From Natural Language to SQL: A Survey on LLM-Based Text-to-SQL Research and Applications

A THESIS SUBMITTED IN PARTIAL
FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE

Submitted by

RISHABH GOPAL SONI

(2K23/AFI/23)

Under the Supervision of
Dr. NIPUN BANSAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

May, 2025

ACKNOWLEDGEMENTS

I have taken efforts in this survey paper. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to **Dr. Nipun Bansal** for his guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing this review paper. I would like to express my gratitude towards the **Head of the Department (Computer Science and Engineering, Delhi Technological University)** for their kind cooperation and encouragement which helped me in the completion of this research survey. I would like to express my special gratitude and thanks to all the Computer Science and Engineering staff for giving me such attention and time.

My thanks and appreciation also go to my colleague in writing the survey paper and the people who have willingly helped me out with their abilities.

Rishabh Gopal Soni
30 May 2025

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CANDIDATE'S DECLARATION

I, Rishabh Gopal Soni, Roll No. 2K23/AFI/23 student of M.Tech (Artificial Intelligence), hereby certify that the work which is being presented in the thesis entitled "**From Natural Language to SQL: A Survey on LLM-Based Text-to-SQL Research and Applications**" in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Artificial Intelligence in the Department of Computer Science and Engineering, Delhi Technological University is an authentic record of my own work carried out during the period from August 2023 to Jun 2025 under the supervision of Dr Nipun Bansal, Asst Prof, Dept of Computer Science and Engineering. The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Place: Delhi **Candidate's Signature**

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CERTIFICATE

Certified that **Rishabh Gopal Soni** (Roll No. 2K23/AFI/23) has carried out the research work presented in the thesis titled "**From Natural Language to SQL: A Survey on LLM-Based Text-to-SQL Research and Applications**", for the award of Degree of Master of Technology from Department of Computer Science and Engineering, Delhi Technological University, Delhi under my supervision. The thesis embodies result of original work and studies are carried out by the student himself and the contents of the thesis do not form the basis for the award of any other degree for the candidate or submit else from the any other University /Institution.

Dr. Nipun Bansal
(Supervisor)

Department of CSE

Date: 30 May 2025

Delhi Technological University

ABSTRACT

Transforming natural language queries into precise SQL queries, a process referred to as Text-to-SQL or Natural Language to SQL (NL-to-SQL), continues to be a complex and challenging problem. This is due to the fact that interpreting user intent, supporting various database schema, and producing proper SQL syntax are challenging tasks.

The application of conventional methods that involve rule-based methods and neural networks has improved significantly, and Pre-trained language models (PLMs) have helped things move forward much faster. However, as the complexity of the natural language query and database schema increases, PLMs with limited parameter numbers lose their accuracy. Such constraints have led to the development of more advanced and domain-specific optimization methods, which in turn limit their generalizability.In recent years, Large language models (LLMs) have shown phenomenal ability to process natural language and offer promising new directions for improving text-to-SQL systems.

This survey presents an extensive overview of methods employing LLMs for text-to-SQL translation. We first outline the history and key technical challenges in the area and then survey key datasets like Spider, WikiSQL, BIRD, and CoSQL, which have played a major role in evaluation efforts. We then survey recent advances, with focus on the role of LLMs and their influence on system effectiveness. Through the identification of recent trends and open research challenges, this paper hopes to guide and stimulate future research towards improving the development of more generalizable and reliable text-to-SQL system based on LLMs.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations

DL	Deep Learning
LLM	Large Language Model
AI	Artificial Intelligence
ML	Machine Learning
DNN	Deep Neural Network
PLM	Pre-trained Language Model
LSTM	Long Short Term Memory
NL	Natural Language
SQL	Structured Query Language

CHAPTER 1

INTRODUCTION

The question-answering in natural language over the database, as performed by Text-to-SQL, is a fundamental task of the research field of natural language processing (NLP). The technology is a form of interaction with structured data in natural language, beneficial particularly to non-tech users, and is the foundation for natural language interfaces to databases (NLIDBs) [15].

Early work in this field concentrated on rule-based systems employing manually designed templates and grammars to translate questions into SQL. Extremely efficient in simple and fixed situations, these systems faltered as they could not keep up with increasing complexity and diversity of requests from users and database schemata. Manual design of rules for every application was time-consuming and impractical [9]. With the shortfalls of rule-based approaches increasingly evident—especially in dealing with complex and heterogeneous user questions—science turned towards neural network-based approaches. Deep learning introduced a significant leap forward, with sequence-to-sequence [25] [19] models proving to be an extremely promising approach. These models significantly improved mapping natural language inputs to relating SQL queries, with increased flexibility and improved generalizability over various linguistic forms [3].

To surmount these limitations, researchers employed deep learning methods. Pre-trained language models (PLMs) such as BERT [5] and T5 [17], as neural networks, were exemplary in semantic comprehension for text-to-SQL tasks. The models were trained in an unsupervised manner and integrated features such as table encoding and schema linking. PLMs were incapable of dealing with complicated queries over wide spans of domains due to them possessing limited model sizes and being contextually sensitive [2].

The advent of large language models like GPT-3 [24], Codex [11], and GPT-4 [7] has been a big driving force for the field. Being large and generalizable, LLMs provide new avenues towards text-to-SQL query generation. LLMs are well-suited to methods like in-context learning and fine-tuning, which allow them to generalize with little task-specific training. Further, LLMs have the ability to minimize hallucination by relying on real database content and thereby becoming promising tools for building reliable and scalable NLIDB systems [16].

Despite these advancements, there are still several issues remaining such as multi-turn conversation interpretation, domain generalizability, and proper synthesis of complex or nested SQL queries. Large datasets labeled by humans such as Spider [23], WikiSQL [6], BIRD [12], and CoSQL [22] have been crucial in pushing the research forward by providing platforms to test the models on a variety of challenging scenario

This survey provides a comprehensive overview of the evolution of text-to-SQL methods, reflecting the transition from traditional to PLM- and LLM-based systems. We discuss basic ideas, main challenges, noteworthy datasets, evaluation measures, and recent best practices in modeling methods. Observing advances in text-to-SQL research, in this article we have emphasized crucial improvements and recommended areas to be further advanced.

