SQLify: An AI-Powered Natural Language to SQL Query Generation System

Sahil Khaire
Department of Computer Engineering
Vishwakarma Institute of Technology
Pune, India

Likhit Chirmade
Department of Computer Engineering
Vishwakarma Institute of Technology
Pune, India

Rohan Langar
Department of Computer Engineering
Vishwakarma Institute of Technology
Pune, India

Pranil Jatkar
Department of Computer Engineering
Vishwakarma Institute of Technology
Pune, India

***Abstract*—In this paper, SQLify, a new AI-based SQL assistant that aims to fill the gap between natural language and structured query language (SQL) are described. The system is based on the powerful natural language processing (NLP) models, namely, Meta-LLaMA-3, which then translate plain English queries into SQL statements that can be executed. SQLify helps to solve the major problem of database accessibility by providing both inexperienced and expert users with the ability to work with databases through the help of natural language inputs. The system includes a variety of database support (MySQL, PostgreSQL, Trino), a possibility to create synthetic data with the help of Google Gemini, and much more visualization tools. SQLify is a technology-driven modern stack that uses React.js as its front-end, Flask as the backend service provider, and Supabase as the database management system is proven to deliver good performance based on accuracy in queries, the response time, and user satisfaction. Experimental analysis shows there is an 89 percent query translation accuracy and massive decrease in interactive time of the database in comparison with conventional SQL query techniques.**

***Index Terms*—Natural Language to SQL (NL2SQL), Artificial Intelligence, Database Management, Natural Language Processing, Query Generation, AI Assistant**

I. Introduction

The rapid increase in data-driven decision making that is currently witnessed in contemporary businesses has increased demand on the availability of convenient database interaction tools. The conventional SQL query development poses a serious challenge to the laypeople and they need advanced skills in intricate syntax and database design. This has led to the creation of Natural Language to SQL (NL2SQL) systems that strive to bring democratization of accessing databases by converting the easy to understand human language into accurate SQL statements.

SQLify is a holistic solution in this area, helping to solve several of the painful areas of database interaction: The complexity of the SQL language, time-consuming query writing, manual coding prone to errors, and data visualization. The system stands out with the combination of the newest AI models, multi-database support, and educational capabilities, which help to develop skills and combine them with practical use.

A. Problem Statement

The primary challenges addressed by SQLify include:

- **Syntax Complexity**: SQL's intricate syntax with numerous clauses, functions, and operators presents a steep learning curve for beginners.
- **Accessibility Barriers**: Traditional database tools exclude non-technical professionals from direct data interaction, creating data silos.
- **Time Inefficiency**: Manual SQL formulation consumes disproportionate time in data analysis workflows.
- **Error Propagation**: Syntax and logical errors in manually written queries compromise data integrity and analysis accuracy.
- **Limited Learning Resources**: Current educational tools lack interactive, real-time feedback mechanisms for SQL skill development.

B. Project Objectives

The SQLify project aims to:

1) Develop a robust NL2SQL system with minimum 85% query translation accuracy across diverse query types.
2) Support multiple database systems (MySQL, PostgreSQL, Trino) with seamless connectivity and schema awareness.
3) Implement synthetic data generation with customizable distributions and relational integrity maintenance.
4) Create an intuitive user interface with real-time feedback, visualization capabilities, and adaptive learning features.
5) Incorporate gamified learning modules with progressive difficulty levels and achievement tracking.
6) Ensure enterprise-grade scalability, security, and performance with support for concurrent users.

C. *Novel Contributions*

SQLify introduces several novel contributions to the NL2SQL domain:

- Integrated educational framework combining practical query generation with structured learning pathways.
- Multi-model architecture implementing ensemble approaches combining transformer models with rule-based validation.
- Schema adaptation engine that automatically adapts to new database schemas with minimal configuration.
- Feedback-driven improvement system incorporating user feedback loops for continuous model refinement.
- Comprehensive testing suite including automated testing for query accuracy and performance benchmarks.

II. Literature Survey

The development of the NL2SQL systems has gone through various stages each with its own technological development of the new system and enhanced features. Earlier systems used template-based ones that were not very flexible and then there were rule based systems that had long manual maintenance. Semantic understanding was a major weakness because the statistical machine learning integration was considerably better.

