REVIEW ARTICLE

A Systematic Literature Review of Text-to-SQL: Performance, Challenges, and Limitations

Muhammad Shahzaib Baig[1], **Tahir Sher**[2,†,*], **Abdul Rehman**[3,†,*] **and Saim Sheikh**[4]

[1] Department of Creative Technologies, Air University, Islamabad 44000, Pakistan
[2] Department of Artificial Intelligence, Korea University, Seoul 02842, Republic of Korea
[3] Convergence Institute of Human Data Technology, Jeonju University, Jeonju 55069, Republic of Korea
[4] Department of Computer Science, Air University, Islamabad 44000, Pakistan

Abstract

This literature review examines the state of Text-to-SQL technology, which translates natural language queries into SQL. It analyzes rule-based, neural, and hybrid approaches, assessing their strengths and weaknesses, and surveys commonly used datasets, benchmarks, and evaluation metrics. The study identifies research gaps concerning generalization, scalability, and interpretability, and suggests integrating user feedback and domain knowledge. To better understand the implementation and potential improvements of machine learning in this domain, we conducted a systematic literature review (SLR) of publications from 2015 to 2023, with select post-2023 works cited as contextual references in the emerging trends discussion. From 439 gathered papers, 23 were identified as highly relevant. The review analyzes these works across four areas: (i) datasets employed, (ii) evolution of learning methods, (iii) development of evaluation procedures, and (iv) a meta-analysis of model performance. The findings confirm significant room for improvement in learning strategies. Persistent research gaps include cross-domain generalization, schema linking for complex databases, a lack of robust multilingual models, and the trade-off between model accuracy and interpretability. We propose future directions such as integrating contrastive schema linking, zero-shot/few-shot learning, explainability-driven design, and developing diverse, large-scale benchmarks that reflect real-world database complexity.

Keywords: Text-to-SQL, systematic literature review, natural language processing, meta analysis.

1 Introduction

In recent times, natural language interfaces have become increasingly popular for querying databases, as they provide a user-friendly and intuitive way for non-experts to interact with complex data systems.

Submitted: 27 June 2025
Revised: 29 July 2025
Accepted: 03 August 2025
Published: 17 November 2025

Vol. 2, **No.** 1, 2026.
10.62762/TACS.2025.497935

***Corresponding authors:**
✉ Tahir Sher
2025010294@korea.ac.kr
✉ Abdul Rehman
a.rehman.jj@jj.ac.kr

† These authors contributed equally to this work

Citation
Baig, M. S., Sher, T., Rehman, A., & Sheikh, S. (2025). A Systematic Literature Review of Text-to-SQL: Performance, Challenges, and Limitations. *ICCK Transactions on Advanced Computing and Systems*, 2(1), 1–24.

© 2025 by the Authors. Published by Institute of Central Computation and Knowledge. This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

The technology that enables this interaction is Text-to-SQL, which aims to translate natural language queries into SQL queries that can be executed on a database [1]. Text-to-SQL has gained significant attention from the research community, resulting in numerous approaches and systems proposed for improving its accuracy and usability [2–4]. However, despite these efforts, the technology still faces significant challenges, including the lack of generalization, scalability, and interpretability [5]. A systematic literature review is needed to provide a comprehensive overview of the state-of-the-art in Text-to-SQL research, including the existing approaches, datasets, evaluation metrics, and research gaps such as model robustness to linguistic variation [6], evaluation methodology [4], and schema linking generalization [7]. The review aims to identify the strengths and limitations of current Text-to-SQL systems and provide insights for future research in this field [8]. The review can serve as a valuable resource for researchers and practitioners working on improving the performance and usability of Text-to-SQL systems. For instance, recursive sketch-based approaches such as RYANSQL [9] demonstrate how structured decomposition of complex SQL queries across cross-domain databases can drive meaningful accuracy improvements, representing the type of technical advancement this review aims to systematically survey.

To achieve the task of converting natural language to SQL queries, researchers have used various methods such as self-attention mechanisms, bi-directional LSTMs, convolutional neural networks, and pre-trained language models such as BERT [10]. More recently, large language models such as Codex have also been evaluated for their Text-to-SQL capabilities [11], demonstrating strong performance through prompt-based approaches without task-specific fine-tuning. Beyond English, efforts have also been made to develop Text-to-SQL datasets for non-English languages, such as DuSQL [12], a large-scale Chinese Text-to-SQL dataset covering 200 databases and 23,797 question-SQL pairs, demonstrating the growing interest in cross-lingual database querying. There are also two main approaches to SQL generation: sketch-based and generation-based. The former involves splitting the SQL generation process into smaller prediction slots, while the latter involves using decoder models that generate SQL through grammar-guided transition actions or sequence decoding [16]. Recent research has also focused on tabular language models, which directly encode table data and natural language text to improve parsing accuracy [3, 17]. Alongside these model advances, multi-turn datasets such as SParC [13] have been developed to support context-dependent parsing across complex, cross-domain database interactions. While some earlier methods relied on hand-crafted techniques, more recent studies have employed deep learning-based models trained on large datasets of natural language sentences and annotations. Early hybrid systems, such as SQLizer [14], combined natural language parsing with type-directed program synthesis to automatically generate SQL queries without requiring users to know the underlying database schema. However, the relatively limited scale of some datasets, such as Spider, which contains 10,181 questions paired with 5,693 unique SQL queries across 200 databases, has constrained the generalization capability of models trained solely on it. Previous models trained on the Spider dataset have primarily used a sequence-to-sequence approach, resulting in relatively low overall accuracy [15]. Therefore, there is a need for larger and more diverse datasets to improve the accuracy of Text-to-SQL systems [18].

Systematic literature review (SLR) has a significant role in meta-analysis, which is the process of combining quantitative data to synthesize evidence [20]. It is also known as research synthesis, pooled analysis, or quantitative review. Meta-analysis uses observational data to answer specific, pre-defined research questions and to identify variations in studies, gaps in knowledge, and future research needs. Meta-analysis can identify heterogeneity in research and data limitations. The availability of datasets and their statistics is an essential factor in improving the accuracy and performance of language models. For example, integrating relational structural information from the database schema into pre-trained Seq2Seq models, as demonstrated by RASAT [21], yields significant accuracy gains on the Spider benchmark, highlighting how both data quality and structural encoding jointly determine model performance. Sparse data in a domain is excellent for training, but cross-domain datasets are often used in such cases [18]. In the following sections, we will discuss the dataset and its statistics in more detail. The second research topic focuses on the core architecture of the models, and recent advancements in encoder and decoder models, such as large pre-trained models, including zero-shot and few-shot learning, have been found to be useful in

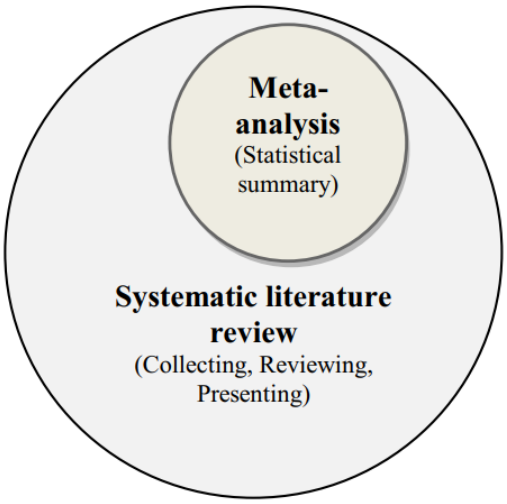


Figure 1. Relationship of SLR and meta-analysis

this area [19]. Table 1 summarizes data-extraction strategies and their implications, providing a clear specification of expectations and results [17].The relationship of SLR and Meta-analysis is shown in Figure 1

2 Research Methodology

The systematic literature review conducted for this investigation involved the development of a review methodology that aimed to address the research questions. In contrast to a traditional literature review, the review methodology was accompanied by selection and rejection criteria that assessed each primary study and a defined search strategy that was comprehensive and unbiased. Quality evaluation criteria were employed to assess the information from each study. The review methodology was also crucial for conducting a quantitative meta-analysis. Additionally, the systematic literature review provided a foundation for identifying research gaps in the chosen field and for positioning new research endeavors.

A preferred reporting item for systematic reviews and meta-analyses (PRISMA) approach as shown in Figure 2 was utilized to conduct a comprehensive systematic literature review, where papers were assessed based on relevance, publication date, and framework accuracy. In total, 439 papers were sourced from various databases and formed the basis of this review. Our focus was on linguistic models, specifically large language models (LLMs), evaluation metrics, dataset availability, and outcomes, as we examined trends and main challenges in this research area. Figure 2 illustrates the review structure definition. To reduce bias, we searched databases such as Google Scholar and included journals and conferences, resulting in a complete collection of pertinent literature without limiting it to a particular research approach. The initial database yielded 439 publications, which were screened and filtered based on several criteria, including publication year and topic relevance, as depicted in Figure 1.

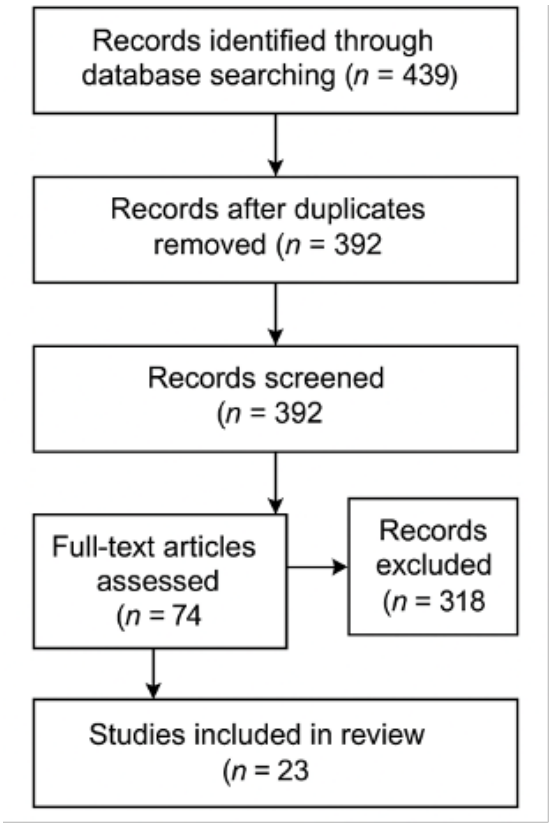


Figure 2. PRISMA Flow Diagram for Study Selection in the Systematic Literature Review.

2.1 Selection OR Rejection Criteria

To fulfill the objectives of the systematic literature review (SLR), certain criteria were established for selecting or rejecting research articles:

(i) Language modeling and the Text-to-SQL model must be the focus of the chosen research studies.

(ii) This SLR requires selected research studies to be published between 2015 and 2023. A small number of post-2023 works are referenced in the Emerging Trends section (Section 4) solely as contextual references and are not counted among the 23 core reviewed studies.

(iii) All research studies chosen for this SLR must be from any reputable journal.

(iv) No duplicate research studies are chosen and if found research was discarded.

The PRISMA Flow Diagram provides a transparent overview of the literature selection process followed in this systematic review. Initially, 439 records were identified through comprehensive database searches. After removing duplicates, 392 unique records remained. These were screened based on their titles and abstracts, resulting in the exclusion of 318 records that did not meet the inclusion criteria. The remaining 74 full-text articles were assessed in detail for eligibility. Following this rigorous screening, 23 studies were ultimately included in the review. This process ensures that the final selection of literature is both methodologically sound and relevant to the research objectives.

2.2 Data Extraction and Synthesis

The data extraction process followed the PRISMA guidelines as illustrated in Figure 3. Table 1 presents a depiction of how data extraction and synthesis are employed to answer research inquiries. In the second section of the table, the bibliography was the first to be extracted, followed by significant findings such as methodology, pre-training strategies (e.g., grammar-augmented pre-training for table semantic parsing [22]), fine-tuning, and outcomes.