MOTIVATION

The impetus for text-to-SQL research results from the desire to take organized data to the masses, most prominently non-technical users who otherwise would not be able to use database query languages. Although early rule-based systems performed well for straightforward and static cases, they soon became unworkable as user queries and database schema increased in diversity and complexity. The advent of deep learning and pre-trained language models (PLMs) such as BERT and T5 improved the ability to understand natural language and translate to SQL but was not good at generalizing over domains and nested or complex queries.

The field has been greatly advanced with the emergence of large language models such as GPT-3 and GPT-4 that are more responsive, precise, and adaptable through in-context learning and low task-specific training. However, there are issues such as serving multi-turn conversation, possessing strong domain adaptation, and being capable of producing advanced SQL queries consistently. Overcoming these roadblocks is key to building natural language interfaces to databases that are both powerful and intuitive, effectively democratizing access to data and enabling more individuals to access structured information in plain language.

CHAPTER 2
EVOLUTION OF TEXT-TO-SQL APPROACHES

The NLP has made excellent improvement in the text-to-SQL space, moving from traditional techniques to deep learning strategies and, currently, to the application of large language models (LLMs) and pre-trained language models (PLMs). An outline of this evolutionary path is shown in Figure 1.

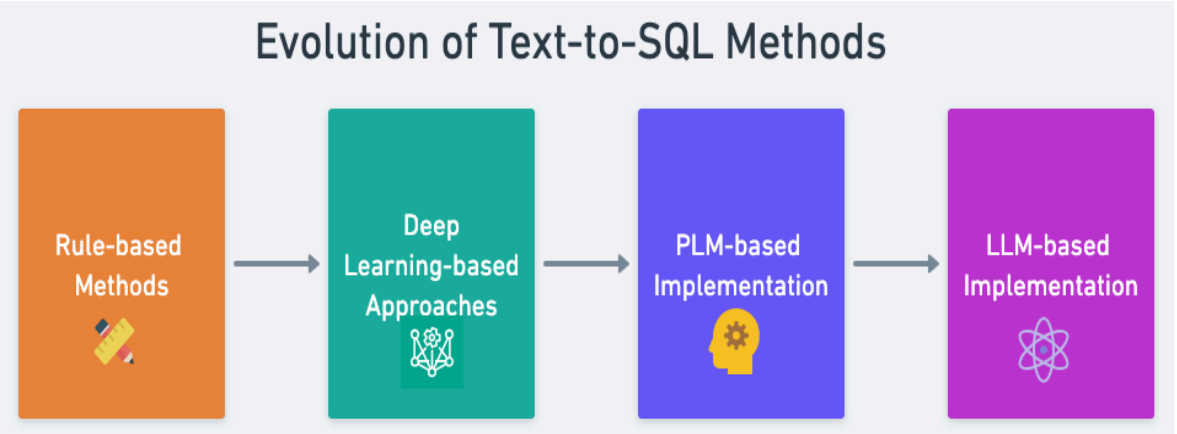


Fig. 1. Advancements in Text-to-SQL Frameworks With Time

2.1 Rule-based Methods

Initial natural language to SQL systems worked on the premise of rule-based systems, converting natural language queries to SQL through hand-crafted rules and heuristics [4]. The systems were effective in generating syntactically correct SQL queries based on pre-designed templates and strict grammatical rules, thereby conforming to SQL structure and syntax. Their determinism provided consistent results for known input configurations, thereby reducing query variability. These systems, however, were incapable of interpreting complex or uncertain natural language queries, particularly nested clauses, coreferences, or ellipses, frequently mapping these features into database schema entities like columns or tables incorrectly. Relying on hand-crafted rules hindered them from adapting to new question styles or schemas in the database, requiring frequent changes across different domains. In addition, mapping complex schemas covering multi-table relationships into rules helpful for consumption took time and was error-prone, scalability being an important issue [9].

2.2 Deep Learning-based Approaches

Deep learning has greatly improved text-to-SQL systems through the application of models such as sequence-to-sequence [25] and encoder-decoder models, including LSTMs [21] and transformers. These facilitate the conversion of natural language inputs into SQL queries, hence providing more flexibility compared to conventional rule-based systems. Nevertheless, they tend to have difficulty in producing either incomplete or syntactically incorrect SQL statements, especially when handling complex operations such as nested subqueries and window functions, which are poorly covered in training data [19].

2.3 PLM-based Implementation

Pre-trained language models (PLM) are now able to efficiently handle Text-to-SQL tasks by tapping into the extensive linguistic and semantic understanding they acquired while pre-training. In order to produce correct SQL queries, previous PLM models like BERT [5] and RoBERTa[17] required fine-tuning the models on conventional text-to-SQL datasets. Through fine-tuning, researchers were able to improve SQL generation from rich semantic representations and language awareness such as PLMs had when pre-trained on big text corpora. Such PLMs, pretrained on big text corpora, had learned well about language and meaning, which the researchers leveraged through fine-tuning to improve SQL generation. Another research direction has been to enhance PLMs with schema information in a way that they would better understand database schema by representing relations and constraints in schemas. This allows schema-aware PLMs to produce more executable SQL queries. While PLMs comparatively perform better than either general deep learning methods in producing correct SQL, they still do not perform well in complex operations like outer joins or aggregations and sometimes produce syntactically incorrect queries. Also, their performance is greatly compromised in cross-domain environments where database schemas or vocabularies are different from training data and need expensive fine-tuning to learn new domains [20].