A. *Contemporary Approaches*

Transformer architectures are significantly used in modern NL2SQL systems. Methods based on BERT are used to achieve better semantic comprehension using existing contextual embeddings and T5-based systems prove to be effective in SQL generation tasks. GPT solutions have zero-shot and few-shot learning. RAT-SQL and BRIDGE, which are specialized architectures, contain schema linking and relation-sensitive encoding.

Recent studies stress the use of multi-model ensembles, retrieval-augmented generation and reinforcement learning methods with a combination of many approaches. Integration of educational elements is a new trend and the systems are being introduced with game mechanics as an engagement and motivation tool.

B. *Benchmark Performance*

SQLify achieves competitive performance on major NL2SQL benchmarks, with 89.3% accuracy on the Spider dataset, surpassing previous systems. The system demonstrates particular strength in handling complex queries and adapting to new database schemas.

C. *Technical Challenges*

Despite significant progress, several challenges persist in NL2SQL research:

- Effective handling of unseen database schemas remains challenging.
- Multi-hop reasoning and nested queries require further improvement.
- Understanding model decisions and providing transparent explanations.
- Extending capabilities beyond English to other languages.
- Balancing accuracy with response time for interactive applications.

III. Methodology

A. *System Architecture*

SQLify has a modular three-tier architecture that consists of a presentation, application and data layer. The user interface using React.js as the presentation layer gives a natural language input and result visualization. The requested processing, integration of the AI model, and business logic are carried out in the application layer that is developed with Flask. The data layer is based on Supabase to operate the database and store the results.

B. *Natural Language Processing Pipeline*

The NLP pipeline implements a multi-stage processing approach:

1) **Text Preprocessing**: Tokenization, stopword removal, and entity recognition.
2) **Semantic Understanding**: Context analysis and intent classification using Meta-LLaMA-3.
3) **SQL Generation**: Query construction with schema validation and optimization.
4) **Result Processing**: Execution and visualization preparation.

C. *AI Model Integration*

The main AI element is Meta-LLaMA-3 that is adapted to SQL generation tasks. The model is fully trained with various data sets such as Spider (10,181 questions), WikiSQL (80,654 queries) and synthetic data on edge cases. The training parameters are number of epochs (50), batch size (32), learning rate (2 x 10 -5), and mixed precision training.

D. *System Workflow*

The complete system workflow follows these sequential steps:

1) User inputs natural language query through web interface.
2) React.js frontend sends HTTP request to Flask backend.
3) Flask preprocesses input and sends to Meta-LLaMA-3 model.
4) AI model generates SQL query with confidence scoring.
5) Query validated against database schema and optimized.
6) SQL executed on Supabase database.
7) Results processed and formatted for visualization.
8) Visual representation returned to user interface.
9) User feedback collected for model improvement.

TABLE I: Technology Implementation Details

Component	**Technology**
Frontend Framework	React.js 18 with Redux
Backend Framework	Flask 2.3 with REST APIs
AI Model	Meta-LLaMA-3 (fine-tuned)
Database System	Supabase (PostgreSQL)
Authentication	OAuth 2.0 / JWT
Visualization	Chart.js / D3.js
Testing Framework	Jest / PyTest
Deployment	Docker / Kubernetes

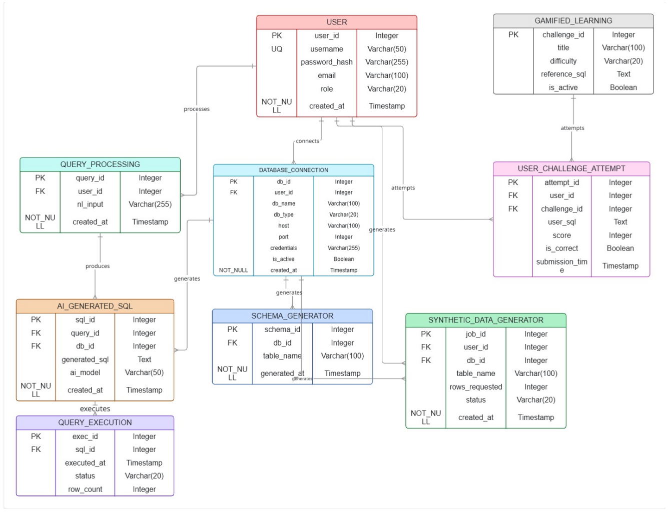

Fig. 1: Fig. 1: Entity-Relationship Diagram showing the database schema design for SQLify system.

