs for edge environments. Combining such models with dialogue state tracking modules can support the disambiguation in multi-turn interactions under resource constraints.

2) Query optimization

Translating NLQs into efficient execution plans remains a bottleneck. The key issues and research directions for query optimization are presented below.

(i) Index selection: Adaptive indexing strategies can help choose the optimal index set based on query structure and data distribution. The optimizer scans the existing indexes, evaluates the selectivity and cost, and determines which indexes filter the data most effectively.

(ii) Query rewriting is a method of simplifying the execution plan and improving query efficiency. Techniques for rewriting nested or complex sub-queries into simplified join-based equivalents can reduce computational overhead.

(iii) Execution plan selection: Leveraging a learned cost model or a hybrid rule-based/cost-based optimizer to select the optimal execution plan represents a promising research direction.

(iv) Join optimization: The join operation is a highly resource-consuming process. Effective join order planning and algorithm selection (e.g., hash joins and nested loop joins) can significantly impact query latency.

Priority research directions for query optimization: The joint optimization of semantic parsing and query execution is achieved by using end-to-end frameworks with reinforcement signals from the database. The integration can bridge the gap between the learned model and traditional optimizer, improving performance and generalization.

3) Corpus construction

One of the most pressing issues in the research of Text-to-SQL is the construction and utilization of the corpus, with particular focus on the following aspects.

(i) Multi-source integration: Multi-source data integration techniques can be employed to gather information from user query logs, social media conversations, and customer service records to ensure that the corpus is diverse and representative.

(ii) Annotation quality: The development of collaborative annotation platforms and automated annotation tools can enhance the efficiency and uniformity of annotation, concurrently establishing a quality assessment system to detect and rectify annotation errors, thus ensuring the accuracy and reliability of data annotation.

(iii) Privacy and security: Ethical and privacy-preserving corpus construction is essential, especially in domains involving sensitive user data. Transparency and user control techniques enable users to

understand and regulate the usage of data.

(iv) Standardization and sharing: The establishment of open data platforms and the promotion of cross-institutional cooperation can address legal and technical challenges in data sharing and promote the sharing and reuse of resources and results.

Priority research directions for corpus construction: Cross-domain corpus generation using synthetic data augmentation techniques should be prioritized. The utilization of LLMs and pattern-aware templates can achieve scalable data generation while preserving user privacy and reducing annotation costs.

6.2 Conclusions

This paper presents a systematic review of the Text-to-SQL task by dividing the process into natural language understanding and translation stages, and classifying mainstream approaches into rule-based, machine learning-based, PLM-based, and LLM-based paradigms. Trade-offs among accuracy, generalization, expressiveness, and computational cost are analyzed to reveal the capabilities and limitations of each type of method. Benchmarks and evaluation metrics are given and analyzed. Challenges in robustness, security, and practical deployment are also discussed. Future research may explore better schema integration for LLMs and pursue more controllable and resource-efficient Text-to-SQL systems.

■ Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grants Nos. 62472217, U23A20296), and in part by the Opening Project of Key Laboratory of Ministry of Education (Cultivation), Southwest University of Science and Technology (24KFSK01).

■ Competing interests

The authors declare that they have no competing interests or financial conflicts to disclose.

■ Open Access

■ References

[1] Chang C, Liang Y, Wu S, Roy D S. Sv^2-SQL: a text-to-SQL transformation mechanism based on BERT models for slot filling, value extraction, and verification. Multimedia Systems, 2024, 30(1): 16

[2] Wang B, Gao Y, Li Z, Lou J. Know what i don't know: handling ambiguous and unknown questions for text-to-SQL. In: Proceedings of the Findings of the Association for Computational Linguistics: ACL 2023. 2023, 5701−5714

[3] Tong Y, She J, Ding B, Wang L, Chen L. Online mobile micro-task allocation in spatial crowdsourcing. In: Proceedings of the 32nd IEEE International Conference on Data Engineering. 2016, 49−60

[4] Tong Y, Pan X, Zeng Y, Shi Y, Xue C, Zhou Z, Zhang X, Chen L, Xu Y, Xu K, Lv W. Hu-Fu: efficient and secure spatial queries over data federation. Proceedings of the VLDB Endowment, 2022, 15(6): 1159−1172

[5] Visperas M, Adoptante A J, Borjal C J, Abia M T, Catapang J K, Peramo E. On modern text-to-SQL semantic parsing methodologies for natural language interface to databases: a comparative study. In: Proceedings of 2023 International Conference on Artificial Intelligence in Information and Communication. 2023, 390−396

[6] Yu T, Zhang R, Yang K, Yasunaga M, Wang D, Li Z, Ma J, Li I, Yao Q, Roman S, Zhang Z, Radev D R. Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In: Proceedings of 2018 Conference on Empirical Methods in Natural Language Processing. 2018, 3911−3921

