Semantic Routing for Hybrid Question Answering: Balancing Deterministic Text-to-SQL and LLM-Based Reasoning

Dylan Robson
Laboratório de Computação de Alto Desempenho
Programa de Pós-Graduação em Informática
Universidade Federal do Espírito Santo
Vitória, Espírito Santo, Brazil
dylan.robson@edu.ufes.br

Aerty Santos
Laboratório de Computação de Alto Desempenho
Programa de Pós-Graduação em Informática
Universidade Federal do Espírito Santo
Vitória, Espírito Santo, Brazil
aerty.santos@edu.ufes.br

Isadora Gotardo
Laboratório de Computação de Alto Desempenho
Programa de Pós-Graduação em Informática
Universidade Federal do Espírito Santo
Vitória, Espírito Santo, Brazil
isadora.gotardo@edu.ufes.br

Elias de Oliveira
Laboratório de Computação de Alto Desempenho
Programa de Pós-Graduação em Informática
Universidade Federal do Espírito Santo
Vitória, Espírito Santo, Brazil
elias@lcad.inf.ufes.br

Abstract

This study investigates a hybrid Question Answering (QA) architecture that synergizes the determinism of structured database queries with the semantic flexibility of Large Language Models (LLMs). We address the limitations of monolithic approaches by introducing a lightweight *Semantic Router* that dynamically dispatches natural language questions to the most appropriate processing pipeline: a deterministic Text-to-SQL module for analytical queries or a Direct-LLM module for open-ended reasoning. Our design minimizes entity misinterpretation and improves robustness by combining schema grounding with a dual-path execution flow. Experimental results on the Spider 1.0 benchmark demonstrate that our pipeline with fully open-source LLMs achieves **79.2%** Execution Accuracy ensuring full reproducibility and significantly reducing inference costs. Furthermore, the routing mechanism attains **99.8%** classification accuracy, proving that a non-parametric scoring system can effectively decouple analytical constraints from conversational flexibility without introducing latency overhead.

Keywords

Large Language Models, Text-to-SQL, Semantic Routing, Hybrid Architecture, Database Question Answering.

1 Introduction

Structured Query Language (SQL) serves as the standard *language* for Relational Database Management Systems (RDBMS), providing a rigorous framework for data definition, manipulation, and control. Within this paradigm, a *SQL query* acts as a formal, declarative statement that specifies desired data retrieval or transformation operations based on relational algebra, strictly adhering to a defined schema. *SQL execution* subsequently refers to the process by which a database engine parses, optimizes, and runs this query against a specific database instance to yield a precise, deterministic result set.

The primary objective of *Text-to-SQL* is to translate natural language questions into these executable SQL queries, thereby treating the database as the authoritative source of truth. Unlike free-form text generation, this approach aims to deliver structured, auditable results [4].

While Large Language Models (LLMs) are powerful, they are fundamentally *probabilistic*: they generate tokens by sampling from a learned conditional distribution. This introduces variability across runs, sensitivity to prompt phrasing, and the risk of producing fluent but unverifiable statements. These issues are magnified when handling the ambiguous or compositional queries common in database QA [2, 7]. Text generated without execution lacks determinism and affords no intrinsic mechanism for validating correctness.

In contrast, traditional SQL-based systems relying on rules, templates, or lexical matching demand detailed schema knowledge from the user or extensive handcrafted patterns. These pipelines are often brittle regarding paraphrasing and schema variations, trading recall for precision while struggling with joins and nested operations across unseen schemas [12, 16]. Consequently, limited coverage and high maintenance overhead emerge as the principal bottlenecks.

SQL execution is *deterministic* with respect to a fixed database instance D: given a well-formed query Q, the result $\text{exec}Q,D$ is uniquely defined and reproducible, enabling auditing and precise error analysis [10]. For questions admitting factual, single-truth answers, a hybrid strategy is therefore preferable. By utilizing an LLM for semantic interpretation and controlled translation into SQL, while delegating the final computation to the database engine, we align robustness (via execution) with flexibility (via language understanding) [3, 11].

We hypothesize that a lightweight semantic routing mechanism based on lexical schema grounding and named entity cues can effectively separate analytical Text-to-SQL queries from open-ended questions. Under this assumption, a hybrid QA architecture can maintain high execution accuracy for structured queries while avoiding unnecessary invocation of expensive Text-to-SQL pipelines for non-analytical inputs.