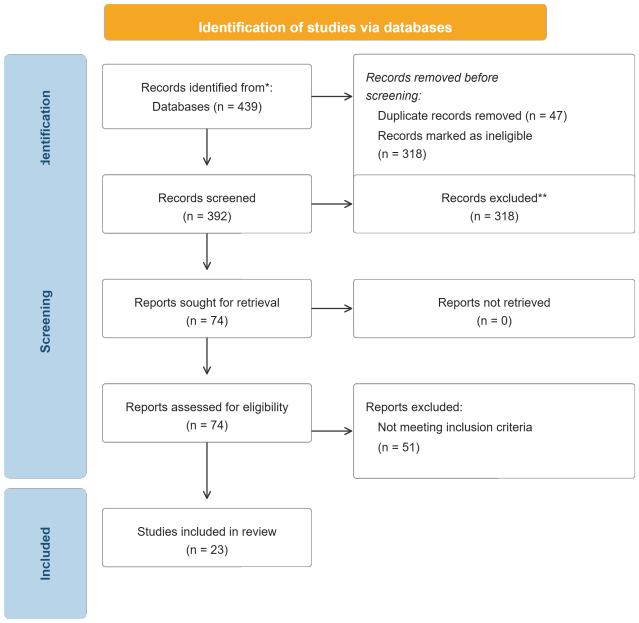


Figure 3. PRISMA data extraction.

2.3 Protocols for Research

The following two issues are the most concerning to the reliability of this review:

(i) Study selection bias and potential errors in data extraction and processing.

Table 1. Research descriptions and outcomes.

Sr. no.	Description	Details
1	Bibliographic data	Authors, title, research type, publication year, and so on.
2	Methodology	The primary goal of our research is to extract the paper's methodology.
3	Pretraining	Pretraining structure is thoroughly examined.
4	Fine-tuning	Fine-tuning structure of each study is thoroughly analyzed.
5	Dataset	Datasets used in the selected studies are identified.

(ii) The search technique, literature sources, selection criteria, and quality criteria all have an impact on how studies are chosen.

We did our literature search using numerous databases, and we describe the search approach we employed in detail so that it may be duplicated in the future. As the initial stage in our search approach, we employed the publication title filter. We only searched for primary research that was primarily focused on finding subjects using predetermined search phrases. As a result, research articles recommending new technologies were required to meet a minimum page length criterion of five pages. The wide list of included research, on the other hand, illustrates the scope of our search. Duplicate primary investigations are avoided by removing grey literature and workshop papers from the literature evaluation.

2.4 Analysis Objective

Determine which empirical research is essential to the process of transforming text to SQL. Using a list of relevant keywords, we retrieved all papers and filtered the results down to just studies published in the recent 10 years. Articles with inappropriate comments are then deleted. The search results are organized further, and the publications are categorized depending on the methodologies and datasets employed.

2.5 Research Questions

The literature review for this study focuses on answering RQ1, which aims to identify the most recent Text-to-SQL models in the literature. RQ2 examines the Machine Learning Setup and is divided into four sub-questions, namely: RQ2.1, which explores the independent factors considered in Text-to-SQL conversions; RQ2.2, which investigates the approaches used for completing the Text-to-SQL task; RQ2.3, which looks at the algorithms used for Text-to-SQL conversion; and RQ2.4, which analyzes the training methodologies proposed in the literature.

Table 2. Breakdown of research questions and their objectives.

Research Question	Motivation
RQ 1 - Literature Review	
RQ 1: What is the current state of the art for Text-to-SQL learning techniques as presented in existing literature?	To explore the current state-of-the-art of the Text-to-SQL conversion.
RQ2 - Learning Techniques	
RQ 2.1: Which factors have been taken into account as independent variables in Text-to-SQL conversion tasks? RQ 2.2: What techniques have been explored to carry out the Text-to-SQL task? RQ 2.3: Which algorithms have been utilized for the conversion of Text-to-SQL? RQ 2.4: What are the proposed approaches for training Text-to-SQL conversion models according to the literature?	The aim of this analysis is to examine previous research on machine learning approaches in Text-to-SQL tasks, including the use of independent and dependent variables, machine learning algorithms, and training methodologies.
RQ 3 - Evaluation Metrics	
RQ 3.1: What methods have been employed to validate Text-to-SQL models in the literature? RQ 3.2: What metrics have been utilized to evaluate the efficacy of Text-to-SQL models? RQ 3.3: Which datasets have been utilized in Text-to-SQL research?	To investigate the techniques used to validate the Text-to-SQL methods proposed, assess their precision, and scrutinize the source code projects.
RQ 4 - Meta-Analysis	
RQ 4.1: What are the independent variables that have shown better performance for Text-to-SQL conversion according to the literature? RQ 4.2: To what extent does the choice of machine learning algorithm impact the performance of Text-to-SQL prediction models? RQ 4.3: How does the choice of training strategy impact the performance of Text-to-SQL prediction models, as reported in the literature?	The study aims to examine how different variables, machine learning methods, and training techniques affect the effectiveness of Text-to-SQL models.

RQ3 investigates the Evaluation Setup used in Text-to-SQL conversion research. The sub-questions include RQ3.1, which explores the validation approaches used; RQ3.2, which examines the assessment metrics used to evaluate Text-to-SQL models; and RQ3.3, which identifies the datasets considered.

RQ4 focuses on Performance Meta-Analysis and includes three sub-questions: RQ4.1, which identifies the most effective independent variables for Text-to-SQL conversion; RQ4.2, which examines the effect of machine learning algorithms on Text-to-SQL prediction model performance; and RQ4.3, which investigates the impact of training strategies on Text-to-SQL prediction model performance. These sub-questions provide insight into the factors that contribute to the success of Text-to-SQL models and can guide future research efforts to improve their performance.

Following the PRISMA protocol and selection criteria, we identified 23 highly relevant studies for in-depth analysis. Table 3 provides a representative sample of 17 of these studies, showcasing the diversity of authors, research focus, and datasets used in the selected literature. The remaining studies are incorporated into the thematic analysis presented in subsequent sections.

To provide a comprehensive overview of the research framework, Table 2 summarizes the breakdown of all research questions and their corresponding objectives, motivations, and mapping relationships discussed in this systematic review.

3 Discussion

After applying the review protocol, we studied the characteristics of the selected publications, focusing on existing models, datasets, training protocols, encoding-decoding techniques, evaluation metrics, and model architectures. The systematic mapping of research questions to their corresponding findings is presented in Table 4, which serves as the analytical framework for this discussion section.

In this section, we will explore the studied characteristics of the literature. The table maps the research questions to their corresponding answers and findings.

3.1 Technical Evolution and Trends in Text-to-SQL

Based on our analysis of the 23 selected studies, several key technical trends emerge in the evolution of Text-to-SQL systems:

Models vary in their ability to perform schema

Table 3. Sample study from the Literature.

ID	Authors	Title	Dataset
S1	Wang et al. [12]	DuSQL: A large-scale and pragmatic Chinese Text-to-SQL dataset	DuSQL
S2	Zhong et al. [17]	Structured queries using reinforcement learning	WikiSQL
S3	Xu et al. [19]	Queries without reinforcement learning	WikiSQL
S4	Yaghmazadeh et al. [14]	SQLizer: query synthesis from natural language	MAS, IMDB, YELP
S5	Li et al. [41]	RESDSQL: decoupling schema linking and skeleton parsing	Spider
S6	Yu et al. [15]	TypeSQL: knowledge-based, type-aware neural SQL	WikiSQL
S7	Dong and Lapata [38]	Coarse-to-fine decoding for semantic parsing	GEOQuery; ATIS; WikiSQL
S8	Cai et al. [25]	Structure-aware dual graph aggregation for cross-domain Text-to-SQL	Spider
S9	Shi et al. [42]	IncSQL with non-deterministic oracles	WikiSQL; filtered ATIS
S10	Hwang et al. [29]	Table-aware contextualisation on WikiSQL	WikiSQL
S11	He et al. [43]	X-SQL: schema representation	WikiSQL
S12	Wang et al. [44]	Execution-guided decoding for robustness	WikiSQL; ATIS; GEOQuery
S13	Lee [45]	Clause-wise decoding for complex generation	Spider
S14	Yu et al. [3]	SyntaxSQLNet with syntax trees	Spider
S15	Lin et al. [30]	Grammar-based neural generation	Spider
S16	Bogin et al. [31]	Graph neural networks for schema	Spider
S17	Lei et al. [32]	Re-examining the role of schema linking in Text-to-SQL	Spider

Table 4. Mapping of research questions to their findings.

Sr. No.	Research Questions	Mapping
1	RQ 1: Literature Considered	Section: State-of-art Models
2	RQ 2: Machine Learning Setup	Section: Encoder-Decoder Techniques and Dataset
3	RQ 3: Evaluation Setup	Section: Evaluation Metrics
4	RQ 4: Performance-Meta Analysis	Section: Meta-Analysis

linking, especially for unseen or unfamiliar databases. Methods such as RAT-SQL integrate relation-aware encoders, while models like SQLova use grammar constraints for increased precision. However, large LLMs such as Codex may outperform traditional models in semantic alignment, but they lack built-in schema awareness [23], making them prone to hallucination unless tightly constrained or fine-tuned.

Transformer-based encoders have largely replaced earlier BiLSTM models due to their superior ability to model global context and parallelize computation. These encoders capture long-range dependencies more effectively, improving the representation of complex input queries.

On the decoding side, models range from autoregressive to sketch-based generation. While sketch-based decoders offer stronger syntactic control by predicting SQL components within a predefined structure, they often struggle with unseen or compositional queries. Autoregressive decoders provide more flexibility but may produce invalid SQL structures when grammar constraints are absent.

The observed performance trends suggest a paradigm shift from rule-based and early neural models toward large pretrained transformers. While these models offer improved fluency and generalization, they introduce trade-offs in terms of interpretability and schema grounding. Furthermore, our review reveals that many evaluation metrics do not capture real-world query complexity or domain adaptation challenges. Thus, future work must consider hybrid metrics, benchmark expansion, and cross-lingual capabilities to bridge this research-to-application gap.

3.2 State-of-the art (SOTA) models

This section addresses RQ1 by reviewing state-of-the-art models for Text-to-SQL, examining their architectures, capabilities, limitations, and challenges.

3.2.1 Seq2SQL

In 2017, a model called Seq2SQL was introduced to generate SQL queries from natural language questions. It comprises an encoder-decoder structure, in which a bidirectional LSTM neural network serves as the encoder to analyze and encode the input question into a series of hidden states. The decoder is an LSTM neural network that generates the SQL query step by step, using the previous tokens and the input question. To highlight the important parts of the input question, the model employs an attention mechanism to calculate a weighted average of the encoder's hidden states based on the present decoder's hidden state. However, Seq2SQL is limited in its ability to handle more complex SQL queries, such as those that require multiple sub-queries or intricate joins. Additionally, a significant amount of training data is required for the model to learn the relationship between natural language questions and their corresponding SQL queries effectively [17].

Structure-grounded pre-training approaches have also been proposed to improve Text-to-SQL by learning text-table alignment from parallel corpora, achieving significant improvements over BERT-based baselines on the Spider benchmark [26].

3.2.2 SQLNet

The model is an encoder-decoder architecture, however, it takes a different approach than Seq2SQL. Instead of an LSTM-based encoder-decoder architecture, SQLNet predicts the SQL query from the input question using a multi-layer perceptron (MLP). When constructing the SQL query, the SQLNet model additionally employs a column attention method to assist it in focusing on the important columns in the database. The column attention method, in particular, computes a weighted sum of the columns depending on the current decoder hidden state, giving more weight to the columns most important to the current decoding phase. SQLNet has the benefit of being able to handle more complicated SQL queries than Seq2SQL [17]. Furthermore, SQLNet requires less training data than Seq2SQL since its sketch-based decomposition and column attention mechanism reduce the output search space, making learning more efficient. SQLNet may still struggle with more sophisticated queries using nested sub-queries or complex database structures.