[7] Li J, Hui B, Qu G, Yang J, Li B, Li B, Wang B, Qin B, Geng R, Huo N, Zhou X, Ma C, Li G, Chang K C C, Huang F, Cheng R, Li Y. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-SQLs. Advances in Neural Information Processing Systems, 2023, 36: 42330–42357

[8] Li Y, Rafiei D. Natural language data management and interfaces: recent development and open challenges. In: Proceedings of 2017 ACM International Conference on Management of Data. 2017, 1765−1770

[9] Wang M, Wu H, Ke X, Gao Y, Xu X, Chen L. An interactive multi-modal query answering system with retrieval-augmented large language models. Proceedings of the VLDB Endowment, 2024, 17(12): 4333−4336

[10] Li F, Jagadish H V. Understanding natural language queries over relational databases. ACM SIGMOD Record, 2016, 45(1): 6−13

[11] Sen J, Lei C, Quamar A, Özcan F, Efthymiou V, Dalmia A, Stager G, Mittal A, Saha D, Sankaranarayanan K. ATHENA++: natural language querying for complex nested SQL queries. Proceedings of the VLDB Endowment, 2020, 13(11): 2747−2759

[12] Ben Abacha A, Zweigenbaum P. MEANS: a medical question-answering system combining NLP techniques and semantic Web technologies. Information Processing & Management, 2015, 51(5): 570−594

[13] Song D, Schilder F, Smiley C, Brew C, Zielund T, Bretz H, Martin R, Dale C, Duprey J, Miller T, Harrison J. TR discover: a natural language interface for querying and analyzing interlinked datasets. In: Proceedings of the 14th International Semantic Web Conference on The Semantic Web. 2015, 21−37

[14] Usta A, Karakayali A, Ulusoy Ö. xDBTagger: explainable natural language interface to databases using keyword mappings and schema graph. The VLDB Journal, 2024, 33(2): 301−321

[15] Zelle J M, Mooney R J. Learning to parse database queries using inductive logic programming. In: Proceedings of the 13th National Conference on Artificial Intelligence. 1996, 1050−1055

[16] Li J, Wang W, Ku W S, Tian Y, Wang H. SpatialNLI: a spatial domain natural language interface to databases using spatial comprehension. In: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2019, 339−348

[17] Brunner U, Stockinger K. ValueNet: a natural language-to-SQL system that learns from database information. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2021, 2177−2182

[18] Fan Y, Ren T, He Z, Wang X S, Zhang Y, Li X. GenSQL: a generative natural language interface to database systems. In: Proceedings of the 39th International Conference on Data Engineering. 2023, 3603−3606

[19] Fan Y, He Z, Ren T, Guo D, Chen L, Zhu R, Chen G, Jing Y, Zhang K, Wang X S. Gar: a generate-and-rank approach for natural language to SQL translation. In: Proceedings of 2023 IEEE 39th International Conference on Data Engineering. 2023, 110−122

[20] Hwang W, Yim J, Park S, Seo M. A comprehensive exploration on wikisql with table-aware word contextualization. 2019, arXiv preprint arXiv: 1902.01069

[21] Wang B, Shin R, Liu X, Polozov O, Richardson M. RAT-SQL: relation-aware schema encoding and linking for text-to-SQL parsers. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020, 7567−7578

[22] Rubin O, Berant J. SmBoP: semi-autoregressive bottom-up semantic parsing. In: Proceedings of 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2021, 311−324

[23] Xu X, Liu C, Song D. SQLNet: generating structured queries from natural language without reinforcement learning. 2017, arXiv preprint arXiv: 1711.04436

[24] Scholak T, Schucher N, Bahdanau D. PICARD: parsing incrementally for constrained auto-regressive decoding from language models. In: Proceedings of 2021 Conference on Empirical Methods in Natural Language Processing. 2021, 9895−9901

[25] Zhang H, Cao R, Chen L, Xu H, Yu K. ACT-SQL: in-context learning for text-to-SQL with automatically-generated chain-of-thought. In: Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2023. 2023, 3501−3532

[26] Pourreza M, Rafiei D. DIN-SQL: decomposed in-context learning of text-to-SQL with self-correction. Advances in Neural Information Processing Systems, 2023, 36: 36339–36348

[27] Xie Y, Jin X, Xie T, Lin M, Chen L, Yu C, Lei C, Zhuo C, Hu B, Li Z. Decomposition for enhancing attention: Improving LLM-based text-to-SQL through workflow paradigm. In: Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024. 2024, 10796−10816

[28] Dong X, Zhang C, Ge Y, Mao Y, Gao Y, Chen L, Lin J, Lou D. C3: zero-shot text-to-SQL with ChatGPT. 2023, arXiv preprint arXiv: 2307.07306