***XVII Computer on the Beach**, Florianópolis, SC, Brazil*
2026.

Early Text-to-SQL systems, such as Seq2SQL[1] and RAT-SQL[2], introduced task-specific modeling and schema-aware encoders. However, they required substantial feature engineering and generalized poorly to new schemas. Recent LLM-assisted pipelines have added structure-aware prompting and guardrails, such as constrained (PICARD) and execution-guided decoding, to improve validity while maintaining control over generation [3, 9, 11, 13]. Our work adopts this philosophy while emphasizing deterministic execution as the ultimate goal.

To mitigate entity ambiguity prior to generation, we incorporate a lightweight Named Entity Recognition (NER) step. This process tags and normalizes mentions (e.g., person name (PER), date, place (PLC)) into typed fields, providing structured signals that support schema grounding and improve intent capture in downstream Text-to-SQL tasks [1, 17].

We evaluate our approach using *execution-match* (EX) on the Spider 1.0 benchmark, a standard dataset for closed-form questions [16]. We compare our method against two primary baselines: LLM-only question answering without SQL execution, and template-based SQL systems driven by lexical matching. "On this benchmark, our hybrid approach achieves competitive Execution Accuracy (79%) relative to the current state-of-the-art proprietary models [3], while strictly utilizing open-source components. Crucially, our architecture distinguishes itself through operational efficiency: by correctly bypassing the expensive SQL generation pipeline for non-analytical queries, we significantly reduce end-to-end latency and computational overhead."

In summary, this paper formulates a principled hybrid approach for deterministic QA and presents a schema-grounded Text-to-SQL pipeline designed for reproducibility and auditability. Furthermore, we provide a comprehensive evaluation protocol contrasting hybrid, LLM-only, and SQL-only regimes, offering an analysis of both robustness and computational cost.

2 Literature Review

To address the scalability and adaptability limitations of previous Large Language Model (LLM)-based NL2SQL models, Wang et al. [14] introduce DBCopilot, a framework designed for natural language querying over large-scale databases. Their architecture decouples the NL2SQL pipeline into a schema routing phase followed by SQL generation. During the routing stage, a lightweight copilot model leverages a differentiable search index and graph-based schema representations to isolate the relevant database tables for a given query. Once the schema is routed, an LLM processes the filtered context to generate the executable SQL statement. Additionally, to obviate the need for extensive manual annotation, the framework incorporates a reverse schema-to-question generation strategy that automatically synthesizes training data, enabling the router to learn semantic mappings across massive database instances.

Empirical evaluations on standard benchmarks—including Spider, BIRD, and Fiben—demonstrate that DBCopilot significantly outperforms retrieval-based and LLM-enhanced baselines. The method achieves up to 19.88% higher recall in schema routing and improves SQL execution accuracy by a margin of 4.43% to 11.22%. These findings suggest that decoupling the routing and generation processes is essential for scaling LLM-based natural language querying systems to heterogeneous, real-world environments.

Gao et al. [3] conduct a systematic benchmark of prompt engineering techniques for LLM-based Text-to-SQL tasks, addressing the lack of comprehensive comparisons in prior research. They dissect the prompt design process into three governing dimensions: question representation (e.g., text versus code-based schema), example selection strategies (based on question or query similarity), and organization formats ranging from full-information to token-efficient structures. Additionally, the study assesses the viability of open-source LLMs, investigating performance differences between raw and supervised fine-tuned (SFT) models while treating token efficiency as a critical constraint for practical deployment.

Synthesizing these insights, the authors propose **DAIL-SQL**, an integrated solution that achieved state-of-the-art performance on the Spider leaderboard with 86.6% execution accuracy. Their methodology combines a selection strategy based on both question and query similarity with an organization format designed to preserve essential question-to-SQL mappings while stripping away token-heavy schema details. A crucial finding regarding open-source models reveals a trade-off: while SFT substantially boosts zero-shot capabilities, it detrimentally impacts in-context learning retention.

Complementary research highlights the value of augmenting keyword-based information retrieval with semantic signals derived from Named Entity Recognition (NER). Izo et al. [5] examine this integration by automatically transforming annotations from the HAREM[3] tagset—specifically Person, Organization, Place, Time, and Value—into first-class index fields. Their central premise is that treating entities as indexable metadata allows search engines to capture user intent more effectively than token matching alone. This is particularly vital under ambiguity, where entity-aware queries (e.g., targeting a specific `PLC` or `PER`) can resolve overlaps that confound standard keyword searches.