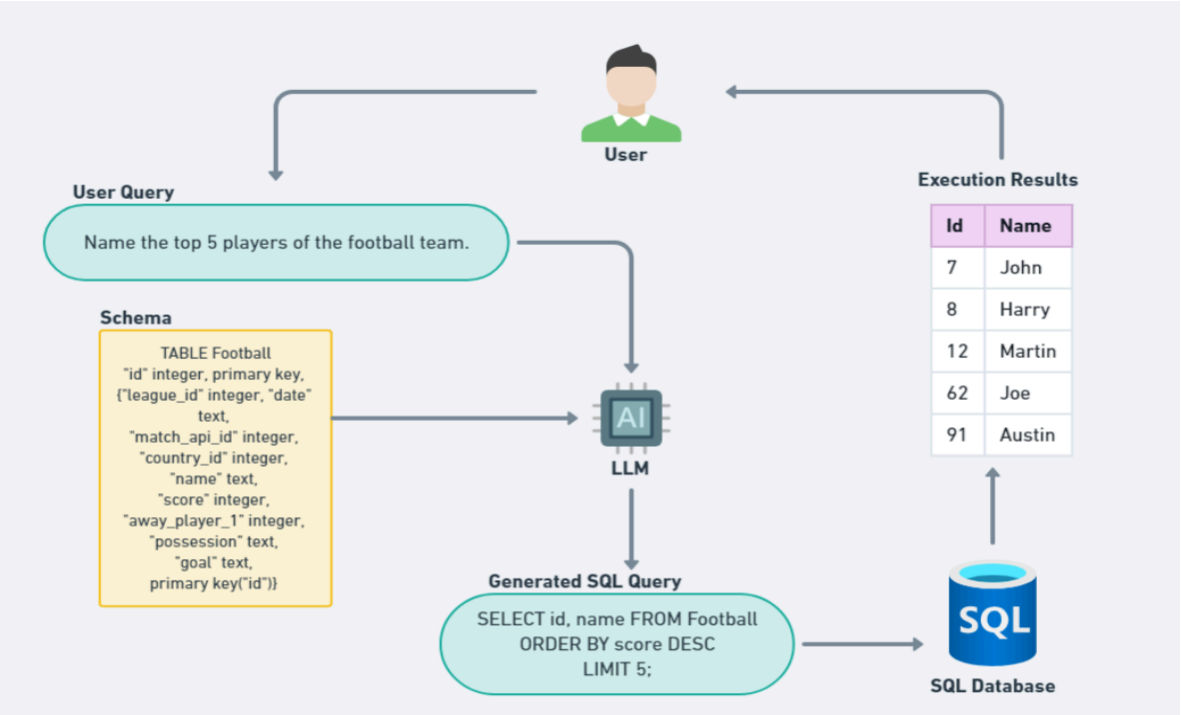


Fig. 2. LLM Framework for Text-to-SQL

2.4 LLM-based Implementation

Large language models(LLM), such as the GPT series [7] [24], have gained widespread attention for their ability to generate fluent and coherent text. Their potential in text-to-SQL tasks is increasingly being explored, leveraging their vast knowledge base and advanced generation capabilities. Common approaches include using prompt engineering to guide proprietary LLMs in generating SQL queries or fine-tuning open-source LLMs on text-to-SQL datasets [16]. Although the use of LLMs in text-to-SQL is still a relatively new area of research, it holds significant promise for further advancements. Current efforts focus on harnessing the reasoning abilities of LLMs ,integrating domain-specific knowledge, and developing more efficient fine-tuning methods [11].

CHAPTER 3
PIPELINE OF TEXT-TO-SQL

Text-to-SQL models are designed to convert everyday language questions into SQL statements, allowing users to interact with databases without needing to understand SQL commands. As seen in Figure 3, these systems are constructed using several essential elements.

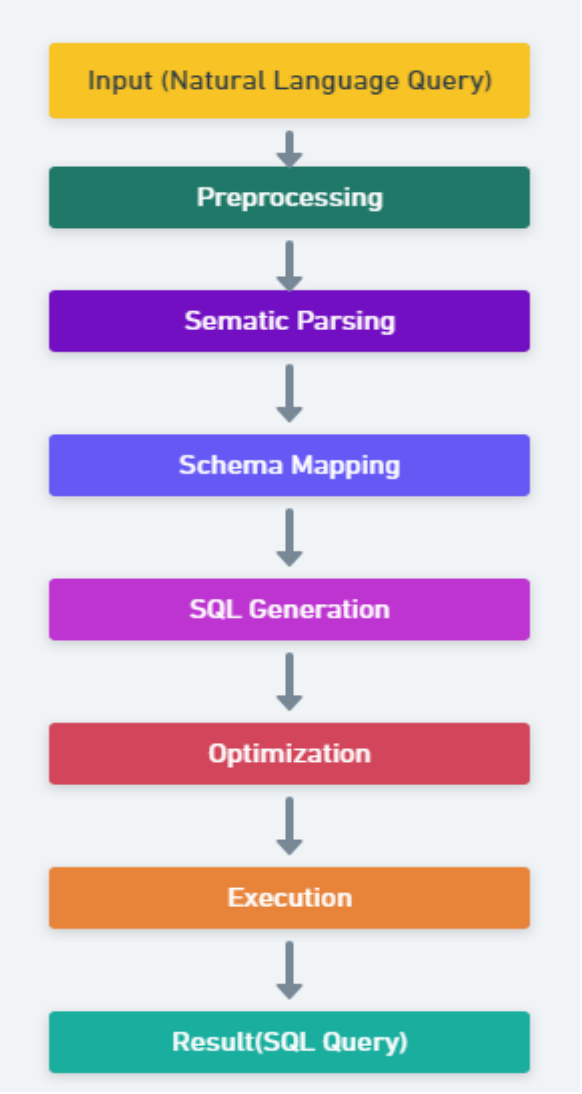


Fig. 3. Overview of the Text-to-SQL Workflow

- **Natural Language Query Understanding (NLQU)**

  The goal of this component is to comprehend the meaning and intent of the user's query by evaluating and interpreting it. To analyse the structure and semantics of the input, it makes use of methods including tokenisation, tagging part-of-speech, and syntactic parsing.

- **Schema Mapping**

  In this stage, words of the natural language query are mapped to the relevant tables and columns in the database structure. To properly match user intent with the database structure, accurate schema mapping is necessary.

- **Semantic Parsing**

  In this stage, the system captures the user's meaning by forming a logical version of their query, which is key to generating the correct SQL.

- **SQL Generation**

  The final phase focuses on converting the structured form into a valid and runnable SQL command. This requires a solid grasp of SQL conventions and ensuring the generated statement aligns properly with the database design [7].

Ongoing developments in artificial intelligence, especially in deep neural networks and language understanding, have greatly enhanced the capabilities of NL-to-SQL models [15]. The adoption of large-scale pre-trained language systems has improved their skill in handling complex questions and producing accurate SQL outputs.

