Evaluating Quantized Small Language Models in a TableRAG Pipeline for Table Question Answering

Eduardo Da Paz Rodrigues[1], **Luciana Bencke**[1]

Institute of Informatics – Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brazil

eduardo.paz@inf.ufrgs.br, luciana.bencke@inf.ufrgs.br

Abstract. *Large Language Models (LLMs) show strong performance in Table Question Answering (TableQA), but their high computational requirements restrict local deployment. This work investigates whether TableRAG-inspired Retrieval-Augmented Generation pipelines can be adapted to quantized Small Language Models (SLMs) in constrained hardware environments. We evaluate multiple SLMs on the WikiTableQuestions (WTQ) and OTT-QA benchmarks, analyzing preprocessing strategies, prompting approaches, and lightweight retrieval mechanisms. The results show that semantic context enrichment improves Text-to-SQL generation, with the Embedding-content-aware strategy achieving the best performance. The 4-bit quantized Gemma-4-E4B (Ollama) model reached 47.9% denotation accuracy on WTQ, while Qwen2.5-Coder (7B) achieved 42.0%. For table retrieval, the proposed dual-stage approach combining bi-encoder retrieval and cross-encoder reranking improved HITS@1 from 64.2% to 70.7% and MRR from 0.73 to 0.78 on OTT-QA. These results indicate that quantized SLMs can be viable for local TableQA applications under constrained environments.*

1. Introduction

Large Language Models (LLMs) achieve high performance in natural language understanding, reasoning, and code generation tasks, enabling new approaches for question answering and semantic parsing. However, the increasing scale of recent models introduces high computational and memory requirements, limiting their deployment in local environments [Bommasani et al. 2022]. Recent advances in Small Language Models (SLMs) and model quantization provide alternatives for reducing these requirements while maintaining performance on specialized tasks [Touvron et al. 2023, Frantar et al. 2023, Dettmers et al. 2023].

Table Question Answering (TableQA) requires models to interpret structured information and transform natural language questions into executable operations over tabular data. Early approaches investigated semantic parsing over semi-structured tables, including the WikiTableQuestions (WTQ) benchmark [Pasupat and Liang 2015]. To support SQL-based evaluation on WTQ, Shi et al. proposed SQUALL, a dataset that enriches WTQ questions with manually annotated SQL equivalents [Shi et al. 2020]. Later works extended this problem by evaluating Text-to-SQL generation across complex database schemas and domains [Yu et al. 2018]. These benchmarks highlight the difficulty of mapping user questions into structured queries while preserving the underlying table semantics.

Retrieval-Augmented Generation (RAG) is an effective strategy to improve language model performance by incorporating external information during inference [Lewis et al. 2020]. In this context, dense retrieval methods enable semantic search over large collections of information [Karpukhin et al. 2020]. Recent approaches extend RAG concepts to structured data, investigating the retrieval of relevant tables as additional context for reasoning models.

The TableRAG framework integrates retrieval-augmented techniques for reasoning over structured documents [Yu et al. 2025]. Similarly, an adaptation for TableQA scenarios combines table retrieval with language models on the WTQ and OTT-QA benchmarks [de Souza et al. 2024]. However, these approaches focus mainly on large-scale models, leaving the behavior of retrieval-augmented TableQA pipelines using compact and quantized models less explored.

Therefore, this work investigates the use of quantized Small Language Models for TableQA under local hardware constraints. Quantization is considered here as a deployment configuration rather than an independent experimental factor. Accordingly, the study does not aim to isolate the causal effect of quantization, but rather to evaluate whether quantized SLMs can provide useful TableQA performance within constrained local environments.

The study addresses three research questions:

- **RQ1:** To what extent can quantized Small Language Models perform TableQA under constrained local hardware?
- **RQ2:** How do prompting and context selection strategies affect Text-to-SQL generation in quantized Small Language Models?
- **RQ3:** Does lightweight two-stage retrieval improve table selection for TableQA?