Evaluated using Precision@k metrics over query pairs contrasting keyword versus semantic forms, the entity-aware engine maintained perfect precision (100%) across all trials. In contrast, the keyword-only baseline degraded significantly under ambiguity, failing to distinguish between proper names and locations (e.g., “Paulo” vs. “São Paulo”) or between named entities and common nouns. These outcomes substantiate entity-centric indexing as a robust strategy for mitigating polysemy and homonymy in heterogeneous corpora, motivating its integration into pipelines requiring precise semantic disambiguation.

3 Proposed Methodology

We propose a hybrid architecture designed to balance the determinism of structured database queries with the semantic flexibility of LLMs. Unlike monolithic approaches that force all queries into a single reasoning modality, our system employs a lightweight semantic router to dispatch user questions to the most appropriate processing pipeline: a *Text-to-SQL* module (powered by Qwen3) for analytical and schema-grounded queries, or a *Direct-LLM* module (Qwen 2.5) for open-ended reasoning. In doing so, the architecture synergizes

[1] Seq2SQL treats Natural Language to SQL as a seq2seq problem and uses reinforcement learning from execution rewards to refine outputs [18].
[2] RAT-SQL uses relation-aware self-attention to jointly encode question and schema, improving schema linking [12].
[3] https://www.linguateca.pt/HAREM/

the robustness and deterministic semantics of database execution with the interpretative capacity of modern LLMs, all underpinned by efficient lightweight pre-processing layers based on NER and schema-grounding heuristics. Figure 1 illustrates the end-to-end data flow.

The architecture positions the User Question and Database Schema as parallel inputs, though their functional roles differ. The Schema serves as a *conditional resource* rather than a strict dependency: it is actively utilized by the Semantic Router to calculate lexical overlap and establish schema grounding. However, for queries classified as open-ended or conversational (the Direct path), the system bypasses this dependency entirely. In such cases, the Schema is effectively disregarded, allowing the LLM to generate responses based solely on its parametric knowledge without requiring database access.

3.1 The Hybrid Routing Mechanism

The core component of our architecture is the semantic router, which classifies an incoming natural language question Q into a binary route $r \in \{\text{SQL}, \text{DIRECT}\}$. To ensure low latency, the router eschews expensive LLM calls during the decision phase, relying instead on a feature-based scoring system derived from NER and lexical schema linking.

As implemented in our pipeline, the routing decision function fQ, S depends on both the question Q and the target database schema S. We employ the spaCy library [8] to extract lightweight named-entity and numeric cues. This choice is intentional: our focus is real-time, low-latency routing suitable for interactive search scenarios. Although heavier transformer-based NER models may offer higher recall, their latency overhead conflicts with the router's design constraints.

The decision process follows three stages: (1) **Entity Extraction and Normalization:** We detect named entities (e.g., `DATE`, `GPE`, `ORG`) and numeric patterns using a pre-trained lightweight model (`en_core_web_sm`). This step grounds the question by identifying potential filter values before SQL generation. (2) **Schema Linking via Lexical Overlap:** Extracted tokens are cross-referenced against schema elements, such as table and column names. We also define a set of domain-specific synonyms (e.g., mapping "song" to "track" or "nation" to "country") to enhance recall. (3) **Heuristic Scoring:** A composite score $Score_{sql}$ determines the final route. Points are awarded for the presence of SQL-specific cues (e.g., "how many", "average", "list") and successful schema linking events. Conversely, open-ended triggers (e.g., "explain", "why") incur a penalty, favoring the Direct-LLM route.

During development, we observed that generic interrogatives such as "What is..." tended to dominate the scoring logic, frequently biasing the decision toward the DIRECT path even when structured computation was required. A representative failure involved the query *"What is the total number of singers?"*, which was initially misrouted because the generic cue "What" overwhelmed the SQL-oriented evidence provided by entity matches.

To mitigate this, we incorporate a *schema-sensitive decay function* that modulates the strength of open-ended cues based on schema grounding. Let *num_hits* denote the count of distinct schema tokens (tables or columns) found in Q. For any open-ended cue w that typically favors the DIRECT route, the effective penalty weight becomes:

$$weight_{\text{DIRECT}} = \frac{base\_w}{2^{num\_hits}}$$

This formulation ensures that generic cues exert high influence only when the question exhibits little to no schema grounding. As soon as meaningful schema overlap is detected, the weight of such cues decays rapidly, allowing SQL-oriented evidence—such as aggregations, numerical terms, and relational markers—to dominate the decision.