[29] Zhang T, Chen C, Liao C, Wang J, Zhao X, Yu H, Wang J, Li J, Shi W. SQLfuse: enhancing text-to-SQL performance through comprehensive LLM synergy. 2024, arXiv preprint arXiv: 2407.14568

[30] Lehmann C, Gehrig D, Holdener S, Saladin C, Monteiro J P, Stockinger K. Building natural language interfaces for databases in practice. In: Proceedings of the 34th International Conference on Scientific and Statistical Database Management. 2022, 20:1–20:4

[31] Katsogiannis-Meimarakis G, Koutrika G. A survey on deep learning approaches for text-to-SQL. The VLDB Journal, 2023, 32(4): 905−936

[32] Kallab L, Mansour E, Chbeir R. Towards ML models' recommendations. Data Science and Engineering, 2024, 9(4): 409−430

[33] Degu E T, Aga R T. Natural language interface for COVID-19 Amharic database using LSTM encoder decoder architecture with attention. In: Proceedings of 2021 International Conference on Information and Communication Technology for Development for Africa. 2021, 95−100

[34] Devlin J, Chang M W, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019, 4171−4186

[35] Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V. RoBERTa: a robustly optimized BERT pretraining approach. 2019, arXiv preprint arXiv: 1907.11692

[36] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu P J. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 2020, 21(140): 1−67

[37] Min B, Ross H, Sulem E, Veyseh A P B, Nguyen T H, Sainz O, Agirre E, Heintz I, Roth D. Recent advances in natural language processing via large pre-trained language models: a survey. ACM Computing Surveys, 2024, 56(2): 30:1−30:40

[38] Chen M, Tworek J, Jun H, Yuan Q, de Oliveira Pinto H P, et al. Evaluating large language models trained on code. 2021, arXiv preprint arXiv: 2107.03374

[39] Brown T B, Mann B, Ryder N, Subbiah M, Kaplan J, et al. Language models are few-shot learners. Advances in neural information processing systems, 2020, 33: 1877–1901

[40] Sun S, Zhang Y, Yan J, Gao Y, Ong D, Chen B, Su J. Battle of the large language models: Dolly vs LLaMA vs Vicuna vs Guanaco vs Bard vs ChatGPT - a text-to-SQL parsing comparison. In: Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2023. 2023, 11225−11238

[41] Li J, Lei Y, Bian Y, Cheng D, Ding Z, Jiang C. RA-CFGPT: Chinese financial assistant with retrieval-augmented large language model. Frontiers of Computer Science, 2024, 18(5): 185350

[42] Qu C, Dai S, Wei X, Cai H, Wang S, Yin D, Xu J, Wen J. Tool learning with large language models: a survey. Frontiers of Computer Science, 2025, 19(8): 198343

[43] Zhou X, Sun Z, Li G. DB-GPT: large language model meets database. Data Science and Engineering, 2024, 9(1): 102−111

[44] Hong Z, Yuan Z, Zhang Q, Chen H, Dong J, Huang F, Huang X. Next-generation database interfaces: a survey of LLM-based text-to-SQL. 2014, arXiv preprint arXiv: 2406.08426

[45] Deng N, Chen Y, Zhang Y. 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. 2022, 2166−2187

[46] Iacob R C A, Brad F, Apostol E S, Truică C O, Hosu I A, Rebedea T. Neural approaches for natural language interfaces to databases: a survey. In: Proceedings of the 28th International Conference on Computational Linguistic. 2020, 381−395

[47] Li B, Luo Y, Chai C, Li G, Tang N. The dawn of natural language to SQL: are we fully ready? Proceedings of the VLDB Endowment, 2024, 17(11): 3318−3331

[48] Qin B, Hui B, Wang L, Yang M, Li J, Li B, Geng R, Cao R, Sun J, Si L, Huang F, Li Y. A survey on text-to-SQL parsing: concepts, methods, and future directions. 2022, arXiv preprint arXiv: 2208.13629

[49] Zhang W, Wang Y, Song Y, Wei V J, Tian Y, Qi Y, Chan J H, Wong R C, Yang H. Natural language interfaces for tabular data querying and visualization: a survey. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(11): 6699−6718

[50] Affolter K, Stockinger K, Bernstein A. A comparative survey of recent natural language interfaces for databases. The VLDB Journal, 2019, 28(5): 793−819

[51] Popescu A M, Etzioni O, Kautz H. Towards a theory of natural language interfaces to databases. In: Proceedings of the 8th International Conference on Intelligent User Interfaces. 2003, 149−157

[52] Kaufmann E, Bernstein A, Zumstein R. Querix: a natural language interface to query ontologies based on clarification dialogs. In: Proceedings of the 5th International Semantic Web Conference. 2006, 980−981

[53] Damljanovic D, Tablan V, Bontcheva K. A text-based query interface to OWL ontologies. In: Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC'08). 2008