3.2.3 Intermediate Representation (IRNet)

The model's architecture is made up of three major components: a natural language parser, a schema encoder, and a query decoder. The natural language parser is in charge of parsing the input question and converting it into an intermediate representation called SemQL, a tree-structured query language designed to bridge natural language and SQL. SemQL is used to extract crucial information including table names, column names, and logical operators, which are then passed on to the model's other components.

The schema encoder is in charge of converting database schema information, such as table and column names, into vector representations, while simultaneously performing schema linking to align question tokens with the corresponding tables and columns in the database. The query decoder component then uses these aligned representations as input.

The query decoder component generates the SQL query based on the supplied question and the encoded database schema information. When creating the SQL query, the query decoder employs an attention method to pay attention to important sections of the input question and database structure.

One of IRNet's [27] primary benefits is that it can handle more sophisticated SQL searches with nested sub-queries or more elaborate database structures. Furthermore, using a parse tree format allows the model to explicitly simulate the syntactic structure of the input query, which can assist increase prediction accuracy. IRNet, on the other hand, may still suffer from queries involving sophisticated joins or nested aggregates.

3.2.4 Spider Baseline Parser

The Spider baseline parser features a bidirectional LSTM network encoder-decoder design [28]. It includes numerous innovative components that are intended to manage the task's problems. The model, for example, has a schema linkage module that connects each word in the input question with a matching column or table in the database schema. This enables the model to incorporate important database schema information more effectively when constructing the SQL query.

A copy mechanism is also used by the Spider model to directly copy relevant terms from the input question to the output SQL query. This enables the model to handle scenarios in which the SQL query is a direct translation of the input question, as opposed to a more sophisticated query requiring many tables or joins.

The Spider model also employs a set of SQL-specific decoding rules to verify that the resulting SQL query is syntactically correct. These decoding rules contain limitations such as ensuring that the columns chosen are valid and that the join requirements are defined appropriately.

Overall, the Spider baseline parser is designed to perform complicated SQL queries with several tables and joins, and it established strong benchmark results on the Spider dataset at the time of its introduction.

3.2.5 SQLova

The model is made up of a hierarchical encoder-decoder design, with the encoder being a two-layer bidirectional LSTM network and the decoder being another two-layer LSTM network.

The SQLova model [29] has many main components that are intended to address the task's problems. The model, for example, has a schema linkage module that connects each word in the input question with a matching column or table in the database schema.

This enables the model to incorporate important database schema information more effectively when constructing the SQL query.

In addition, when creating the SQL query, the SQLova model employs a sequence-to-set attention mechanism that attends to important words in the input question as well as the associated database schema components. As a result, the model may better represent the links between the various items in the input question and the database design. To verify that the resulting SQL query is syntactically legitimate, the SQLova model employs a set of SQL-specific decoding rules. These decoding rules contain limitations such as ensuring that the columns chosen are valid and that the join requirements are defined appropriately.

Overall, the SQLova model is designed to handle complex SQL queries with multiple tables and joins, and achieved competitive results on several benchmark datasets at the time of its publication. Additionally, the hierarchical architecture of the model allows it to better capture the relationships between different elements in the input question and the database schema, which can help improve the accuracy of the predictions.

3.2.6 *Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers (RAT-SQL)*

RAT-SQL [24] is a contemporary approach for Text-to-SQL conversion, which is used for translating natural language queries into SQL queries. The model comprises two primary components: a question encoder and a SQL query decoder. The question encoder adopts a relation-aware self-attention mechanism to encode both the input question and database schema jointly into context-aware representations. The SQL query decoder then generates the SQL query from these representations using an LSTM-based decoder guided by schema-linking constraints and beam search. RAT-SQL achieves strong generalization to unseen database schemas through explicit schema linking, making it a widely adopted baseline for Text-to-SQL conversion [24]. Additionally, S^2SQL [33] further extends this line by injecting syntactic dependency information of questions into the question-schema interaction graph encoder, achieving improved performance on the Spider benchmark.

3.2.7 *Graphix-T5: Mixing Pre-trained Transformers with Graph-Aware Layers*

Graphix-T5 [34] is a hybrid Text-to-SQL model that augments the pre-trained T5 encoder-decoder architecture with specially designed graph-aware layers. The core motivation is to introduce structural inductive bias into the T5 backbone, enabling the model to better reason over the relational structure of database schemas and multi-hop dependencies between question tokens and schema elements.

The graph-aware layers are interleaved with the standard transformer layers of T5, allowing the model to jointly capture both the sequential context from pre-training and the structural relationships encoded in the schema graph. This hybrid design preserves the generalization capability of large pre-trained models while equipping them with explicit structural reasoning capacity that vanilla T5 lacks.

Graphix-T5 was evaluated across four Text-to-SQL benchmarks: Spider, Spider-SYN, Spider-Realistic, and Spider-DK. It surpasses all other T5-based parsers by a significant margin, with Graphix-T5-large outperforming the original T5-large by 5.7% on exact match (EM) accuracy and 6.6% on execution accuracy (EX). This demonstrates that integrating graph-aware structural components into pre-trained transformers is an effective strategy for improving both accuracy and robustness in cross-domain Text-to-SQL parsing.

3.2.8 *PICARD - Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models* [35]

PICARD model [35] is designed to address the problem of generating structured output sequences that satisfy complex syntactic and semantic constraints. It uses an auto-regressive decoding strategy to generate output sequences incrementally, while also incorporating a parser to ensure that each step in the decoding process satisfies the specified constraints. PICARD is built on top of a pre-trained T5 model and introduces a constrained decoding mechanism that incrementally validates each generated token against SQL grammar rules, rejecting invalid continuations at each decoding step to ensure syntactic correctness of the output SQL query. During fine-tuning, the model is trained to predict both the next token in the output sequence and the parsing actions that need to be taken to ensure that the output sequence satisfies the specified constraints.

3.2.9 *TURL (Table Understanding through Representation Learning)*

TURL [36] introduces a pre-training and fine-tuning paradigm for relational Web tables. The framework learns deep contextualized representations of tabular

data in a self-supervised manner, employing a Masked Entity Recovery (MER) objective that trains the model to reconstruct masked table cell entities using the surrounding table structure and associated natural language descriptions.

TURL is evaluated on six table understanding tasks: entity linking, column type annotation, relation extraction, row population, cell filling, and schema augmentation. Its universal model design allows it to be fine-tuned across these tasks with minimal task-specific modifications, substantially outperforming prior methods. While TURL does not directly target SQL query generation, its ability to learn structured representations of table schemas and cell-level entity semantics makes it a representative example of table-aware pre-training methods, an important class of techniques in the broader landscape of models designed to bridge natural language and structured relational data.

3.2.10 *SeaD: End-to-end Text-to-SQL Generation with Schema-aware Denoising*

SeaD [37] (End-to-end Text-to-SQL Generation with Schema-aware Denoising) is a neural network-based end-to-end approach that generates SQL queries directly from natural language inquiries. The SeaD model is made up of two primary parts: a denoiser and a SQL generator. The denoiser is in charge of cleaning up the input question and deleting any extraneous or distracting data. This is accomplished by using the table structure, which adds context and limits the sorts of queries that may be created. The SQL generator component then constructs the matching SQL query from the denoised question. The generator has a transformer-based design with encoder and decoder components to capture the intricate links between the input question and the destination SQL query. The SeaD model is optimized during training using a mix of maximum likelihood estimation and execution-guided training, which uses the execution results of generated SQL queries as feedback signals to improve generation accuracy. As a result, the model is able to properly manage the trade-off between creating grammatically good SQL queries and ensuring that they are semantically right and valid in relation to the table structure. SeaD can successfully handle noisy and ambiguous input questions and create correct and acceptable SQL queries that fulfill the restrictions of the database schema by including a schema-aware denoiser.

3.2.11 *SADGA: Structure-Aware Dual Graph Aggregation Network for Text-to-SQL*

SADGA (Structure-Aware Dual Graph Aggregation Network) [25] is a neural network-based model used in Text-to-SQL jobs to generate SQL queries from natural language inquiries. The SADGA model is intended to manage the complicated structure of input questions and SQL queries, which frequently have several components and subqueries. The model includes a dual graph aggregation network that successfully captures the interactions between the different input and output structure components. A question graph and a SQL graph comprise the dual graph aggregation network. The structure of the input question is represented by the question graph, while the structure of the output SQL query is represented by the SQL graph. The model then employs a number of graph convolutional layers to successfully collect information from each graph's numerous nodes and edges, allowing it to represent the intricate connections between the input and output structures. The SADGA model is optimized during training using cross-entropy loss on the target SQL sequences, and is evaluated using both exact match accuracy and execution accuracy to assess whether generated queries are both syntactically correct and semantically executable on the underlying database. Overall, the SADGA model has demonstrated strong performance across a variety of tests and benchmarks, and it represents a substantial advancement in the field of Text-to-SQL generation. SADGA is able to manage the intricate connections between the input and output structures and create accurate and valid SQL queries that fulfill the restrictions of the underlying database by using a structure-aware dual graph aggregation network.

3.3 Encoding-Decoding Techniques

This section addresses RQ2, exploring the machine learning setup in the context of encoding-decoding techniques used in Text-to-SQL models, along with datasets, challenges, and limitations.

3.3.1 *Encoding Techniques*

In the context of machine learning and natural language processing (NLP), the act of transforming text-based information into numerical values is known as encoding. The purpose of encoding is to restructure unstructured text data into a structured format that can be processed by machine learning algorithms. There exist several encoding methods that have been utilized in recent algorithmic developments.

Transformer-based Encoders The transformer-based encoder is a novel neural network architecture that has been widely adopted in Text-to-SQL systems, with pre-trained models such as BERT [10] serving as powerful backbone encoders for jointly representing natural language questions and database schemas. One notable application is SADGA [25], which employs a structure-aware dual graph aggregation network built upon transformer-based encoding to improve SQL generation accuracy. It is commonly used for sequence-to-sequence applications such as text summarization, machine translation, and question-answering, including Text-to-SQL. Transformer-based encoders contain several self-attention layers and feed-forward neural networks. Self-attention enables the model to gain contextual information from the input text by attending to various parts of the text. Feed-forward neural networks help transform the encoded information into a structured format suitable for downstream applications. During the encoding process, each input token is transformed into a dense vector representation, which is then fed through the self-attention layer. The self-attention layer calculates a weighted sum of all input vectors, where the weights depend on the similarity between each input vector and all the other input vectors. This allows the model to grasp the relationships between various tokens in the input text. The output from the self-attention layer then proceeds through a feed-forward neural network, which applies a non-linear transformation to the input vector. This process is repeated several times, with each subsequent layer building on the output from the preceding layer. The final output of the Transformer-based encoder is a fixed-length vector representation of the input text, which can be utilized as input for downstream tasks like Text-to-SQL. The Transformer-based encoder has demonstrated exceptional performance on a range of NLP tasks, including Text-to-SQL. While Transformer-based encoders have shown impressive results in various NLP tasks, including Text-to-SQL, they also have some limitations in this context. Some of the limitations of Transformer-based encoders in Text-to-SQL are:

(i) Limited ability to model sequential information: Unlike RNNs, which have a natural ability to model sequential information, Transformer-based encoders do not have an explicit recurrence and are limited in their ability to capture sequential dependencies. This can be a limitation in Text-to-SQL, where the order of the input tokens may be important for accurately predicting the corresponding SQL query.

(ii) Difficulty in handling out-of-vocabulary (OOV) tokens: Transformer-based encoders rely on pre-trained word embeddings to represent input tokens. However, when the input text contains OOV tokens, which are not present in the pre-trained vocabulary, the model may struggle to encode them accurately, leading to reduced performance.

(iii) Limited interpretability: Transformer-based encoders are often described as black-box models because it can be difficult to interpret how they arrive at their output. In Text-to-SQL, this may be a limitation when trying to understand the model's decision-making process, which is important for model debugging and error analysis.

