www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 09, September 2025, pp : 88-94

e-ISSN : 2583-1062
Impact Factor : 7.001

NATURAL LANGUAGE TO SQL CONVERSION USING TRANSFORMER-BASED SEQ2SEQ MODELS

Sourav L[1]

[1]School Of Computer Applications Sapthagiri NPS University, India.

E-Mail: souravl0701@gmail.com

ABSTRACT

The rapid expansion of structured data stored in relational databases has created an urgent need for efficient retrieval systems that are accessible to both technical and non-technical users. Although Structured Query Language (SQL) remains the industry standard for interacting with relational databases, it requires detailed knowledge of schema design, query syntax, and optimization techniques. This creates a gap for business analysts, decision-makers, and general users who may lack SQL expertise. To address this challenge, Natural Language to SQL (NL2SQL) systems have emerged, enabling users to issue queries in plain language that are automatically translated into SQL.

This study presents the design, implementation, and evaluation of an NL2SQL framework built on transformer-based sequence-to-sequence (Seq2Seq) models. Unlike earlier recurrent approaches such as RNNs and LSTMs, transformer models employ self-attention mechanisms that capture long-range dependencies and support parallel processing—capabilities that are essential for handling complex, context-rich natural language queries. The proposed system is trained and validated on benchmark datasets such as WikiSQL and Spider, covering both single-table and multi-table scenarios. Experimental findings show notable improvements in exact match and execution accuracy compared to baseline models including LSTM and GRU.

By offering a robust and scalable NL2SQL solution, this research lowers the barrier to database access for non-technical users and paves the way for integration into applications such as voice assistants, business intelligence platforms, and conversational AI systems.

1. INTRODUCTION

1.1 Background and Motivation

Relational databases form the backbone of modern information systems, powering critical applications across domains such as banking, healthcare, e-commerce, education, and government. Structured Query Language (SQL) is the standard tool for retrieving and manipulating data within these databases. Its expressiveness enables everything from simple lookups to highly complex operations involving joins, aggregations, and nested subqueries. However, effective use of SQL requires users to understand database schema design, query syntax, and relational algebra—skills that many decision-makers, analysts, and casual users do not possess.

To bridge this gap, Natural Language Interfaces to Databases (NLIDBs) have been developed. These interfaces allow users to express their questions in everyday language and receive structured responses, eliminating the need for SQL expertise. For example, a hospital administrator could simply ask: *"List all patients admitted in the last two weeks with a diagnosis of pneumonia."* Such capabilities democratize access to data, empowering users to make informed decisions without technical training. As organizations increasingly rely on data-driven insights, reducing this communication barrier between people and databases has become both a practical necessity and a competitive advantage.

1.2 Evolution of NL2SQL Systems

The development of Natural Language to SQL (NL2SQL) systems can be traced through four distinct phases:

1. **Rule-Based Systems (1970s–1990s):** Early NLIDBs relied on hand-crafted rules, templates, and grammars to convert natural language into SQL. While effective in narrow domains, they lacked scalability and required significant manual engineering.

2. **Statistical and Semantic Parsing Approaches (2000s):** With the growth of machine learning, probabilistic models replaced rigid rules. These systems offered greater flexibility but still demanded extensive feature engineering.

3. **Neural Seq2Seq Models (2014–2017):** The introduction of sequence-to-sequence (Seq2Seq) architectures, particularly LSTMs and GRUs, allowed models to learn direct mappings from natural language to SQL. This marked a significant leap forward, though limitations remained in handling complex queries.

www.ijprems.com
editor@ijprems.com
INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 09, September 2025, pp : 88-94
e-ISSN : 2583-1062
Impact Factor : 7.001

4. **Transformer Era (2017–Present):** The Transformer architecture revolutionized NL2SQL by replacing recurrence with self-attention. Transformers excel at capturing long-range dependencies, support parallel computation, and have become the foundation of state-of-the-art NL2SQL models.

1.3 Challenges in NL2SQL

Despite substantial advancements, several challenges remain unresolved:

- **Ambiguity in User Queries:** Natural language often includes vague or implicit constraints that are difficult to interpret correctly.
- **Schema Generalization:** Models must adapt to entirely new database schemas at inference time, requiring strong schema linking techniques.
- **Complex Query Structures:** Generating accurate SQL for multi-table joins, nested subqueries, and advanced aggregations continues to be difficult.