Although the router defaults to spaCy for entity extraction, it degrades gracefully when the model cannot be loaded or executed within latency budgets. In such instances, a regex-based fallback mechanism detects coarse numeric and temporal expressions (e.g., four-digit years, cardinal numbers) and feeds them into the same linking pipeline. This preserves routing functionality under constrained deployments while sacrificing some recall and granularity.

Beyond the binary classification r, the router emits auxiliary *hints* to enhance prompting and auditing for downstream components. Specifically, it returns a surface-marked version of the question alongside a shortlist of table candidates (`tables_hint`). Furthermore, it identifies both lexically matched columns (`cols_lex`) and columns inferred from coarse type cues (`cols_by_type`, such as numerical or temporal fields). These signals serve to constrain retrieval or structure the TEXT-TO-SQL prompt, providing clear observability for debugging and error analysis.

To reduce false negatives during linking, both question and schema tokens are normalized: lowercased, accent-stripped, and augmented with the synonym map. This normalization improves robustness across heterogeneous database naming conventions while keeping the router free from expensive embedding models.

Let $Score_{\text{sql}}Q, S$ denote the final composite score derived from the weighted cues described above. The routing rule is a single-threshold classifier:

$$rQ, S = \begin{cases} \text{SQL}, & \text{if } Score_{\text{sql}}Q, S \geq \theta, \\ \text{DIRECT}, & \text{otherwise.} \end{cases}$$

In our implementation, we adopt a small, interpretable scoring scheme with a default threshold of $\theta = 1$. The score components and threshold can be tuned on a held-out dataset to reflect domain-specific preferences—such as favoring higher precision over coverage for SQL routing—while preserving the latency and debuggability benefits of a non-parametric decision rule.

Formally, if $Score_{\text{sql}}Q, S \geq \theta$, the query is routed to the TEXT-TO-SQL pipeline (Qwen3). Otherwise, it is routed to the Direct-LLM module (Qwen 2.5).

3.2 Schema-Grounded TEXT-TO-SQL Generation

Our TEXT-TO-SQL module adopts the architectural principles of the DAIL-SQL framework [3], prioritizing structured question representation, schema-grounded prompting, and example organization that preserves the mapping between natural language questions and SQL skeletons. However, we replace the proprietary GPT-4 model—used as the core generator in the original work—with the fully open-source Qwen3 family. This substitution serves a dual purpose: it eliminates dependencies on closed commercial APIs, which pose

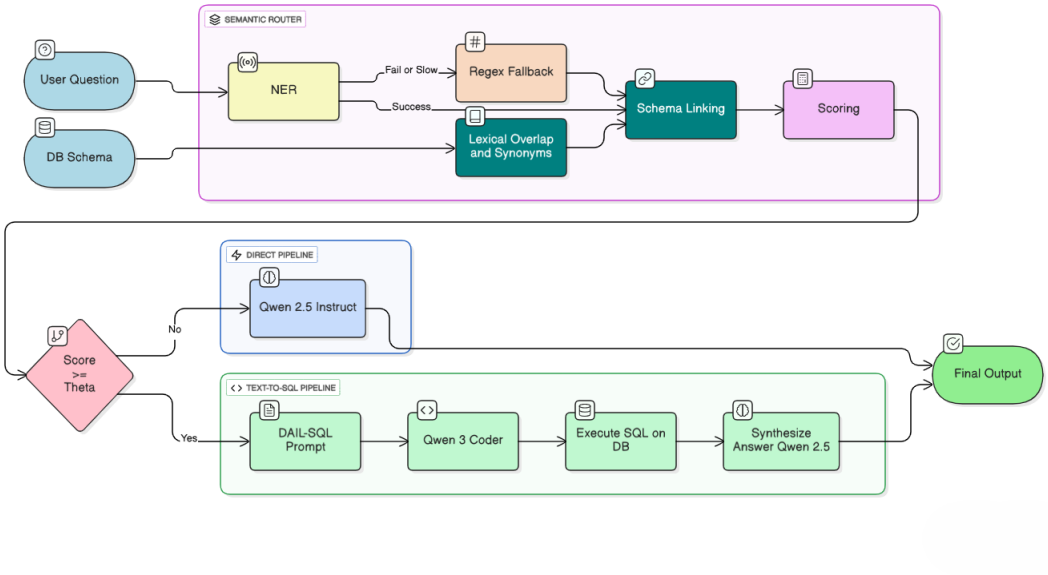

Figure 1: Architectural Overview of the Hybrid System. The User Question (Q) is the primary driver, while the Database Schema (S) acts as a conditional context. The Semantic Router evaluates Q against S to determine intent; if the question is schema-grounded, it follows the TEXT-TO-SQL path (green). Conversely, if the schema is irrelevant or absent, the query follows the Direct LLM path (blue), treating the schema as optional.

risks regarding accessibility and cost, while simultaneously ensuring that the entire pipeline remains reproducible, auditable, and deployable in local environments without external service calls.