(iv) Computationally expensive: Transformer-based models typically require a large amount of computation, both during training and inference. This can make them more challenging to deploy in production systems, especially in resource-constrained environments.

(v) Difficulty in handling long input sequences: Transformer-based models are known to struggle with long input sequences, which can be a limitation in Text-to-SQL, where the input text may be quite long. While techniques such as attention and masked self-attention are designed to handle this issue, there are still limitations in terms of the maximum input sequence length that can be processed efficiently.

Convolutional Neural Networks (CNNs) In Text-to-SQL, CNNs can be used as encoders to transform the input text into a fixed-length vector representation that can be used as input to downstream tasks. The input text is first preprocessed, which involves tokenization and optionally, padding or truncating the input text to a fixed length. The tokenized text is then embedded into a dense vector representation, where each token is represented as a vector of fixed size. The CNN encoder then applies a series of convolutional filters to the embedded input text. The convolutional filters scan over the input text and apply a mathematical operation to a window of adjacent tokens at a time. The operation applied by the convolutional filters is typically a dot product between the filter weights and the input text window. The output of the convolutional filters is passed through a non-linear activation function, such as ReLU, to introduce non-linearity into the

model. This process is repeated with multiple filters of different sizes, which allows the model to capture different patterns and features in the input text. After the convolutional layers, the output is flattened into a fixed-length vector representation, which can be used as input to downstream tasks such as Text-to-SQL. The final output vector represents the encoded information from the input text, which captures important features and patterns relevant to the downstream task. CNN encoders have been shown to be effective in natural language processing tasks including text classification, sentiment analysis, and Text-to-SQL. They are particularly useful for capturing local features and patterns in the input text, and can be used in combination with other encoding techniques such as recurrent neural networks (RNNs) and Transformer-based models for improved performance. While CNNs have shown promising results in various NLP tasks, including Text-to-SQL, they also have some limitations in this context. Some of the limitations of CNNs in Text-to-SQL are:

(i) Limited ability to model sequential information: Unlike RNNs, which have a natural ability to model sequential information, CNNs are limited in their ability to capture sequential dependencies. This can be a limitation in Text-to-SQL, where the order of the input tokens may be important for accurately predicting the corresponding SQL query.

(ii) Difficulty in capturing long-range dependencies: CNNs process input through local convolutional windows, which limits their ability to capture long-range dependencies between distant tokens. In Text-to-SQL, relationships between question words and schema elements that are far apart in the input sequence may be missed, leading to inaccurate SQL generation.

(iii) Limited interpretability: Like Transformer-based encoders, CNNs are often described as black-box models, which can be a limitation in Text-to-SQL when trying to understand the model's decision-making process.

(iv) Difficulty in handling out-of-vocabulary (OOV) tokens: CNNs rely on pre-trained word embeddings to represent input tokens. When the input text contains OOV tokens, which are not present in the pre-trained vocabulary, the model may struggle to encode them accurately, leading to reduced performance.

(v) Limited ability to capture long-range dependencies: CNNs are designed to capture local dependencies in the input sequence, which can be a limitation in Text-to-SQL, where long-range dependencies between tokens may be important for accurately predicting the corresponding SQL query.

Recurrent Neural Networks (RNNs) In the context of Text-to-SQL, recurrent neural networks (RNNs) are utilized as encoders to transform the input text into a fixed-length vector representation suitable for downstream tasks. Prior to encoding, the input text undergoes preprocessing, which includes tokenization and optionally, padding or truncating the input to a fixed length. The tokenized text is then transformed into a dense vector representation, where each token is assigned a fixed-size vector. The RNN encoder processes the input tokens individually while maintaining a hidden state that preserves context and information from preceding tokens. The hidden state is updated at each time step by combining the current input token with the previous hidden state. The RNN encoder's output is a sequence of hidden states, with one for each input token. The final hidden state, representing the encoded information from the entire input text, is commonly obtained by performing a pooling operation like max-pooling or average-pooling on the sequence of hidden states. RNN encoders are highly effective in natural language processing tasks such as sentiment analysis, text classification, and Text-to-SQL, specifically in modeling sequential data and capturing long-term dependencies in the input text. While RNNs have been widely used in various NLP tasks, including Text-to-SQL, they also have some limitations in this context. Some of the limitations of RNNs in Text-to-SQL are:

(i) Difficulty in modeling long-term dependencies: Although RNNs are designed to model sequential information and capture dependencies between tokens, they can struggle to model long-term dependencies in Text-to-SQL, where there may be a large distance between relevant tokens.

(ii) Vanishing Gradient Problem: The vanishing gradient problem is a common issue with RNNs, which occurs when the gradients become too small during backpropagation, leading to slow learning or even a complete halt in learning. This can be a limitation in Text-to-SQL, where the model needs to learn complex dependencies between tokens.

(iii) Difficulty in retaining long-range dependencies: Unlike CNNs, RNNs can process variable-length input sequences, which is advantageous for Text-to-SQL tasks where question lengths vary. However, RNNs

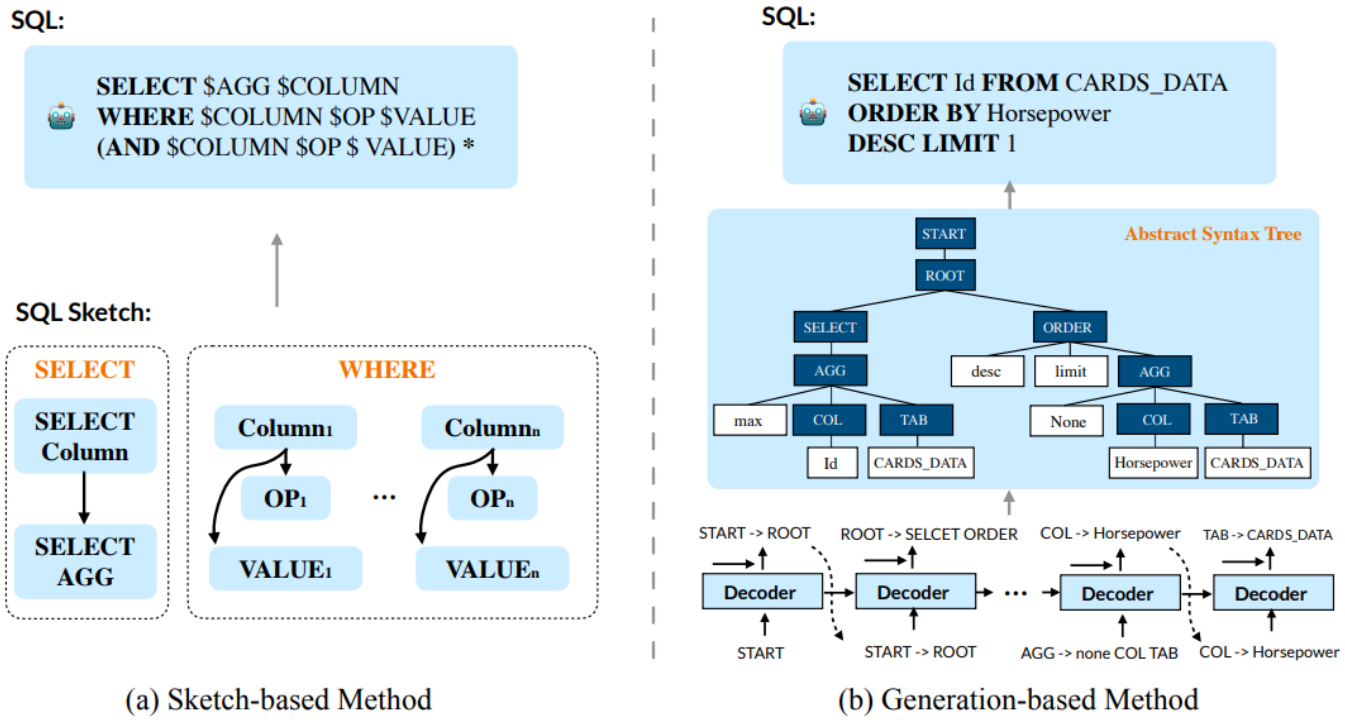


Figure 4. Sketch-based SQL generation framework, as adopted in SQLNet [17] and TypeSQL [15].

can struggle to retain information over very long sequences due to vanishing gradients, which limits their effectiveness on complex, multi-clause SQL generation tasks.

(iv) Limited ability to capture parallelism: Unlike CNNs, RNNs are inherently sequential and cannot easily capture parallelism in the input text. This can be a limitation in Text-to-SQL, where the input text may contain parallel or overlapping clauses that are important for accurately predicting the corresponding SQL query.

(v) Difficulty in handling noisy data: RNNs are sensitive to noise and may struggle to handle noisy or erroneous input data, which can be a limitation in Text-to-SQL, where the input text may contain errors or typos.

Graph Neural Networks (GNNs) GNNs can be utilized for encoding purposes, to convert the input text into a fixed-length vector representation, suitable for downstream tasks. First, the input text is tokenized and converted into a graph representation, where nodes correspond to tokens or schema elements and edges represent their syntactic or semantic relationships. Then, the tokens are embedded into a dense vector representation, where each token is represented as a vector with a constant size. The GNN encoder creates a graph from the embedded input text, where nodes represent the tokens and edges signify the relationships between them based on syntactic or semantic attributes. Semantic parsing approaches that exploit hierarchical decomposition of the input structure [38] have similarly informed the design of graph-based encoders, motivating the use of structured representations that capture both local and global dependencies in the question and schema. Afterward, the encoder applies graph convolutional layers to the input graph, which operates on nodes and their neighbors. The resulting output is a sequence of node embeddings, one for each node in the input graph. Finally, a pooling operation, such as max-pooling or average-pooling, is applied to the node embeddings to obtain a fixed-length vector that serves as input for downstream tasks. GNN encoders have demonstrated effectiveness in various natural language processing tasks, including Text-to-SQL, as they capture syntactic and semantic relationships between tokens in the input text. They can be used in conjunction with other encoding techniques such as CNNs and RNNs for improved performance. They also have some limitations in this context. Some of the limitations of GNNs in Text-to-SQL are:

(i) Difficulty in modeling complex graph structures: In Text-to-SQL, the input text can be represented as a

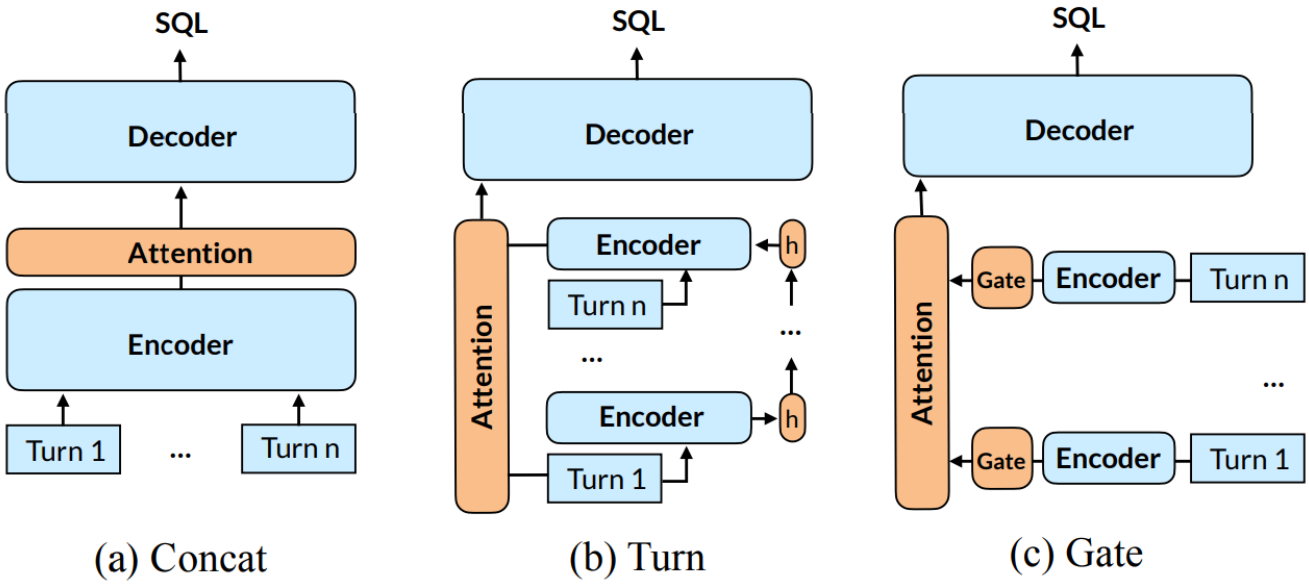


Figure 5. Text question encoding techniques for multi-turn Text-to-SQL: (a) Concatenation-based encoding, (b) Turn-based hierarchical encoding, and (c) Gate-based selective encoding.

graph where each node represents a word or a part of the sentence, and the edges represent relationships between the nodes. However, modeling complex graph structures can be a challenging task for GNNs.