However, challenges persist, particularly in managing complex queries and ensuring cross-domain adaptability. Current research efforts are aimed at enhancing the robustness and flexibility of text-to-SQL systems to overcome these obstacles.

CHAPTER 4
RECENT BENCHMARKS, MODELS, AND DATASETS

Evaluating text-to-SQL solutions requires comprehensive datasets and strong evaluation standards, with several prominent examples available in the field.

4.1 Benchmark Datasets

- **Spider**

  This benchmark is a large-scale, diverse, and challenging text-to-SQL dataset created to test how wellmodels generalize across various database schemas and query formats [23].

- **Spider 2.0**

  Expanding upon its predecessor, this updated benchmark includes 632 challenging NL-to-database tasks derived from actual business use cases. These datasets, typically stored on cloud platforms such as Snowflake and BigQuery, feature schemas with more than 1,000 fields. Spider 2.0 pushes the limits by demanding that models process multi-step SQL sequences exceeding100 lines, making it especially suitable for evaluating large models in enterprise scenarios [8].

- **CSpider**

  As the Chinese adaptation of the original Spider dataset, CSpider includes over 10,000 questions and nearly 5,700 distinct SQL queries across 200 databases. It aims to support the development of natural language interfaces tailored to Chinese-language databases [13].

- **BIRD**

  BIRD is a comprehensive benchmark comprising 12,751 question-query pairs over 95 databases, having an overall size of 33.4 GB. Spanning 37 professional fields—from blockchain to education—it is designed to assess how effectively systems manage large-scale databases and incorporate external domain-specific knowledge [12].

- **CoSQL**

  CoSQL is a dialogue-focused dataset designed to evaluate multi-turn exchanges. It features upwards of 30,000 conversational turns and over 10,000 labeled SQL statements, sourced from 3,000 authentic dialogues spanning 200 distinct databases [15]. By prioritising natural,interactive query evolution and refinement, it serves as an excellent resource for testing conversational database technologies [22].

- **WikiSQL**

  This dataset, constructed from Wikipedia tables, contains 80,000 pairs of natural language queries and their equivalent SQL queries. It is specifically designed to assess the effectiveness of models on relatively straightforward SQL query generation tasks [6].

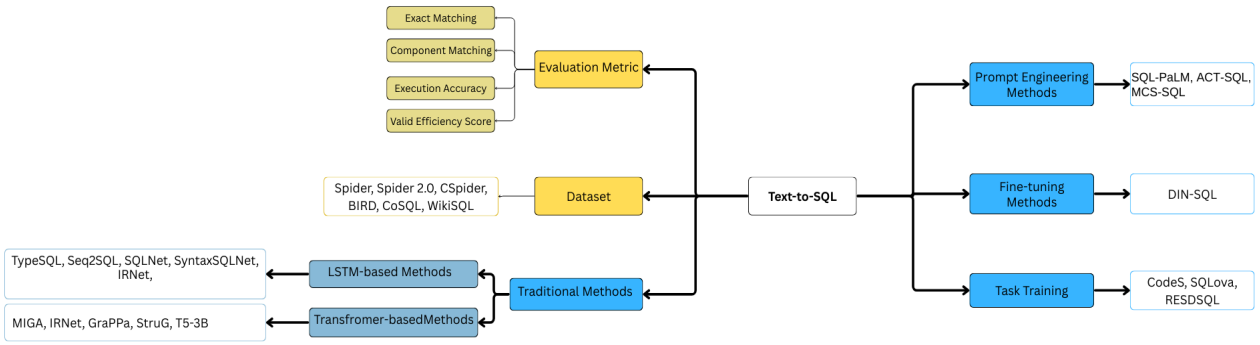


Fig 4. Overview of Text-to-SQL metrics, datasets,and methods.

4.2 Models

The growth of text-to-SQL models has been shaped via a number of significant advancements (see Table I):

- **Seq2SQL**

  An early sketch-based model for the WikiSQL dataset, Seq2SQL generates a predefined SQL structure and fills it using reinforcement learning. It focuses on execution accuracy by optimising the SQL query outcomes [25].

- **TypeSQL**

  Built upon SQLNet, TypeSQL incorporates type information such as dates, numbers, and named entities to enhance column prediction. It is trained on WikiSQL and improves semantic understanding between the query and schema [19].

- **SQLNet**

  SQLNet is a sketch-based model for WikiSQL that eliminates reinforcement learning. It fills SQL query slots using column-wise attention, simplifying the generation process while maintaining strong performance [18].

- **SyntaxSQLNet**

  Designed for Spider, SyntaxSQLNet handle complex and nested SQL queries by using an LSTM encoder and a syntax tree-based decoder. It integrates SQL grammar into the decoding process to improve structural accuracy [21].

- **IRNet**

  IRNet combines graph neural networks with intermediate representation decoders to map natural language query to SQL. Trained on Spider, it excels at representing question-schema relationships for complex queries [3].

- **GraPPa**

  GraPPa is a RoBERTa-based model pretrained on SQL-natural language pairs. It improves schema linking and contextual learning on the Spider dataset through enhanced pretraining techniques [20].

- **StruG**

  A structure-aware Transformer model for Spider, StruG integrates SQL grammar constraints directly into the decoding process. It ensures syntactically valid SQL generation using grammar-aware decoding [1].

- **MIGA**

  MIGA employs a RoBERTa encoder with multi-granularity alignment between question tokens and schema elements. It enhances representation quality and improves SQL accuracy on Spider [2].

- **T5-3B**

  Based on the large-scale T5 transformer (3B parameters), this model treats SQL generation as a text-to-text task. Fine-tuned on Spider and CoSQL, it demonstrates strong generalisation across domains [17].

- **SQLova**

  SQLova merges BERT embeddings with a pointer network and column attention for effective SQL slot filling. It is specifically tailored for WikiSQL and achieves high execution accuracy [5].

- **RESDSQL**

  RESDSQL leverages T5 or RoBERTa backbones with improved schema linking and ensemble decoding. It is trained on Spider and CoSQL and emphasises robust natural language understanding and decoding [10].

