MARMo: An Evidence-Guided and Memory-Augmented Multi-Agent Repair Framework for Accurate Text-to-NoSQL Parsing

Anonymous ACL submission

Abstract

Text-to-NoSQL aims to extend natural language interfaces (NLIs) beyond traditional Text-to-SQL, enabling intuitive querying of NoSQL systems such as MongoDB—widely adopted in modern data-intensive applications. While recent advances in Text-to-NoSQL have yielded benchmark datasets, existing systems still suffer from erroneous query generation and a lack of robust mechanisms for error detection, explanation and iterative correction. In this paper, we propose MARMo: a novel evidence-guided, memory-augmented multi-agent repair framework for accurate Text-to-NoSQL parsing (focusing on MongoDB). The framework employs a modular, plug-and-play architecture that coordinates multiple specialized agents to perform iterative error diagnosis and adaptive refinement. Extensive experiments on the TEND benchmark validate that MARMo, as a modular and interpretable framework, achieves overall improvement on repair accuracy and generalization for Text-to-NoSQL. Our work not only enhances the reliability of natural language-driven NoSQL query systems but also paves the way for future research into robust, human-in-the-loop NLIs for unstructured and semi-structured data management.

1 Introduction

Natural language interfaces, by lowering the barrier to data access for non-technical users, have catalyzed significant research interest in NLP and databases, particularly in the Text-to-SQL task of translating plain-language queries into executable statements over structured relational data (Qin et al., 2022; Iacob et al., 2020; Katsogiannis-Meimarakis et al., 2023). However, despite the remarkable progress in Text-to-SQL modeling, existing efforts are largely confined to relational databases, which presume rigid schemas and normalized tables (Zhong et al., 2017; Yu et al., 2018). In contrast, NoSQL databases such as MongoDB

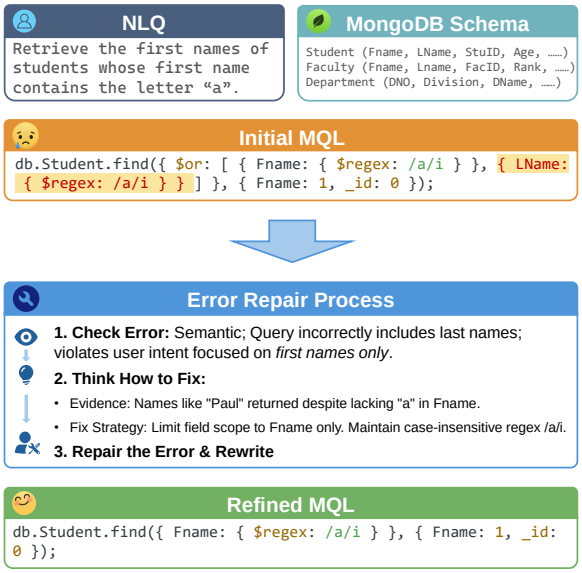

Figure 1: An illustrative example of the MQL repair task that simulates a realistic human correction process, including check errors in the initial query, formulating repair strategies based on contextual evidence, and rewriting the query to better align with the user's intent.

are increasingly adopted in modern applications for their schema-less design, hierarchical data representation, and flexibility in modeling diverse, dynamic data. Although several studies have proposed benchmarks for the Text-to-NoSQL task, the interaction gap between natural language and NoSQL systems remains significantly underexplored, as current methods predominantly adopt few-shot generation paradigms with limited diversity, hindering the broader vision of democratized access to heterogeneous data platforms (Zhang et al., 2024a,b; Lu et al., 2025; Qin et al., 2025).

Recent advances in large language models (LLMs) have demonstrated remarkable capabilities in understanding complex instructions and generating structured outputs, including executable database queries (Achiam et al., 2023; Xie et al., 2024b; Zhu et al., 2024; Chung et al., 2025). Mean-

while, multi-agent systems have emerged as a promising paradigm for decomposing and coordinating complex tasks through the collaboration of specialized agents each responsible for subtasks such as planning, reasoning, or correction. Within the context of NLIs for NoSQL databases, the synergistic combination of LLMs and multi-agent architectures offers a promising solution to the limitations of one-shot generation (Wang et al., 2025; Cen et al., 2025). This hybrid approach not only enhances interpretability and modularity, but also enables dynamic feedback integration and adaptive repair, paving the way for more robust and extensible query generation pipelines.

In this paper, we propose **MARMo**, a novel evidence-guided and **M**emory-**A**ugmented **M**ulti-**A**gent **R**epair framework for accurate Text-to-NoSQL parsing (**Mo**ngoDB), which operates in a feedback-driven refinement loop, consisting of three key agents: the Error Checker Agent, the Feedback Manager Agent and the Error Repair Agent. As shown in Figure 1, this repair workflow is inspired by the realistic human correction process when people find errors. Given a natural language query (NLQ), MARMo refines the corresponding MQL (MongoDB Querying Language) output, the Error Checker Agent first detects different errors by leveraging analysis from an MQL execution tool. In the setting of MQL application, we categorize MQL errors into three main types: the syntax error, schema error, and semantic error. The Feedback Manager Agent invokes different sub-agents based on the error type and generate the feedback based on the aggregated information. The Error Repair Agent finally leverages this aggregated feedback to retrieve and adapt similar error-repair patterns from memory to repair and regenerate the refined MQL, which is then re-executed for validation, allowing for iterative correction if necessary. In all, MARMo is designed to be pluggable, extensible, and generalizable, making it applicable across diverse scenarios for Text-to-NoSQL models and a promising step toward robust natural language interfaces for data system diversity.

In our experiments, we evaluate MARMo on the Text-to-MongoDB benchmark TEND (Lu et al., 2025), which contains diverse natural language queries paired with corresponding MQL annotations. We assess the overall performance of MARMo and compare it against a set of competitive baseline methods. Experimental results show that our framework consistently outperforms prior approaches, highlighting the benefits of multi-agent collaboration and error-aware iterative repair in the context of Text-to-NoSQL generation. The modular design of MARMo enables flexible and robust handling of diverse error types. In particular, the incorporation of database schema and query execution feedback facilitates more accurate query refinement, contributing to improved structural and semantic alignment between generated queries and gold references.