(ii) Difficulty in handling large graphs: GNNs can struggle with large graphs because of the computational complexity involved in processing large amounts of data. In Text-to-SQL, the input graph can be large and complex, especially for longer queries, which can be a limitation for GNNs.

(iii) Difficulty in capturing long-term dependencies: While GNNs are designed to capture information from neighboring nodes, they may struggle to capture long-term dependencies between nodes that are farther apart in the graph.

(iv) Limited interpretability: GNNs are known for their lack of interpretability, which can be a limitation in Text-to-SQL, where it is important to understand how the model generates the SQL query based on the input text.

(v) Limited availability of pre-trained models: Unlike Transformer-based and RNN-based models, there are relatively fewer pre-trained GNN models available for Text-to-SQL tasks, which can make it difficult to leverage transfer learning and limit the overall performance of the model.

3.3.2 *Decoding Techniques*

In the process of Text-to-SQL parsing, there are two types of decoder models: those that rely on sketches, and those that use generation-based techniques. Each of these decoder designs is discussed in detail below.

Sketch-based Methods Text-to-SQL solutions can be approached through a sketch-based method, which involves dividing the SQL generation process into sub-modules, also known as "slots," responsible for predicting different parts of the SQL query. A typical sketch structure is illustrated in Figure 4, showing how SQL queries are decomposed into predictable components. These modules include the SELECT column, AGG function, WHERE clause value, and others. Various models have been proposed for predicting these slots, including SQLNet, which uses a SQL sketch with a separate model for each slot. SDSQL and SQLova propose a modified syntax-guided sketch with six prediction modules. Meanwhile, TypeSQL reduces the number of modules by merging the select-column and where-column modules into a single module. Execution-guided decoding approaches are also used to eliminate non-executable partial SQL queries. Sketch-based solutions are a widely-used approach to Text-to-SQL that offer precise control over the SQL generation process by decomposing query construction into modular prediction steps organized within a pipeline architecture [40]. More recently, RESDSQL [41] proposed a ranking-enhanced

encoding and skeleton-aware decoding framework to decouple schema linking from SQL skeleton parsing, achieving state-of-the-art robustness on Spider and its variants.

Generation-based Methods For more complex SQL queries, Seq2Seq models with a generation-based approach may be better than sketch-based strategies. This approach uses a decoder, such as that introduced in Bridge [39], which includes an LSTM-based indicator generator with multi-head attention and a copy mechanism for directly incorporating schema elements into the generated SQL query. Graph-based schema encoding methods [31] have also been combined with generation-based decoders to better capture structural relationships between database tables and natural language question tokens. During decoding, the decoder generates either a token from the vocabulary, a token copied from the input query, or a schema element selected from the database schema. However, previous generation-based methods have had difficulty generating SQL queries with correct grammar, so more advanced methods generate SQL queries in a first-pass order within an AST abstract syntax tree. This involves the LSTM decoder performing three operations: APPLY-RULE to convert the latest point into a grammar rule, SELECT-COLUMN to pick a column from the schema when filling in the leaf node, and SELECT-TABLE to select the table. The choice between sketch-based and generation-based approaches for Text-to-SQL depends on the complexity and variability of SQL queries in the domain, as well as the quality and size of the available training data. The comparison of accuracy is given in Figure 5.

3.4 Datasets

The amount and quality of examples are critical components of the model's training. The quality of the datasets influences model performance. Single-turn Text-to-SQL datasets with discrete questions and multi-turn Text-to-SQL datasets with multi-turn successive questions are the two basic types of datasets.

To provide a comprehensive overview of dataset characteristics used in Text-to-SQL research, Table 5 presents detailed statistics of major datasets, including the number of questions, SQL queries, databases, domains, and tables for each benchmark.

We categorize benchmark datasets based on schema complexity (flat vs. nested), domain specificity (general-purpose vs. task-specific), and support for multi-turn dialogues. For instance, Spider features complex and unseen schemas, whereas WikiSQL contains simpler, single-table queries. Datasets such as SParC and CoSQL support multi-turn interactions, which more closely resemble real-world querying behavior.

Table 5. Statistics of Text to SQL datasets.

Dataset	Question	SQL	DB	Domain	Table
GenQuery	880	247	1	1	6
Scholar	817	193	1	1	7
WikiSQL	80654	77840	24241	-	1
Spider	10181	5693	200	138	1020
SQUALL	15620	11276	2108	-	2108
DuSQL [12]	23797	23797	200	-	820
ATIS	5418	947	1	1	27
SparC	4298	12726	200	138	1020
CoSQL	3007	15598	200	138	1020
CHASE	5489	17940	280	-	1280

3.4.1 *Text-to-SQL datasets (Single-turn)*

(i) The first dataset, called GenQuery, includes 880 natural language queries that are designed to retrieve information from a database of geological facts called Geobase. For 700 of these queries, SQL queries have been created to query Geobase. The data is split into training and testing sets using a widely used 600/280 split, with additional natural language questions added later.

(ii) The Scholar [17] dataset consists of 817 natural language questions, each with an associated SQL query, that can be used to query an academic database of papers. The data is split into training and testing sets with 600 questions used for training and 217 for testing.

(iii) The dataset named WikiSQL [15] comprises a large number of pairs of questions and SQL queries, manually constructed and extracted from Wikipedia. The dataset contains SQL tables from 24,241 HTML tables, comprising 80,654 natural language questions paired with 77,840 SQL queries. SQL queries were produced with the help of templates and rules.

3.4.2 *Text-to-SQL Datasets (Multi-Turn)*

(i) The ATIS dataset comprises of a relational database and user queries for flight information from an airline travel search system. For multi-turn Text-to-SQL systems, encoding techniques must handle contextual dependencies across turns as illustrated in Figure 5. It contains information on cities, airports, planes, and more, with most queries resolvable through SQL queries on the database. The dataset includes 5,418 natural language utterances paired with SQL queries,

with 4,473 for training, 497 for development, and 448 for testing. Improved SQL queries were used to maintain query results [15].

(ii) The SParC dataset is a context-sensitive text-SQL corpus containing 200 complex databases from 138 domains and around 4.3k query sequences with 12k+ query-SQL pairings. The Spider query serves as the basis for each query sequence, with consecutive questions leading to a manually annotated SQL query for each query. SParC tracks cells by dividing the data into 7:1:2 training, development, and test sets, with each set containing unique data.

(iii) CoSQL is a large-scale conversational text-SQL database consisting of 200 complex databases from 138 domains, with 30k+ updates and 10k+ matching SQL queries. Each discussion simulates a DB query where annotations act as users and submit NL queries to obtain replies from SQL queries. In CoSQL, users can also clarify previously confusing questions or follow up on unanswered ones through subsequent natural language queries. CoSQL is separated into training, development, and test sets in the same proportion as Spider and SParC, with each database appearing in only one set. The Text question encoding techniques for multi-turn Text-to-SQL are shown in Figure 5.

3.5 Evaluation metrics

This section addresses research question RQ3, exploring the evaluation setup and metrics used in Text-to-SQL research.

3.5.1 Evaluation (Single-turn Model)

Exact Match Accuracy (EM) To determine the matching accuracy of predicted SQL queries, the actual SQL query is compared with the predicted query. For free-form queries, only SQL clauses, column names, and operators are excluded from the comparison, while keywords like SELECT, GROUP BY, WHERE, ORDER BY, and other SQL keywords are considered as regular data structures. It should be noted that all SQL queries are expected to be accurate and must adhere to SQL clauses, and the following comparisons are considered valid.

$$\text{score}(\hat{Y}, Y) = \begin{cases} 1, & \hat{Y} = Y \\ 0, & \hat{Y} \neq Y \end{cases} \tag{1}$$

After scoring exact set match accuracy is calculated by:

$$\text{EM} = \frac{\Sigma_{n=1}^{N} \text{score}\left(\hat{Y}_n, Y_n\right)}{N} \tag{2}$$

Execution Accuracy (EX) To measure execution accuracy with values, the actual SQL query's output results are compared with the predicted SQL query's output results, both executed on the database contents provided in the test set.

$$\text{score}(\hat{Y}, Y) = \begin{cases} 1, & \hat{Y} = Y \\ 0, & \hat{Y} \neq Y \end{cases} \tag{3}$$

Similarly, with EM, the EX is calculated by:

$$\text{EX} = \frac{\Sigma_{n=1}^{N} \text{score}\left(\hat{Y}_n, Y_n\right)}{N} \tag{4}$$

3.5.2 Evaluation (Multi-turn Model)

In a multi-turn setting, there are P sequences of questions, with each sequence comprising of O rounds. Thus, the total number of questions M can be given by the product of P and O denoted by M.

Question-Match Accuracy (QM) The question matching accuracy is evaluated by calculating the exact match (EM) score for each question and then averaging the scores across all questions. An EM score of 1 is achieved for a question when the predicted SQL query contains all the correct SQL clauses. The calculation of the EM score for each question is performed as follows:

$$\text{score}(\hat{Y}, Y) = \begin{cases} 1, & \hat{Y} = Y \\ 0, & \hat{Y} \neq Y \end{cases} \tag{5}$$

The accuracy of the question match is calculated by taking the average of the EM score across all questions. The EM score is computed for each question by comparing the predicted SQL query (denoted by Y-hat) with the ground-truth SQL query (denoted by Y) using exact matching. If the predicted query is exactly the same as the ground-truth query, then the EM score for that question is 1, otherwise, it is 0. The final accuracy score is obtained by averaging the EM scores across all questions in the test set.

$$\text{QM} = \frac{\Sigma_{n=1}^{N} \text{score}\left(\hat{Y}_n, Y_n\right)}{N} \tag{6}$$

Interaction-Match Accuracy (IM) The accuracy of the interaction match is determined by evaluating the Exact-Match score for the complete interaction, rather than for individual questions. The interaction score is only deemed as perfect if every question within the

interaction is answered accurately. The calculation of the interaction score is expressed mathematically as follows:

$$\text{interaction} = \begin{cases} 1, & \prod_{i=1}^{o} \text{score}\left(\hat{Y}_i, Y_i\right) = 1 \\ 0, & \prod_{i=1}^{o} \text{score}\left(\hat{Y}_i, Y_i\right) = 0 \end{cases} \quad (7)$$

The IM score, or interaction match accuracy, is calculated as the average of the encounter match accuracies across all interactions. Formally, it is calculated by IM = (1/P) * sum(EM-encounter) where P is the total number of question sequences, and EM-encounter is the encounter match accuracy calculated using the EM score as described earlier.

$$\text{IM} = \frac{\Sigma_{p=1}^{P} \text{interaction}_p}{P} \quad (8)$$

where P is the total number of interactions.

