Resolving Ambiguity in NL2SQL

— A Reproduction and Analysis of the Sphinteract Framework

Group Information

This section provides details about the group members who conducted this reproduction study.

Name	Undergraduate Major
WU Mengfei	Economics
LIU Zichun	Automation
ZHONG Zhiyuan	Mathematics
SONG Jiafeng	Psychology

Contribution Statement

A contribution statement clarifies the specific roles and responsibilities of each team member, ensuring transparency in collaborative academic work.

Work	Member
Video	ZHONG Zhiyuan
Code	LIU Zichun
Report	WU Mengfei, LIU Zichun, SONG Jiafeng

1. Project Overview

This section details the replicated academic paper and the core motivation for selecting it, establishing the foundation of this reproduction study. The central challenge in the Natural Language to SQL (NL2SQL) domain is the inherent ambiguity of human language, which often leads even powerful Large Language Models (LLMs) to misinterpret user intent and generate incorrect queries. This project focuses on a novel interactive framework designed to address this fundamental problem.

1.1 The Replicated Paper

The work reproduced and analyzed in this report is the paper titled **"Sphinteract: Resolving Ambiguities in NL2SQL Through User Interaction"** by Fuheng Zhao, Shaleen Deep, Fotis Psallidas, Avrilia Floratou, Divyakant Agrawal, and Amr El Abbadi. It was published in **PVLDB, 18(4): 1145 - 1158, 2024.**

1.2 Motivation for Selection

The core problem in the NL2SQL domain is that a user's intent is often ambiguous, leading LLMs to generate incorrect SQL queries. Traditional non-interactive, unidirectional approaches are forced to guess at user intent, and these guesses frequently do not align with the user's expectations.

For example, consider the question: *"Where is the first 'BWR' type power plant built and located?"* This seemingly simple query contains multiple points of ambiguity:

- **Temporal Ambiguity:** Does "first" refer to the earliest construction start date (`ConstructionStartAt`) or the earliest operational date (`OperationalFrom`)?
- **Output Ambiguity:** Does "located" refer to the country and name (`Country`, `Name`) or the precise geographical coordinates (`Latitude`, `Longitude`)?

An LLM must make an assumption for each of these points, and an incorrect assumption at any step results in a failed query. The Sphinteract framework's novel interactive approach to systematically resolving these ambiguities presents a compelling and important area for reproduction and analysis. By engaging the user in a clarification dialogue, the system can move from guesswork to confirmed intent, dramatically improving the reliability of NL2SQL systems.

2. Technical Design and Implementation Strategy

This section deconstructs the original Sphinteract framework's technical approach and then details the specific strategy employed in this reproduction study, including the key methodological deviation from the original paper's evaluation.

2.1 The Sphinteract Framework

Sphinteract is a framework designed to help LLMs resolve ambiguity in NL2SQL tasks by interactively soliciting user feedback. Its core innovation lies in guiding the LLM's reasoning process through a structured prompt paradigm called **"Summarize, Review, Ask" (SRA)**. This paradigm enables the LLM to iteratively refine its understanding of the user's needs.

The SRA process involves three key steps:

1. **Summarize:** Consolidate all known information based on the user's question and previous feedback.
2. **Review:** Identify any remaining points of confusion or ambiguity.
3. **Ask:** Generate a targeted, multiple-choice clarification question to resolve a specific ambiguity.