- **DIN-SQL**

  DIN-SQL uses a dual representation approach to model questions and schema independently. Built on PLMs and trained on Spider, it achieves strong alignment for accurate SQL generation [14].

- **SQL-PaLM**

  SQL-PaLM utilizes Google's PaLM LLM and chain-of-thought prompting to tackle complex SQL generation. It is trained on Spider and performs well in reasoning-heavy tasks [16].

- **Code-S**

  Code-S is built on a Codex-style GPT model, focusing on code synthesis and in-context learning for SQL. It is trained on Spider and is effective in few-shot learning settings [11].

- **ACT-SQL**

  ACT-SQL combines GPT-3 with adaptive prompting strategies like chain-of-thought reasoning. Designed for Spider, it dynamically adjusts its generation process based on the question context [24].

- **MCS-SQL**

  MCS-SQL applies GPT-4 step by step to produce SQL queries for databases such as Spider. The procedure begins with identifying the concerned tables and columns of the database schema. It proceeds to create different possible queries. It then, in the decision step, compares them based on feedback during execution and confidence levels and chooses the best one using multiple-choice strategy. The procedure makes sure the output of SQL queries is correct and reliable [7].

TABLE 1 Performance Comparison Across Text-to-SQL Architectures

Model	Dataset	Training Method	Accuracy
Seq2SQL [25]	WikiSQL	Seq2Seq + Reinforcement Learning	59.4%
TypeSQL [19]	WikiSQL	SQLNet + Type Information	82.6%
SQLNet [18]	WikiSQL	Sketch-based Seq2Seq	63.2%
SyntaxSQLNet [21]	Spider	LSTM + Syntax Tree Decoder	60.5%
IRNet [3]	Spider	Graph Neural Network + IR Decoder	61.9%
GraPPa [20]	Spider	RoBERTa (pre-trained on SQL & NL)	76%
StruG [1]	Spider	Transformer + SQL Grammar Constraints	75%
MIGA [2]	Spider	RoBERTa / Encoder-Decoder	73%
T5-3B [17]	Spider, CoSQL	Google T5 (3B parameters)	70.0%
SQLova [5]	WikiSQL	BERT + Pointer Network	83.6%
RESDSQL [10]	Spider, CoSQL	T5 / RoBERTa	80.5%
DIN-SQL [14]	Spider	PLM + Dual Representation Encoder	76%
SQL-PaLM [16]	Spider	PaLM (Google) + Chain-of-Thought	80%
Code-S [11]	Spider	CodeX (Codex-style GPT variant)	77%
ACT-SQL [24]	Spider	GPT-3 + Chain-of-Thought	78%
MCS-SQL [7]	Spider	GPT-4 + In-context learning	89.5%

4.3 Evaluation Metrics

To evaluate the performance of text-to-SQL systems, we consider four commonly employed metrics. The metrics are broadly divided into two categories: **content-based metrics**, which examine the structure of SQL queries, and **execution-based metrics**, which assess the result that is achieved by executing the queries on a database [15].

1 Content-Based Metrics

These metrics measure how accurately the structure and syntax of the predicted SQL queries reflect those of the correct queries [4].

- **Component Matching (CM)**:
  This metric analyzes SQL queries by dividing them into components such as **SELECT**, **WHERE**, **GROUP BY**, **ORDER BY**, and **KEYWORDS**. It evaluates if each predicted segment aligns with the actual answer by applying the **F1 score**. Because the sequence of these elements is often irrelevant in SQL, CM treats them as sets for comparison.

- **Exact Matching (EM)**:
  EM evaluates whether the generated SQL query is an **exact match** with the reference query. For a prediction to be deemed accurate, all elements evaluated by CM must align perfectly with the truth of the ground.

2 Execution-Based Metrics

Execution-based metrics check whether the predicted queries produce the correct output when run on the database [4].

- **Execution Accuracy (EX)**:
  This measures whether executing the predicted SQL query gives the **same result** as the correct query. It's a direct test of whether the model's output is functionally correct.

- **Valid Efficiency Score (VES)**:
  VES evaluates both the **accuracy and runtime performance** of valid SQL statements. This metric reflects not just the precision of the queries, but also how efficiently they are executed. It is determined by comparing the execution durations of predicted queries against those of reference queries, with the final value averaged over several trials to ensure consistency [15].

These metrics offer valuable information about the **precision**, **reliability**, and **real-world applicability** of text-to-SQL systems across various contexts.

CHAPTER 5
APPLICATION AND USE CASES OF TEXT-TO-SQL

Text-to-SQL models are now essential tools in most applications, enabling users to interact with structured databases using natural language. Converting everyday language to executable SQL queries, the models simplify data access and its interpretation without the need for technical expertise. Additionally,Text-to-SQL technology allows organisations to automate processes, enhance data transparency, and gain actionable insights from complex data sets—efficiently addressing data-related problems in industries with greater efficiency and accuracy.[15].

5.1. Medical Informatics and Patient Data Access

- **Clinical Assistance Tools**

Supports healthcare professionals by enabling accurate queries of patient data for diagnosis, treatment planning, or epidemiological research.

- **Efficient Patient Information Retrieval**

Enables fast retrieval of patient records, such as specific diagnoses or lab results, to aid critical decision-making.

- **Research Enablement**

Facilitates large-scale data extraction for medical studies focused on public health trends, drug effectiveness, and disease analysis.

5.2. Intelligent Learning and Academic Support

- **Personalised Learning Platforms**

Uses student information stored in databases to adjust educational content for individualised learning.

- **Educational Data Mining**

Allows institutions to analyse student performance and improve academic planning, curriculum design, and intervention strategies.

- **Student Data Access Portals**

Empowers students to retrieve course information, grades, and library resources independently using natural language queries.

5.3. Financial Intelligence and Operations

- **Dynamic Financial Analysis**

Helps analysts generate real-time reports and risk assessments by querying financial datasets using natural language.

- **Customer Insights and Resolution**

Assists support agents in delivering accurate and personalized information to clients through fast data access.

- **Fraud Monitoring Mechanisms**

Enables early detection of suspicious activities by monitoring transactional data for abnormal patterns.

5.4. Enterprise Analytics and Strategic Decision-Making

- **Consumer Behaviour and Sales Analysis**

Facilitates market trend analysis and sales tracking by allowing non-technical users to access business data easily.