Conceptually, our implementation mirrors the methodology of [3]. The database schema is encoded through SQL DDL snippets (Code Representation Prompt), and relevant cross-domain examples are selected via masked question similarity combined with SQL skeleton matching. The final prompt—constructed using DAIL-style question–query pairs—drives Qwen3 to generate a semantically grounded SQL query.

Upon generation, the SQL statement is executed against a target database to yield deterministic results. These outputs are subsequently forwarded to the response generation stage for natural language rendering. This decoupling of SQL generation from the final response preserves the interpretability emphasized in [3] while upholding a fully open and reproducible TEXT-TO-SQL workflow.

3.3 Direct LLM Interaction

For queries classified as open-ended, conversational, or lacking sufficient schema grounding, the system bypasses the SQL generation pipeline and routes the input directly to a large language model. This trajectory empowers the system to address qualitative, evaluative, or explanatory inquiries that relational operators cannot resolve—addressing an inherent limitation of purely SQL-driven baselines [15]. Typical examples include requests for summarization, comparative analysis, recommendations, or reasoning-based insights where no direct relational mapping or execution result exists.

In our implementation, this component is handled by the open-source `Qwen/Qwen2.5-7B-Instruct` model. We selected this specific architecture because it is optimized for instruction following and free-form generation—making it well-suited for open-enxded queries—while simultaneously remaining fully reproducible and deployable independent of proprietary APIs. Upon invocation, the Direct-LLM module receives a contextually enriched prompt containing router-generated metadata, such as detected entities, lexical hints, and coarse semantic tags. This allows the model to leverage partial structural signals without necessitating full schema grounding. Consequently, it generates a natural-language response directly, ensuring robustness in scenarios where structured querying is impossible or inappropriate.

4 Experiments and Results

In this section, we evaluate the efficacy of our hybrid pipeline relative to standard baselines. To contextualize our findings, we adopt the benchmarks established by [3] as the current state-of-the-art, with the primary objective of surpassing their performance on the shared evaluation dataset.

Our analysis assesses system performance across three principal dimensions. EX serves as the primary accuracy metric, calculating the percentage of queries where the execution result aligns perfectly with the ground truth. To evaluate architectural efficiency and correctness, we measure *Routing Precision*—defined as the ratio of questions correctly dispatched to their respective pipelines—and *Latency*, which tracks the total end-to-end processing time.

4.1 Experimental Setup

We evaluated our approach on the Spider 1.0 benchmark [16], a large-scale cross-domain Text-to-SQL dataset comprising complex multi-table queries across 200 databases. This benchmark is particularly well-suited for assessing generalization, schema grounding, and robustness to domain shifts—properties that are central to our hybrid architecture and routing strategy. All experiments were conducted using the router described in Section 3.1, utilizing the Qwen3-based Text-to-SQL generator alongside the Qwen2.5 model for answer rendering and direct LLM interaction.

To ensure reproducibility and eliminate dependencies on proprietary APIs, all models were executed locally using the open-source `Qwen/Qwen3-Coder-30B-A3B-Instruct` and `Qwen/Qwen2.5-7B-Instruct` checkpoints. We adhered strictly to the original Spider evaluation protocol, EX over the development set without utilizing database-specific demonstrations. This experimental design enables us to isolate the distinct contributions of the semantic router, the DAIL-style schema-grounded prompting adapted for Qwen3, and the dual-path architecture that combines SQL execution with direct LLM reasoning.

4.2 Execution Accuracy Results

For comparison purposes, Table 1 presents the results of the EX experiment on the development set of the Spider 1.0 benchmark. This test evaluates the model's ability to generate SQL queries whose execution on the database returns exactly the same result expected by the ground truth. Unlike structural metrics such as EM, EX measures the functional accuracy of the system by verifying whether the final answer produced by the generated query is correct. In this way, Table 1 directly and rigorously quantifies the performance of the different Text-to-SQL pipelines evaluated, enabling a comparison between the effectiveness of our hybrid method and the reference configurations used by [3].

