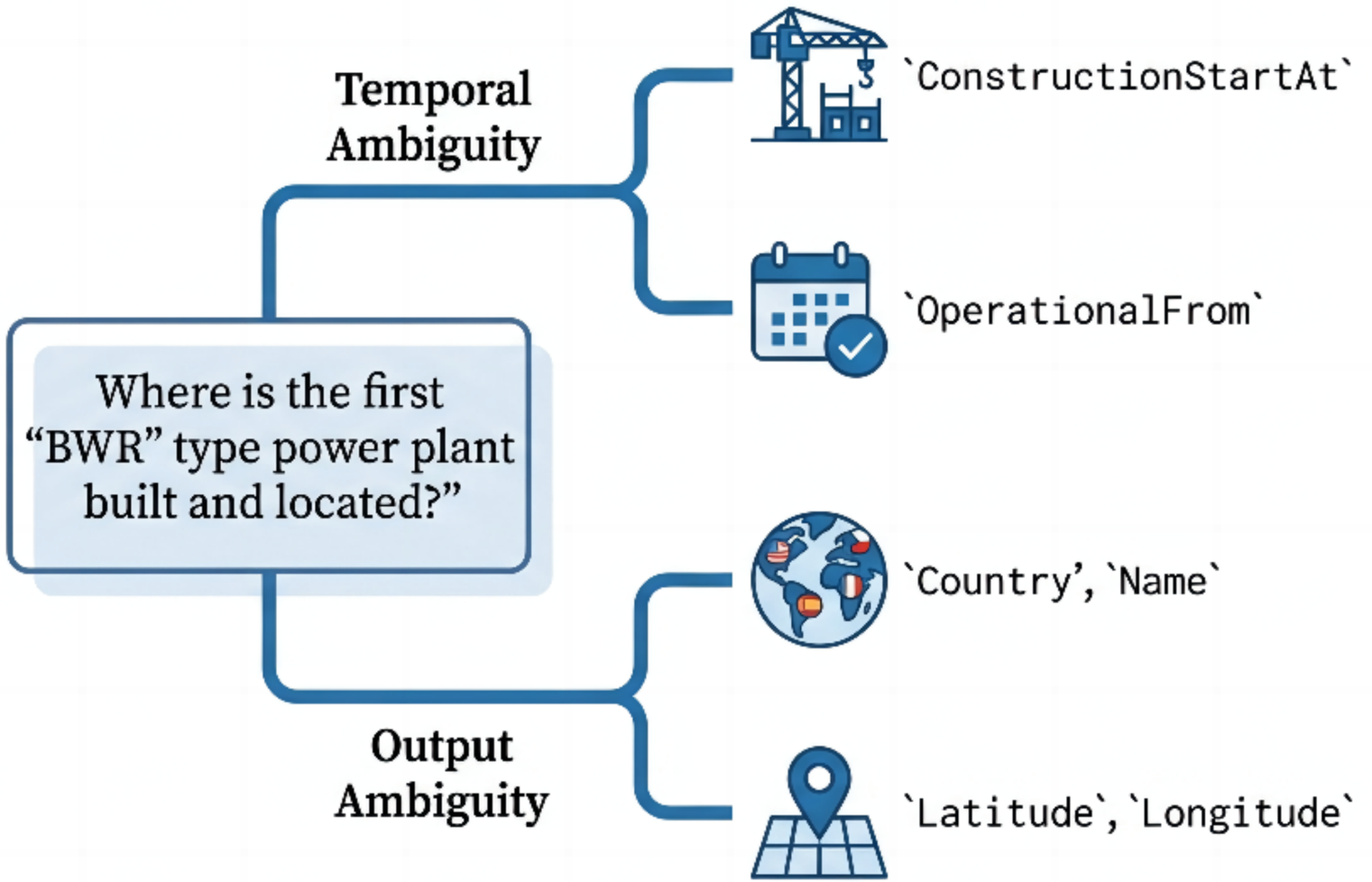
Resolving Ambiguity in NL2SQL: A Fproduction and Analysis of the Sphinteract Framework

WU Mengfei, LIU Zichun, ZHONG Zhiyuan, SONG Jiafeng

NL2SQL's Persistent Challenge: User Intent is Often Ambiguous

Problem Statement: While Large Language Models (LLMs) are powerful for Natural Language to SQL (NL2SQL) tasks, they frequently fail when a user's question has multiple valid interpretations.

Takeaway: Non-interactive approaches guess at user intent, often leading to incorrect SQL queries that do not match the user's expectation.

