Enhancing Text-to-SQL with In-Context Learning: A Multi-Agent Approach Based on CHESS

Renato O. Miyaji[1], Rafael M. Fernandes[2], Krysthian F. Martins[3], Jorge Melegati[4], Pedro L. P. Corrêa[1]

[1]Escola Politécnica – Universidade de São Paulo (USP)

[2]Faculdade de Filosofia, Letras e Ciências Humanas – Universidade de São Paulo (USP)

[3]Banco Safra

[4]Faculty of Engineering - Free University of Bozen-Bolzano

{re.miyaji,rafael.macario,krysthian,pedro.correa}@usp.br

{jorge}@jmelegati.com

Abstract. *Text-to-SQL has gained increasing attention with Large Language Models (LLMs). While existing architectures have demonstrated the potential of multi-agent systems there remains significant room for improvement. In this work, we extend the CHESS framework by integrating In-Context Learning (ICL) techniques into the Candidate Generator module, evaluating three strategies: Zero-Shot, Few-Shot Learning, and Retrieval-Augmented Generation (RAG). We implement the system using GPT-4o, and perform experiments on the financial dataset from BIRD-SQL. Results show that Few-Shot Learning and RAG significantly outperform the standard approach. Compared to Zero-Shot (59.31% Execution Accuracy (EX), 0.412 ROUGE-1), RAG significantly boosted performance, increasing EX to 69.48% and ROUGE-1 to 0.652.*

1. Introduction

The task of code generation by Large Language Models (LLMs) gained significant research interest in the fields of Software Engineering and Databases. A particularly prominent scope within this domain is the generation of code for database queries, especially Structured Query Language (SQL), based on natural language instructions. This task, commonly referred to as Text-to-SQL [Hong et al. 2024], seeks to bridge the gap between human language and machine-readable query languages, facilitating more intuitive and accessible data interaction [Katsogiannis-Meimarakis and Koutrika 2023]. However, several challenges hinder the broader and more effective adoption of LLMs in this context. Among these challenges are the need to process large volumes of metadata, the difficulty of performing reasoning to correctly interpret user intent, the necessity of generating valid and executable queries, and the inherent ambiguities in natural language instructions [Talaei et al. 2024].

To address these limitations, various methods have been proposed in the literature. One of the state-of-the-art approaches is Contextual Harnessing for Efficient SQL Synthesis (CHESS) [Talaei et al. 2024], which distinguishes itself from other methods through its modular architecture and favorable cost-performance ratio. Unlike other heavy computational frameworks, CHESS manages to achieve competitive performance while

maintaining lower computational demands, making it suitable for real-world applications. CHESS operates as a multi-agent architecture that integrates several techniques designed to tackle the key challenges in the Text-to-SQL pipeline. It consists of four independent agents, each responsible for a distinct subtask: the Information Retriever (IR) identifies and extracts relevant contextual data, the Schema Selector (SS) narrows down large schemas to relevant subsets, the Candidate Generator (CG) generates and iteratively refines query candidates, and the Unit Tester (UT) validates the generated queries using unit tests [Talaei et al. 2024]. This modular decomposition enhances flexibility and interpretability, and allows for targeted improvements in each component.

Despite the strengths of the CHESS framework, significant challenges persist—particularly in the Candidate Generation phase. This stage involves producing candidate SQL queries that are not only syntactically correct but also semantically aligned with the user's intent. The complexity of this task increases considerably when dealing with databases that have intricate schemas, requiring a deeper level of reasoning and context understanding [Hong et al. 2024]. These challenges highlight the need for more effective techniques that can enhance the generation process under such demanding conditions.

To improve the Candidate Generation step in the Text-to-SQL task, this work explores In-Context Learning techniques—Few-Shot Learning and Retrieval-Augmented Generation (RAG) [Lewis et al. 2020]. These methods enrich the model's input with contextual examples, enhancing its ability to handle complex queries without additional training. While Few-Shot Learning guides generation through curated prompts, RAG dynamically retrieves relevant examples, improving accuracy and reducing hallucinations in ambiguous scenarios.