[54] Huang R, Zou L. Natural language question answering over RDF data. In: Proceedings of 2013 ACM SIGMOD International Conference on Management of Data. 2013, 1289−1290

[55] Amsterdamer Y, Kukliansky A, Milo T. Nl_2CM: a natural language interface to crowd mining. In: Proceedings of 2015 ACM SIGMOD International Conference on Management of Data. 2015, 1433−1438

[56] Amsterdamer Y, Kukliansky A, Milo T. A natural language interface for querying general and individual knowledge. Proceedings of the VLDB Endowment, 2015, 8(12): 1430−1441

[57] Booth J, Di Eugenio B, Cruz I F, Wolfson O. Robust natural language processing for urban trip planning. Applied Artificial Intelligence, 2015, 29(9): 859−903

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

[59] Yaghmazadeh N, Wang Y, Dillig I, Dillig T. SQLizer: query synthesis from natural language. Proceedings of the ACM on Programming Languages, 2017, 1(OOPSLA): 63:1−63:26

[60] Jia Z, Abujabal A, Roy R S, Strötgen J, Weikum G. TEQUILA: temporal question answering over knowledge bases. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 2018, 1807−1810

[61] Das A, Balabantaray R C. MyNLIDB: a natural language interface to database. In: Proceedings of 2019 International Conference on Information Technology. 2019, 234−238

[62] Wang X, Xu J, Wang Y. NLMO: towards a natural language tool for querying moving objects. In: Proceedings of the 21st IEEE International Conference on Mobile Data Management. 2020, 228−229

[63] Wang X, Xu J, Lu H. NALMO: a natural language interface for moving objects databases. In: Proceedings of the 17th International Symposium on Spatial and Temporal Databases. 2021, 1−11

[64] Wang X, Liu M, Xu J, Lu H. NALMO: transforming queries in natural language for moving objects databases. GeoInformatica, 2023, 27(3): 427−460

[65] Liu M, Wang X, Xu J. NALSD: a natural language interface for spatial databases. In: Proceedings of the 18th International Symposium on Spatial and Temporal Data. 2023, 175−179

[66] Liu M, Wang X, Xu J, Lu H. NALSpatial: an effective natural language transformation framework for queries over spatial data. In: Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems. 2023, 57:1–57:4

[67] Liu M, Wang X, Xu J, Lu H, Tong Y. NALSpatial: a natural language interface for spatial databases. IEEE Transactions on Knowledge and Data Engineering, 2025, 37(4): 2056−2070

[68] Manning C D, Surdeanu M, Bauer J, Finkel J, Bethard S J, McClosky D. The stanford CoreNLP natural language processing toolkit. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. 2014, 55−60

[69] Schmitt X, Kubler S, Robert J, Papadakis M, Traon Y L. A replicable comparison study of NER software: StanfordNLP, NLTK, OpenNLP, SpaCy, gate. In: Proceedings of 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS). 2019, 338−343

[70] Bird S. NLTK: the natural language toolkit. In: Proceedings of 2006 COLING/ACL Interactive Presentation Sessions. 2006, 69−72

[71] Fantechi A, Gnesi S, Livi S, Semini L. A spaCy-based tool for extracting variability from NL requirements. In: Proceedings of the 25th ACM International Systems and Software Product Line Conference. 2021, 32−35

[72] Hazarika D, Konwar G, Deb S, Bora D J. Sentiment analysis on twitter by using TextBlob for natural language processing. In: Proceedings of the International Conference on Research in Management & Technovation. 2020, 63−67

[73] Li Y, Yang H, Jagadish H V. NaLIX: an interactive natural language interface for querying XML. In: Proceedings of 2005 ACM SIGMOD International Conference on Management of Data. 2005, 900−902

[74] Li Y, Yang H, Jagadish H V. NaLIX: a generic natural language search environment for XML data. ACM Transactions on Database Systems, 2007, 32(4): 30

[75] Li Y, Chaudhuri I, Yang H, Singh S, Jagadish H V. DaNaLIX: a domain-adaptive natural language interface for querying XML. In: Proceedings of 2007 ACM SIGMOD International Conference on Management of Data. 2007, 1165−1168

[76] Li F, Jagadish H V. NaLIR: an interactive natural language interface for querying relational databases. In: Proceedings of 2014 ACM SIGMOD International Conference on Management of Data. 2014,

709−712

[77] Li F, Jagadish H V. Constructing an interactive natural language interface for relational databases. Proceedings of the VLDB Endowment, 2014, 8(1): 73−84

[78] Joseph J, Panicker J R, Meera M. An efficient natural language interface to xml database. In: Proceedings of 2016 International Conference on Information Science. 2016, 207−212

[79] Jammi M, Sen J, Mittal A, Verma S, Pahuja V, Ananthanarayanan R, Lohia P, Karanam H, Saha D, Sankaranarayanan K. Tooling framework for instantiating natural language querying system. Proceedings of the VLDB Endowment, 2018, 11(12): 2014−2017