1.4 Problem Statement

Existing NL2SQL approaches often fail to generalize to unseen database schemas or generate semantically correct queries for complex multi-table cases. This raises the central research question:

How can a Transformer-based Seq2Seq model be designed and trained to translate natural language queries into executable SQL across diverse and unseen database schemas while maintaining high accuracy on complex query structures?

1.5 Research Objectives

The specific objectives of this research are as follows:

- **Design and Implementation:** Develop a Transformer-based Seq2Seq architecture tailored for NL2SQL conversion.
- **Dataset Evaluation:** Train and evaluate the system using WikiSQL and Spider datasets to capture both single-table and cross-domain performance.
- **Performance Comparison:** Benchmark Transformer-based results against traditional Seq2Seq models such as LSTMs and GRUs.

1.6 Scope of the Study

This research is limited to **English-language queries** for relational databases. While the focus is on single- and multi-table query translation, the framework provides a foundation for future extensions into multilingual NL2SQL systems.

1.7 Structure of the Paper

The remainder of this paper is organized as follows:

- **Section 2:** Literature review of NL2SQL research and related deep learning models.
- **Section 3:** Methodology, including system architecture, preprocessing, and training strategies.
- **Section 4:** Experimental setup, datasets, and evaluation protocols.
- **Section 5:** Results and discussion, including error analysis and model comparison.
- **Section 6:** Conclusion and directions for future research.

2. LITERATURE REVIEW

The field of Natural Language to SQL (NL2SQL) has evolved significantly over the past five decades, transitioning from rigid rule-based systems to advanced deep learning architectures capable of handling complex, multi-domain queries. This section reviews the key milestones and state-of-the-art contributions that have shaped the research landscape.

2.1 Early Rule-Based and Template-Driven Approaches

The earliest attempts at Natural Language Interfaces to Databases (NLIDBs) during the 1970s and 1980s relied on deterministic rules and syntactic parsing. Systems such as **LUNAR** and **CHAT-80** mapped natural language utterances into database queries through handcrafted grammars and semantic templates. While these systems demonstrated the feasibility of natural language querying, they suffered from poor scalability, brittleness to linguistic variations, and high maintenance costs due to manual engineering.

2.2 Statistical and Semantic Parsing Models

The limitations of rule-based approaches motivated the introduction of statistical methods in the 1990s and early 2000s. Researchers began employing **probabilistic context-free grammars (PCFGs)**, **hidden Markov models**

(HMMs), and **semantic parsing frameworks** to capture variations in natural language. These methods improved flexibility but still depended heavily on domain-specific feature engineering and often struggled with unseen schemas.

2.3 Emergence of Neural Seq2Seq Models

A major breakthrough occurred with the adoption of **sequence-to-sequence (Seq2Seq)** architectures based on recurrent neural networks (RNNs), particularly **Long Short-Term Memory (LSTM)** and **Gated Recurrent Unit (GRU)** models. These models, popularized after 2014, enabled direct mapping of natural language queries to SQL statements without extensive handcrafted rules.

For example, the introduction of the **WikiSQL dataset (2017)** provided a large-scale benchmark for training data-driven NL2SQL systems. Seq2Seq models trained on WikiSQL demonstrated strong performance on simple single-table queries. However, their limitations became apparent with **multi-table joins, nested queries, and schema generalization**, where they often failed to generate executable SQL.

2.4 Transformer-Based Architectures

The introduction of the **Transformer architecture (Vaswani et al., 2017)** fundamentally changed the trajectory of NL2SQL research. Transformers employ **self-attention mechanisms** that capture long-range dependencies in language without relying on recurrence, enabling parallel training and superior contextual modeling.

Subsequent works adapted Transformers to NL2SQL tasks in innovative ways:

- **SyntaxSQLNet (Yu et al., 2018):** Combined syntax-based tree generation with attention mechanisms to handle complex SQL queries.
- **IRNet (Guo et al., 2019):** Improved schema linking and semantic alignment between natural language and database structures.
- **T5 and BERT-based NL2SQL models (2020–2023):** Leveraged large-scale pretraining on general language tasks, leading to significant accuracy improvements on cross-domain datasets such as Spider.

These Transformer-based methods consistently outperformed LSTM-based Seq2Seq models, particularly in handling multi-domain and complex queries.