The main contributions of this work are **(i)** an empirical analysis of quantized Small Language Models for Table Question Answering, evaluating different models and execution settings under constrained computational resources; **(ii)** an investigation of prompting and context selection strategies for Text-to-SQL generation, including schema-based, content-aware, embedding-based retrieval, and schema-linking approaches; **(iii)** an analysis of retrieval effectiveness and generation errors, including table retrieval performance, semantic errors, and the trade-off between accuracy and inference latency.

These results provide insights into the feasibility of deploying lightweight TableQA systems based on Small Language Models under local resource constraints.

2. Related Work

Table Question Answering (TableQA) requires models to combine natural language understanding with reasoning over structured information. Early benchmarks investigated the problem of answering questions directly over semi-structured tables, such as WikiTableQuestions (WTQ) [Pasupat and Liang 2015]. More recent benchmarks, such as Spider, extended this research direction by evaluating Text-to-SQL systems in cross-domain database scenarios, highlighting the challenges involved in translating natural language questions into executable database queries [Yu et al. 2018].

Retrieval-Augmented Generation (RAG) has emerged as an approach to improve language model responses by incorporating external information retrieved during infer-

ence [Lewis et al. 2020]. Dense retrieval methods further enabled semantic matching between queries and large information collections, allowing relevant contexts to be selected before generation [Karpukhin et al. 2020]. Following this direction, recent studies have explored retrieval mechanisms for structured reasoning tasks, including table-based question answering.

Yu et al. proposed a retrieval-augmented framework for heterogeneous document reasoning, demonstrating that retrieving structured information can improve the reasoning capabilities of large language models [Yu et al. 2025]. Subsequently, de Souza et al. adapted this idea specifically for TableQA by combining table retrieval with language model generation and evaluating the approach on WTQ and OTT-QA benchmarks [de Souza et al. 2024]. However, these approaches mainly rely on large-scale language models, leaving the behavior of retrieval-augmented table reasoning with smaller and locally executable models insufficiently explored.

Recent advances in Small Language Models (SLMs) have demonstrated that compact architectures can achieve performance while reducing computational requirements. Models such as LLaMA showed that relatively compact architectures can provide strong language understanding capabilities [Touvron et al. 2023], while quantization techniques, such as QLoRA, further reduce memory consumption and enable efficient deployment of language models in constrained environments [Dettmers et al. 2023].

Studies in resource-constrained scenarios have also investigated the applicability of SLMs for specialized tasks. Recent work has evaluated quantized SLM agents executed locally for literature screening tasks, comparing different prompting and agent-based strategies against larger language models [Freitas Filho and Fernandes 2025]. The results show that SLMs can achieve performance under specific configurations, although effectiveness may vary depending on the complexity of the adopted strategy. These findings highlight the potential of SLMs as efficient alternatives for scenarios where computational resources and local execution are important considerations.

Despite these advances, the combination of quantized SLMs with TableRAG-inspired pipelines remains an open research direction. In particular, it is still unclear whether the benefits observed with large language models can be maintained when retrieval mechanisms, context selection strategies, and prompting approaches are applied to smaller locally executable models.

Beyond evaluating only task-level performance, recent research has also emphasized the importance of analyzing qualitative failures and limitations of language models. Sousa et al. [Sousa et al. 2025] investigated structural issues in code generated by LLMs, showing that functional correctness does not necessarily imply high-quality outputs. Similar challenges appear in Text-to-SQL scenarios, where generated queries may be syntactically valid but contain semantic errors, incorrect schema references, or hallucinated attributes.

This work addresses these gaps by evaluating retrieval strategies, prompting approaches, and generation behavior of quantized Small Language Models in a local TableQA scenario. In addition to measuring task performance, we analyze retrieval effectiveness and generation failures to better understand the limitations and opportunities of lightweight TableRAG-inspired systems.

3. Methodology

Following the evaluation methodology introduced by de Souza et al. [de Souza et al. 2024], this work implements a TableRAG-inspired pipeline adapted for quantized Small Language Models. The framework is reproduced under constrained hardware conditions, with modifications focused on model scale, inference environment, and prompt construction strategies. The architecture consists of two stages: an offline indexing stage, responsible for table preparation and embedding generation, and an online inference stage, responsible for retrieving relevant information, constructing prompts, and generating SQL queries.