Overall, the proposed enhancement to the CHESS architecture represents a promising direction for improving Text-to-SQL performance. By combining the strengths of modular multi-agent design with the adaptive capabilities of modern prompt-based learning methods, we aim to achieve more reliable and contextually aware SQL query generation. To validate our approach, we conducted experiments using the financial schema from the BIRD-SQL dataset, a benchmark known for its complex queries and diverse schema elements. We evaluated three prompting strategies using Execution Accuracy and ROUGE-1 as performance metrics. The results demonstrate a consistent improvement with the integration of ICL techniques: Execution Accuracy increased from 59.31% in the Zero-Shot setting to 65.26% with Few-Shot Learning and further to 69.48% with RAG. Similarly, ROUGE-1 scores improved from 0.412 to 0.618 and 0.652, respectively. These findings confirm that contextual prompting significantly enhances the quality and correctness of SQL query generation.

2. Related Works

The task of Text-to-SQL using Language Models (LMs) and LLMs has been extensively explored in the literature [Hong et al. 2024]. Earlier approaches primarily relied on classical transformer-based models such as BERT [Yin et al. 2020] and T5 [Li et al. 2023], which were fine-tuned on labeled datasets to learn the mapping from natural language questions to SQL queries. With the emergence of more powerful LLMs, particularly from 2023 onward, researchers began to explore ICL as an alternative to fine-tuning.

ICL enables pre-trained LLMs to solve downstream tasks by conditioning on examples provided within the prompt, without modifying the model's weights [Hong et al. 2024]. Reasoning techniques such as Chain-of-Thought (CoT) prompting have been proposed to enhance the model's ability to perform logical inference by explicitly breaking down the reasoning process into interpretable steps [Wei et al. 2022]. In addition, task decomposition strategies have gained traction. An example is DIN-SQL [Pourreza and Rafiei 2023], which decomposes the Text-to-SQL pipeline into three phases, allowing each subtask to be addressed more effectively by specialized prompts and reasoning pathways.

Another important contribution in this domain is the C3 framework [Dong et al. 2023], which incorporates three core components: Clear Prompting, Calibration with Hints, and Consistent Output. This multi-stage pipeline improves robustness and correctness, especially in complex query scenarios. Moreover, combinations of DIN-SQL and C3 techniques have also been proposed to further enhance performance [Nascimento and Casanova 2024].

More recently, new architectures have emerged that take a multi-agent perspective. These methods typically involve agents responsible for understanding the natural language question, interpreting the database schema, generating SQL queries, and validating the outputs [Hong et al. 2024]. By assigning specific roles to each agent, these approaches aim to modularize the task and reduce the cognitive load on individual components [Talaei et al. 2024]. Despite these advances, there remains substantial room for improving specific stages of the Text-to-SQL process. Candidate generation, in particular, continues to pose challenges when dealing with complex schemas or ambiguous natural language instructions [Hong et al. 2024]. Therefore, research focused on refining and enhancing individual components has the potential to significantly contribute to the development of more accurate, scalable, and generalizable Text-to-SQL systems.

3. Methodology

3.1. Proposed Method

The proposed method builds upon the CHESS architecture [Talaei et al. 2024], which employs a multi-agent system to modularize and optimize the Text-to-SQL pipeline. CHESS consists of four distinct agents, each responsible for a specific subtask in the generation of SQL queries from natural language instructions. This structured decomposition enables fine-grained control and targeted improvements at each stage of the process.

The first agent, the Information Retriever (IR), is responsible for gathering relevant information from the input and the database catalog using three key tools: keyword extraction, entity retrieval, and context retrieval. To begin, the agent identifies primary keywords and key phrases in the natural language query. These keywords are then used in the entity retrieval phase, where the agent searches for semantically and syntactically similar values within the database. This is achieved using a combination of edit distance and a hierarchical retrieval strategy that leverages Locality Sensitive Hashing (LSH) and embedding-based similarity, allowing the agent to effectively match keywords with relevant database values and their associated columns. Finally, the context retrieval tool enables the agent to access the database catalog, which contains valuable schema metadata that is stored in a vector database and queried using semantic similarity to ensure

that the most relevant contextual information is retrieved and passed to the model, significantly improving its understanding of the query and reducing the likelihood of errors [Talaei et al. 2024].

Following the IR, the Schema Selector (SS) is designed to reduce the complexity of large database schemas by selecting only the tables and columns that are relevant to the user's query. To achieve this, the SS agent employs three tools: filter column, select tables, and select columns [Talaei et al. 2024].

The third agent, the Candidate Generator (CG), plays a central role by generating high-quality SQL query candidates and refining them iteratively [Talaei et al. 2024]. In our work, we implemented three different variations of the CG to evaluate the impact of distinct prompting strategies. The first variation applies a standard Zero-Shot Learning approach, where the LLM attempts to generate a query without additional examples. The second variation introduces Few-Shot Learning, incorporating 10 curated examples directly into the prompt to guide the model's reasoning and structure. The third and most advanced approach employs Retrieval-Augmented Generation (RAG) [Lewis et al. 2020], using a vector database to retrieve the top 10 similar examples based on the embedding similarity of the input question.