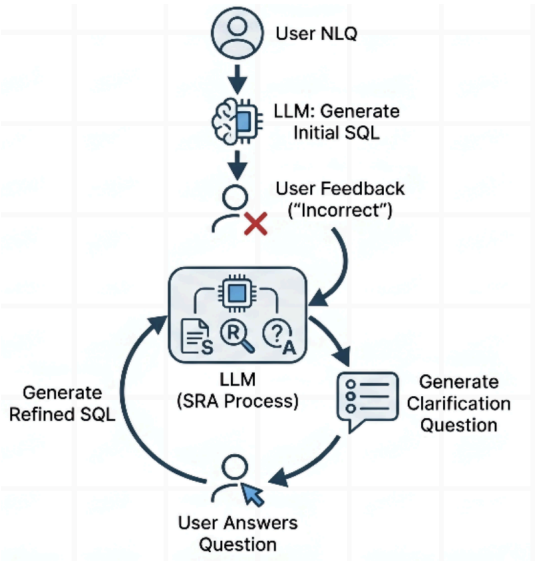

FIG. 1 SRA Process

To structure its reasoning and question generation, Sphinteract categorizes ambiguities into four primary types. This taxonomy provides a systematic checklist for the LLM to diagnose and resolve specific points of confusion.

- **AmbQuestion:** The question's phrasing is inherently vague (e.g., 'most productive' could mean by count, sales, etc.).
- **AmbColumn:** There is an unclear mapping from entities in the question to specific database columns (e.g., does 'position' map to a `notes` or `category` column?).
- **AmbOutput:** The desired output columns, format, or ordering are not specified.
- **AmbValue:** There is uncertainty about the correct predicate values for a `WHERE` clause (e.g., is the state 'TX' or 'TEXAS'?).

2.2 Our Reproduction Strategy

The primary objective of this study was to reproduce the three progressive interaction models proposed in the paper (M1, M2, and M3) and validate their performance on the KaggleDBQA dataset.

A key methodological choice was made to deviate from the original paper's random sampling approach. Instead, we curated a specialized test set of **30 highly ambiguous questions**. These questions were pre-filtered and identified as 'ambiguous' by an LLM evaluator.

The rationale for this focused approach was to create a challenging benchmark specifically designed to **stress-test the framework's core value proposition**—resolving ambiguity. By concentrating on difficult cases, this strategy provides a clearer and more potent signal of the interactive system's true value, as opposed to a random sample which may be dominated by simpler, unambiguous queries.

This strategic deviation from random sampling to targeted stress-testing forms the core of our reproduction's experimental design, setting the stage for a detailed examination of the implemented algorithms.

3. Core Algorithm Implementation

This section focuses on the implementation details of the core algorithms from the Sphinteract paper. It covers the three distinct interaction models that were built and tested, as well as the methodology used to programmatically validate the correctness of the SQL queries they generated.

3.1 The Three Interaction Models

The study implemented and evaluated three progressive interaction models, each building upon the last in complexity and capability.

1. **M1 (Baseline): Simple Feedback** In this baseline model, the user provides a simple binary signal of 'correct' or 'incorrect' for a generated query. If the query is incorrect, the LLM is prompted again with the history of incorrect queries and instructed to self-correct and generate a new attempt.
2. **M2 (Sphinteract): Clarification Questions (CQs)** This model implements the core Sphinteract framework. The LLM uses the "Summarize, Review, Ask" (SRA) paradigm to analyze the user's request, identify specific ambiguities, and generate targeted, multiple-choice clarification questions. The user's answer is then used to generate a refined SQL query.
3. **M3 (Sphinteract+ES): CQs with Early Stopping** This model is an enhancement of M2. The LLM is given an additional instruction: if it determines that no significant ambiguity remains, it can decide to stop asking questions and output "NO AMBIGUITY". This mechanism aims to optimize the interaction cost by avoiding unnecessary clarification rounds.

3.2 Correctness Validation

To validate the correctness of the generated SQL, a function named `evalfunc` was implemented. This function programmatically executes both the LLM-generated SQL query and the ground-truth (gold) SQL query against the target database.

Correctness is determined if the execution results of the two queries are identical. The comparison logic uses a simple but effective heuristic:

- If the ground-truth query contains an `ORDER BY` clause, a **strict-order comparison** is performed.
- Otherwise, the results from both queries are sorted before comparison, allowing for **order-independence**.

This automated validation process ensures objective and consistent evaluation across all three models and experimental settings.