During offline processing, tables from the datasets are normalized, converted into SQLite databases, and serialized into textual representations. These representations are then used to generate dense embeddings for table retrieval and content selection. During the online inference stage, the user's query is first processed by the retrieval module, which identifies the most relevant tables. Subsequently, the selected tabular information is incorporated into the prompt, enabling the quantized SLM to generate the corresponding SQL query, which is then executed by the SQLite engine.

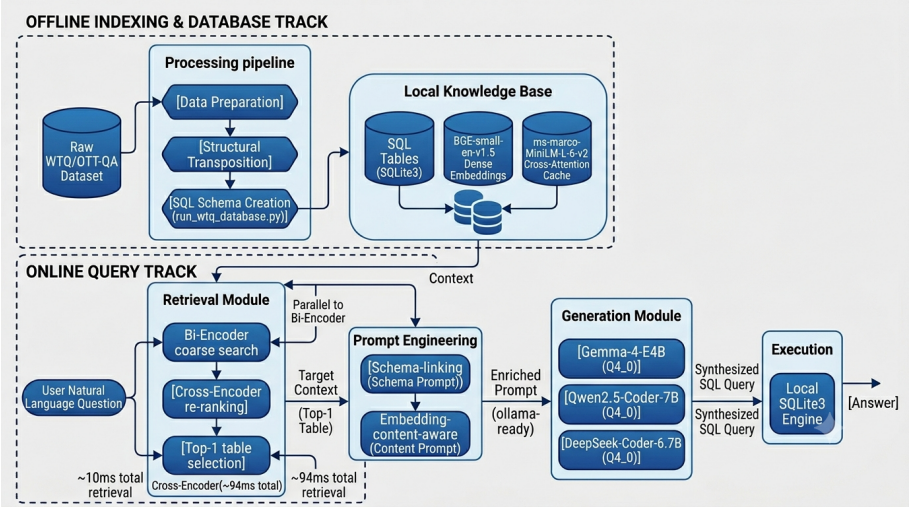


Figure 1. Overview of the TableRAG approach and experimental design. Retrieval and Text-to-SQL generation are evaluated separately rather than as an end-to-end pipeline.

3.1. Table Retrieval

The retrieval module identifies relevant tables before SQL generation. Initially, each table is transformed into a textual representation containing its schema information and metadata. These representations are encoded into dense vectors using the BGE-small-en-v1.5 embedding model [Xiao et al. 2023], a compact general-purpose encoder optimized for semantic retrieval tasks.

Given a natural language question, the system computes cosine similarity between the query embedding and the indexed table embeddings. The top-20 candidates are selected as input for the second retrieval stage.

To improve ranking quality, a cross-encoder re-ranking step is applied using the ms-marco-MiniLM-L-6-v2 model [Reimers and Gurevych 2019], which jointly encodes

query and document pairs to produce more precise relevance estimates than bi-encoders. After re-ranking, the final candidate list is used for evaluation.

This two-stage strategy provides a balance between retrieval efficiency and ranking accuracy, allowing the system to perform semantic filtering while maintaining compatibility with local execution environments.

3.2. Prompting Strategies

To evaluate the influence of contextual information on Text-to-SQL generation, four prompting strategies were investigated. The approaches progressively introduce additional information from the table while controlling the amount of context provided to the model. Table 1 summarizes the evaluated strategies.

Table 1. Prompting strategies evaluated in this work.

Method	Schema Information	Additional Context
Zero-shot	Complete	None
Content-aware	Complete	Random sampled rows
Embedding-content-aware	Complete	Semantically retrieved rows
Schema-linking	Filtered	Relevant columns + retrieved rows

Zero-shot. The baseline strategy provides only the database schema and the user question. The schema contains table names and column names, but no examples from the table contents are included.

Content-aware. This strategy extends the baseline by adding randomly sampled table rows. The objective is to provide the model with examples of possible values and improve the connection between natural language expressions and table attributes.

Embedding-content-aware. Instead of randomly selecting rows, this approach retrieves table tuples based on their semantic similarity with the user question. The selected rows are added to the prompt as contextual examples, reducing irrelevant information while preserving query-related content.