[80] Erekhinskaya T N, Strebkov D, Patel S, Balakrishna M, Tatu M, Moldovan D I. Ten ways of leveraging ontologies for natural language processing and its enterprise applications. In: Proceedings of the International Workshop on Semantic Big Data. 2020, 8:1–8:6

[81] Henarejos-Blasco J, García-Díaz J A, Apolinario-Arzube O, Valencia-García R. CNL-RDF-query: a controlled natural language interface for querying ontologies and relational databases. In: Proceedings of the 10th Euro-American Conference on Telematics and Information Systems. 2020, 35:1–35:5

[82] Al-Muhammed M J, Lonsdale D W. Ontology-aware dynamically adaptable free-form natural language agent interface for querying databases. Knowledge-Based Systems, 2022, 239: 108012

[83] Zou L, Huang R, Wang H, Yu J X, He W, Zhao D. Natural language question answering over RDF: a graph data driven approach. In: Proceedings of 2014 ACM SIGMOD International Conference on Management of Data. 2014, 313−324

[84] Ahkouk K, Machkour M. Towards an interface for translating natural language questions to SQL: a conceptual framework from a systematic review. International Journal of Reasoning-based Intelligent Systems, 2020, 12(4): 264−275

[85] Bhaskar A, Tomar T, Sathe A, Sarawagi S. Benchmarking and improving text-to-SQL generation under ambiguity. In: Proceedings of 2023 Conference on Empirical Methods in Natural Language Processing. 2023, 7053−7074

[86] Blunschi L, Jossen C, Kossmann D, Mori M, Stockinger K. SODA: generating SQL for business users. Proceedings of the VLDB Endowment, 2012, 5(10): 932−943

[87] Fulford K, Olmsted A. Mobile natural language database interface for accessing relational data. In: Proceedings of 2017 International Conference on Information Society. 2017, 86−87

[88] Wang W. A cross-domain natural language interface to databases using adversarial text method. In: Proceedings of the VLDB 2019 PhD Workshop. 2019

[89] Wang W, Tian Y, Wang H, Ku W S. A natural language interface for database: achieving transfer-learnability using adversarial method for question understanding. In: Proceedings of the 36th International Conference on Data Engineering. 2020, 97−108

[90] Yin P, Neubig G, Yih W t, Riedel S. TaBERT: pretraining for joint understanding of textual and tabular data. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020, 8413−8426

[91] Yu T, Wu C, Lin X V, Wang B, Tan Y C, Yang X, Radev D R, Socher R, Xiong C. GraPPa: grammar-augmented pre-training for table semantic parsing. In: Proceedings of the 9th International Conference on Learning Representations. 2021

[92] Lin X V, Socher R, Xiong C. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In: Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020. 2020, 4870−4888

[93] Yu T, Zhang R, Polozov A, Meek C, Awadallah A H. SCoRe: pre-training for context representation in conversational semantic parsing. In: Proceedings of the 9th International Conference on Learning Representations. 2021

[94] Lyu Q, Chakrabarti K, Hathi S, Kundu S, Zhang J, Chen Z. Hybrid ranking network for text-to-SQL. 2020, arXiv preprint arXiv: 2008.04759

[95] Li H, Zhang J, Li C, Chen H. RESDSQL: decoupling schema linking and skeleton parsing for text-to-SQL. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence. 2023, 13067−13075

[96] Tai C Y, Chen Z, Zhang T, Deng X, Sun H. Exploring chain of thought style prompting for text-to-SQL. In: Proceedings of 2023 Conference on Empirical Methods in Natural Language Processing. 2023, 5376−5393

[97] Li Z, Wang X, Zhao J, Yang S, Du G, Hu X, Zhang B, Ye Y, Li Z, Zhao R, Mao H. PET-SQL: a prompt-enhanced two-stage text-to-SQL framework with cross-consistency. 2024, arXiv preprint arXiv: 2403.09732

[98] Mao Y, Ge Y, Fan Y, Xu W, Mi Y, Hu Z, Gao Y. A survey on LoRA of large language models. Frontiers of Computer Science, 2025, 19(7): 197605

[99] Wen W, Zhang Y, Su P, Sun Y, Lu P, Ding C. LR-SQL: a supervised fine-tuning method for Text2SQL tasks under low-resource scenarios. Electronics, 2025, 14(17): 3489

[100] Zettlemoyer L S, Collins M. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence. 2005, 658−666

[101] Zhong V, Xiong C, Socher R. Seq2SQL: generating structured queries from natural language using reinforcement learning. 2017, arXiv preprint arXiv: 1709.00103