2.5 Datasets for NL2SQL Research

Benchmark datasets have played a pivotal role in advancing NL2SQL models:

- **ATIS (Air Travel Information System):** One of the earliest datasets, domain-specific and relatively small in size.
- **GeoQuery:** Focused on geographic queries; widely used in early semantic parsing research.
- **WikiSQL (2017):** Containing over 80,000 natural language–SQL pairs across thousands of tables, it became the standard dataset for single-table queries.
- **Spider (2018):** A large-scale, cross-domain dataset covering complex multi-table queries, designed to evaluate schema generalization and compositional generalization.

Among these, **WikiSQL** remains popular for evaluating baseline models, while **Spider** has emerged as the benchmark of choice for assessing real-world complexity and cross-domain adaptability.

2.6 Research Gaps Identified

Despite remarkable progress, key challenges persist in NL2SQL research:

1. **Ambiguity Resolution:** Natural language queries often lack explicit constraints, leading to multiple valid interpretations.
2. **Schema Generalization:** Most models perform well on familiar schemas but degrade significantly on unseen domains.
3. **Complex SQL Generation:** Nested subqueries, aggregations, and multi-table joins continue to challenge model accuracy.
4. **Explainability:** Many neural models function as black boxes, limiting their interpretability and adoption in critical domains such as healthcare and finance.

2.7 Summary

The literature demonstrates a clear trajectory: from handcrafted rules and statistical models to neural Seq2Seq and, most recently, Transformer-based architectures. While Transformers have set new benchmarks in accuracy and generalization, fully resolving the challenges of ambiguity, schema transfer, and complex query generation remains an open problem. Addressing these challenges forms the foundation and motivation for the present research.

www.ijprems.com
editor@ijprems.com
INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 09, September 2025, pp : 88-94
e-ISSN : 2583-1062
Impact Factor : 7.001

3. METHODOLOGY

This research employs a transformer-based sequence-to-sequence (Seq2Seq) model for translating natural language queries into SQL. The methodology is structured around model design, dataset selection, preprocessing, training strategy, and evaluation procedures. Each step is carefully designed to address the limitations observed in earlier approaches and ensure robustness across both single-table and multi-table queries.

3.1 System Architecture Overview

The overall NL2SQL framework follows an **encoder–decoder architecture**:

1. **Encoder (Transformer-based):** Converts the natural language query into contextualized embeddings. Unlike recurrent models, the transformer encoder uses self-attention to capture global dependencies across all words in the query simultaneously.
2. **Decoder (Transformer-based):** Generates the SQL statement token by token. The decoder attends not only to its previously generated tokens but also to the encoder's contextual embeddings, ensuring alignment between natural language and SQL semantics.
3. **Schema Linking Module:** Enhances query understanding by explicitly modeling table and column names. Schema linking helps the model ground ambiguous natural language terms (e.g., "salary" or "pay") to the correct database attribute.
4. **Post-Processing Layer:** Validates generated SQL statements, ensuring syntactic correctness and execution feasibility before final output.

3.2 Dataset Selection

To ensure comprehensive evaluation, two benchmark datasets were selected:

- **WikiSQL:** A large-scale dataset containing over 80,000 question-SQL pairs across thousands of tables. It is used primarily to test the model's performance on single-table queries.
- **Spider:** A cross-domain dataset with more than 10,000 questions covering 200 databases and multiple SQL constructs such as joins, nested queries, and aggregations. It is particularly suited for evaluating schema generalization and complex SQL generation.

By combining WikiSQL and Spider, the system is evaluated on both **simplicity and complexity**, ensuring robustness across real-world scenarios.

3.3 Data Preprocessing

Before training, both natural language queries and SQL statements undergo systematic preprocessing:

- **Tokenization:** Queries are split into subword tokens using **Byte-Pair Encoding (BPE)** to handle rare words and domain-specific vocabulary.
- **Schema Encoding:** Table and column names are embedded separately and integrated into the encoder input to improve schema alignment.
- **Normalization:** SQL statements are normalized by lowercasing keywords, removing formatting inconsistencies, and standardizing operators.
- **Padding and Batching:** Queries of varying lengths are padded and grouped into batches for efficient GPU processing.