To summarize, the main contributions of this work are as follows:

- We introduce MARMo, a modular multi-agent repair framework for Text-to-NoSQL query generation. MARMo systematically integrates error diagnosis, feedback, and iterative repair, marking the first framework in this domain that empowers agents to not only generate but also correct NoSQL queries.

- We design a structured error taxonomy and a corresponding feedback mechanism that distinguish syntax, schema, and semantic errors in Text-to-NoSQL tasks. To address schema-related issues, we develop an Evidence Agent for precise error localization and informed repair, and a global repair memory tailored to the Text-to-NoSQL setting, which supports iterative corrections and robust MQL regeneration.

- Through extensive experiments on the TEND benchmark, we demonstrate the effectiveness of MARMo and provide empirical insights that point toward promising future directions for robust and accurate Text-to-NoSQL systems.

2 Related Works

Text-to-SQL has long been a core research problem at the intersection of natural language processing and databases (Hong et al., 2024). Early systems were limited to narrow domains and simple schemas, often relying on rule-based grammars or templates (Li and Jagadish, 2014; Kim et al., 2020; Gkini et al., 2021). The introduction of large-scale benchmarks, such as WikiSQL (Zhong et al., 2017), Spider and its variants (Yu et al., 2018; Gan et al., 2021a,b), and BIRD (Li et al., 2023), has driven the development of more robust and generalizable models. Pre-trained language models have further advanced the field, enabling improved contextual

reasoning, schema linking, and handling of complex SQL constructs, including in conversational settings (Cao et al., 2021; Li et al., 2021; Scholak et al., 2021; Fan et al., 2023). More recently, LLMs and multi-agent paradigms have been investigated to decompose Text-to-SQL into subtasks such as planning, reasoning, and verification, enhancing both robustness and interpretability (Pourreza and Rafiei, 2023; Zhang et al., 2023; Wang et al., 2025; Li et al., 2024a; Ren et al., 2024; Mao et al., 2024; Pourreza et al., 2024; Yun and Lee, 2025; Cen et al., 2025; Shkapenyuk et al., 2025; Dönder et al., 2025). In parallel, growing attention has been given to error analysis and repair, with new benchmarks and methods targeting error classification and correction in SQL generation (Liu and Tan, 2024; Xu et al., 2025; Liu et al., 2025; Shen et al., 2025).

NoSQL Databases play an indispensable role in modern infrastructures due to their scalability, flexibility, and support for semi-structured and heterogeneous data (Dritsas and Trigka, 2025). Unlike relational databases, NoSQL systems adopt dynamic, hierarchical representations, spanning document stores, key-value stores, and graph databases (Zhang et al., 2024b). Recent efforts have increasingly addressed Text-to-NoSQL translation, most focusing on graph-based systems such as Neo4j and Cypher (Zhou et al., 2024; Tran et al., 2024; Lin et al., 2024; Gao et al., 2024). A key milestone is TEND (Lu et al., 2025), which provides the first end-to-end pipeline for constructing Text-to-NoSQL datasets and introduces a MongoDB-specific benchmark. Its multilingual extension, MultiTEND (Qin et al., 2025), underscores the significance of cross-lingual generalization. Building on these foundations, our work expands the solution space by introducing novel architectural components and refinement strategies that improve the accuracy and robustness of Text-to-NoSQL parsing in MongoDB.

LLM-based Agents have recently become a dominant paradigm for complex, multi-step reasoning tasks, from programming to document understanding (Yang et al., 2023; Nunez et al., 2024; Ishibashi and Nishimura, 2024; Xiao et al., 2024; Li et al., 2024b, 2025; Han et al., 2025). In Text-to-SQL, agent-based approaches decompose parsing into sub-modules such as intent detection, schema linking, and query generation (Wang et al., 2025; Xie et al., 2024a; Cen et al., 2025; Deng et al., 2025), improving modularity and interpretability. Specifically, recent work in Text-to-SQL and code generation has demonstrated that many agent-based approaches combine error detection and self-repair mechanisms with tool-use strategies to achieve further performance improvements. For instance, CRITIC (Gou et al., 2024) introduces a framework that enables LLMs to iteratively validate and refine their own outputs using external tools, thereby improving reliability and correctness in tasks such as code generation. MAC-SQL (Wang et al., 2025) introduces a multi-agent, tool-assisted framework that boosts Text-to-SQL performance on large databases, achieving new SOTA results on BIRD. SQLFixAgent (Cen et al., 2025) further enhances Text-to-SQL reliability by focusing specifically on detecting and repairing semantic errors in generated SQL.

However, prior work has largely focused on query construction, devoting limited attention to systematic error detection and repair. We address this gap by introducing a multi-agent repair framework that integrates query generation with robust error analysis and iterative refinement for Text-to-NoSQL tasks. This framework not only advances Text-to-NoSQL research but also provides a foundation for systematic exploration of natural language interaction with semi-structured data.

3 Overview

In this section, we provide an overview of the problem formulation and our proposed framework. We first formalize the Text-to-NoSQL and MQL Repair tasks, highlighting the unique challenges posed by MongoDB's semi-structured data model. We then briefly introduce the overall architecture of MARMo, a modular, agent-based system designed to iteratively detect, diagnose, and repair errors in generated MQL queries.