Schema-linking. This strategy reduces schema complexity by selecting the most relevant columns before prompt construction. Column relevance is estimated using embedding similarity between the question and column names. The selected columns are combined with semantically retrieved rows to create the final prompt representation. This approach is inspired by schema-linking strategies investigated in Text-to-SQL research, where filtering irrelevant schema elements before generation has been shown to improve SQL accuracy [Gao et al. 2023].

4. Experimental Setup

4.1. Datasets

The evaluation follows the decoupled methodology proposed by de Souza et al. [de Souza et al. 2024], where retrieval and generation components are evaluated independently. This separation allows analyzing the individual contribution of table retrieval and Text-to-SQL generation.

WikiTableQuestions (WTQ). WTQ is used to evaluate the Text-to-SQL generation capability of the evaluated models. Since the benchmark provides the corresponding table for each question, the retrieval stage is bypassed and the model receives the correct table context directly. The generated SQL queries are executed in SQLite, and the resulting answers are compared against the expected denotations.

Because WTQ tables originate from web sources, their schemas may contain inconsistencies such as invalid column names, duplicated attributes, and heterogeneous data types. Before database creation, a normalization procedure was applied to ensure SQL compatibility.[1]

Open Text-to-Table Question Answering (OTT-QA)[Chen et al. 2021]. OTT-QA is used exclusively to evaluate the retrieval component. Unlike WTQ, the correct table is not provided with the question, requiring the system to identify the relevant table from a larger collection. Retrieval performance is measured using HITS@K and Mean Reciprocal Rank (MRR).

4.2. Hardware and Models

Experiments were conducted on a local machine with an NVIDIA GeForce RTX 3060 GPU (12 GB VRAM), 16 GB RAM, and 1 TB storage, running Ubuntu 24.04. SQLite3 was used for table storage and execution of generated SQL queries.

Models were selected to cover different parameter scales (3.2B–120B), general-purpose and code-specialized architectures, and locally deployable versus API-based inference. Llama 3.2 and Gemma-4-E4B represent compact general-purpose models, while Qwen2.5-Coder and DeepSeek-Coder represent code-specialized alternatives. Llama 3.3 (70B) and GPT-OSS (120B) serve as larger-scale reference models and are not directly comparable to the local models due to their different execution environments.

Table 2 summarizes the evaluated models, execution environments, and quantization schemes. Local models use 4-bit quantization (Q4_0 or Q4_K_M), while the API-served models use FP8 (Llama 3.3) and MXFP4 (GPT-OSS). Quantization is treated as a deployment configuration rather than as an independent experimental factor; therefore, the study compares models across different scales and quantization formats rather than measuring quantization-induced degradation relative to full-precision models.

4.3. Evaluation Protocol

All models were evaluated with temperature set to zero to ensure deterministic generation and reproducibility across runs. For WTQ evaluation, a randomly selected subset of 1,000 question-answer pairs was used due to computational constraints, with each sample evaluated under all four prompting strategies described in Section 3.2. For OTT-QA, retrieval performance was evaluated over the full set of 2,214 validation samples containing 8,891 candidate tables, comparing bi-encoder-only retrieval against the proposed dual-stage bi-encoder plus cross-encoder configuration.

[1] In total, 13,445 columns were processed: 4,676 renamed, 476 numeric conversions, 166 duplicate resolutions, and 75 unnamed column corrections.

Table 2. Specifications of all evaluated models. Local models run via Ollama on the hardware described above; API models run on external infrastructure exceeding local memory capacity.

Model	Params	Context	Quant.	Env.
Llama 3.2	3.2B	131,072	Q4_K_M	Local
DeepSeek-Coder	7B*	16,384	Q4_0	Local
Qwen2.5-Coder	7.6B	32,768	Q4_K_M	Local
Gemma-4-E4B**	8.0B	131,072	Q4_K_M	Local
Llama 3.3 (70B)	70B	131,072	FP8	API
GPT-OSS (120B)	120B	128,000	MXFP4	API

*Commonly referred to as 6.7B in the DeepSeek-Coder family naming; Ollama reports 7B raw parameters.

**The "E4B" designation refers to effective parameter count under Gemma 3n's per-layer embedding scheme; raw parameter count is 8.0B.