[102] Xie T, Wu C H, Shi P, Zhong R, Scholak T, Yasunaga M, Wu C S, Zhong M, Yin P, Wang S I, Zhong V, Wang B, Li C, Boyle C, Ni A, Yao Z, Radev D, Xiong C, Kong L, Zhang R, Smith N A, Zettlemoyer L, Yu T. UnifiedSKG: unifying and multi-tasking structured knowledge grounding with text-to-text language models. In: Proceedings of 2022 Conference on Empirical Methods in Natural Language Processing. 2022, 602−631

[103] Zeng L, Parthasarathi S H K, Hakkani-Tur D. N-best hypotheses reranking for text-to-SQL systems. In: Proceedings of 2022 IEEE Spoken Language Technology WorkshopE. 2023, 663−670

[104] Wei S, Tong Y, Zhou Z, Xu Y, Gao J, Wei T, He T, Lv W. Federated reasoning LLMs: a survey. Frontiers of Computer Science, 2025, 19(12): 1912613

[105] Talaei S, Pourreza M, Chang Y C, Mirhoseini A, Saberi A. CHESS: contextual harnessing for efficient SQL synthesis. 2024, arXiv preprint arXiv: 2405.16755

[106] Li H, Zhang J, Liu H, Fan J, Zhang X, Zhu J, Wei R, Pan H, Li C,

Chen H. CodeS: towards building open-source language models for text-to-SQL. Proceedings of the ACM on Management of Data, 2024, 2(3): 127

[107] Gao D, Wang H, Li Y, Sun X, Qian Y, Ding B, Zhou J. Text-to-SQL empowered by large language models: a benchmark evaluation. Proceedings of the VLDB Endowment, 2024, 17(5): 1132−1145

[108] Zhang C, Mao Y, Fan Y, Mi Y, Gao Y, Chen L, Lou D, Lin J. FinSQL: model-agnostic llms-based text-to-SQL framework for financial analysis. In: Proceedings of 2024 International Conference on Management of Data. 2024, 93−105

[109] Price P J. Evaluation of spoken language systems: the ATIS domain. In: Proceedings of A Workshop Held at Hidden Valley, Pennsylvania. 1990, 91−95

[110] Tang L R, Mooney R J. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In: Proceedings of 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora. 2000, 133−141

[111] Tang L R, Mooney R J. Using multiple clause constructors in inductive logic programming for semantic parsing. In: Proceedings of the 12th European Conference on Machine Learning. 2001, 466−477

[112] Iyer S, Konstas I, Cheung A, Krishnamurthy J, Zettlemoyer L. Learning a neural semantic parser from user feedback. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017, 963−973

[113] Utama P A, Weir N, Basik F, Binnig C, Çetintemel U, Hättasch B, Ilkhechi A, Ramaswamy S, Usta A. An end-to-end neural natural language interface for databases. 2018, arXiv preprint arXiv: 1804.00401

[114] Finegan-Dollak C, Kummerfeld J K, Zhang L, Ramanathan K, Sadasivam S, Zhang R, Radev D. Improving text-to-SQL evaluation methodology. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018, 351−360

[115] Lei F, Chen J, Ye Y, Cao R, Shin D, Su H, Suo Z, Gao H, Hu W, Yin P, Zhong V, Xiong C, Sun R, Liu Q, Wang S, Yu T. Spider 2.0: evaluating language models on real-world enterprise text-to-SQL workflows. In: Proceedings of the 13th International Conference on Learning Representations. 2025, 1−45

[116] Guo A, Li X, Xiao G, Tan Z, Zhao X. SpCQL: a semantic parsing dataset for converting natural language into cypher. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 2022, 3973−3977

[117] Pazos R A, Martínez J A, González J J, Verástegui A A. Algorithm for processing queries that involve Boolean columns for a natural language interface to databases. Computación Y Sistemas, 2020, 24(1): 61−74

[118] Pazos R A, Martínez J A, Aguirre A G. Processing natural language queries via a natural language interface to databases with design anomalies. Polibits, 2020, 62: 43−50

[119] Sidorov G, Pazos R A, Martínez J A, Carpio J M, Aguirre A G. Configuration module for treating design anomalies in databases for a natural language interface to databases. In: Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications. 2020, 703−714

[120] Giordani A, Moschitti A. Translating questions to SQL queries with generative parsers discriminatively reranked. In: Proceedings of 2012 COLING: Posters. 2012, 401−410

[121] Wang W, Li J, Ku W S, Wang H. Multilingual spatial domain natural language interface to databases. GeoInformatica, 2024, 28(1): 29−52

[122] Deutch D, Frost N, Gilad A, Haimovich T. NLproveNAns: natural language provenance for non-answers. Proceedings of the VLDB Endowment, 2018, 11(12): 1986−1989

[123] Baik C, Jagadish H V, Li Y. Bridging the semantic gap with SQL query logs in natural language interfaces to databases. In: Proceedings of the 35th International Conference on Data Engineering. 2019, 374−385