3.1 Task Formulation

3.1.1 Text-to-NoSQL

The Text-to-NoSQL task aims to translate a natural language question into an executable NoSQL query, conditioned on a target database and its schema. Formally, let the natural language input be denoted as a token sequence $x = \{x_1, x_2, \ldots, x_n\}$, $x \in \mathcal{N}$, where $\mathcal{N}$ is NLQ sets. Let the NoSQL database be represented as $\mathcal{D} = \{C_1, C_2, \ldots, C_k\}$, where each C_i is a collection and consists of a set of fields $F_i = \{f_{i1}, f_{i2}, \ldots, f_{im_i}\}$. The database schema $\mathcal{S}$ can thus be viewed as the structural metadata $\mathcal{S} = \{(C_i, F_i)\}_{i=1}^{k}$. The goal of the task is to generate

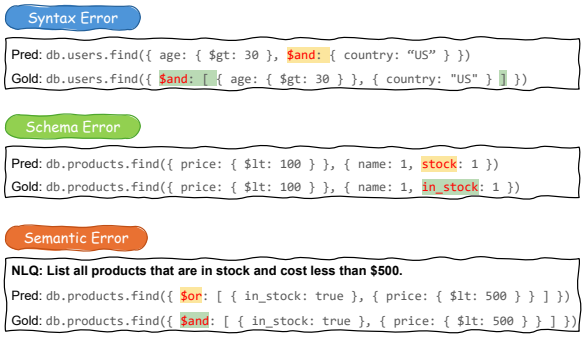

Figure 2: Three error types: Syntax Error: misuse of $\$and$ operator corrected by wrapping conditions in an array; Schema Error: invalid field $stock$, replaced with the correct schema in_stock; Semantic Error: misuse of $\$or$ instead of $\$and$, leading to a mismatch with the user intent.

a MQL $y = \{y_1, y_2, \ldots, y_m\}$, where $y \in \mathcal{Y}$ is a sequence of tokens from the target MQLs, such that y faithfully captures the semantics of x and is executable against $\mathcal{D}$.

Unlike SQL, NoSQL queries typically operate over semi-structured or hierarchical data without fixed relational schemas. For MongoDB, especially, its nested document-oriented data model introduces unique challenges such as inferring collection names, nested field paths, and handling heterogeneous document structures. As a result, solving this task requires not only understanding the natural language semantics but also adapting to the dynamic, schema-less nature of NoSQL databases.

3.1.2 MQL Repair Task

The MQL Repair task focuses on identifying and correcting errors in initially generated MongoDB queries, with the ultimate goal of producing executable MQL statements that accurately reflect the user's intent.

Within the context of our framework MARMo, the MQL Repair task functions as a plug-and-play module that can be integrated with existing Text-to-NoSQL systems. By post-editing their generated queries, it improves the overall generation accuracy and enhances system usability. This modular repair mechanism ensures robustness in practical deployment, enabling seamless and reliable end-to-end workflows.

3.1.3 Error Types

To address the challenges in robust MQL repair, we design a fine-grained error categorization scheme along with corresponding identification and repair strategies. Specifically, we classify MQL errors into three primary types: syntax errors, schema errors, and semantic error, as illustrated in Figure 2.

(1) **Syntax Error** refers to cases where the generated MQL query violates the formal grammar rules of the query language, rendering it unparsable or unexecutable.

(2) **Schema Error** arises when the structural form of the query does not match the expected template, such as incorrect nesting, misplaced clauses, or missing components, regardless of specific value content.

(3) **Semantic Error** occurs when the query structure is correct but the intent it conveys deviates from the user's original question, often due to incorrect predicates, misused operators, or improper value conditions.

3.2 Framework Overview

MARMo operates in a feedback-driven refinement loop comprising three core agents: the *Error Checker Agent*, *Feedback Manager Agent*, and *Error Repair Agent* shown in Figure 3. First, error information is leveraged by the Error Checker Agent to detect and classify MQL failures (e.g., syntactic, schema-related, or semantic). Based on the identified error type, the Feedback Manager Agent selects an appropriate reasoning path to activate different sub-agents and generate targeted repair suggestions, selectively incorporating evidence such as schema metadata, domain constraints, or self-reflection strategies. Finally, the Error Repair Agent synthesizes corrected MQL queries guided by this feedback. The algorithm is shown as Algorithm 1.

This iterative, agent-based design enables progressive query refinement and modular error correction. As a plug-and-play architecture, MARMo enhances MQL generation across diverse upstream systems by enabling fine-grained error resolution and offering transparent diagnostic traces for each repair step.

4 Methodology

In this section, a detailed description of MARMo is presented by systematically introducing its three main agents, the Error Checker Agent, the Feedback Manager Agent, and the Error Repair Agent, in the order they operate. Each component is designed to emulate distinct stages of a realistic hu-