4.4. Evaluation Metrics

We evaluate Text-to-SQL generation using denotation accuracy on WTQ and table retrieval using HITS@1, HITS@5, HITS@10 and MRR on OTT-QA. Denotation accuracy measures answer correctness after SQL execution, while HITS@K and MRR assess retrieval coverage and ranking quality, respectively.

5. Results and Discussion

5.1. Overall TableQA

The Text-to-SQL evaluation investigates how contextual information affects the generation capability of quantized SLMs. Table 3 presents the denotation accuracy obtained under different prompting strategies.

Table 3. Denotation accuracy (%) on WikiTableQuestions for different prompting strategies.

Model	Zero-shot	Content-aware	Embedding-content-aware	Schema-linking
Llama 3.2 (3B)	14.4	24.0	**28.0**	24.3
DeepSeek-Coder (6.7B)	21.7	29.1	**32.6**	28.4
Qwen2.5-Coder (7B)	25.3	37.8	**42.0**	36.4
Gemma-4-E4B (Ollama)	19.8	38.9	**47.9**	42.2
Llama 3.3 (70B)	22.4	31.8	**47.1**	41.9
GPT-OSS (120B)	24.0	34.4	**38.4**	33.1

Across all evaluated models, the Embedding-content-aware strategy achieved the highest accuracy. This suggests that selecting table rows according to their semantic similarity with the question provides more useful information than randomly sampled content.

The comparison between Content-aware and Embedding-content-aware also indicates that additional context alone is not sufficient. The quality and relevance of the retrieved examples play an important role in improving SQL generation.

Among local models, Gemma-4-E4B (Ollama) achieved the highest accuracy, reaching 47.9%. This result was obtained under the enriched prompting strategy and

indicates that compact models can benefit substantially from carefully selected table context. However, direct comparisons with large-scale reference models should be interpreted carefully because models were executed under different computational settings and configurations.

Qwen2.5-Coder (7B) also showed strong performance, achieving 42.0% accuracy. Its results suggest that models optimized for code-related tasks can be useful for Text-to-SQL generation, even when operating with fewer parameters than general-purpose large models.

5.2. Table Retrieval

The retrieval evaluation investigates whether the implemented lightweight retrieval pipeline can identify relevant tables before SQL generation. Experiments were conducted on 2,214 OTT-QA validation samples containing 8,891 candidate tables.

Table 4 compares the bi-encoder retrieval baseline with the proposed dual-stage approach including cross-encoder re-ranking.

Table 4. Retrieval performance on the OTT-QA dataset.

Configuration	HITS@1	HITS@5	HITS@10	MRR	Latency
Bi-Encoder Only	0.6423	0.8369	0.8916	0.7250	10.44 ms
Bi + Cross-Encoder	**0.7069**	**0.8708**	**0.9124**	**0.7780**	94.95 ms

Cross-encoder re-ranking improves all four retrieval metrics, increasing HITS@1 from 64.2% to 70.7% and MRR from 0.73 to 0.78. The improvement suggests that re-ranking is particularly useful when several candidate tables are semantically similar.

The improvement comes with an increase in latency, from 10.44 ms to 94.95 ms per query. However, the retrieval stage remains below 100 ms, suggesting that the proposed approach is compatible with interactive local applications.

5.3. Efficiency Analysis

The latency analysis evaluates the relationship between model accuracy and execution cost under local hardware constraints. Figure 2 presents the average inference latency and the highest denotation accuracy obtained by each model.

The results show a trade-off between accuracy and computational cost. Gemma-4-E4B (Ollama) achieved the highest accuracy among evaluated local models but presented higher latency, with an average execution time of 7.37 seconds and a P_{95} latency of 21.07 seconds.

Conversely, Qwen2.5-Coder (7B) provided a more balanced behavior, achieving 42.0% accuracy with lower inference cost. These results indicate that model selection for local TableQA systems should consider both reasoning capability and deployment constraints.