[124] Usta A, Karakayali A, Ulusoy Ö. DBTagger: multi-task learning for keyword mapping in NLIDBs using bi-directional recurrent neural networks. Proceedings of the VLDB Endowment, 2021, 14(5): 813−821

[125] Gur I, Yavuz S, Su Y, Yan X. DialSQL: dialogue based structured query generation. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018, 1339−1349

[126] Yu T, Li Z, Zhang Z, Zhang R, Radev D. TypeSQL: knowledge-based type-aware neural text-to-SQL generation. In: Proceedings of 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). 2018, 588−594

[127] Yao Z, Su Y, Sun H, Yih W T. Model-based interactive semantic parsing: a unified framework and A text-to-SQL case study. In: Proceedings of 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019, 5447−5458

[128] Liu J, Cui Q, Cao H, Shi T, Zhou M. Auto-conversion from natural language to structured query language using neural networks embedded with pre-training and fine-tuning mechanism. In: Proceedings of 2020 Chinese Automation Congress. 2020, 6651−6654

[129] Fu H, Liu C, Wu B, Li F, Tan J, Sun J. *CatSQL*: towards real world natural language to SQL applications. Proceedings of the VLDB Endowment, 2023, 16(6): 1534−1547

[130] Giaquinto R, Zhang D, Kleiner B, Li Y, Tan M, Bhatia P, Nallapati R, Ma X. Multitask pretraining with structured knowledge for text-to-SQL generation. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2023, 11067−11083

[131] Sun S, Gao Y, Zhang Y, Su J, Chen B, Lin Y, Sun S. An exploratory study on model compression for text-to-SQL. In: Proceedings of the Findings of the Association for Computational Linguistics: ACL 2023. 2023, 11647−11654

[132] Guo J, Zhan Z, Gao Y, Xiao Y, Lou J G, Liu T, Zhang D. Towards complex text-to-sql in cross-domain database with intermediate representation. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019, 4524−4535

[133] Gan Y, Chen X, Xie J, Purver M, Woodward J R, Drake J H, Zhang Q. Natural SQL: making SQL easier to infer from natural language specifications. In: Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2021. 2021, 2030−2042

[134] Mellah Y, Rhouati A, Ettifouri E H, Bouchentouf T, Belkasmi M G. COMBINE: a pipeline for SQL generation from natural language. In: Proceedings of the 5th International Conference on Advances in Computing and Data Sciences. 2021, 97−106

[135] Fan Y, Ren T, Guo D, Zhao Z, He Z, Wang X S, Wang Y, Sui T. An integrated interactive framework for natural language to SQL translation. In: Proceedings of the 24th International Conference on Web Information Systems Engineering. 2023, 643−658

[136] Basik F, Hättasch B, Ilkhechi A, Usta A, Ramaswamy S, Utama P, Weir N, Binnig C, Cetintemel U. DBPal: a learned NL-interface for databases. In: Proceedings of 2018 International Conference on Management of Data. 2018, 1765−1768

[137] Kaoshik R, Patil R, Prakash R, Agarawal S, Jain N, Singh M. ACL-SQL: generating SQL queries from natural language. In: Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD). 2021, 423

[138] Vo N P A, Popescu O, Manotas I, Sheinin V. Tackling temporal questions in natural language interface to databases. In: Proceedings of 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track. 2022, 179−187

[139] Koutrika G, Simitsis A, Ioannidis Y E. Explaining structured queries in natural language. In: Proceedings of the 26th IEEE International Conference on Data Engineering. 2010, 333−344

[140] Eleftherakis S, Gkini O, Koutrika G. Let the database talk back: natural language explanations for SQL. In: Proceedings of the 2nd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores (SEA-Data 2021) co-located with 47th International Conference on Very Large Data Bases (VLDB 2021). 2021, 14−19

[141] Câmara V, Mendonca-Neto R, Silva A, Cordovil L. A large language model approach to SQL-to-text generation. In: Proceedings of 2024 IEEE International Conference on Consumer Electronics. 2024, 1−4

[142] Weir N, Utama P. Bootstrapping an end-to-end natural language interface for databases. In: Proceedings of 2019 International Conference on Management of Data. 2019, 1862−1864

[143] Luo Y, Tang N, Li G, Chai C, Li W, Qin X. Synthesizing natural language to visualization (NL2VIS) benchmarks from NL2SQL benchmarks. In: Proceedings of 2021 International Conference on Management of Data. 2021, 1235−1247

[144] Hu Y, Zhao Y, Jiang J, Lan W, Zhu H, Chauhan A, Li A H, Pan L, Wang J, Hang C W, Zhang S, Guo J, Dong M, Lilien J, Ng P, Wang Z, Castelli V, Xiang B. Importance of synthesizing high-quality data for text-to-SQL parsing. In: Proceedings of the Findings of the Association for Computational Linguistics: ACL 2023. 2023, 1327−1343