4. Evaluation and Results

This section presents the empirical results of the reproduction experiment. It details the experimental setup, defines the performance metrics, and provides a thorough analysis of the findings regarding the accuracy and efficiency of the different interaction models.

4.1 Experimental Setup

The experiment was designed as a controlled comparison of the three interaction models on a challenging dataset.

- **Dataset:** A curated set of **30 highly ambiguous questions** selected from the KaggleDBQA benchmark.
- **Methods Compared:**
    - **M1** (Baseline: Simple Feedback / Self-Correction)
    - **M2** (Sphinteract: Clarification Questions)
    - **M3** (Sphinteract + Early Stopping)
- **Evaluation Settings:** Both **Zero-shot** (no examples provided) and **3-shot** (three examples provided) SQL generation.
- **Core Metrics:**
    - **Execution Accuracy:** The percentage of generated SQL queries that produce the correct result when executed against the database.
    - **Average Interaction Rounds:** The mean number of user interactions (clarification questions or feedback rounds) required per question.
- **Feedback Simulation:** An oracle using **GPT-4o** was employed to automatically answer clarification questions based on the ground-truth SQL. This is a standard practice in interactive systems research to ensure consistent and scalable evaluation.

4.2 Performance Analysis: Accuracy and Efficiency

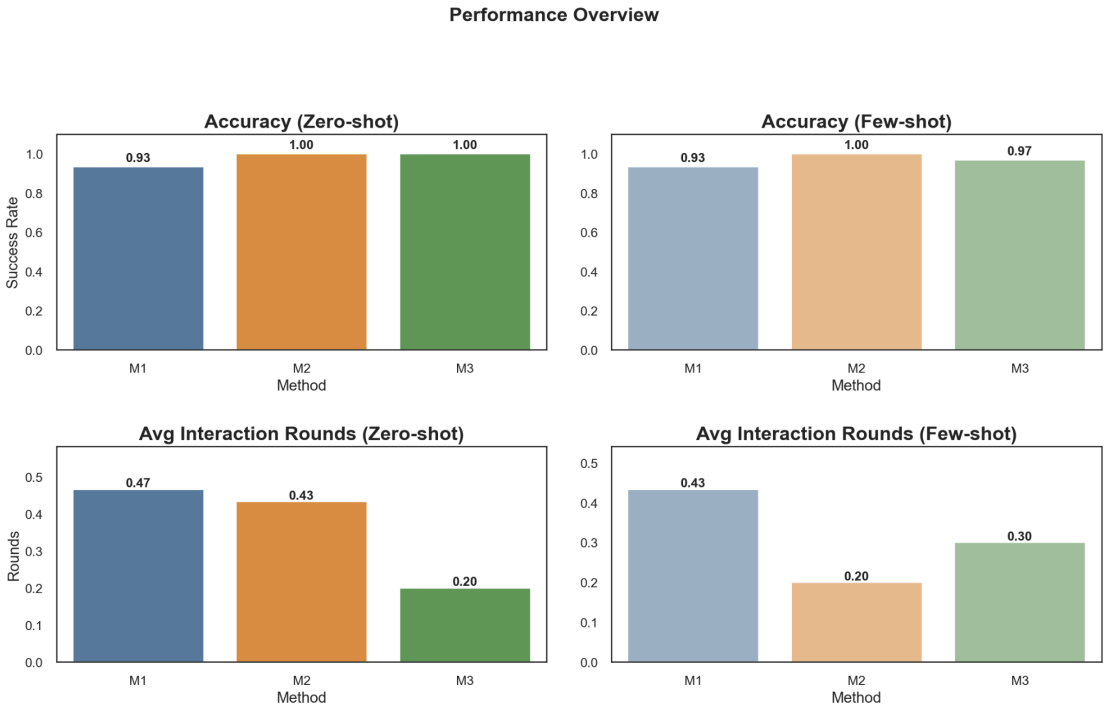

FIG. 2 Performance Overview of Methods 1-3 with Zero-shot & Few-shot

The results demonstrate a clear performance advantage for the interactive Sphinteract models, especially on this high-ambiguity dataset.