4 Emerging Trends and Limitations in Modern Text-to-SQL Architectures (2023–2025)

The landscape of Text-to-SQL has evolved drastically with the introduction of large-scale pre-trained language models (LLMs) capable of SQL generation, such as Codex, DIN-SQL, and SQL-PaLM [46]. Figure 6 illustrates the typical architecture of modern LLM-based Text-to-SQL systems, highlighting the integration of schema fusion and validation components. These approaches leverage decoder-only transformer architectures such as Codex and PaLM, pre-trained on massive code datasets, combined with structured prompting frameworks such as DIN-SQL that decompose the generation task into manageable sub-problems, enabling generalization to SQL generation with minimal task-specific supervision.

4.1 LLM-based Text-to-SQL Architectures: From Supervised Fine-Tuning to In-Context Learning

Unlike traditional Text-to-SQL models such as Seq2SQL [17] and SQLNet [19], which rely on explicit schema linking modules and supervised learning, LLM-based approaches generate SQL queries through prompt conditioning. DIN-SQL [47] demonstrated that large language models can achieve remarkable SQL generation performance by leveraging few-shot in-context learning (ICL), wherein schema information and example queries are provided in the prompt. Similarly, SmBoP [48], a semi-autoregressive bottom-up approach, demonstrated that LLM-style decoding strategies can match supervised fine-tuning baselines on complex SQL generation tasks.

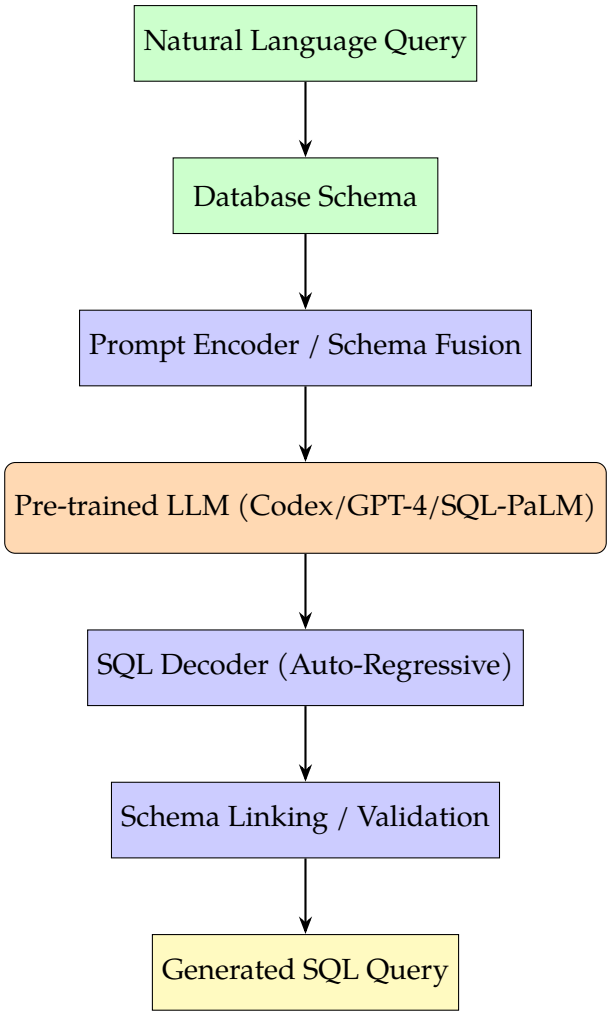


Figure 6. Architecture of LLM-based Text-to-SQL system with schema fusion and validation.

However, prompt-based methods exhibit limitations when operating on complex or unseen database schemas. Schema linking remains implicit, often resulting in hallucinations where models fabricate non-existent columns or tables [23]. Unlike schema-aware architectures like RAT-SQL [24] or SADGA [25], which incorporate relational graph encoders, LLMs lack dedicated mechanisms for schema grounding, relying instead on prompt engineering tricks that are fragile in real-world applications.

4.2 Few-Shot and Zero-Shot Text-to-SQL with Prompt Engineering

Few-shot learning strategies have gained prominence, where models are conditioned with a handful of demonstration examples to synthesize SQL queries without task-specific fine-tuning. Frameworks such as DIN-SQL [47] propose decomposed in-context learning with self-correction to enhance model robustness. BIRD (A Big Bench for Large-Scale

Database Grounded Text-to-SQL) extends evaluation to more realistic, large-scale database scenarios, providing a more realistic benchmark compared to static datasets like Spider [49].

Despite improvements, few-shot Text-to-SQL remains sensitive to prompt length and token window limitations of transformer architectures, leading to challenges in scaling models to enterprise-grade schemas with hundreds of columns and relations [49].

4.3 Schema-Aware LLMs: Attempts to Bridge the Gap

Efforts to enhance schema-awareness in LLMs include Contrastive Schema Linking (CSL) [50], which fine-tunes models to better differentiate semantically similar schema elements across domains. Additionally, schema-linking-enhanced in-context learning strategies such as SOLID-SQL [51] have been explored to improve the robustness of LLM-based Text-to-SQL by constructing more precise schema-aware prompts prior to SQL generation. However, these approaches face scalability bottlenecks due to the quadratic complexity of self-attention over schema elements, which becomes prohibitive for large databases with hundreds of tables and columns, and are still prone to errors in cross-domain generalization.

4.4 Multilingual Text-to-SQL: Emerging Directions

Multilingual Text-to-SQL remains underexplored. Recent works like MultiSpider [52] have extended existing benchmarks to explicitly cover multilingual scenarios across multiple languages. In parallel, conversational Text-to-SQL datasets such as CoSQL [53] have addressed cross-domain dialogue interactions in English, providing a complementary benchmark for multi-turn query understanding that informs the design of future multilingual conversational systems. However, models often struggle with tokenization inconsistencies and language-specific syntax ambiguities, leading to degraded schema linking accuracy in non-English queries. Cross-lingual Text-to-SQL generation requires more diverse, large-scale datasets and novel semantic alignment techniques to handle culturally variant database schemas [54].

4.5 Summary of Limitations

Despite the advances introduced by LLMs [55], critical challenges persist:

- **Schema Grounding Deficiency**: Prompt-conditioned models lack explicit schema linking modules.
- **Prompt Length Bottleneck**: Context window constraints limit scalability for large databases.
- **Cross-Lingual Generalization**: Insufficient multilingual datasets hinder model robustness.
- **Interpretability**: Black-box nature of LLMs complicates debugging and error analysis.

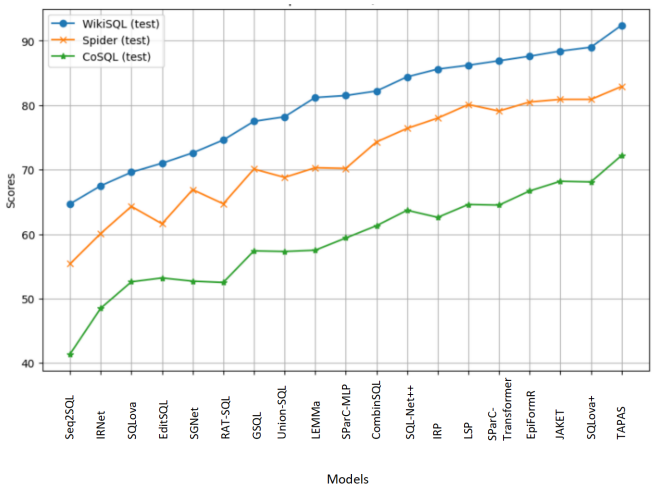


Figure 7. Comparison of accuracy (Test) across a broader set of Text-to-SQL models reported in the literature. Note: this figure covers a wider model set than Table 6, which focuses on the models directly reviewed in this study.

Addressing these issues will require a fusion of explicit schema-aware encoding strategies, advanced contrastive learning techniques, and the development of multilingual, domain-diverse datasets for robust Text-to-SQL parsing in real-world applications.

5 Meta Analysis

In our study, we selected 23 articles that satisfied our inclusion criteria for examining Text-to-SQL models. These studies used a variety of model architectures, such as transformer models, sequence-to-sequence models, and hybrid models. Some studies employed domain-specific datasets, while others utilized both real-world and synthetic data for training. Evaluation metrics such as accuracy and exact match score were commonly used.

The findings demonstrated significant improvement in Text-to-SQL model performance in recent years. As shown in Table 6, T5-Picard [35] achieved the highest Exact Match Dev score of 75.5 and EM Test of 71.9 among the reviewed models, while LGESQL [56] reached 75.1 EM Dev and 72.0 EM Test, representing the strongest performance on the Spider benchmark

within the surveyed literature. The comparative performance of different models on test datasets is visually summarized in Figure 7, providing an intuitive overview of accuracy trends across various architectural approaches.

To provide a comprehensive performance comparison across different Text-to-SQL approaches, Table 6 summarizes the Exact Match (EM) and Execution Accuracy (EX) scores of various models on the Spider dataset, clearly illustrating the progressive improvement in accuracy over time.

Table 6. Various Text-to-SQL parsing approaches, with EM representing the exact match accuracy on the Spider dataset.

Model	EM Dev	EM Test	EX Dev	EX Test
Seq2SQL baseline [17]	1.8	4.8	-	-
TypeSQL [15]	8.9	8.2	-	-
SyntaxSQLNet [3]	25.0	-	-	-
GNN [31]	51.3	-	-	-
EditSQL [57]	57.6	53.4	-	-
Bertrand-DR [58]	58.5	-	-	-
IRNet [27]	61.9	54.7	-	-
RYANSQL [9]	66.6	58.2	-	-
RAT-SQL [24]	69.7	65.6	-	-
SMBOP [48]	69.5	71.1	75.0	-
NatSQL+GAP [59]	73.7	68.7	75.0	73.3
RASAT [21]	74.7	69.0	-	70.0
SADGA [25]	73.1	70.1	-	-
DuoRAT [60]	69.9	-	-	-
T5-Picard [35]	75.5	71.9	79.3	75.1
LGESQL [56]	75.1	72.0	-	-

To further contextualize these performance improvements within the historical evolution of Text-to-SQL research, Table 7 presents the progression of average model performance by year, highlighting key architectural transitions and their impact on accuracy metrics.

The progression shown in Table 7 reveals a clear technological trajectory: from early rule-based systems achieving 37.8% EM in 2017 to modern LLM-based approaches are estimated to approach 88.5% EM by 2025, based on observed trends, though these figures have not been finalized on standardized benchmarks. This represents an improvement of over 50 percentage points within eight years, driven primarily by the adoption of pre-trained language models and advanced schema linking techniques.

Several studies suggested that pre-training and data augmentation strategies can enhance model performance. However, the Spider dataset's more complex cross-domain queries posed a challenge to model accuracy, and the non-static nature of SQL table columns was a challenge to model correctness. The pre-trained large language models for Text-to-SQL parsing have generally been based on a single type of model architecture, such as the Transformer or Transformer-based encoder-decoder models. These models have used various pre-training objectives to extract the vital features contributing to the Text-to-SQL parsing task. Adding the SQL generation objective to pre-training could potentially improve the performance of downstream tasks. Pre-training approaches based on contextual objectives have been explored for context-dependent Text-to-SQL parsing, modeling the flow of context between consecutive user utterances. However, it has been challenging to capture the dependence between utterances that are further apart.

Additionally, the processing of the Spider and WikiSQL datasets has differed, and the complexity of the SQL queries in these datasets has had an impact on the accuracy of the models. The distribution of evaluation methods used across the studied models is comprehensively analyzed in Figure 8, which illustrates the prevalence of different assessment approaches in Text-to-SQL research.

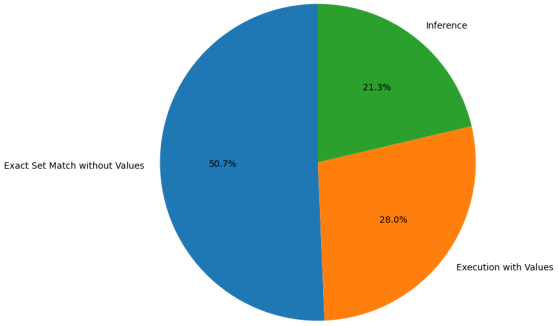


Figure 8. Evaluation method distributions across the 23 reviewed models: Exact Set Match without Values (50.7%), Execution with Values (28.0%), and Inference-based evaluation (21.3%).