Table 1: Comparison of EX on the Spider Development Set (%).

Strategy	Score
DAIL-SQL + Qwen3-Coder-30B + Self-Consistency	79.2%
DAIL-SQL + GPT-4 + Self-Consistency	86.6%

While GPT-4 retains advantages in semantic generalization—particularly in complex compositional queries—the Qwen3 adaptation demonstrates that the
DAIL-SQL methodology [3] transfers effectively to open-source LLMs. A critical factor in this comparison is computational viability: considering that GPT-3 already contained 175 billion parameters, GPT-4 is estimated to be significantly larger, implying a prohibitive inference cost if deployed on comparable local hardware. Moreover, the proprietary nature of such models means specific versions may become unavailable or deprecated, compromising long-term reproducibility. Consequently, when weighing these resource constraints against benefits in controllability, cost, and accessibility, this trade-off remains acceptable for research and production settings that prioritize transparency over maximum absolute accuracy.

To better understand the remaining performance gap, we manually inspected failure cases. Several recurring patterns emerged. A substantial portion of errors were *schema-mismatch errors*, where the model referenced nonexistent columns (e.g., `no such column: T2.Year`, `T1.MakeId`, or `T1.ship_id`). These cases often arise from subtle variations in column naming conventions or incorrect alignment between question entities and schema tables. Another prominent category involved *syntax errors*, including malformed joins or misplaced table aliases (e.g., `near "T2": syntax error`). We also observed a small number of *gold query errors* present in the dataset itself, such as type comparison failures (string vs. integer) or encoding issues in text fields (e.g., invalid UTF-8 in `last_name` values from the WTA_1 database). These gold errors do not reflect model limitations but instead highlight edge cases in the benchmark's reference SQL.

Overall, the error distribution indicates that most failures stem from structural mismatches in SQL generation rather than fundamental reasoning errors by the LLM. This suggests that integrating additional schema-normalization steps or targeted post-processing—such as alias validation or table-level sanity checks—may further reduce EX loss without requiring heavier models or additional training.

4.3 Latency and Efficiency Analysis

Operational efficiency is a central design principle of our hybrid architecture. To quantify the performance benefits of our routing strategy, we measured the inference latency and token consumption across the pipeline's distinct stages. These experiments were conducted on a single NVIDIA A100 (80 GB) GPU, 2 AMD EPYC 7513 32-Core Processor and 512Gb of RAM to reflect performance under high-end hardware constraints.

Table 2 presents the average processing time and token usage per query. The data reveals a significant computational disparity between the stages. The *Text-to-SQL* generation is the dominant source of latency, requiring approximately 139 seconds per query. This intensive computational cost stems from the DAIL-SQL prompting strategy, which necessitates long context windows and multi-sample self-consistency decoding ($n > 1$) to ensure accuracy.

Table 2: Average Latency and Token Consumption per Query.

Pipeline Stage	Time (s)	Token In	Token Out
Router Decision (CPU)	0.004	14.410	0.00
SQL Gen. (Qwen3-30B)	138.871	759.400	29.467
LLM Answer (Qwen2.5-7B)	3.555	195.475	130.370

In sharp contrast, the semantic router operates with negligible overhead, executing in under 5 milliseconds on CPU with minimal token consumption. Similarly, the Direct-LLM answer generation (Qwen2.5) remains highly efficient, averaging just 3.56 seconds.

These findings validate the architectural necessity of the routing mechanism. By diverting non-analytical, conversational, or qualitative questions away from the resource-heavy Text-to-SQL pipeline, the system avoids the 139-second bottleneck for a large subset of user queries. Consequently, the router acts as a critical optimization

layer, making the system viable for interactive, high-throughput deployments without sacrificing the depth required for complex SQL analysis.

4.4 Routing Performance

4.4.1 Router-Only Evaluation. Following the integration of schema-sensitive exponential decay into the scoring function (Section 3.1), the router achieves 100% accuracy on the initial 200-question calibration set (comprising 100 SPIDER development questions and 100 DIRECT queries). This refinement successfully resolved all previously identified misclassifications, particularly the ambiguity inherent in generic interrogatives such as "*What is...*".