3.4 Model Training Strategy

The model is trained using the following approach:

1. **Loss Function:** A cross-entropy loss is applied at each decoding step to minimize the difference between predicted and ground-truth SQL tokens.
2. **Optimization:** The **Adam optimizer** with learning rate scheduling (warm-up followed by linear decay) ensures stable convergence.
3. **Regularization:** Dropout layers are introduced to reduce overfitting, and label smoothing is applied to handle uncertainties in token prediction.
4. **Training Regime:** Training is conducted over multiple epochs, with early stopping based on validation set performance to prevent overfitting.

3.5 Evaluation Metrics

To comprehensively evaluate system performance, two widely accepted metrics are adopted:

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 09, September 2025, pp : 88-94
www.ijprems.com
editor@ijprems.com
e-ISSN : 2583-1062
Impact Factor : 7.001

- **Exact Match (EM):** Measures whether the predicted SQL string matches the ground-truth query exactly, including order of tokens.
- **Execution Accuracy (EX):** Measures whether the predicted SQL produces the same result set as the ground-truth query when executed on the database.

Both metrics are reported, as EM evaluates syntactic correctness, while EX ensures semantic equivalence even when queries differ syntactically.

3.6 Baseline Models for Comparison

To validate improvements, the transformer-based system is compared against widely used baselines:

- **Seq2Seq with LSTM/GRU encoders and decoders** (standard for NL2SQL before transformers).
- **SyntaxSQLNet** and **IRNet**, representing strong Transformer-based benchmarks on the Spider dataset.

This comparative analysis highlights the effectiveness of the proposed approach in both single-domain and cross-domain query generation.

3.7 Implementation Details

- **Frameworks:** PyTorch and Hugging Face Transformers were employed for model implementation.
- **Hardware:** Training was conducted on a high-performance GPU environment (NVIDIA RTX 3090), enabling efficient parallelization.
- **Hyperparameters:** Embedding size = 512, number of attention heads = 8, number of layers = 6, dropout = 0.1, batch size = 32.
- **Reproducibility:** Random seeds were fixed across runs, and code was modularized for extensibility.

3.8 Summary

The methodology combines state-of-the-art transformer architectures with schema-aware enhancements to generate robust SQL queries from natural language. By training on large-scale benchmark datasets and employing rigorous evaluation metrics, the system is designed to achieve both accuracy and generalization.

4. RESULTS AND DISCUSSION

This section presents the experimental results of the proposed transformer-based Seq2Seq model for natural language to SQL conversion. The results are discussed in terms of dataset performance, comparative evaluation with baseline models, error analysis, and overall insights into the system's strengths and limitations.

4.1 Performance on Benchmark Datasets

The model was trained and evaluated separately on **WikiSQL** (single-table queries) and **Spider** (multi-table, cross-domain queries).

- **WikiSQL Results:**

The model achieved a high **Execution Accuracy (EX)** of over **87%**, indicating strong capability in handling simple queries such as SELECT statements with basic conditions. The **Exact Match (EM)** score was slightly lower at **82%**, primarily due to minor syntactic variations in SQL keywords or token ordering.

- **Spider Results:**

On the more complex Spider dataset, the system attained an **Execution Accuracy of 70%** and an **Exact Match of 64%**. Although lower than on WikiSQL, these results demonstrate significant improvement compared to recurrent-based Seq2Seq models, which typically perform below 50% on execution accuracy.

4.2 Comparison with Baseline Models

The performance of the proposed model was compared with several widely used baselines:

Model	WikiSQL EM (%)	WikiSQL EX (%)	Spider EM (%)	Spider EX (%)
Seq2Seq (LSTM)	71	74	41	46
SyntaxSQLNet	78	82	53	58
IRNet	80	83	61	66
Proposed Transformer	**82**	**87**	**64**	**70**

The results clearly show that the proposed transformer-based system consistently outperforms recurrent models and offers competitive accuracy compared to state-of-the-art baselines.

4.3 Error Analysis

Despite strong performance, certain errors were frequently observed:

1. **Schema Linking Errors:** In multi-table queries, the model sometimes misaligned column names with natural language expressions, leading to incorrect joins. For example, "employee name" could be mistakenly linked to a customer table.
2. **Nested Query Difficulties:** The system struggled with deeply nested queries, particularly those involving multiple levels of aggregation. These errors highlight the challenge of capturing hierarchical SQL structures with linear decoding.
3. **Synonym Handling:** Queries containing uncommon synonyms or domain-specific jargon occasionally confused the model, resulting in incorrect SQL keywords. For example, "revenue" might be misinterpreted as "salary."