- **Supply Chain Oversight**

Aids in tracking inventory levels, managing logistics, and optimizing supply chain operations.

- **Human Resource Metrics**

Supports workforce analytics by enabling natural language queries on productivity, turnover, and training data.

CHAPTER 6
CHALLENGES AND FUTURE DIRECTIONS

While LLM-driven text-to-SQL research has come a longway, there are still a number of obstacles that need attention. Inthe following section, we will look at these ongoing challenges and consider possible paths forward for future directions [15].

6.1. Real-World Adaptability of Text-to-SQL Systems

- **Challenges**

Text-to-SQL systems often fail in real-world use due to vague, incomplete, or typo-filled user queries that differ from clean training data. This causes a mismatch between user language and database schema. Local datasets may also be inconsistent, leading to incorrect SQL generation—around 40% in some evaluations.

- **Future Directions**

Future work should focus on training models with noisy data and using data augmentation to address limited real-world datasets. Fine-tuning open-source models on smaller, domain-specific data can improve adaptability. Expanding to multilingual and multimodal capabilities will make these systems more practical and inclusive [26].

6.2. Balancing Cost and Performance in Text-to-SQL Systems

- **Challenges**

The balance between efficiency and expense in text-to-SQL solutions is largely shaped by how quickly they generate results and how much computational power they require. As databases become more complex, the length of the database schema tokens grows, leading to higher costs and potential limitations of the model. This issue is especially prominent with proprietary LLMs or open-source models that have short context lengths, which may exceed token limits. Moreover, feeding the entire database schema as input introduces redundancy, making the process less efficient and more costly, thus affecting both practical applications and research outcomes.

• **Future Directions**

To counterbalance cost and performance, such techniques as schema filtering, wherein only the necessary components of the schema are presented to the model to decrease both redundancy and cost, must be investigated more. Though in-context learning has boosted accuracy, performance must be weighed against efficiency. Techniques of limiting API calls and inference time, such as shortening input lengths and decreasing implementation steps, can increase system performance. In addition, seeking architectural improvements for indigenous LLMs can deliver further acceleration and efficiency benefits [4].

6.3. Ensuring Data Security and Model Transparency in Text-to-SQL Systems

• **Challenges**

Text-to-SQL models have significant data security and modeling transparency challenges. Local control of sensitive data by databases through proprietary APIs has the potential for data leakage, especially when confidentiality is a critical requirement. Despite the ability of local fine-tuning methods to reduce some of these challenges, existing fine-tuning practices perform less than optimally. In addition, model explainability, an established deep learning challenge, remains a concern in LLM-based natural language query to SQL databases. The explanatory shortfall in comparison to the way models produce SQL queries—particularly for prompt-based learning and fine-tuning methods—restricts comprehension and trust.

• **Future Directions**

To meet these issues, follow-up research should entail the designing of fine-tuning frameworks for text-to-SQL models that place emphasis on data protection and minimize possible vulnerabilities. Model development in explainability is also essential, which can be achieved through the embedding of step-by-step query generation procedures and procedures from deep learning research dedicated to improving model interpretability. Model decision-making transparency enhancement, especially in database knowledge, will be instrumental in enhancing the system performance as well as the trust of the users in real use [4].

6.4. Expanding Capabilities of Text-to-SQL Systems

- **Challenges**

Text-to-SQL models struggle with generalizing their use beyond the generation of SQL. Text-to-SQL models are like those which are utilized in code generation and language comprehension but, in being pushed to other uses—such as natural language translation to other code forms—means that they must be highly engineered in order to deal with numerous programming regulations. Further, the addition of execution-aware mechanisms and cross-modal knowledge transfer from database to applications such as question answering is accompanied by integration challenges. These challenges underscore the intricacy of applying text-to-SQL approaches to an immense range of NLP and coding tasks.

- **Future Directions**

Future research should explore how text-to-SQL frameworks can be adapted for the generation of natural language code by leveraging execution-based learning and modular extensions. Additionally, integrating database-driven structured information into LLM-based question answering could enhance factual accuracy and reasoning. These cross-domain extensions have the potential to make text-to-SQL models more versatile. Continued development in these areas will support broader applications, such as knowledge-based QA systems and programming assistants, bridging structured data with general-purpose language understanding [26].

CHAPTER 7

CONCLUSION

In conclusion, text-to-SQL systems are essential to make the database accessible to everybody, especially those without technical knowledge. These systems make it possible to query and retrieve data using natural language, thus removing most of the hurdle that was there earlier when working with databases. This is because they will be able to access the information they require in a simple manner without having to master sophisticated query languages like SQL. The success reached so far along this direction, particularly with the assistance of large language models and big benchmark datasets, has significantly increased the efficiency and utility of text-to-SQL systems. Nevertheless, there are certain issues that remain to be solved. Some of the fundamental problems are that such systems tend to have a hard time answering questions from certain domains or industries since they tend to get trained on generic data sets. This might result in errors or misinterpretations when the system attempts to translate specialized jargon or terms.

These issues need to be fixed through future research that goes into developing more domain-specific data sets that take into account the special language and requirements of various industries. This would help improve the ability of the models to identify and process industry-specific questions and queries. This should also be enhanced to improve the generalization capacity of the systems so that they can work optimally even when faced with new or unknown types of data and queries. Another significant area for advancement is the integration of a richer understanding of context, including the user's intent, question history, and database schema being queried. Additionally, advancement in these areas may empower text-to-SQL systems to be more powerful and flexible and useful to a broader class of users and applications.

In total, much progress has already been made, but much work needs to be put in to make text-to-SQL systems truly functional and universally accessible. By continuing to work on domain-specific problems, model generalization, and contextualization, researchers can assist in driving the next wave of text-to-SQL technology that will further simplify data access and bring it within everyone's reach. This will ultimately enable more individuals to tap into the power of data, making better decisions and driving innovation across all walks of society.

REFERENCES

[1] Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. *Structure-grounded pretraining for text-to-sql*. arXiv preprint arXiv:2010.12773, 2020.

[2] Yingwen Fu, Wenjie Ou, Zhou Yu, and Yue Lin. *Miga: A unified multi-task generation framework for conversational text-to-sql*. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12790–12798, 2023.