To rigorously assess robustness and generalization at scale, we constructed an extended evaluation corpus totaling 2,034 questions. This dataset combines the complete Spider 1.0 development set (1,034 questions) with 1,000 questions sampled from *TriviaQA-unfiltered* [6]. TriviaQA-unfiltered is an open-domain QA dataset containing factoid-style inquiries paired with evidence documents. We selected this dataset as an adversarial stress test because many of its questions contain linguistic features that superficially mimic analytical SQL intent (e.g., numerical comparisons, aggregations, temporal references). However, since these questions lack a corresponding database schema, they must intrinsically be routed to the DIRECT pipeline. This setup effectively evaluates the router's ability to distinguish between surface-level lexical similarity and genuine, schema-grounded analytical intent.

Table 3 details the routing accuracy across this combined corpus.

Table 3: Router accuracy on the extended evaluation corpus (2,034 questions).

Evaluation Subset	Size	Routing Accuracy
SPIDER 1.0 (dev)	1,034	**100.0%**
TriviaQA-unfiltered	1,000	**99.6%**
Combined Set	**2,034**	**99.8%**

The router attains an aggregate accuracy of **99.8%**, misclassifying only four instances out of 2,034. A representative error from the TriviaQA subset is the query:

> *"Out of 11 series of prime time seasons, how many times did Happy Days make the Nielsen Top Twenty?"*

Despite lacking an associated schema, the question's explicit numerical framing ("11 series", "how many times") and ranking structure ("Top Twenty") activated strong SQL-oriented heuristic cues, creating a lexical decoy that tipped the decision boundary toward the SQL route. Such edge cases illustrate the challenge of disambiguating complex open-domain factoid questions from analytical database queries when they share significant syntactic overlap.

Nevertheless, these results confirm that the routing mechanism maintains high reliability across both structured (Spider) and open-domain (TriviaQA) distributions. Crucially, it achieves this near-perfect discrimination while preserving the architectural benefits of interpretability, negligible computational overhead, and independence from learned parametric models.

4.4.2 Downstream Pipeline Evaluation. Table 4 delineates the downstream accuracy achieved for both categories following router classification.

Table 4: Breakdown of answer accuracy by category following router classification.

Question Category	Accuracy
DIRECT	**92%**
TEXT-TO-SQL	78%

The system demonstrates superior reliability on DIRECT questions compared to TEXT-TO-SQL queries, achieving accuracies of 92% and 86%, respectively. This performance disparity is anticipated: direct queries leverage the robust instruction-following capabilities of Qwen2.5, whereas Text-to-SQL queries traverse a multi-stage pipeline where errors may propagate across semantic parsing, SQL generation, database execution, and answer rendering.

A granular inspection of the 30 unanswered questions (22 TEXT-TO-SQL, 8 DIRECT) reveals distinct failure modes. Among the 22 TEXT-TO-SQL errors, a significant subset involves complex aggregations and filters—specifically, the correct identification of schema elements (e.g., `edispl`) and the application of relational constraints. Additional failures include challenges with entity linking, negation-based logic, metadata-style queries, and multi-condition reasoning. These patterns confirm that the dominant bottleneck lies not in routing, but in the inherent semantic and relational complexity required for accurate SQL generation.

Regarding the 100 items processed via the DIRECT pipeline, no errors were attributable to misrouting. Consequently, the observed deficiencies—representing 8% of the Direct subset (or 4% of the total evaluation set)—reflect genuine limitations in the model's domain-specific granularity. These failures were characterized by subtle technical hallucinations, such as the inversion of false-negative logic in probabilistic structures, or the propagation of generalized engineering myths (e.g., regarding the irreversibility of DDL operations). This indicates that while the model achieves high semantic alignment, it occasionally sacrifices rigorous technical precision for plausible-sounding approximations.

In summary, while the router now operates with near-perfect accuracy even in large-scale heterogeneous settings, the overall system performance remains bounded by the limitations of the Text-to-SQL component. Future optimization efforts should therefore prioritize enhanced schema reasoning, aggregation handling, and robustness to multi-condition queries within the SQL generation stage.

4.5 Future Improvements

Although the current heuristic-based router proves lightweight and effective, future research will investigate replacing the rule-based logic with a distilled BERT-based classifier. This evolution aims to enhance sensitivity to complex intent shifts while maintaining a minimal latency footprint.

Furthermore, we aim to integrate an iterative refinement framework. This mechanism would empower the system to dynamically re-route queries—switching from the Direct to the Text-to-SQL

pipeline if initial reasoning is found to lack grounding—and to leverage execution feedback loops, enabling the model to self-correct SQL queries in response to database errors.