- **Accuracy (Zero-shot):** Both M2 (Sphinteract) and M3 (Sphinteract+ES) achieved **100% accuracy**, showcasing a powerful capability to resolve errors through interaction. In contrast, the M1 baseline stalled at **93.3% (28/30)**, failing on two of the thirty queries.
- **Efficiency (Few-shot):** Providing few-shot examples primarily improved interaction efficiency rather than peak accuracy. For the M2 model, 3-shot prompting **cut the average interaction rounds by more than half**, from 0.43 in the zero-shot setting to just 0.20. Conversely, few-shot prompting showed a mixed result for M3, as its accuracy slightly decreased to 96.7% (one error) and its average interaction rounds increased to 0.30, suggesting that in-context examples are not a universal benefit for all interactive models on this specific dataset.

- **Optimal Trade-off (Zero-shot):** The M3 model (with early stopping) provided the best overall performance in the zero-shot setting. It matched M2's perfect 100% accuracy while achieving the **lowest interaction cost** of all methods (0.20 rounds), successfully identifying when further questions were unnecessary.

4.3 Analysis of Error Recovery

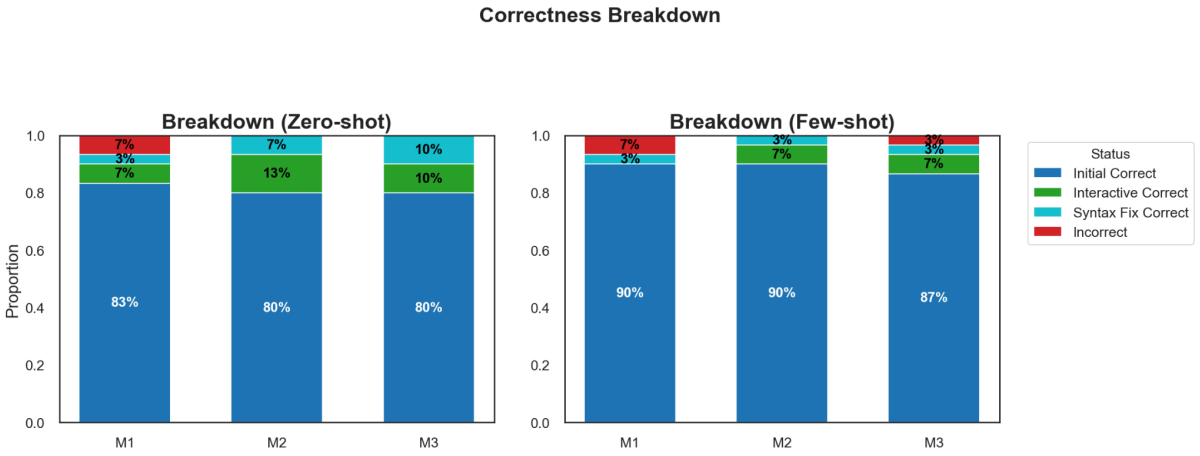

FIG. 3 Correctness Breakdown among Four Status

A deeper analysis of how correctness was achieved reveals the true strength of the interactive models.

- **Observation 1:** All methods achieved a high rate of correctness on the first attempt ('Initial Correct'). In the zero-shot setting, both M2 and M3 generated the correct query on the first try for 24 out of 30 samples (80%), indicating strong baseline LLM performance with gpt-3.5-turbo.
- **Observation 2:** The key differentiator was **error recovery**. In the zero-shot setting, the interactive models **M2 and M3 successfully corrected 100% of the initially failed queries** through either user interaction or automated syntax fixes.
- **Observation 3:** In contrast, the M1 baseline, relying only on simple self-correction, failed to correct 2 out of the 30 queries, leaving them as final errors in both the zero-shot and few-shot settings.

This breakdown clearly shows that the interactive mechanism of Sphinteract acts as a highly reliable safety net for difficult, ambiguous cases where a non-interactive model would otherwise fail. These findings highlight the critical role of interaction in achieving robust NL2SQL performance, leading to a reflection on the broader challenges encountered.