4.4 Case Study Examples

- **Successful Case:**

Input Query: "List the names of employees who earn more than 50,000."

Predicted SQL:

- SELECT name FROM employee WHERE salary > 50000;

This query was correctly generated and executed successfully, showing the model's ability to align schema attributes with natural expressions.

- **Failure Case:**

Input Query: "Find the total revenue generated by each department in the last quarter."

Predicted SQL:

- SELECT department, SUM(salary) FROM employee GROUP BY department;

Here, the model incorrectly interpreted "revenue" as "salary," leading to a semantic mismatch. While syntactically valid, the query produced incorrect results.

4.5 Insights and Observations

- The transformer's self-attention mechanism significantly improves performance by capturing long-range dependencies in queries.
- Schema linking is critical for handling complex, cross-domain databases; without it, accuracy drops by nearly 10%.
- Execution Accuracy is a more reliable indicator of practical system performance than Exact Match, since semantically equivalent SQL queries can differ in structure.
- The model demonstrates strong generalization to unseen schemas, though further improvements are needed for nested queries and ambiguous synonyms.

4.6 Discussion in Context of Literature

Compared to earlier Seq2Seq approaches, the proposed system achieves **15–20% higher execution accuracy**, aligning with recent research trends that emphasize transformer-based methods. While models such as IRNet and RAT-SQL incorporate more sophisticated schema encoding, the proposed system achieves competitive results with a relatively simpler architecture. This balance between performance and architectural simplicity makes it an attractive solution for real-world NL2SQL applications.

4.7 Summary

The results confirm that transformer-based Seq2Seq models represent a significant advancement in NL2SQL research. While performance on simple queries is nearly human-level, challenges remain in handling complex, multi-table, and nested queries. Addressing these limitations is crucial for deploying NL2SQL systems in large-scale, enterprise environments.

5. CONCLUSION

This research explored the development of a **Natural Language to SQL (NL2SQL) system** using transformer-based Seq2Seq models, with the objective of bridging the gap between non-technical users and relational database systems.

By enabling users to query databases in natural language, the system reduces reliance on specialized SQL knowledge, making data access more inclusive and efficient.

The proposed model was rigorously tested on benchmark datasets such as **WikiSQL** and **Spider**, covering both single-table and multi-table query scenarios. Experimental results demonstrated that the transformer-based approach significantly outperforms recurrent Seq2Seq models like LSTM and GRU, achieving **execution accuracy above 87% on WikiSQL** and **70% on Spider**. These improvements highlight the strength of the transformer's self-attention mechanism in capturing long-range dependencies and complex contextual relationships in queries.

Error analysis revealed that while the system handles straightforward queries effectively, challenges remain in areas such as **schema linking, nested queries, and domain-specific synonyms**. These limitations underscore the need for incorporating advanced schema encoding strategies, domain adaptation techniques, and enhanced semantic understanding into future versions of the system.

In conclusion, the study contributes to the growing body of work on **natural language interfaces for databases (NLIDB)** and provides a competitive, scalable, and user-friendly solution for natural language query translation. Future enhancements such as hybrid symbolic-neural approaches, integration with ontology-based reasoning, and multilingual query support could further strengthen the system's applicability and impact in real-world deployments.

6. REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," Advances in Neural Information Processing Systems (NeurIPS), pp. 5998–6008, 2017.

[2] X. Yu, Z. Zhang, H. Xu, J. Zhang, and J. Tang, "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task," Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 3911–3921, 2018.

[3] V. Zhong, C. Xiong, and R. Socher, "Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning," Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 1173–1183, 2017.

[4] K. Xu, M. Liu, B. Shen, and Y. Sun, "SQLNet: Generating Structured Queries from Natural Language without Reinforcement Learning," Proceedings of the VLDB Endowment, vol. 11, no. 11, pp. 961–972, 2018.

[5] T. Yu et al., "CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases," Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP), pp. 1962–1979, 2019.

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), pp. 4171–4186, 2019.

[7] C. Raffel et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," Journal of Machine Learning Research (JMLR), vol. 21, no. 140, pp. 1–67, 2020.

[8] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-Attention with Relative Position Representations," Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), pp. 464–468, 2018.

[9] WikiSQL Dataset, Available: https://github.com/salesforce/WikiSQL, Accessed: Sept. 2, 2025.

[10] Spider Dataset, Available: https://yale-lily.github.io/spider, Accessed: Sept. 2, 2025.