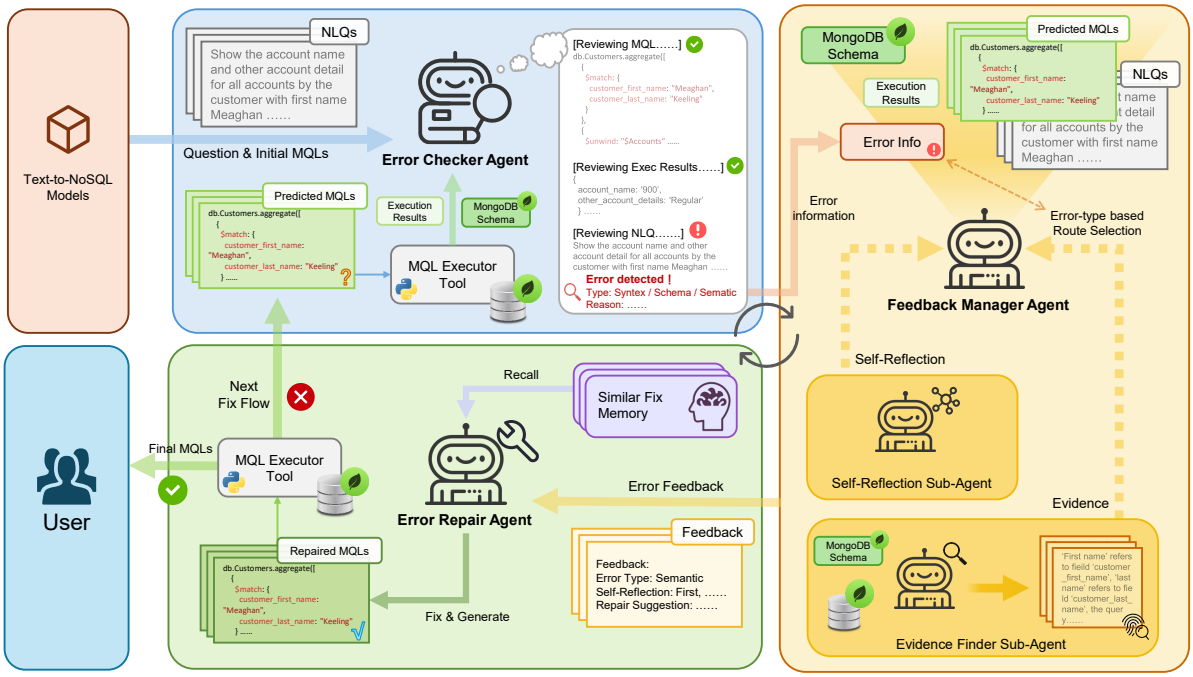

Figure 3: An overview of **MARMo** (**Mo**ngoDB-focused, **M**emory-**A**ugmented, **M**ulti-**A**gent **R**epair), an evidence-guided framework for accurate Text-to-NoSQL parsing. It features an iterative workflow with three specialized agents: the *Error Checker Agent* detects errors via execution traces and semantic cues; the *Feedback Manager Agent* orchestrates evidence-aware reasoning to produce targeted correction signals; and the *Error Repair Agent* integrates feedback and contextual memory to iteratively refine queries toward syntactic validity, schema compliance, and intent alignment.

man correction process, from error detection to guided repair.

4.1 Error Checker Agent

The Error Checker Agent is responsible for identifying and classifying errors in the predicted MQL query based on its execution behavior and contextual information. Given a natural language query and a corresponding MQL query generated by an external model, the system attempts to determine whether the MQL is correct or contains errors.

Specifically, the Error Checker Agent first invokes a Python-based executor of MQL to run the query against the MongoDB instance. The execution result including returned documents or error messages is then parsed and analyzed. Upon detecting a failure, the Error Detector uses a hybrid strategy: it combines rule-based inspection of execution logs with LLM-based reasoning over both the original input and query to determine the error type. Errors are categorized into syntactic (e.g., invalid query structure), schema-related (e.g., incorrect collection or field names), or semantic (e.g., mismatched intent). The Error Checker Agent performs comprehensive analysis and validation of the generated MQL query against the input question and schema, ultimately identifying the error type by detecting syntax error, schema mismatch or logical misalignment with the user intent and providing a rationale for the classification. Finally, the agent outputs a structured error report including the detected error type and contextual clues for downstream repair of the Feedback Manager Agent.

The motivation behind this design is to integrate execution-based error detection with the semantic reasoning capabilities of large language models, thereby enhancing the accuracy of query correctness assessment. Beyond mere error detection, the agent aims to precisely identify the type and underlying cause of each error, offering structured feedback to support downstream repair mechanisms.

4.2 Feedback Manager Agent

The Feedback Manager Agent is a composite agent composed of two collaborative sub-agents that work in tandem to produce structured and informative feedback for query repair. Taking the original natural language query, the initial MQL and the error type identified by the Error Checker Agent as

Algorithm 1 The Algorithm of MARMo

In: Natural Language Question N, Initial MQL Query $\hat{Q}$, MongoDB database D
Out: Correct MQL $\hat{Q}_c$

Let iteration $i = 0$.
while $i < Max$ **do**
  ExecResult $r = ExecTool(D, \hat{Q})$.
  ErrorType $e = ErrorChecker(N, r, \hat{Q})$.
  if $e = correct$ **then**
    $\hat{Q}_c = \hat{Q}$.
    Break.
  end if
  Feedback $f = Feedback(N, e, D, \hat{Q})$.
  MQL $\hat{Q}_r = ErrRefiner(N, f, D, \hat{Q})$.
  ExecResult $r^{'} = ExecTool(D, \hat{Q}_r)$.
  if $r^{'} = PASS$ **then**
    $\hat{Q}_c = \hat{Q}_r$.
    Break.
  end if,
  $\hat{Q} = \hat{Q}_r$ ($\rightarrow$ Next Iteration)
end while
return final MQL $\hat{Q}_c$

input, it initiates a reasoning process to analyze and contextualize the failure. The Feedback Manager Agent selectively determines the processing route based on the identified error type, invoking specialized sub-agents to generate type-specific feedback, which is subsequently aggregated and integrated into a unified correction feedback response.

(1) **Self-Reflection Sub-Agent** performs a detailed, step-by-step inference process on the faulty query by leveraging chain-of-thought prompting. This approach enables the agent to systematically analyze each component of the query and reason through potential inconsistencies. The primary goal is to accurately localize the source of the error, whether it stems from semantic misunderstandings or structural flaws within the query.

(2) **Evidence-Finder Sub-Agent** is to extract schema-related information, such as field types, collection names, nesting hierarchies, and field frequency—from the MongoDB context to facilitate schema-aware reasoning. Inspired by the effectiveness of evidence provided by the BIRD (Li et al., 2023; Yun and Lee, 2025) in supporting query generation, we introduce this agent to similarly enhance the generation and optimization of MQL queries. By grounding reasoning in concrete schema evidence, the agent helps improve both accuracy and robustness of the query construction process.

Together, these collaborative sub-agents enable the Feedback Manager to deliver precise, context-aware, and actionable feedback, thereby enhancing the effectiveness of downstream query repair and optimization.