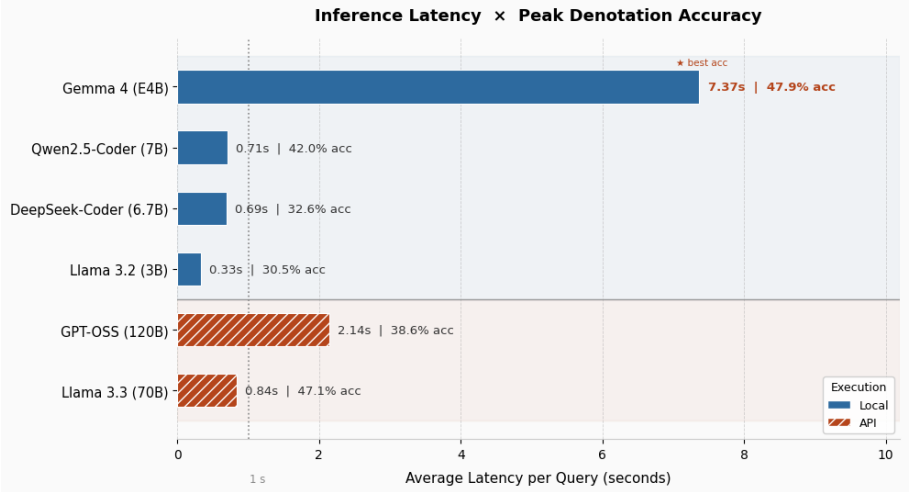


Figure 2. Accuracy and latency comparison of evaluated models under local execution conditions.

5.4. Error Analysis

To analyze generation failures, schema hallucination errors were measured across evaluated models. Table 5 presents the number of generated queries containing invalid schema references.

Table 5. Schema hallucination analysis on WTQ.

Model	Samples	Errors	Rate (%)
GPT-OSS (120B)	4000	44	0.88
Gemma-4-E4B (Ollama)	4000	119	2.38
DeepSeek-Coder (6.7B)	4000	129	2.58
Qwen2.5-Coder (7B)	4000	150	3.00
Llama 3.2 (3B)	4000	495	9.90
Llama 3.3 (70B)	4000	595	11.90

The results indicate that smaller code-oriented models generated fewer schema-related failures than some general-purpose models in this evaluation setting. Gemma-4-E4B (Ollama) and DeepSeek-Coder (6.7B) achieved the lowest error rates among locally executed models.

Schema hallucination — where models reference column or table names absent from the actual schema — is a known failure mode in structured generation tasks, as documented in hallucination evaluation benchmarks for LLMs [Li et al. 2023]. To complement the aggregate results, we examined representative Text-to-SQL outputs. Table 6 shows an example from Qwen2.5-Coder (7B) for the same WTQ question.

The example shows that the model produces executable SQL but returns an incorrect answer across all strategies. The error is related to the representation of the population values: applying an explicit numeric conversion to the values produces the correct result of 4. This illustrates that valid SQL does not necessarily imply correct semantic interpretation.

Table 6. Representative Qwen2.5-Coder (7B) outputs for NT-1010.

Strategy	Generated SQL	Pred.
Content-aware	`COUNT(*) WHERE population > 2000`	3
Embedding-content-aware	`COUNT(*) WHERE population > 2000`	3
Schema-linking	`COUNT(*) WHERE population > 2000`	3
Zero-shot	`COUNT(township) WHERE population > 2000`	3
Gold answer:		**4**

5.5. Comparison with Large-Scale Reference Models

[de Souza et al. 2024] report 72% denotation accuracy using GPT-4 with SQL-based generation on WTQ. In comparison, Gemma-4-E4B (Ollama) and Qwen2.5-Coder (7B) achieved 47.9% and 42.0%, respectively, under the Embedding-content-aware strategy. These results are not directly comparable due to differences in model scale, evaluation subsets, and prompting protocols, but provide a reference point for the performance gap between quantized SLMs and large-scale models.

6. Limitations and Future Work

The evaluation has two main limitations. First, WTQ was evaluated on a single random subset of 1,000 examples, so sampling variance and statistical significance were not assessed. Second, quantization was treated as a deployment condition rather than an independent experimental factor; therefore, the study does not isolate the degradation caused by quantization relative to full-precision models.