[3] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. *Towards complex text-to-sql in cross-domain database with intermediate representation*. arXiv preprint arXiv:1905.08205, 2019.

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

[5] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. *A comprehensive exploration on wikisql with table-aware word contextualization*. arXiv preprint arXiv:1902.01069, 2019.

[6] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. *A comprehensive exploration on wikisql with table-aware word contextualization*. arXiv preprint arXiv:1902.01069, 2019.

[7] Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. *Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation*. arXiv preprint arXiv:2405.07467, 2024.

[8] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, et al. *Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows*. arXiv preprint arXiv:2411.07763, 2024.

[9] Fei Li and Hosagrahar V. Jagadish. *Constructing an interactive natural language interface for relational databases*. *Proceedings of the VLDB Endowment*, 8(1):73–84, 2014.

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

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

[12] Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, et al. *Can llm already serve as a database interface. A big bench for large-scale database grounded text-to-sqls*. CoRR abs/2305.03111, 2023.

[13] Qingkai Min, Yuefeng Shi, and Yue Zhang. *A pilot study for chinese sql semantic parsing*. arXiv preprint arXiv:1909.13293, 2019.

[14] Mohammadreza Pourreza and Davood Rafiei. *Din-sql: Decomposed in-context learning of text-to-sql with self-correction*. *Advances in Neural Information Processing Systems*, 36:36339–36348, 2023.

[15] Aditi Singh, Akash Shetty, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. *A survey of large language model-based generative ai for text-to-sql: Benchmarks, applications, use cases, and challenges*. In *2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 00015–00021. IEEE, 2025.

[16] Ruoxi Sun, Sercan Ö. Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, et al. *Sql-palm: Improved large language model adaptation for text-to-sql (extended)*. arXiv preprint arXiv:2306.00739, 2023.

[17] Albert Wong, Lien Pham, Young Lee, Shek Chan, Razel Sadaya, Youry Khmelevsky, Mathias Clement, Florence Wing Yau Cheng, Joe Mahony, and Michael Ferri. *Translating natural language queries to sql using the t5 model*. In *2024 IEEE International Systems Conference (SysCon)*, pages 1–7. IEEE, 2024.

[18] Xiaojun Xu, Chang Liu, and Dawn Song. *Sqlnet: Generating structured queries from natural language without reinforcement learning*. arXiv preprint arXiv:1711.04436, 2017.

[19] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. *Type-sql: Knowledge-based type-aware neural text-to-sql generation*. arXiv preprint arXiv:1804.09769,

[20] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. *Grappa: Grammar-augmented pre-training for table semantic parsing*. arXiv preprint arXiv:2009.13845, 2020.

[21] Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. *Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-SQL task*. arXiv preprint arXiv:1810.05237, 2018.

[22] Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. *Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases*. arXiv preprint arXiv:1909.05378, 2019.

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

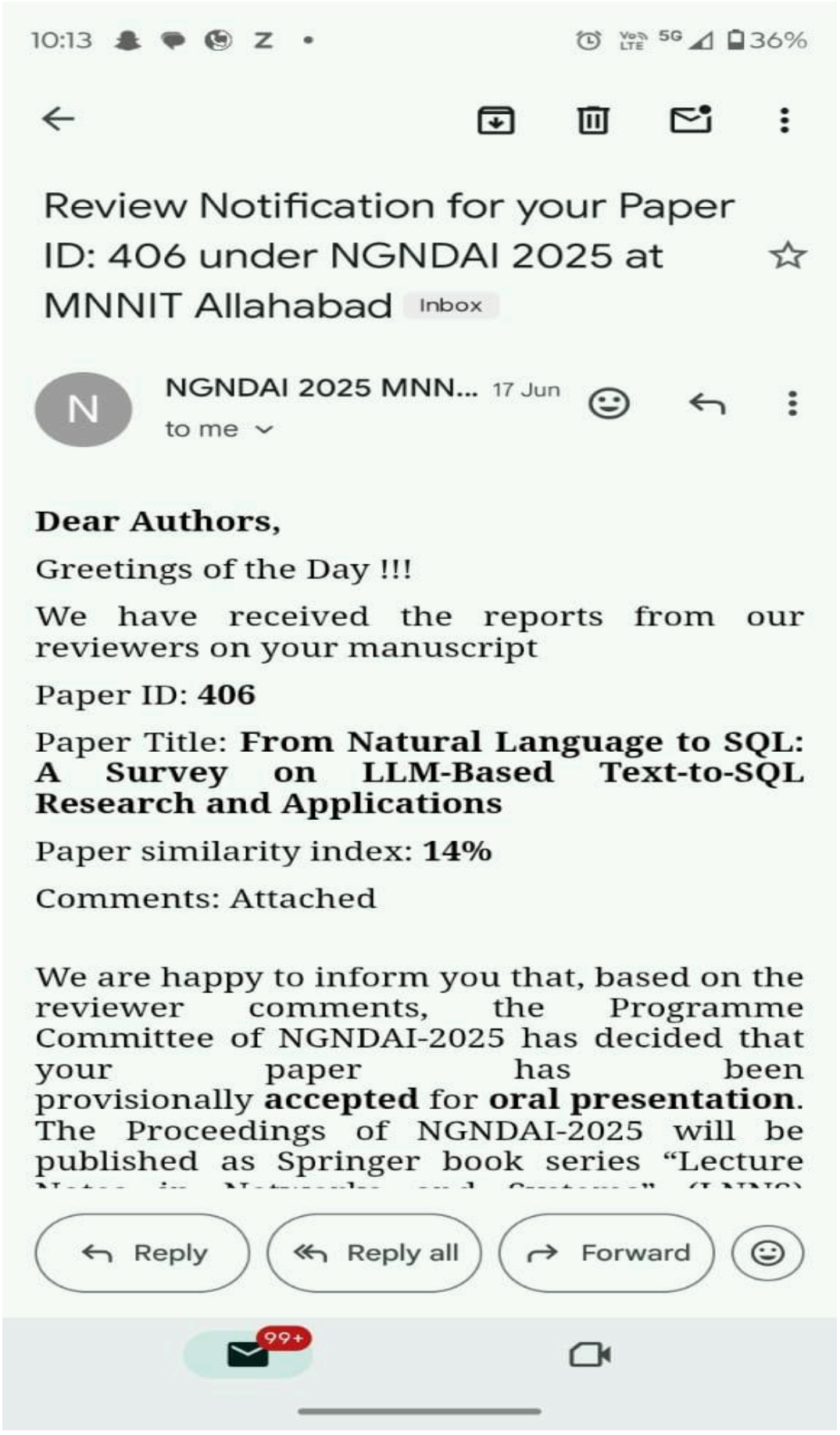
[24] Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. *Act-sql: In-context learning for text-to-sql with automatically-generated chain-of-thought*. arXiv preprint arXiv:2310.17342, 2023.