4.3 Error Repair Agent

The Error Repair Agent is tasked with fixing and synthesizing corrected MQL queries based on structured feedback generated by upstream components in the pipeline, primarily the Feedback Manager Agent. Its core objective is to transform faulty queries into executable forms that align with both the database schema and the user intent.

To this end, this agent employs a similar fix memory mechanism and execution-guided judgment. It selects past cases that closely resemble the current query failure, and these retrieved examples serve as in-context demonstrations, providing inductive signals to guide the repair process. In parallel, the agent integrates the structured feedback to further constrain and inform the generation of candidate MQL queries, ensuring that the revisions are both semantically appropriate and structurally valid. Finally, each repaired MQL query is evaluated by the MQL executor to determine whether it satisfies the execution criteria as the final output or requires further refinement in the next repair iteration. This design allows the Error Repair Agent to effectively integrate retrieved patterns with contextual reasoning, enabling accurate and adaptable MQL query correction.

5 Experiments

In this section, we present the experimental setup and main results of MARMo, our proposed multi-agent repair framework for Text-to-NoSQL query generation. We conduct comprehensive experiments on the benchmark TEND to assess the framework's performance in handling erroneous predictions, including its ability to detect, classify, and refine faulty queries. The results clearly demonstrate that MARMo improves the accuracy and robustness of the query generation process compared to existing baselines.

For implementation of MARMo, the backbone LLM of the whole framework and different agent roles is "DeepSeek-V3" with the parameter setting 'temperature = 0.0' for relatively more precise and stable generation of MQLs. The number of repair

Method	Query-based Metric Results			Execution-based Metric Results		
	EM	QSM	QFC	EX	EFM	EVM
Instructing LLM	5.91%	50.77%	64.32%	35.06%	53.05%	58.23%
Few-shot LLM	10.41%	48.40%	55.46%	35.82%	66.99%	68.25%
RAG for LLM	16.32%	63.06%	70.05%	53.26%	73.95%	62.02%
Fine-tuned LLaMA	20.54%	56.50%	67.68%	53.12%	84.36%	68.11%
SMART	21.77%	60.65%	72.86%	59.57%	**84.90%**	**69.73%**
MARMo (Ours)	**22.09%**	**62.81%**	**74.52%**	**62.34%**	83.71%	66.88%

Table 1: The overall performance comparison of different methods on the TEND benchmark, including both query-based and execution-based evaluation metrics.

iterations is set to be 3. The memory mechanism is constructed from the training set. Similarity is computed via vector-based semantic similarity by text-embedding-ada-002.

5.1 Dataset

We evaluate our approach on the TEND benchmark (Lu et al., 2025), a large-scale benchmark for complex NoSQL query understanding especially on MongoDB. As shown in Table 2, it comprises 154 MongoDB databases across 105 domains, with 347 collections and 5,960 unique fields. The dataset includes 17,020 natural language and NoSQL query pairs, with each NoSQL query aligned to five NLQs. This diversity in both schema and query structure makes TEND a challenging and comprehensive testbed for evaluating Text-to-NoSQL models on tasks that demand nuanced understanding of both natural language and structured query generation.

Property	Value
Databases	154
Domains	105
Collections	347
Unique fields	5,960
NL-MQL pairs	17,020

Table 2: Basic statistics of the TEND dataset.

5.2 Baselines and Metrics

We compare our proposed framework MARMo against a range of strong baselines which are aligned with the experimental configurations in (Lu et al., 2025), whose details are available in Appendix A.1, ensuring fair and consistent evaluation. Our main experiments compare different paradigms of closed-source LLM utilization, including zero-shot prompting, few-shot in-context learning, and retrieval-augmented generation (RAG), as well as domain-adapted open-source LLMs via fine-tuning, alongside the most recent SMART framework. Collectively, these baselines provide a comprehensive spectrum of methods, enabling us to rigorously position MARMo against both prior neural solutions and the latest advances in LLM-based Text-to-NoSQL generation.

Meanwhile, to comprehensively evaluate model performance on the Text-to-NoSQL task, we adopt six complementary metrics, which can be broadly categorized into query-based and execution-based evaluations, following the settings of TEND, and details are elaborated in the Appendix A.2. Specifically, Query-based metrics emphasize structural fidelity, measuring how well the generated NoSQL queries align with the gold-standard annotations at the syntactic and representational levels. In contrast, execution-based metrics assess functional correctness, focusing on whether the generated queries return the intended results when executed against the underlying databases. This dual perspective allows us to jointly capture surface-level accuracy and end-task utility.

5.3 Experimental Results

Table 1 summarizes the performance of MARMo against representative baselines on the TEND benchmark. Evaluation is conducted using six complementary metrics that jointly capture query-level fidelity and execution-level correctness.

MARMo the method achieves the overall balance across all evaluated metrics, highlighting the effectiveness of its repair-oriented design. In terms of query-based metrics, MARMo achieves an Exact Match (EM) of 22.09%, which, despite the strictness of surface-form alignment, demonstrates the framework's ability to generate queries that are structurally close to gold references. For structural

evaluation, MARMo attains 62.81% Query Stages Match (QSM) and 74.52% Query Fields Coverage (QFC), reflecting its strength in reconstructing logical steps and correctly referencing schema elements. On execution-based metrics, MARMo reaches 62.34% Execution Accuracy (EX), showing robust end-to-end performance in producing executable queries. However, the proposed method exhibits a certain degradation on the EFM and EVM metrics compared to SMART. By analyzing the execution results of the golden queries and the queries predicted by MARMo, we observe that this issue may stem from inconsistencies in field naming introduced during the repair process. Specifically, different models may adopt non-unified naming conventions when representing the same semantic fields. For example, for the question `Give me the names of schools and their associated drivers for all school bus routes.`, the golden query uses the fields `School` and `Name`, whereas the repaired predicted query employs `School` and `Driver_Name`. By explicitly using `Driver_Name`, the repaired query avoids the semantic ambiguity associated with the generic field name `Name`. Taken together, these results indicate that MARMo achieves improvements primarily at the symbolic and structural levels of query formulation, while generally preserving correctness in execution-based evaluation. The overall performance gains validate the utility of its modular multi-agent repair strategy, underscoring its potential as a foundation for advancing Text-to-NoSQL query generation.