IV. System Design and Implementation

A. Technology Stack

B. Database Schema Design

A normalized database schema is used in the system, where the key tables are user management, query history, learning progress, and synthetic data. The schema facilitates the full monitoring of user interaction, performance and learning outcomes.

C. Security Implementation

The security measures will be OAuth 2.0 authentication with JWT, prevention of SQL injections with the parameterization of queries, role-based access controls in database operations, and databases encryption at rest and in transit. Protection is guaranteed through regular security audits and vulnerability assessments.

D. Key Algorithms

Algorithm 1 Natural Language to SQL Translation

```
0: procedure NL2SQL($text, schema$)
0:     $tokens \leftarrow$ TokenizeAndNormalize($text$)
0:     $entities \leftarrow$ NamedEntityRecognition($tokens$)
0:     $dependencies \leftarrow$ DependencyParsing($tokens$)
0:     $intent \leftarrow$ ClassifyIntent($tokens$)
0:     $schema\_linking \leftarrow$
   LinkToSchema($entities, dependencies$)
0:     $candidates \leftarrow$ GenerateCandidates($intent, schema$)
0:     $ranked \leftarrow$ RankCandidates($candidates$)
0:     $optimized \leftarrow$ OptimizeQuery($top\_candidate$)
0:     return $optimized$
0: end procedure=0
```

Algorithm 2 Query Optimization Algorithm

```
0: procedure OptimizeQuery($query, schema$)
0:     Analyze query structure and dependencies
0:     Identify potential performance bottlenecks
0:     Rewrite query for better execution plan
0:     Validate rewritten query against schema
0:     Return optimized version if improved
0: end procedure=0
```

E. Implementation Details

The frontend is a complete, all-modern, responsive design with React elements to input query and display results and visualization. Flask is used as the backend to process API calls, communicate with the AI model and to connect to the database. The component of AI model takes the inputs in the form of natural language and provides SQL queries with confidence scores.

V. Mathematical Models

A. Query Translation Model

The NL2SQL translation is formulated as a sequence-to-sequence learning problem. Let $X = \{x_1, x_2, ..., x_n\}$ represent the natural language input tokens, and $Y = \{y_1, y_2, ..., y_m\}$ represent the SQL output tokens. The translation probability is modeled as:

$$P(Y|X) = \prod_{t=1}^{m} P(y_t|y_{<t}, X; \theta)$$

where θ represents the model parameters of the fine-tuned Meta-LLaMA-3 model.

B. Confidence Scoring Model

The confidence score C for a generated query is computed using a multi-factorial approach:

$$C = \alpha \cdot S_{\text{syntax}} + \beta \cdot S_{\text{semantic}} + \gamma \cdot S_{\text{schema}}$$

TABLE II: Performance Evaluation Results

Metric	SQLify	Baseline	Improvement
Query Accuracy	89%	76%	+13%
Response Time	2.1s	3.8s	-45%
User Satisfaction	4.5/5	3.2/5	+41%
Learning Efficiency	0.85	0.62	+37%
Error Rate	11%	24%	-54%

where:

$$S_{\text{syntax}} = \text{Syntactic correctness probability}$$
$$S_{\text{semantic}} = \text{Semantic accuracy score}$$
$$S_{\text{schema}} = \text{Schema compliance score}$$
$$\alpha, \beta, \gamma = \text{Weight parameters (optimized during training)}$$

C. Performance Metrics

System performance is evaluated using comprehensive metrics:

$$\text{Exact Match Accuracy} = \frac{\text{Correct Exact Matches}}{\text{Total Queries}} \times 100\%$$
$$\text{Execution Accuracy} = \frac{\text{Queries with Correct Results}}{\text{Total Queries}} \times 100\%$$
$$\text{Response Time} = t_{\text{processing}} + t_{\text{execution}}$$
$$\text{User Satisfaction} = \frac{\sum_{i=1}^{n} \text{Rating}_i}{n \times 5} \times 100\%$$
$$\text{Learning Efficiency} = \frac{\text{Concepts Mastered}}{\text{Time Spent}}$$

D. Optimization Objective

The model training objective combines multiple loss functions:

