Text-to-SQL Conversion Strategies: Enhancing Accuracy Through Schema Linking and Modular Prompting

Candidate: Tahsin Jawwad
Supervisor: Dr. Ramon Lawrence
Course: COSC 448 – Directed Studies in Computer Science
Term: Winter 2024/25 Term 2

Introduction

Text-to-SQL is the process of converting natural language questions into SQL queries, enabling users to retrieve database information without explicit SQL knowledge. This capability is crucial for database query tools, educational platforms, and various business applications. My Directed Studies project under Dr. Ramon Lawrence specifically aimed at exploring and enhancing Text-to-SQL methodologies, using Generative AI models, primarily GPT-3.5-turbo, and focusing on schema linking and modular prompting strategies to address limitations found in current methods.

The goal, as outlined in the course description, was to investigate different Generative AI approaches, replicate existing experiments, and propose improvements by integrating structural and semantic metadata into the translation process. Collaborations within the database research group provided valuable insights and resources. This research aimed to address current shortcomings in Text-to-SQL systems, specifically schema ambiguity and the limitations of modular prompt-based query generation methods using GPT-3.5.

Overview of Research and Development (January–April 2025)

Literature Review and Initial Replication

The first phase involved intensive research into recent Text-to-SQL approaches, focusing on LangChain, C3, and DIN-based strategies mentioned in Nascimento et al. (2024) in an Oracle database.

LangChain-Based Strategies

LangChain offers modular frameworks to simplify the implementation of natural language processing with Large Language Models (LLMs) for database querying tasks. Specifically, it provides three predefined methods:

- **SQLQueryChain:** Automatically extracts database metadata and creates prompts to facilitate direct conversion of natural language into SQL queries.

- **SQLDatabaseSequentialChain:** Determines relevant tables from user queries before invoking SQLQueryChain, making it effective for large databases by narrowing the search space for the LLM.

- **SQLDatabaseAgent:** Interacts flexibly with databases, incorporating schema awareness and error recovery mechanisms to correct and regenerate SQL queries dynamically when execution errors occur.

C3-Based Strategies

The C3 approach relies on careful prompt engineering with GPT models. It addresses schema complexity and ambiguities by employing:

- **Clear Prompting (CP):** Selectively includes only relevant tables and columns in prompts to reduce token usage and confusion.

- **Calibration with Hints (CH):** Provides explicit hints to mitigate common GPT biases such as unnecessary column selections and incorrect SQL constructs.

- **Consistent Output (CO):** Generates multiple reasoning paths and employs voting mechanisms to choose the most consistent SQL query from multiple LLM outputs, thereby reducing variability and improving accuracy.

DIN-Based Strategies

DIN-SQL decomposes the text-to-SQL conversion process into clearly defined, sequentially executed steps, enhancing accuracy and clarity:

- **Schema Linking:** Employs a chain-of-thought prompting method to match natural language elements explicitly to database schema components (tables, columns, foreign keys).

- **Query Classification and Decomposition:** Classifies queries into easy (single-table), non-nested (joins required), and nested (complex subqueries), using tailored prompts for each type.

- **SQL Generation:** Utilizes intermediate representations such as NatSQL for non-nested complex queries and structured multi-step prompts for nested queries.

- **Self-Correction:** Implements a correction mechanism to identify and rectify minor SQL syntax and logic errors through additional prompts, enhancing overall reliability.

I replicated experiments from Nascimento et al. (2024), involving setup and execution of tests on the Mondial database, verifying reproducibility and identifying initial areas of improvement. The original findings can be found in the references. Here is an overview of the replicated findings:

	Accuracy								
Model/LLM	Simple	Medium	Complex	Overall	Input Tkn	Output Tkn	Tokens	Est. Cost	Exec Time
Manual Prompt Chain + DANKE									
SQLCoder	0.42	0.39	0.21	0.34					
SQLQueryChain	0.82	0.61	0.35	0.59	595,825	3,642	599,467	$ 1.80	0:01:50
SQLQueryChain - with samples	0.73	0.64	0.38	0.58	811,725	2,442	814,167	$ 2.44	0:02:27
SQLDatabaseSequentialChain	0.61	0.36	0.24	0.40	75,291	3,297	78,588	$ 0.12	0:01:44
SQLDatabaseSequentialChain - with samples	0.64	0.42	0.24	0.43	65,688	3,424	69,112	$ 0.11	0:01:43
SQL Database Agent	0.55	0.39	0.18	0.37					0:04:52
SQL Database Agent - with samples	0.55	0.48	0.21	0.48					0:06:40
DIN									
C3	0.76	0.36	0.27	0.46	177,696	448,447	626,143	$ 1.16	2:44:42
C3 + DIN									

The manual prompt chain + DANKE was not run due to lack of access to the DANKE keyword search tool, whereas DIN-based strategies were shown to be costly. All experiments were run using gpt-3.5-turbo-16k. Generally, the replicated findings resembled that of the paper, with a simple SQLQueryChain and C3 performing the best.