5.4 Case Study

To illustrate the effectiveness of MARMo in repairing incorrect Text-to-NoSQL predictions, we present a representative case study from our experimental dataset, as summarized in Table 3. The natural language query (NLQ) posed was 'What is the predominant age of editors in the dataset?'. The initially predicted query computed the average age of all editors rather than identifying the most frequent age. This mismatch demonstrates a typical semantic error where the model captures a general statistical aggregation rather than the mode, reflecting a subtle misalignment between the NLQ and the predicted aggregation logic. MARMo successfully detected this semantic discrepancy through its error classification and feedback generation modules. This case highlights MARMo's capability to not only identify incorrect aggregation functions but also systematically transform the query to align with the user intent expressed in natural language.

NLQ	**What is the predominant age of editors in the dataset?**
Target NoSQL	`db.editor.aggregate([ { $group: { _id: "$Age", count: { $sum: 1 } } }, { $sort: { count: -1 } }, { $limit: 1 }, { $project: { _id: 0, Age: "$_id" } } ]);` Target: Success to find the most frequent Age among editors, returns 1 result. Result: [{"Age": 20}]
Before Fix	`db.editor.aggregate([ { $group: { _id: null, avg_Age: { $avg: "$Age" } } }, { $project: { _id: 0, avg_Age: 1 } } ]);` Error: Wrong computing the average age instead of finding the most frequent age (mode), leading to unexpected results. Result: [{"avg_Age": 32.66666}]
After Fix by MARMo (Ours)	`db.editor.aggregate([ { $group: { _id: "$Age", count: { $sum: 1 } } }, { $sort: { count: -1 } }, { $limit: 1 }, { $project: { _id: 0, Age: "$_id" } } ]);` Correction: Find the semantic mismatch of NLQ and query, fix and generate the correct query. Result: [{"Age": 20}]

Table 3: The shown example is a natural language query, the original incorrect NoSQL query, and the corrected query by MARMo, along with their execution results.

6 Conclusions

In this paper, we propose MARMo, a novel multi-agent repair framework designed to address Text-to-NoSQL query fix generation, with a focus on MongoDB. MARMo systematically handles prediction errors by incorporating dedicated agents for error detection, classification, feedback generation, and query refinement. Extensive experiments demonstrate that MARMo enhances the accuracy and reliability of query generation overall, outperforming existing approaches. This work highlights the potential of collaborative agent-based architectures in improving the interpretability and robustness of natural language interfaces to NoSQL databases. Future work will explore MARMo to broader NoSQL paradigms, such as document-graph hybrids and multi-modal interfaces.

Limitations

We propose MARMo, a novel multi-agent repair framework for enhancing the accuracy of Text-to-NoSQL parsing, inspired by the human process of error correction. MARMo employs an efficient error detection mechanism coupled with a type-aware repair strategy implemented in a plug-and-play fashion. Through extensive comparisons with competitive baselines and detailed case studies, we demonstrate that MARMo consistently achieves a high level of repair performance.

Although the empirical results are promising, the current evaluation is primarily conducted on the TEND dataset, with a relatively limited set of baselines. Due to data availability and incomplete literature preparation, we were unable to obtain the complete MultiTEND and DocSpider datasets. To ensure reproducibility and consistency, these datasets were not included in the main experiments. MARMo was originally developed and implemented with the TEND Benchmark in mind, incorporating optimizations specific to MongoDB semantics, error types, and repair patterns. Extending the framework to other datasets would require additional engineering effort. As one of the major benchmarks in the Text-to-NoSQL domain, TEND's complexity and diversity allow MARMo's capabilities to be effectively demonstrated. These choices allow us to maintain a controlled and focused experimental setting, while also emphasizing the extensibility of MARMo to Text-to-NoSQL parsing tasks and the novelty of integrating multi-agent collaboration with error repair in this context. Meanwhile, to ensure the accuracy of our evaluation stability, the current implementation of the fixed memory in MARMo relies solely on the training set and does not incorporate any data from the evaluation set and real-time fix process.

For future work, we plan to broaden the empirical scope by applying MARMo to a wider variety of datasets, different NoSQL databases and incorporating more diverse baseline systems, which will further assess the generalizability and robustness of our framework.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: Line graph enhanced text-to-SQL model with mixed local and non-local relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2541–2555, Online. Association for Computational Linguistics.

Jipeng Cen, Jiaxin Liu, Zhixu Li, and Jingjing Wang. 2025. Sqlfixagent: Towards semantic-accurate text-to-sql parsing via consistency-enhanced multi-agent collaboration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 49–57.

Yeounoh Chung, Gaurav T. Kakkar, Yu Gan, Brenton Milne, and Fatma Özcan. 2025. Is long context all you need? leveraging llm's extended context for nl2sql. *Proc. VLDB Endow.*, 18(8):2735–2747.

Minghang Deng, Ashwin Ramachandran, Canwen Xu, Lanxiang Hu, Zhewei Yao, Anupam Datta, and Hao Zhang. 2025. Reforce: A text-to-sql agent with self-refinement, consensus enforcement, and column exploration. *arXiv preprint arXiv:2502.00675*.

Yusuf Denizay Dönder, Derek Hommel, Andrea W Wen-Yi, David Mimno, and Unso Eun Seo Jo. 2025. Cheaper, better, faster, stronger: Robust text-to-sql without chain-of-thought or fine-tuning. *arXiv preprint arXiv:2505.14174*.

Elias Dritsas and Maria Trigka. 2025. Database systems in the big data era: Architectures, performance, and open challenges. *IEEE Access*, 13:95068–95084.