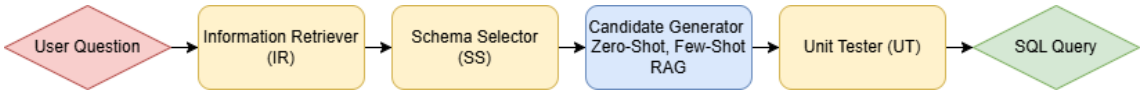

Figure 1. Flowchart of the proposed Text-to-SQL system architecture, highlighting the modified Candidate Generator module incorporating Zero-Shot, Few-Shot, and RAG strategies.

We implemented the RAG system using the Chroma vector database, enabling efficient retrieval of contextually similar examples. The retrieval was performed via vector similarity search with K = 10, ensuring that the prompt was enriched with relevant past examples that could assist in query formulation [Lewis et al. 2020]. This dynamic augmentation of the input context aims to improve generation accuracy, especially for complex or ambiguous queries where prior knowledge is essential. Finally, the Unit Tester (UT) validates the generated SQL queries using natural language unit tests guided by an LLM. This agent simulates the execution of queries and checks for semantic alignment with the original question, ensuring not only syntactic validity but also that the outputs would satisfy the user's intent [Talaei et al. 2024]. Figure 1 presents the proposed method.

We implemented the entire multi-agent system using the LangGraph framework, which facilitates orchestrating agent interactions with clear state transitions. We used GPT-4o [OpenAI 2025], OpenAI's latest generation LLM, across all components due to its strong performance on language understanding and reasoning tasks, making it well-suited for Text-to-SQL applications.

3.2. Experiments

For the experimental evaluation, we adopted the BIRD-SQL dataset [Li et al. 2024], which is widely used in the literature for the Text-to-SQL task. Unlike traditional benchmarks such as Spider [Yu et al. 2018], BIRD-SQL emphasizes more realistic user-generated questions and includes richer, more diverse database schemas. It also incorporates more complex natural language instructions that often involve nested queries,

aggregations, and joins, making it a more challenging and representative benchmark for testing the capabilities of modern LLM-based systems.

As a proof of concept, we conducted experiments on the financial schema from the BIRD-SQL dataset. This schema, one of the most complex in the benchmark, includes 8 tables and 50 columns, and features natural language questions that typically require multi-step reasoning, complex joins, and advanced data manipulation. Its high level of schema complexity and query diversity makes it a suitable choice for stress-testing the proposed multi-agent Text-to-SQL system.

For evaluation, we adopted Execution Accuracy (EX) as our primary metric. Execution Accuracy measures the percentage of generated SQL queries that produce the same result as the gold-standard query when executed against the target database [Hong et al. 2024]. This metric provides a robust assessment of the semantic correctness of the generated queries, independently of their syntactic structure. In addition to Execution Accuracy, we also computed the Recall-Oriented Understudy for Gisting Evaluation (ROUGE-1) between the generated and gold SQL queries. While not sufficient alone to guarantee semantic equivalence, ROUGE-1 helps assess lexical and structural similarity, offering additional insight into the model's performance.

Finally, to gain a deeper understanding of the model's behavior, we conducted an error analysis using LLMs to identify and classify common types of mistakes in the generated SQL queries. This qualitative analysis complements the quantitative results and provides actionable insights for future improvements in specific stages of the generation process.

4. Results

We conducted experiments using the system with three variations of the Candidate Generator (CG): Zero-Shot Learning, Few-Shot Learning, and Retrieval-Augmented Generation (RAG). Table 1 presents the evaluation metrics Execution Accuracy (EX) and ROUGE-1 for each of the approaches.

Table 1. EX and ROUGE-1 with Zero-Shot, Few-Shot, and RAG in financial BIRD-SQL dataset

Approach	EX	ROUGE-1
Zero-Shot	59.31%	0.412
Few-Shot	65.26%	0.618
RAG	69.48%	0.652

We observed that the application of Few-Shot Learning significantly improved both the EX and ROUGE-1 metrics compared to the standard Zero-Shot Learning approach, with Execution Accuracy (EX) increasing from 59.31% to 65.26% and ROUGE-1 rising from 0.412 to 0.618. This result highlights the importance of In-Context Learning (ICL) techniques when working with LLMs. Furthermore, with the application of Retrieval-Augmented Generation (RAG), which enables dynamic and contextually appropriate retrieval, performance improved even further, reaching 69.48% in EX and 0.652 in ROUGE-1.