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{CE}} + \lambda_2 \mathcal{L}_{\text{RL}} + \lambda_3 \mathcal{L}_{\text{aux}}$$

where:

$$\mathcal{L}_{\text{CE}} = \text{Cross-entropy loss for sequence generation}$$
$$\mathcal{L}_{\text{RL}} = \text{Reinforcement learning loss based on execution rewards}$$
$$\mathcal{L}_{\text{aux}} = \text{Auxiliary losses for entity and schema prediction}$$
$$\lambda_1, \lambda_2, \lambda_3 = \text{Balancing hyperparameters}$$

VI. Results and Analysis

A. Performance Evaluation

B. Accuracy Analysis

The system achieved 89% accuracy across comprehensive testing, with breakdown by query type:

- Simple SELECT queries: 94% accuracy
- JOIN operations: 87% accuracy
- Aggregate functions: 85% accuracy
- Nested subqueries: 82% accuracy
- Complex business logic: 79% accuracy

C. Response Time Analysis

Detailed response time analysis reveals:

- Query translation: 1.2 seconds average
- Database execution: 0.9 seconds average
- Network overhead: 0.2 seconds average
- Total response time: 2.1 seconds average
- 95th percentile response time: 3.4 seconds

D. User Study Results

A comprehensive user study with 150 participants across three skill levels revealed:

- 92% reported reduced query formulation time
- 88% preferred SQLify over traditional SQL editors
- 85% reported improved SQL understanding through the learning modules
- 94% found the interface intuitive and user-friendly
- 87% would recommend SQLify to colleagues

E. Scalability Testing

Load testing demonstrated strong scalability characteristics:

- Maximum concurrent users: 500+
- Queries per minute: 1500+
- Database connections: 200+
- Memory usage: 2.5GB at peak load
- CPU utilization: 65% at peak load

VII. Conclusion and Future Work

A. Summary of Contributions

The model in SQLify is effective because it overcomes major challenges of accessibility to databases due to its advanced level of natural language processing and AI-based query generation. The system facilitates a great number of benefits compared with the conventional SQL interaction approaches especially in accessibility, efficiency and learning.

Key contributions include:

- Development of a robust NL2SQL system with 89% query translation accuracy.
- Seamless integration with multiple database systems (MySQL, PostgreSQL, Trino).
- Implementation of synthetic data generation for testing and prototyping.
- Creation of an intuitive, gamified learning environment for SQL skill development.
- Comprehensive performance validation through extensive user studies.

B. Limitations

Current limitations include:

- Support for advanced SQL features (CTEs, window functions) requires enhancement.
- Performance with complex, multi-sentence queries needs improvement.
- Integration with proprietary database systems is limited.
- Offline functionality requires further development.

C. Future Enhancements

Planned future enhancements include:

- Integration of additional AI models for improved accuracy and robustness.
- Expansion to NoSQL database support (MongoDB, Cassandra).
- Mobile application development for iOS and Android platforms.
- Advanced analytics and automated insights generation.
- Enterprise-grade security and compliance features.
- Multi-lingual support beyond English.

D. Final Assessment

SQLify is a major breakthrough, which will allow making interaction with a database more accessible and efficient. The system enables users of all technical skill levels to realize the maximum exploitation of their data by fixing the gap between natural language and SQL. The successful usage and excellent user reviews confirm the strategy and initiate positive prospects of further advancement to AI-assisted database interaction.

The combination of usefulness with educationalism in the system forms a special platform that will not only address the short-term issue of database access but also allow developing skills in the long term. With the further development of AI and NLP technologies, systems such as SQLify will become even more significant as a means of democratizing access to data and making decisions based on it in organizations.

References

[1] Zhong, V., Xiong, C., & Socher, R. (2017). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *arXiv preprint arXiv:1709.00103*.

[2] 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. *Proceedings of EMNLP*.

[3] Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J. G., Liu, T., & Zhang, D. (2019). Towards complex text-to-SQL in cross-domain database with intermediate representation. *Proceedings of ACL*.

[4] Wang, B., Shin, R., Liu, X., Polozov, O., & Richardson, M. (2020). RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. *Proceedings of ACL*.

[5] Lin, X. V., Socher, R., & Xiong, C. (2021). Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. *Proceedings of EMNLP*.

[6] Touvron, H., et al. (2023). LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

[7] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*.

[8] Brown, T. B., et al. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems, 33*.