Yuankai Fan, Tonghui Ren, Zhenying He, X Sean Wang, Ye Zhang, and Xingang Li. 2023. Gensql: A generative natural language interface to database systems. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 3603–3606. IEEE.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021a. Towards robustness of text-to-SQL models against synonym substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2505–2515, Online. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021b. Exploring underexplored limitations of cross-domain text-to-SQL generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8926–8931, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yingqi Gao, Yifu Liu, Xiaoxia Li, Xiaorong Shi, Yin Zhu, Yiming Wang, Shiqi Li, Wei Li, Yuntao Hong,

Zhiling Luo, and 1 others. 2024. Xiyan-sql: Amulti-generator ensemble framework for text-to-sql. *arXiv preprint arXiv:2411.08599*.

Orest Gkini, Theofilos Belmpas, Georgia Koutrika, and Yannis Ioannidis. 2021. An in-depth benchmarking of text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data*, pages 632–644.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Siwei Han, Peng Xia, Ruiyi Zhang, Tong Sun, Yun Li, Hongtu Zhu, and Huaxiu Yao. 2025. Mdocagent: A multi-modal multi-agent framework for document understanding. *arXiv preprint arXiv:2503.13964*.

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. *arXiv preprint arXiv:2406.08426*.

Radu Cristian Alexandru Iacob, Florin Brad, Elena-Simona Apostol, Ciprian-Octavian Truică, Ionel Alexandru Hosu, and Traian Rebedea. 2020. Neural approaches for natural language interfaces to databases: A survey. In *proceedings of the 28th International Conference on Computational Linguistics*, pages 381–395.

Yoichi Ishibashi and Yoshimasa Nishimura. 2024. Self-organized agents: A llm multi-agent framework toward ultra large-scale code generation and optimization. *arXiv preprint arXiv:2404.02183*.

George Katsogiannis-Meimarakis, Mike Xydas, and Georgia Koutrika. 2023. Natural language interfaces for databases with deep learning. *Proceedings of the VLDB Endowment*, 16(12):3878–3881.

Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. Natural language to sql: Where are we today? *Proceedings of the VLDB Endowment*, 13(10):1737–1750.

Bingxuan Li, Yiwei Wang, Jiuxiang Gu, Kai-Wei Chang, and Nanyun Peng. 2025. METAL: A multi-agent framework for chart generation with test-time scaling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30054–30069, Vienna, Austria. Association for Computational Linguistics.

Fei Li and Hosagrahar V Jagadish. 2014. Nalir: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 709–712.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024a. Codes: Towards building open-source language models for text-to-sql. *Proceedings of the ACM on Management of Data*, 2(3):1–28.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2023. 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*, 36:42330–42357.

Yuntao Li, Hanchu Zhang, Yutian Li, Sirui Wang, Wei Wu, and Yan Zhang. 2021. Pay more attention to history: A context modelling strategy for conversational text-to-sql. *arXiv preprint arXiv:2112.08735*.

Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, and 1 others. 2024b. Autokaggle: A multi-agent framework for autonomous data science competitions. *arXiv preprint arXiv:2410.20424*.

Zhisheng Lin, Yifu Liu, Zhiling Luo, Jinyang Gao, and Yu Li. 2024. Momq: Mixture-of-experts enhances multi-dialect query generation across relational and non-relational databases. *arXiv preprint arXiv:2410.18406*.

Xinyu Liu, Shuyu Shen, Boyan Li, Nan Tang, and Yuyu Luo. 2025. Nl2sql-bugs: A benchmark for detecting semantic errors in nl2sql translation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, page 5662–5673, New York, NY, USA. Association for Computing Machinery.

Xiping Liu and Zhao Tan. 2024. Epi-sql: Enhancing text-to-sql translation with error-prevention instructions. *arXiv preprint arXiv:2404.14453*.

Jinwei Lu, Yuanfeng Song, Zhiqian Qin, Haodi Zhang, Chen Zhang, and Raymond Chi-Wing Wong. 2025. Bridging the gap: Enabling natural language queries for nosql databases through text-to-nosql translation. *arXiv preprint arXiv:2502.11201*.

Wenxin Mao, Ruiqi Wang, Jiyu Guo, Jichuan Zeng, Cuiyun Gao, Peiyi Han, and Chuanyi Liu. 2024. Enhancing text-to-sql parsing through question rewriting and execution-guided refinement. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2009–2024.

Ana Nunez, Nafis Tanveer Islam, Sumit Kumar Jha, and Peyman Najafirad. 2024. Autosafecoder: A multi-agent framework for securing llm code generation through static analysis and fuzz testing. *arXiv preprint arXiv:2409.10737*.

Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O Arik. 2024. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql. *arXiv preprint arXiv:2410.01943*.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36:36339–36348.

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, and 1 others. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629*.

Zhiqian Qin, Yuanfeng Song, Jinwei Lu, Yuanwei Song, Shuaimin Li, and Chen Jason Zhang. 2025. MultiTEND: A multilingual benchmark for natural language to NoSQL query translation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24632–24657, Vienna, Austria. Association for Computational Linguistics.

Tonghui Ren, Yuankai Fan, Zhenying He, Ren Huang, Jiaqi Dai, Can Huang, Yinan Jing, Kai Zhang, Yifan Yang, and X Sean Wang. 2024. Purple: Making a large language model a better sql writer. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 15–28. IEEE.

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*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiawei Shen, Chengcheng Wan, Ruoyi Qiao, Jiazhen Zou, Hang Xu, Yuchen Shao, Yueling Zhang, Weikai Miao, and Geguang Pu. 2025. A study of in-context-learning-based text-to-sql errors. *arXiv preprint arXiv:2501.09310*.

Vladislav Shkapenyuk, Divesh Srivastava, Theodore Johnson, and Parisa Ghane. 2025. Automatic metadata extraction for text-to-sql. *arXiv preprint arXiv:2505.19988*.