Regarding the error analysis, they were categorized into five main groups: Reference to Nonexistent Column/Table, Incorrect Use of JOINs, Error in Conditions (WHERE, HAVING, etc.), Incorrect Grouping or Aggregation, and Others. In the Zero-Shot Learning approach, the most common error categories were: Reference to Nonexistent Column/Table (35%), Incorrect Use of JOINs (27%), Error in Conditions (WHERE, HAVING, etc.) (20%), Incorrect Grouping or Aggregation (5%), and Others (13%). With the RAG approach, the absolute number of errors decreased by 25.4%. However, proportionally, the percentage of Incorrect Use of JOINs errors dropped to 22%, while Reference to Nonexistent Column/Table errors increased to 40%.

The error analysis reveals important insights into the impact of different prompting strategies on the quality of SQL generation. Under the Zero-Shot Learning approach, the high incidence of errors related to referencing nonexistent columns or tables and incorrect use of JOINs indicates difficulties in schema understanding and query structure formulation without contextual guidance. The application of RAG significantly reduced the total number of errors, demonstrating its effectiveness in enhancing query generation through contextually relevant examples. Notably, the decrease of JOIN-related errors suggests an improvement in relational logic. However, the rise in Reference to Nonexistent Column/Table errors indicates a new challenge introduced by RAG—while it improves reasoning and structure, it may occasionally retrieve misleading schema references from similar, but incorrect, examples. This highlights the importance of further refining retrieval mechanisms and schema disambiguation strategies in future work.

5. Conclusions

In this work, we presented an extension of the CHESS architecture for the Text-to-SQL task by incorporating In-Context Learning (ICL) techniques into the Candidate Generator component. By evaluating three prompting strategies—Zero-Shot Learning, Few-Shot Learning, and Retrieval-Augmented Generation (RAG)—we demonstrated that enriching the model with contextual examples leads to significant improvements in both Execution Accuracy and ROUGE-1 scores. Among the approaches, RAG yielded the best overall performance, highlighting the effectiveness of dynamic, context-aware retrieval in complex query generation. Additionally, our error analysis provided insights into the most common failure modes, guiding future improvements in query synthesis and schema interpretation.

As future work, we plan to expand the evaluation of our system to include all schemas available in the BIRD-SQL dataset, which would allow for a more comprehensive assessment of its generalizability across diverse domains and query types. Furthermore, we intend to investigate the impact of using different LLMs within each agent, comparing performance, cost-efficiency, and robustness across models. Moreover, the proposed contributions could be evaluated with other state-of-the-art Text-to-SQL methods.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

References

Dong, X., Zhang, C., Ge, Y., Mao, Y., Gao, Y., Chen, I., Lin, J., and Lou, D. (2023). C3: Zero-shot text-to-sql with chatgpt. *ArXiv*.

Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., and Huang, X. (2024). Next-generation database interfaces: A survey of llm-based text-to-sql. *ArXiv*.

Katsogiannis-Meimarakis, G. and Koutrika, G. (2023). A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, 32:905—-936.

Lewis, P., Perez, E., Piktus, A., Petroni, F., and Karpukhin, V. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Li, J., Hui, B., Cheng, R., Qin, B., and Ma, C. (2023). Graphix-t5: mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press.

Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., Wang, B., Qin, B., Geng, R., Huo, N., et al. (2024). Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Nascimento, E. and Casanova, M. A. (2024). Querying databases with natural language: The use of large language models for text-to-sql tasks. In *Anais Estendidos do XXXIX Simpósio Brasileiro de Bancos de Dados*, pages 196–201, Porto Alegre, RS, Brasil. SBC.

OpenAI (2025). Gpt-4o. Available on: `https://platform.openai.com/docs/models/gpt-4o`. Accessed on 21 April 2025.

Pourreza, M. and Rafiei, D. (2023). Din-sql: decomposed in-context learning of text-to-sql with self-correction. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Talaei, S., Pourreza, M., Chang, Y., Mirhoseini, A., and Saberi, A. (2024). Chess: Contextual harnessing for efficient sql synthesis. *ArXiv*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Yin, P., Neubig, G., Yih, W., and Riedel, S. (2020). Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceeding of the 58th Annual Meeting of the Association for Computational Linguistics*.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., and Wang, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In Riloff, E., editor, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.