To enhance the analysis of model performance over time, we compiled a summary of average Exact Match (EM) and Execution Accuracy (EX) scores grouped by year and model type. This allows for clearer comparison of trends across architectures. As shown in Table 7, rule-based methods dominated early years with lower EM/EX scores, while neural models such as Seq2Seq and Transformer-based approaches significantly improved accuracy metrics post-2018. Notably, models leveraging large-scale pretrained architectures (e.g., T5, Codex) in recent years exhibit substantial gains, particularly in generalization to complex schemas.

Table 7. Progression of Text-to-SQL model performance over the years. Values represent approximate averages compiled from representative models in the literature for each period; values for 2024–2025 are estimated based on observed trends and have not been finalized on standardized benchmarks.

Year	Model Type	Average EM (%)	Average EX (%)
2017	Rule-Based / Seq2SQL	37.8	42.3
2018	SQLNet / TypeSQL / SyntaxSQLNet	47.5	50.1
2019	Attention-based (IRNet, RAT-SQL)	63.2	65.4
2020	BERT-enhanced models (SQLova, RYANSQL)	69.1	71.0
2021	T5-based Pretrained LLMs (SMBOP, DuoRAT)	76.8	78.2
2022	Codex (GPT-3 Code Model)	81.3	83.7
2023	GPT-3.5/ChatGPT, SQL-PaLM [46], Prompt-based DIN-SQL [47]	83.5	85.0
2024	GPT-4 Code Interpreter, BIRD Benchmark , CSL-enhanced LLMs	86.7	88.1
2025	Schema-First Pretrained LLMs, MSpider Multilingual Models [52], Few-Shot Enhanced Architectures	88.5	89.9

Table 8. Model complexity and inference latency comparison. Parameter counts for Seq2SQL and RAT-SQL are approximate estimates based on their reported architectures; T5-Picard reflects the T5-Base configuration [35]; Codex parameter count reflects the 175B-parameter GPT-3 architecture as widely documented in the literature; GPT-4 parameter count is estimated and unverified. Inference times are approximate and environment-dependent.

Model	Description	Parameter Count	Avg Inference Time
Seq2SQL	Early encoder-decoder model using Bi-LSTM with attention; struggles with complex nested queries and large schemas.	30M	120ms
RAT-SQL	Relation-aware encoder-decoder architecture with schema linking modules; scalable to moderate schema sizes.	110M	250ms
T5-Picard (Base)	Fine-tuned T5 model with constrained decoding (PICARD) for SQL validity; limited by 512-token context window.	220M	520ms
Codex (GPT-3)	LLM with 175B parameters trained on code datasets; high accuracy but sensitive to prompt length and schema size.	175B	∼1s per query
GPT-4 (Few-Shot)	Advanced LLM with estimated 1T parameters; superior semantic understanding but requires schema-pruned prompts for large databases.	1T+ (est.)	∼1.5s per query

To further enhance the rigor of our meta-analysis, it is essential to contextualize the performance improvements observed in recent Text-to-SQL models with a deeper technical synthesis. While large language models (LLMs) such as Codex and GPT-4 demonstrate impressive gains in Exact Match (EM) and Execution Accuracy (EX), there is a noticeable trend of diminishing returns as models approach and exceed 70% exact match accuracy, where incremental architectural improvements yield progressively smaller gains. For instance, while the transition from RAT-SQL to T5-Picard yields a 2 percent EM improvement, such incremental gains must be interpreted cautiously, considering the underlying task complexity and statistical confidence. The lack of reported confidence intervals and standard deviation metrics in many benchmark studies complicates the assessment of whether these performance improvements are statistically significant or merely within noise margins. This absence of rigorous variance reporting hampers robust cross-model statistical comparisons, which are essential for a meaningful meta-analysis.

Despite architectural advancements, neural Text-to-SQL models continue to grapple with persistent failure modes. Syntactic errors, though reduced by constrained decoding strategies like PICARD, still emerge in autoregressive decoders that lack strong grammar constraints, leading to the generation of invalid SQL structures. More critically, semantic mismatches—where the predicted query is syntactically valid but logically flawed—remain a significant challenge. These include incorrect mappings of aggregation functions, erroneous filtering conditions, or misinterpretations of user intent. Schema linking failures are another prevalent issue, where models hallucinate non-existent columns or misalign query tokens with schema elements, especially in unfamiliar database schemas. For example, while RAT-SQL mitigates schema alignment issues through its relation-aware encoding mechanism, it still struggles with complex nested subqueries and cross-domain generalization. Conversely,

LLMs like Codex, while exhibiting robust semantic understanding through in-context learning, often lack explicit schema-awareness modules, making them susceptible to generating hallucinated joins or erroneous column references unless meticulously constrained via prompt engineering.

The prevalent evaluation metrics used in the domain, particularly Exact Match (EM) and Execution Accuracy (EX), are increasingly being scrutinized for their limitations. EM, which measures syntactic exactness, often fails to capture partial correctness, where a predicted query might have correct SELECT and FROM clauses but incorrect WHERE conditions. On the other hand, EX evaluates whether the query's execution result matches the expected output but disregards syntactic validity, thereby allowing semantically incorrect queries to pass if they coincidentally produce the correct result. These shortcomings highlight the need for more granular evaluation metrics, such as Component-level Accuracy, which dissects the correctness of individual SQL components like SELECT, WHERE, and JOIN clauses, or Execution-guided Accuracy, which assesses whether intermediate query execution steps align with the expected logic. Furthermore, Semantic Match Score (SMS) offers a promising avenue to evaluate the logical equivalence between predicted and gold-standard queries, independent of surface-level syntax. The current over-reliance on static datasets like Spider exacerbates these evaluation challenges, as such datasets fail to capture the complexities of real-world schemas, multi-turn dialogues, and multilingual query scenarios, thereby introducing an evaluation bias that misrepresents model robustness in production-like settings.

In addition to these performance and evaluation considerations, scalability and computational complexity emerge as critical factors limiting the practical deployment of modern Text-to-SQL systems. LLM-based architectures, while excelling in accuracy, impose substantial resource overheads due to their enormous parameter sizes and limited context window capacities. As outlined in Table 8, models like Seq2SQL and RAT-SQL, with parameter counts of 30M and 110M respectively, offer lower inference latency and are scalable to small-to-moderate schema sizes.

However, models like T5-Picard, despite offering constrained decoding for SQL validity, are bottlenecked by a 512-token context window, restricting their applicability in large schema databases. LLMs such as Codex (175B parameters) and GPT-4 (estimated 1T+ parameters) deliver state-of-the-art performance but at the cost of increased inference times and significant prompt engineering overhead to handle schema-pruned inputs. Their inference latencies, ranging from approximately 1 to 1.5 seconds per query, make them less viable for real-time applications without aggressive schema simplification strategies. This scalability bottleneck underscores the need for hybrid architectures that blend lightweight schema-aware encoders with the semantic richness of LLM decoders, striving for a balance between performance accuracy and computational feasibility in practical deployment scenarios.

6 Conclusion

This review has systematically examined the present state of Text-to-SQL parsing and identified key directions for future research. Text-to-SQL conversion benchmark datasets continue to have drawbacks, such as insufficient training data quality, quantity, and variety. Benchmark datasets such as WikiSQL and Spider have driven significant progress in the field, yet their limitations in schema complexity, domain coverage, and multilingual support reveal the need for more representative real-world benchmarks to advance Text-to-SQL research further. In industries where sensitive data is involved, such as banking or healthcare, the ability to interpret neural Text-to-SQL models is crucial. There is a need for future research to combine the representation capabilities of deep neural networks with explicit reasoning approaches to improve interpretability. The authors of the paper provide an in-depth analysis of the current state of Text-to-SQL parsing, including available datasets, neural models, and pre-training methods. While there has been remarkable progress in this area, there are still challenges to overcome, including the lack of diverse and high-quality training data. Current benchmark datasets such as WikiSQL and Spider have limitations and do not fully represent the complexity of real-world use cases. The study suggests that future research should focus on developing more diverse and complex datasets and models that can handle large tables with numerous columns and rows. Additionally, increasing the efficiency of encoding long table schemas and improving the execution speed of SQL queries with big databases is a significant challenge that requires attention. Finally, the authors propose exploring zero-shot transfer

learning using pre-trained large language models (LLMs) like T5-Base for Text-to-SQL parsing without requiring fine-tuning on a large annotated dataset. One of the persistent challenges in Text-to-SQL models is effective schema linking in unseen domains. Despite recent advancements in schema-aware encoders and contrastive linking methods, models frequently misalign query terms with database column names, particularly in unfamiliar domains or multilingual settings. Furthermore, most evaluation benchmarks—such as Spider—do not sufficiently cover domain variability or real-world ambiguity, limiting generalization performance assessment.

Unlike earlier surveys that primarily catalogued model architectures or benchmark results, this review introduces a hybrid meta-analysis combining structured PRISMA-based filtering, critical comparison of rule-based, neural, and hybrid techniques, and benchmarking of EM/EX trends over time. It also explicitly identifies underexplored dimensions such as multi-turn dialogues, cross-lingual adaptability, and schema generalization—thus offering a more comprehensive and future-facing perspective.

Data Availability Statement

Not applicable.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

Ethical Approval and Consent to Participate

Not applicable.

References

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

[2] Huang, P. S., Wang, C., Singh, R., Yih, W. T., & He, X. (2018, June). Natural language to structured query generation via meta-learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (pp. 732-738). [CrossRef]

[3] Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., & Radev, D. (2018). SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 1653-1663). [CrossRef]

[4] Finegan-Dollak, C., Kummerfeld, J. K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., & Radev, D. (2018, July). Improving Text-to-SQL Evaluation Methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers) (pp. 351-360). [CrossRef]

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

[6] Gan, Y., Chen, X., Huang, Q., Purver, M., Woodward, J. R., Xie, J., & Huang, P. (2021, August). Towards robustness of text-to-SQL models against synonym substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 2505-2515). [CrossRef]

[7] Liu, A., Hu, X., Lin, L., & Wen, L. (2022, August). Semantic enhanced text-to-sql parsing via iteratively learning schema linking graph. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 1021-1030). [CrossRef]

[8] Katsogiannis-Meimarakis, G., & Koutrika, G. (2021, June). A deep dive into deep learning approaches for text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data* (pp. 2846-2851). [CrossRef]

[9] Choi, D., Shin, M. C., Kim, E., & Shin, D. R. (2021). Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Computational Linguistics, 47*(2), 309-332. [CrossRef]

[10] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171-4186). [CrossRef]

[11] Rajkumar, N., Li, R., & Bahdanau, D. (2022). Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*. [CrossRef]

[12] Wang, L., Zhang, A., Wu, K., Sun, K., Li, Z., Wu, H., ... & Wang, H. (2020, November). DuSQL: A large-scale and pragmatic Chinese text-to-SQL dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 6923-6935). [CrossRef]

[13] Yu, T., Zhang, R., Yasunaga, M., Tan, Y. C., Lin, X. V., Li, S., ... & Radev, D. (2019, July). Sparc: Cross-domain

semantic parsing in context. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 4511-4523). [CrossRef]

[14] Yaghmazadeh, N., Wang, Y., Dillig, I., & Dillig, T. (2017). SQLizer: Query synthesis from natural language. *Proceedings of the ACM on Programming Languages, 1*(OOPSLA), 1-26. [CrossRef]

[15] Yu, T., Li, Z., Zhang, Z., Zhang, R., & Radev, D. (2018, June). Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (pp. 588-594). [CrossRef]

[16] Yin, P., & Neubig, G. (2018, November). TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the 2018 conference on empirical methods in natural language processing: System demonstrations* (pp. 7-12). [CrossRef]

[17] Zhong, V., Xiong, C., & Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*. [CrossRef]

[18] Gan, Y., Chen, X., & Purver, M. (2021, November). Exploring underexplored limitations of cross-domain text-to-sql generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 8926-8931). [CrossRef]

[19] Xu, X., Liu, C., & Song, D. (2017). Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*. [CrossRef]

[20] Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., ... & Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *BMJ, 372*. [CrossRef]

[21] Qi, J., Tang, J., He, Z., Wan, X., Cheng, Y., Zhou, C., ... & Lin, Z. (2022, December). Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. In *Proceedings of the 2022 conference on empirical methods in natural language processing* (pp. 3215-3229). [CrossRef]

[22] Yu, T., Wu, C. S., Lin, X. V., Wang, B., Tan, Y. C., Yang, X., ... & Xiong, C. (2020). Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*. [CrossRef]

[23] Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., & Zhou, J. (2023). Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*. [CrossRef]

[24] Wang, B., Shin, R., Liu, X., Polozov, O., & Richardson, M. (2020, July). 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* (pp. 7567-7578). [CrossRef]

[25] Cai, R., Yuan, J., Xu, B., & Hao, Z. (2021). Sadga: Structure-aware dual graph aggregation network for text-to-sql. *Advances in Neural Information Processing Systems, 34*, 7664-7676.