Quoc-Bao-Huy Tran, Aagha Abdul Waheed, and Sun-Tae Chung. 2024. Robust text-to-cypher using combination of bert, graphsage, and transformer (cobgt) model. *Applied Sciences*, 14(17):7881.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2025. MAC-SQL: A multi-agent collaborative framework for text-to-SQL. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 540–557, Abu Dhabi, UAE. Association for Computational Linguistics.

Yihang Xiao, Jinyi Liu, Yan Zheng, Xiaohan Xie, Jianye Hao, Mingzhi Li, Ruitao Wang, Fei Ni, Yuxiao Li, Jintian Luo, and 1 others. 2024. Cellagent: An llm-driven multi-agent framework for automated single-cell data analysis. *arXiv preprint arXiv:2407.09811*.

Wenxuan Xie, Gaochen Wu, and Bowen Zhou. 2024a. Mag-sql: Multi-agent generative approach with soft schema linking and iterative sub-sql refinement for text-to-sql. *arXiv preprint arXiv:2408.07930*.

Yuanzhen Xie, Xinzhou Jin, Tao Xie, Matrixmxlin Matrixmxlin, Liang Chen, Chenyun Yu, Cheng Lei, Chengxiang Zhuo, Bo Hu, and Zang Li. 2024b. Decomposition for enhancing attention: Improving LLM-based text-to-SQL through workflow paradigm. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10796–10816, Bangkok, Thailand. Association for Computational Linguistics.

Bo Xu, Shufei Li, Hongyu Jing, Ming Du, Hui Song, Hongya Wang, and Yanghua Xiao. 2025. Boosting text-to-sql through multi-grained error identification. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4282–4292.

Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir 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*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Janghyeon Yun and Sang-goo Lee. 2025. Seed: Enhancing text-to-sql performance and practical usability through automatic evidence generation. *arXiv preprint arXiv:2506.07423*.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023. ACT-SQL: In-context learning for text-to-SQL with automatically-generated chain-of-thought. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532, Singapore. Association for Computational Linguistics.

Wenlong Zhang, Chengyang He, Guanqun Yang, Dipankar Bandyopadhyay, Tian Shi, and Ping Wang. 2024a. Natural language querying on domain-specific nosql database with large language models. In *2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 5174–5181. IEEE.

Wenlong Zhang, Tian Shi, and Ping Wang. 2024b. Natural language querying on nosql databases: Opportunities and challenges [vision paper]. In *2024 IEEE International Conference on Big Data (BigData)*, pages 3353–3357. IEEE.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Yuhang Zhou, Yu He, Siyu Tian, Yuchen Ni, Zhangyue Yin, Xiang Liu, Chuanjun Ji, Sen Liu, Xipeng Qiu, Guangnan Ye, and Hongfeng Chai. 2024. r^3-NL2GQL: A model coordination and knowledge graph alignment approach for NL2GQL. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13679–13692, Miami, Florida, USA. Association for Computational Linguistics.

Yizhang Zhu, Shiyin Du, Boyan Li, Yuyu Luo, and Nan Tang. 2024. Are large language models good statisticians? *Advances in Neural Information Processing Systems*, 37:62697–62731.

A Appendix

A.1 Baselines

We compare our proposed framework against several strong baselines, including both traditional neural models and recent LLM-based approaches for Text-to-NoSQL generation. Baselines settings follow (Lu et al., 2025).

- **Zero-shot LLM** leverages the inherent zero-shot capabilities of LLMs to generate contextually appropriate responses without requiring task-specific examples or fine-tuning.
- **Few-shot LLM** employs in-context learning by incorporating a limited number of static task-relevant examples into the prompt, enabling LLMs to adapt to domain-specific tasks with context.
- **RAG for LLM** enhances LLM performance through Retrieval-Augmented Generation, which dynamically incorporates external knowledge into prompts, mitigating hallucinations caused by incomplete or ambiguous input.
- **Finetuned LLaMA** (Grattafiori et al., 2024) refers to the LLaMA model finetuned with TEND training set to predicted MQL. In practice we use LLaMA-3.2-1B-Instruct to implement the Text-to-NoSQL task.
- **SMART** is the first framework targeting Text-to-NoSQL generation, SMART integrates SLMs and RAG to enable schema prediction, query generation, refinement via retrieved examples, and optimization based on execution feedback.
- **MARMo** is the first solution that applies multi-agent collaboration architecture to Text-to-NoSQL. It introduces iterative refinement process and three dedicated agents responsible for error detection and classification, feedback synthesis, and query refinement, respectively. In addition, its modular and pluggable design allows it to be seamlessly integrated into a wide range of Text-to-NoSQL systems as an auxiliary repair component.

A.2 Metrics

To comprehensively evaluate model performance on the Text-to-NoSQL task, we adopt six complementary metrics, grouped into **query-based** and **execution-based** evaluations, following TEND. Query-based metrics focus on structural fidelity:

- **Exact Match (EM)** requires the generated query to exactly match the gold query in both structure and content. It provides a strict measure of syntactic and semantic alignment.
- **Query Stages Match (QSM)** evaluates the correctness of the query pipeline by comparing stage types (e.g. match, group, lookup) and order with these of the gold query.
- **Query Fields Coverage (QFC)** measures whether all relevant fields—either from the schema or derived—are correctly included, reflecting schema comprehension. It evaluates the performance of methods on database schema coverage.

On the other side, execution-based metrics assess functional correctness:

- **Execution Accuracy (EX)** checks if the generated query yields fully identical results to the gold query when executed. EX is the most critical performance metric for evaluating Text-to-NoSQL models.
- **Execution Fields Match (EFM)** evaluates whether the field set of output matches the correct set of field names, as the JSON-like query result of document-based database like MongoDB has the field part (key) and the value part (value).
- **Execution Value Match (EVM)** verifies the correctness of output values -.

Together, these metrics offer a holistic view of model performance, from structural precision to executable reliability.