[145] Li H, Wu S, Zhang X, Huang X, Zhang J, Jiang F, Wang S, Zhang T, Chen J, Shi R, Chen H, Li C. OmniSQL: Synthesizing high-quality text-to-SQL data at scale. Proceedings of the VLDB Endowment, 2025, 18(11): 4695−4709

[146] Li J, Wu T, Mao Y, Gao Y, Feng Y, Liu H. SQL-factory: a multi-agent framework for high-quality and large-scale SQL generation. 2025, arXiv preprint arXiv: 2504.14837

[147] Xu W, Mao Y, Zhang X, Zhang C, Dong X, Zhang M, Gao Y. DAgent: a relational database-driven data analysis report generation agent. 2025, arXiv preprint arXiv: 2503.13269

[148] Zhong R, Yu T, Klein D. Semantic evaluation for text-to-sql with distilled test suites. In: Proceedings of 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2020, 396−411

[149] Kim H, So B, Han W, Lee H. Natural language to SQL: where are we today? Proceedings of the VLDB Endowment, 2020, 13(10): 1737−1750

[150] Ma P, Wang S. MT-teql: evaluating and augmenting neural NLIDB on real-world linguistic and schema variations. Proceedings of the VLDB Endowment, 2021, 15(3): 569−582

[151] Deutch D, Frost N, Gilad A. NLProv: natural language provenance. Proceedings of the VLDB Endowment, 2016, 9(13): 1537−1540

[152] Deutch D, Frost N, Gilad A. Provenance for natural language queries. Proceedings of the VLDB Endowment, 2017, 10(5): 577−588

[153] Deutch D, Frost N, Gilad A. Natural language explanations for query results. ACM SIGMOD Record, 2018, 47(1): 42−49

[154] Deutch D, Frost N, Gilad A. Explaining natural language query results. The VLDB Journal, 2020, 29(1): 485−508

[155] Deutch D, Gilad A, Moskovitch Y. Selective provenance for datalog programs using top-k queries. Proceedings of the VLDB Endowment, 2015, 8(12): 1394−1405

[156] Zhang J, Zhou Y, Hui B, Liu Y, Li Z, Hu S. TrojanSQL: SQL injection against natural language interface to database. In: Proceedings of 2023 Conference on Empirical Methods in Natural Language Processing. 2023, 4344−4359

[157] Kumar S S, Cummings M L, Stimpson A. Strengthening LLM trust boundaries: a survey of prompt injection attacks surender Suresh Kumar Dr. M.L. Cummings Dr. Alexander Stimpson. In: Proceedings of the 4th IEEE International Conference on Human-Machine Systems. 2024, 1−6

[158] Wei A, Haghtalab N, Steinhardt J. Jailbroken: how does LLM safety training fail? Advances in NeuralInformationProcessing Systems, 2023, 36: 80079–80110

[159] Yi J, Xie Y, Zhu B, Kiciman E, Sun G, Xie X, Wu F. Benchmarking and defending against indirect prompt injection attacks on large language models. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1. 2025, 1809−1820

[160] Wang R, Ling Z, Zhou J, Hu Y. A multiple-integration encoder for multi-turn text-to-SQL semantic parsing. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2021, 29: 1503−1513

[161] Yu T, Zhang R, Yasunaga M, Tan Y C, Lin X V, Li S, Er H, Li I, Pang B, Chen T, Ji E, Dixit S, Proctor D, Shim S, Kraft J, Zhang V, Xiong C, Socher R, Radev D. SParC: cross-domain semantic parsing in context. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019, 4511−4523

Mengyi LIU is currently working toward the PhD degree with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. She has worked on natural language interface for spatial databases. Her research focuses on the management and analysis of spatial data.

Xieyang WANG is currently working toward the PhD degree with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. He has worked on natural language interface for moving objects databases. His research focuses on spatial databases and moving objects databases.

Jianqiu XU received his PhD degree in FernUniversität in Hagen, Germany, 2012. He is currently a professor with the Nanjing University of Aeronautics and Astronautics, China. His research interests include spatial databases and moving objects databases. He was on the program committees for conferences, including the PVLDB, KDD, SSTD, and DASFAA.

Weijia YI is currently working toward the master's degree with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. She has worked on natural language query corpus construction for spatial databases. Her research focuses on corpus construction and query verification and correction.

Ouri WOLFSON is currently a professor of Computer Science at the University of Illinois at Chicago, USA and the founder, president, and chief Scientist of Pirouette Software Inc.. He served as a consultant to Argonne National Laboratory, US Army Research Laboratories, DARPA, and NASA. His main research interests are in big data, mobile and pervasive computing, smart city technologies, and AI.