5 Conclusion

In this work, we introduced a novel strategy that combines NLP, NER, and LLM techniques to generate SQL queries for databases, thereby narrowing the search results for Q&A resolutions. Our approach demonstrated a 79.2% performance, utilizing fully open-sources LLMs for better reproducibility.

In our approach, a heuristic-based router filters out questions that cannot reasonably be converted into SQL procedures. We argue that this strategy accounts for approximately 99.8% of the overall performance gain. Besides, although the current heuristic-based router is lightweight and effective, future work will explore additional strategies to better handle more complex cases. Potential directions include: (a) enhancing the heuristic to incorporate additional named entities and (b) training a small BERT-based classifier to replace the rule-based logic. These approaches aim to further improve sensitivity to complex intent shifts without significantly increasing latency.

The results presented in this work demonstrate an alternative to relying solely on an LLM to answer general-context questions, showing that some queries are better handled through complementary strategies.

Acknowledgments

This study was financed by FAPES. Dylan Faria Robson is supported by FAPES (238/2025). Isadora Potiguara Gotardo is supported by FAPES (252/2025).

References

[1] Krisztian Balog. 2018. *Entity-Oriented Search.* Springer. doi:10.1007/978-3-319-93935-3

[2] Adithya Bhaskar, Tushar Tomar, Ashutosh Sathe, and Sunita Sarawagi. 2023. Benchmarking and Improving Text-to-SQL Generation under Ambiguity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 7053–7074. doi:10.18653/V1/2023.EMNLP-MAIN.436

[3] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proc. VLDB Endow.* 17, 5 (2024), 1132–1145. doi:10.14778/3641204.3641221

[4] Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2024. Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL. *CoRR* abs/2406.08426 (2024). arXiv:2406.08426 doi:10.48550/ARXIV.2406.08426

[5] Flávio Izo, Elias Oliveira, and Claudine Badue. 2021. Named Entities as a Metadata Resource for Indexing and Searching Information. In *Intelligent Systems Design and Applications - 21st International Conference on Intelligent Systems Design and Applications (ISDA 2021) Held During December 13-15, 2021 (Lecture Notes in Networks and Systems, Vol. 418)*, Ajith Abraham, Niketa Gandhi, Thomas Hanne, Tzung-Pei Hong, Tatiane Nogueira Rios, and Weiping Ding (Eds.). Springer, 838–848. doi:10.1007/978-3-030-96308-8_78

[6] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, Vancouver, Canada, 1601–1611. doi:10.18653/v1/P17-1147

[7] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2025. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net. https://openreview.net/forum?id=XmProj9cPs

[8] Ines Montani, Matthew Honnibal, Adriane Boyd, Sofie Van Landeghem, and Henning Peters. 2023. explosion/spaCy: v3.7.2: Fixes for APIs and requirements. https://doi.org/10.5281/zenodo.10009823. doi:10.5281/zenodo.10009823 Version v3.7.2, Zenodo.

[9] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/72223cc66f63ca1aa59edaec1b3670e6-Abstract-Conference.html

[10] Raghu Ramakrishnan and Johannes Gehrke. 2003. *Database management systems (3. ed.)*. McGraw-Hill.

[11] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 9895–9901. doi:10.18653/V1/2021.EMNLP-MAIN.779

[12] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 7567–7578. doi:10.18653/V1/2020.ACL-MAIN.677

[13] Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. 2018. Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100* (2018).

[14] Tianshu Wang, Xiaoyang Chen, Hongyu Lin, Xianpei Han, Le Sun, Hao Wang, and Zhenyu Zeng. 2023. DBCopilot: Natural Language Querying over Massive Databases via Schema Routing. *arXiv e-prints*, Article arXiv:2312.03463 (Dec. 2023), arXiv:2312.03463 pages. arXiv:2312.03463 [cs.CL] doi:10.48550/arXiv.2312.03463

[15] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/forum?id=1PL1NIMMrw

[16] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 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, Brussels, Belgium, October 31 - November 4, 2018*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, 3911–3921. doi:10.18653/V1/D18-1425

[17] Nikita Zhiltsov, Alexander Kotov, and Fedor Nikolaev. 2015. Fielded Sequential Dependence Model for Ad-Hoc Entity Retrieval in the Web of Data. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, Ricardo Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto (Eds.). ACM, 253–262. doi:10.1145/2766462.2767756

[18] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR* abs/1709.00103 (2017). arXiv:1709.00103 http://arxiv.org/abs/1709.00103