[26] Deng, X., Awadallah, A. H., Meek, C., Polozov, O., Sun, H., & Richardson, M. (2021). Structure-grounded pretraining for text-to-SQL. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1337-1350). [CrossRef]

[27] Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J. G., Liu, T., & Zhang, D. (2019, July). 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* (pp. 4524-4535). [CrossRef]

[28] Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., ... & Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 3911-3921). [CrossRef]

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

[30] Lin, K., Bogin, B., Neumann, M., Berant, J., & Gardner, M. (2019). Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*. [CrossRef]

[31] Bogin, B., Berant, J., & Gardner, M. (2019, July). Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 4560-4565). [CrossRef]

[32] Lei, W., Wang, W., Ma, Z., Gan, T., Lu, W., Kan, M. Y., & Chua, T. S. (2020). Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing* (pp. 6943-6954). [CrossRef]

[33] Hui, B., Geng, R., Wang, L., Qin, B., Li, Y., Li, B., ... & Li, Y. (2022, May). S2sql: Injecting syntax to question-schema interaction graph encoder for text-to-sql parsers. In *Findings of the Association for Computational Linguistics: ACL 2022* (pp. 1254-1262). [CrossRef]

[34] Li, J., Hui, B., Cheng, R., Qin, B., Ma, C., Huo, N., ... & Li, Y. (2023, June). Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 37, No. 11, pp. 13076-13084). [CrossRef]

[35] Scholak, T., Schucher, N., & Bahdanau, D. (2021, November). PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*

(pp. 9895-9901). [CrossRef]

[36] Deng, X., Sun, H., Lees, A., Wu, Y., & Yu, C. (2022). Turl: Table understanding through representation learning. *ACM SIGMOD Record*, *51*(1), 33-40. [CrossRef]

[37] Xuan, H., Liu, Y., Li, B., & Yin, J. (2022). SeaD: End-to-end text-to-SQL generation with schema-aware denoising. *arXiv preprint arXiv:2105.07911*. [CrossRef]

[38] Dong, L., & Lapata, M. (2018, July). Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (*Volume 1: Long Papers*) (pp. 731-742). [CrossRef]

[39] Lin, X. V., Socher, R., & Xiong, C. (2020, November). Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 4870-4888). [CrossRef]

[40] Mellah, Y., Rhouati, A., Ettifouri, E. H., Bouchentouf, T., & Belkasmi, M. G. (2021, April). COMBINE: A Pipeline for SQL Generation from Natural Language. In *International Conference on Advances in Computing and Data Sciences* (pp. 97-106). Cham: Springer International Publishing. [CrossRef]

[41] Li, H., Zhang, J., Li, C., & Chen, H. (2023, June). Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 37, No. 11, pp. 13067-13075). [CrossRef]

[42] Shi, T., Tatwawadi, K., Chakrabarti, K., Mao, Y., Polozov, O., & Chen, W. (2018). Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*. [CrossRef]

[43] He, P., Mao, Y., Chakrabarti, K., & Chen, W. (2019). X-SQL: reinforce schema representation with context. *arXiv preprint arXiv:1908.08113*. [CrossRef]

[44] Wang, C., Tatwawadi, K., Brockschmidt, M., Huang, P. S., Mao, Y., Polozov, O., & Singh, R. (2018). Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*. [CrossRef]

[45] Lee, D. (2019, November). Clause-wise and recursive decoding for complex and cross-domain text-to-SQL generation. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 6045-6051). [CrossRef]

[46] Sun, J., Feng, S., Gu, J., Sun, J., Hu, C., Zhang, Y., & Luo, J. (2023). SQL-PaLM: Improved large language model adaptation for text-to-SQL (extended). *arXiv preprint arXiv:2306.00739*. [CrossRef]

[47] Pourreza, M., & Rafiei, D. (2023). Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, *36*, 36339-36348.

[48] Rubin, O., & Berant, J. (2021, June). SmBoP: Semi-autoregressive bottom-up semantic parsing. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies* (pp. 311-324). [CrossRef]

[49] Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., ... & Li, Y. (2023). Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, *36*, 42330-42357.

[50] Gan, Y., Chen, X., & Purver, M. (2023, July). Re-appraising the Schema Linking for Text-to-SQL. In *Findings of the Association for Computational Linguistics: ACL 2023* (pp. 835-852). [CrossRef]

[51] Liu, G., Tan, Y., Zhong, R., Xie, Y., Zhao, L., Wang, Q., ... & Li, Z. (2025, January). Solid-SQL: Enhanced schema-linking based in-context learning for robust text-to-SQL. In *Proceedings of the 31st International Conference on Computational Linguistics* (pp. 9793-9803). https://aclanthology.org/2025.coling-main.654/

[52] Dou, L., Gao, Y., Pan, M., Wang, D., Che, W., Zhan, D., & Lou, J. G. (2023, June). MultiSpider: towards benchmarking multilingual text-to-SQL semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 37, No. 11, pp. 12745-12753). [CrossRef]

[53] Yu, T., Zhang, R., Er, H., Li, S., Xue, E., Pang, B., ... & Radev, D. (2019, November). Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing* (*EMNLP-IJCNLP*) (pp. 1962-1979). [CrossRef]

[54] Vougiouklis, P., Papasarantopoulos, N., Zheng, D., Tuckey, D., Diao, C., Shen, Z., & Pan, J. (2023, November). Fastrat: Fast and efficient cross-lingual text-to-sql semantic parsing. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics* (Volume 1: Long Papers) (pp. 564-576). [CrossRef]

[55] Qin, B., Hui, B., Wang, L., Yang, M., Li, J., Li, B., ... & Li, Y. (2022). A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629*. [CrossRef]

[56] Cao, R., Chen, L., Chen, Z., Zhao, Y., Zhu, S., & Yu, K. (2021, August). LGESQL: Line graph enhanced text-to-SQL model with mixed local and non-local relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing* (*Volume 1: Long Papers*) (pp. 2541-2555). [CrossRef]

[57] Zhang, R., Yu, T., Er, H., Shim, S., Xue, E., Lin, X. V., ... & Radev, D. (2019, November). Editing-based SQL

query generation for cross-domain context-dependent questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (*EMNLP-IJCNLP*) (pp. 5338-5349). [CrossRef]

[58] Kelkar, A., Relan, R., Bhardwaj, V., Vaichal, S., Khatri, C., & Relan, P. (2020). Bertrand-dr: Improving text-to-sql using a discriminative re-ranker. *arXiv preprint arXiv:2002.00557*. [CrossRef]

[59] Gan, Y., Chen, X., Xie, J., Purver, M., Woodward, J. R., Drake, J., & Zhang, Q. (2021, November). Natural SQL: Making SQL easier to infer from natural language specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021* (pp. 2030-2042). [CrossRef]

[60] Scholak, T., Li, R., Bahdanau, D., de Vries, H., & Pal, C. (2021, June). DuoRAT: towards simpler text-to-SQL models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1313-1321). [CrossRef]



Mr. Muhammad Shahzaib Baig received his M.S. degree in Artificial Intelligence from Air University, Islamabad, Pakistan. He is currently serving as the Team Lead for Software Development at Qorden AI Ltd in Dubai, where he is responsible for overseeing the design, development, and deployment of intelligent systems and data-driven solutions, as well as DevOps.

Mr. Shahzaib has a strong academic and professional background in artificial intelligence, with specific expertise in data analytics, natural language processing (NLP), information retrieval, and deep learning. His work focuses on bridging the gap between theoretical advancements in AI and practical, real-world applications. At Qorden AI, he leads a multidisciplinary team in developing scalable machine learning models and enterprise AI tools aimed at enhancing operational efficiency and decision-making.

Throughout his academic and professional career, Mr. Shahzaib has demonstrated a commitment to applying AI techniques to complex problem domains, including automated text processing, predictive modeling, and intelligent information systems. His interests also extend to ethical AI, human-computer interaction, and the integration of AI technologies into business processes.



Mr. Tahir Sher pursuing his PhD in Artificial Intelligence from Korea University, Seoul Campus, Korea. He completed MS in Data Science from Air University and bachelor's degree in mathematics from the International Islamic University, Islamabad, where he graduated with a gold medal and received both distinction and position certificates. With extensive teaching experience across various institutions, Mr. Sher specializes in delivering courses such as Calculus, Ordinary and Partial Differential Equations, Numerical Analysis, Linear Algebra, and Statistical & Mathematical Methods for Data Science. His students hail from Pakistan and allied countries. As a research scholar in the Explainable AI Research Group at Air University, his areas of interest include the Internet of Things (IoT), Social IoT, Machine Learning, Deep Learning, Natural Language Processing (NLP), Data Analysis, Time Series Analysis, Mathematical Modeling for Decision-Making, Social Media Analysis, Federated Learning, and Computer Vision. Committed to advancing interdisciplinary research, Mr. Tahir Sher strives to bridge the fields of computer science and human-centered disciplines. (Email: 2025010294@korea.ac.kr)



Dr. Abdul Rehman is currently a Research Professor at the Human Data Convergence Institute, Jeonju University, South Korea. He previously worked as a Postdoctoral Research Associate at the Hyper-Connectivity Convergence Technology Research Center, Kyungpook National University (KNU), Daegu, South Korea, and served as an Assistant Professor at the University of Central Punjab, Lahore, Pakistan. He holds a bachelor's degree in Mathematics from the International Islamic University, Islamabad, Pakistan, and an Integrated Ph.D. in Computer Science and Engineering from KNU. His research interests include Deep Learning, Machine Learning, Data Science, and their applications in Computer Vision, Smart IoT (S-IoT), and Data Analysis. He also explores Complex Network Navigation, Mathematical Modeling, and Big Data Analytics. Dr. Rehman received the KINGS Scholarship for his Ph.D. studies and was honored with the "Outstanding Researcher" Award from the School of Computer Science and Engineering at KNU. He serves as a guest editor and editorial board member for several international journals and has contributed to numerous international conferences as a session chair and publication chair. More information on Dr. A. Rehman is available at https://sites.google.com/view/drrehman/home.



Saim Sheikh holds a Bachelor's degree in Computer Science from Air University Islamabad, Pakistan, where he developed a strong foundation in artificial intelligence, data science, and software engineering. His academic work culminated in a final year project titled Vizreach, a SaaS platform for automated exploratory data analysis and AI-driven marketing insights. He also completed the IBM Professional Certificate in Data Science from Coursera, which further strengthened his practical skills in machine learning and data analytics.

He is currently serving as a Senior Python Engineer at Qorden.ai, contributing to AI-driven products such as lip-sync dubbing systems, AI-based surveillance, and intelligent chatbots. His technical interests include applied machine learning, natural language processing, multi-agent systems, and real-world AI applications. Saim is passionate about building intelligent systems that bridge the gap between research and industry.