Integration of New Features

From further research, I introduced two new strategies:

- mschema: a semi-structure representation of database schema used by a leading framework in Text-to-SQL research (XiYan-SQL). This can be found here.
- Few-Shot Similar Pairs: a way to fetch and give the LLM similar NL questions and its corresponding SQL query.

Experiment results:

	Accuracy								
Model/LLM	Simple	Medium	Complex	Overall	Input Tkn	Output Tkn	Tokens	Est. Cost	Exec Time
Few-Shot + SQLQueryChain - with samples	0.80	0.67	0.67	0.70	124,227	508	124,735	$ 0.46	0:00:19
Few-Shot50 + SQLQueryChain - with samples	0.71	0.65	0.56	0.64	310,284	1,531	311,815	$ 0.94	0:01:19
Mschema + SQLQueryChain - with samples	0.76	0.61	0.41	0.59	1,561,725	3,539	1,565,264	$ 4.70	0:06:19

The Few-Shot was carried out by splitting the given 100 queries in the mondial dataset into test and train groups. The first experiment involved 80 queries in the train, so the experiment was only carried out in the remaining 20. The second experiment involved 50 queries in the train. Both of these experiments were implemented using the simplest best performing framework SQLQueryChain to gain an understanding of the possible improvements. The experiment yielded better results, but this could be biased due to the smaller queries set.

The mschema was integrated in a similar fashion, replacing the normal schema representation in SQLQueryChain with the mschema representation. This resulted in a similar result as without mschema, with the same overall accuracy as the best experiment replication earlier but performing better on complex queries.

Advanced Schema Linking and Validation

Key contributions during this period included:

- Refactoring schema linking processes to fully incorporate mschema.
- Combining C3 and DIN, as suggested in Nascimento et al. (2024)
- Building a validator for common SQL syntax errors.
- Embedding-based retrieval for schema elements following recent advances (Pourrezza et al., 2024).

	Accuracy								
Model/LLM	Simple	Medium	Complex	Overall	Input Tkn	Output Tkn	Tokens	Est. Cost	Exec Time
C3Plus	0.79	0.48	0.26	0.51	2,157,185	390,062	2,547,247	$ 8.03	01:14:48

This implementation actually saw a slight increase in performance from the base C3, but still was disappointing. Some of the potential problems include:

- Not feeding the pipeline the error messages given by the SQL validator
- Not correctly handling extraction errors in the schema linking section
- Mschema integration into the pipeline required a lot of function changes and was not fully and accurately integrated
- Embedding-based retrieval yielded disappointing results in test runs

Research Findings

Importance of Schema Linking

Robust schema linking significantly enhances the accuracy and reliability of Text-to-SQL outputs by preventing common mistakes such as incorrect column or table identification. Methods employing clear prompts and step-by-step reasoning (chain-of-thought) demonstrated substantial accuracy improvements, particularly seen in DIN-SQL and RESDSQL implementations. The emphasis should be on correctly reducing large schemas into smaller, more manageable schemas.

BASE-SQL and RESDSQL Evaluation

- **BASE-SQL:** An effective baseline, highlighting simplicity and ease of implementation. Although I was not able to produce the code, trying to research and replicate the pipeline found here can be beneficial.

- **RESDSQL:** Provided sophisticated schema matching and demonstrated high potential for managing complex schemas through substring matching with source code which can be found here.

Limitations Encountered

- Usage of GPT-3.5-turbo-16k exclusively restricted comprehensive comparative analysis with models such as GPT-4 and LLaMA.

- The complexity of schemas, especially in large industrial databases, posed significant accuracy challenges, demonstrating the necessity of adaptive schema handling methods.

Conclusions and Future Directions

This research successfully incorporated state-of-the-art schema linking and modular prompting into GPT-3.5-driven Text-to-SQL frameworks, showing modest yet promising enhancements. For continued advancement, the following strategies are recommended:

- Exploring and benchmarking additional AI models, particularly GPT-4 and LLaMA, for broader insight and potential accuracy gains.

- Deeper and more extensive integration of RESDSQL to leverage its advanced schema matching capabilities.

- Address scalability and robustness for real-world databases, focusing on adaptive schema handling methods.

- Incorporation of Few-Shot Similar Pairs as it has shown an increase in accuracy in the experiments.

- Use of a keyword-search method that stores table and column names along with its possible NL references

Through careful documentation and structured experimentation, this project provides a solid foundation for subsequent research aimed at bridging existing accuracy gaps in real-world Text-to-SQL applications.

References

- Nascimento, E. et al. (2024). "Text-to-SQL Meets the Real-World." *26th International Conference on Enterprise Information Systems.* https://www.scitepress.org/Papers/2024/125552/125552.pdf

- Gao, Y. et al. (2025). “A Preview of XiYan-SQL: A Multi-Generator Ensemble Framework for Text-to-SQL” *arXiv.* arXiv:2411.08599

- Pourrezza, M. et al. (2024). “SQL-Encoder: Improving NL2SQL In-Context Learning Through a Context-Aware Encoder”. *arXiv*. https://arxiv.org/html/2403.16204v1