Future work will address these limitations by evaluating different quantization levels and measuring their effect on Text-to-SQL accuracy and error types. A second priority is to evaluate a complete end-to-end TableRAG pipeline that jointly performs table retrieval, context construction, and Text-to-SQL generation.

7. Conclusion

This work investigated quantized Small Language Models in a TableRAG-inspired pipeline for Table Question Answering under constrained computational environments. We evaluated Text-to-SQL generation, context construction strategies, and lightweight table retrieval.

Regarding RQ1, the results show that quantized SLMs can perform TableQA when provided with appropriate context. Gemma-4-E4B (Ollama) and Qwen2.5-Coder (7B) achieved strong results among the evaluated local models, indicating that smaller models can benefit substantially from query-relevant context.

For RQ2, semantic table content retrieval was more effective than providing additional schema information. The Embedding-content-aware strategy consistently improved Text-to-SQL accuracy, showing the importance of query-relevant table content for SQL generation.

Regarding RQ3, combining dense retrieval with cross-encoder re-ranking improved HITS@1 from 64.2% to 70.7% and MRR from 0.73 to 0.78 on OTT-QA. Despite the additional latency, retrieval remained below 100 ms per query, supporting its use in interactive local applications.

The error analysis further showed that semantic errors remain a major source of failure, particularly for smaller models. This indicates that the main challenge is not only generating executable SQL, but correctly mapping natural language intent to database operations.

Overall, the results demonstrate that quantized SLMs can be viable for local TableQA when combined with effective context construction and retrieval strategies. Future work will focus on quantization degradation analysis and on evaluating a complete end-to-end TableRAG pipeline that jointly integrates retrieval, context construction, and SQL generation.

8. Artifact Availability

The source code, experiment configurations, and preprocessing scripts developed in this study will be publicly available at: `https://github.com/dapaz98/tablerag-slm` The repository includes instructions for reproducing the experiments and generating the reported results.

References

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2022). On the opportunities and risks of foundation models. *ACM Computing Surveys*, 55(12):1–45.

Chen, W., Chang, M.-W., Schlinger, E., Wang, W., and Cohen, W. W. (2021). Open question answering over tables and text. In *International Conference on Learning Representations (ICLR)*.

de Souza, E. A., da Silva, P. F., Gomes, D., Batista, V., Batista, E., and Pacheco, M. (2024). TableRAG: A novel approach for augmenting LLMs with information from retrieved tables. In *Anais do Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana (STIL)*, pages 182–191, Belém, PA, Brazil. Sociedade Brasileira de Computação.

Dettmers, T. et al. (2023). Qlora: Efficient finetuning of quantized llms. *NeurIPS*.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2023). GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*.

Freitas Filho, J. G. M. d. and Fernandes, S. R. (2025). Do slm agents screen papers as well as llms? In *Anais do Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*.

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., and Zhou, J. (2023). Text-to-SQL empowered by large language models: A benchmark evaluation. arXiv preprint arXiv:2308.15363.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In Webber, B., Cohn, T., and He, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Lewis, V., Riedel, S., Rocktäschel, T., Yogatama, D., Welbl, D., Chang, K.-W., Goyal, N., Macdonald, R., Lewis, K., Leary, R., Fan, A., Joulin, A., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474. Curran Associates, Inc.

Li, J., Cheng, X., Zhao, W. X., Nie, J.-Y., and Wen, J.-R. (2023). HaluEval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464, Singapore. Association for Computational Linguistics.

Pasupat, P. and Liang, P. (2015). Compositional semantic parsing on semi-structured tables. In *ACL*.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shi, T., Zhao, C., Boyd-Graber, J., Daumé III, H., and Lee, L. (2020). On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Sousa, E., Santos, E., Sampaio, A., and Bezerra, C. (2025). Comparing structural quality of code generated by llms: A static analysis of code smells. In *Anais do Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Xiao, S., Liu, Z., Zhang, P., and Muennighoff, N. (2023). C-pack: Packaged resources to advance general Chinese embedding. arXiv preprint arXiv:2309.07597.

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

Yu, X., Jian, P., and Chen, C. (2025). TableRAG: A Retrieval Augmented Generation Framework for Heterogeneous Document Reasoning. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10023–10037, Miami, Florida, USA. Association for Computational Linguistics.