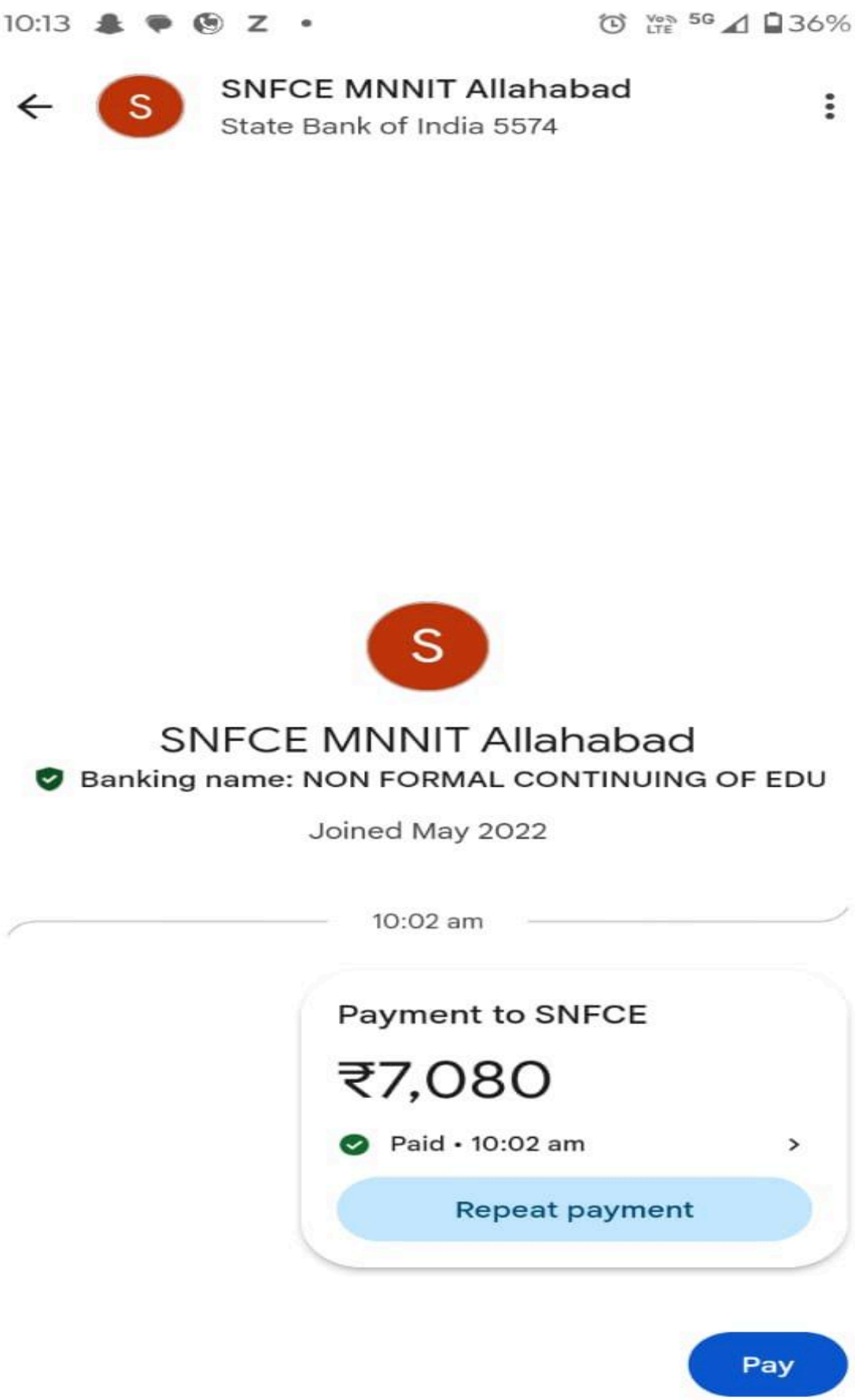
[25] Victor Zhong, Caiming Xiong, and Richard Socher. *Seq2sql: Generating structured queries from natural language using reinforcement learning*. arXiv preprint arXiv:1709.00103, 2017.

[26] Xiaohu Zhu, Qian Li, Lizhen Cui, and Yongkang Liu. *Large language model enhanced text-to-sql generation: A survey*. arXiv preprint arXiv:2410.06011, 2024.

List of Publications

1. **From Natural Language to SQL: A Survey on LLM-Based Text-to-SQL Research and Applications. (Accepted)**


10:13
36%
Review Notification for your Paper ID: 406 under NGNDAI 2025 at MNNIT Allahabad
Inbox
N
NGNDAI 2025 MNN...
17 Jun
to me
Dear Authors,
Greetings of the Day !!!
We have received the reports from our reviewers on your manuscript
Paper ID: 406
Paper Title: From Natural Language to SQL: A Survey on LLM-Based Text-to-SQL Research and Applications
Paper similarity index: 14%
Comments: Attached
We are happy to inform you that, based on the reviewer comments, the Programme Committee of NGNDAI-2025 has decided that your paper has been provisionally accepted for oral presentation. The Proceedings of NGNDAI-2025 will be published as Springer book series "Lecture
Reply
Reply all
Forward
99+


10:13
36%
SNFCE MNNIT Allahabad
State Bank of India 5574
S
SNFCE MNNIT Allahabad
Banking name: NON FORMAL CONTINUING OF EDU
Joined May 2022
10:02 am
Payment to SNFCE
₹7,080
Paid • 10:02 am
Repeat payment
Pay