5. Challenges and Lessons Learned

This section reflects on the practical engineering challenges faced during the reproduction study and distills the key insights gained from overcoming them. Reproducing state-of-the-art research, particularly in the LLM domain, is as much an engineering endeavor as it is a scientific one.

5.1 Engineering Challenges

Several practical hurdles had to be overcome to ensure the reliability and efficiency of the experimental pipeline.

1. **API Instability & Rate Limiting** Frequent 502 (Bad Gateway), 429 (Too Many Requests), and timeout errors from the LLM API threatened to disrupt long-running experiments. The solution

was to implement **robust error handling with exponential backoff** and an **automatic fallback mechanism** to alternative models (e.g., `gpt-4o-mini`, `gpt-4o`), ensuring that experiments could run to completion without manual intervention.
2. **Efficient Ambiguous Sample Selection** Manually identifying 30 ambiguous samples from the dataset would have been slow and subjective. To automate this, we developed a **parallelized batch processor** that used an LLM to evaluate samples for ambiguity. An **early-stopping mechanism** was crucial, halting the process as soon as the target of 30 samples was found, saving significant time and API costs.
3. **Nuanced Performance Analysis** A simple "correct" or "incorrect" label was insufficient to understand *how* queries were successfully resolved. To gain deeper insight, we implemented **distinct status tags**—`Initial Correct`, `Interactive Correct`, and `Syntax Fix Correct`—to differentiate between queries that were correct from the start, those fixed through user feedback, and those corrected via automated syntax repair.

5.2 Key Insights and Takeaways

Overcoming these challenges yielded valuable lessons about both the Sphinteract framework and the process of LLM-based research reproduction.

- **Confirmation:** Our results, generated on a high-ambiguity dataset, strongly validate the central claim of the Sphinteract paper: interactive feedback is a highly effective method for resolving ambiguity and achieving near-perfect accuracy in NL2SQL.
- **Methodological Insight:** We found that testing on a curated, high-ambiguity subset provides a much clearer and more powerful signal of an interactive system's value compared to random sampling. The latter approach can be dominated by simpler, unambiguous queries, which may mask the true benefit of the interactive mechanism.
- **Engineering Takeaway:** Reproducing state-of-the-art LLM research is a significant engineering challenge. Success requires building robust and resilient systems with thoughtful solutions for API instability, efficient data processing, and precise metric tracking to ensure the reliability and reproducibility of experimental results.

Successfully navigating these engineering hurdles was essential for validating the paper's claims and arriving at a conclusive summary of the project's findings.

6. Conclusion

This report successfully reproduced the core claims of the Sphinteract framework, demonstrating its efficacy in resolving ambiguity in NL2SQL tasks. The framework, particularly its "Summarize, Review, Ask" (SRA) paradigm, provides a robust and effective methodology for aligning LLM-generated SQL queries with true user intent.

Our reproduction, conducted on a challenging, ambiguity-focused dataset, confirms the framework's ability to achieve exceptional accuracy by systematically clarifying ambiguities where non-interactive methods fail. The results show that interactive models like Sphinteract (M2) and its efficiency-optimized variant (M3) can correct 100% of initial query failures on this difficult subset, serving as a critical safety net.

As identified by the original paper, promising avenues for future work include improving the user experience (UX) beyond multiple-choice questions, developing methods for learning user preferences over time to personalize the clarification process, and exploring asynchronous system architectures to improve performance and reduce user wait times.

References

Zhao, F., Deep, S., Psallidas, F., Floratou, A., Agrawal, D., & El Abbadi, A. (2024). Sphinteract: Resolving Ambiguities in NL2SQL Through User Interaction. *PVLDB, 18*(4), 1145 - 1158. doi:10.14778/3717755.3717772

Links to Deliverables

- **Source Code Repository:** GitHub Repository
- **Video Demonstration:** Link to Video