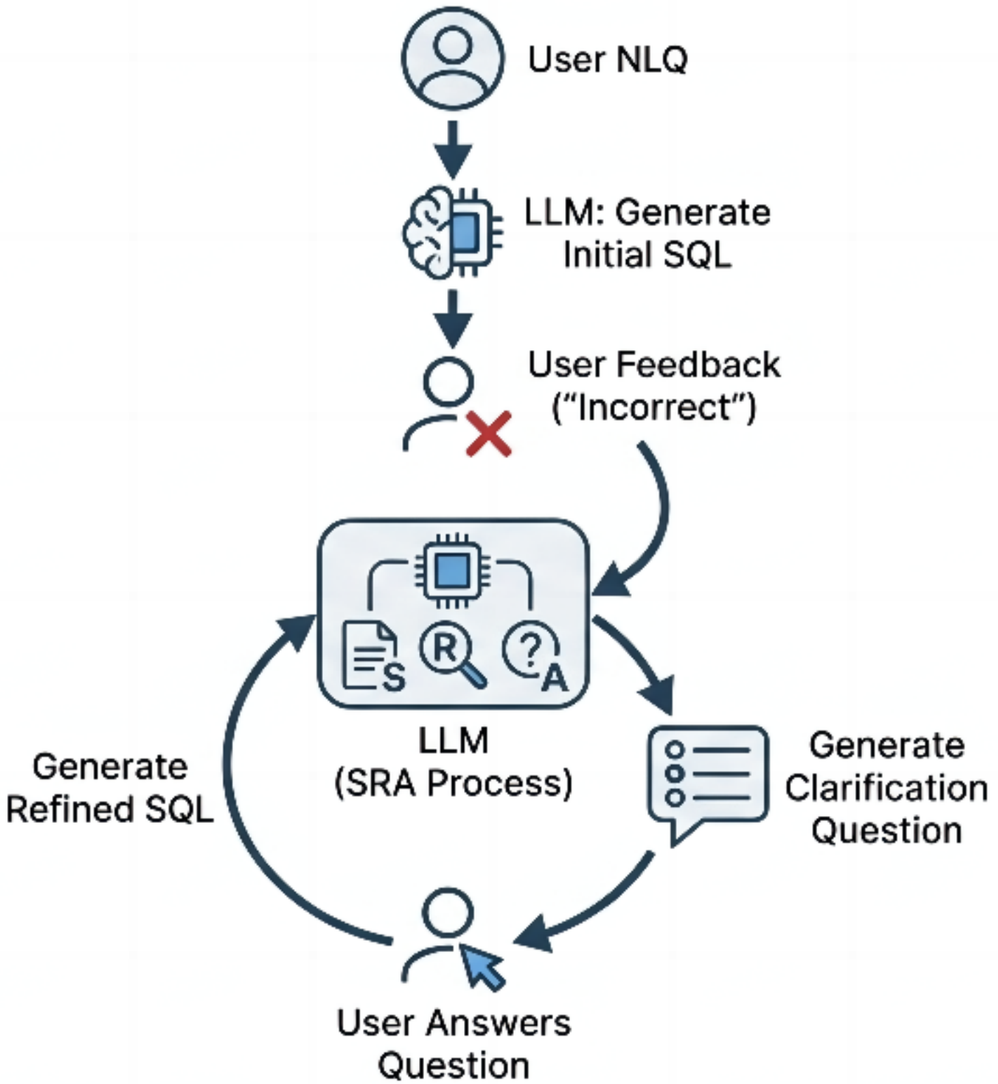


Sphinteteract: Resolving Ambiguity through Iter Inrectively User Interaction

Concept: A framework from Zhao et al. (PVLDB 2024) that helps LLMs resolve ambiguity by interactively soliciting user feedback.

Core Paradigm: It uses a structured prompt called **SRA (Summarize, Review, Ask)** to guide the LLM's reasoning:

- **Summarize:** Consolidate known information.
- **Review:** Identify remaining ambiguities.
- **Ask:** Generate a targeted clarification question.



A Taxonomy of Ambiguity to Guide Clarification

Sphinteract categorizes ambiguities into four primary types to structure its reasoning and question generation:

AmbQuestion

The question's phrasing is inherently vague (e.g., 'most productive' could mean by count, sales, etc.).

AmbColumn

Unclear mapping from entities in the question to specific database columns (e.g., does 'position' map to `notes` or `category` column?).

AmbOutput

The desired output columns, format, or ordering are not specified.

AmbValue

Uncertainty about the correct predicate values for a `WHERE` clause (e.g., is the state 'TX' or 'TEXAS'?).

This taxonomy provides a systematic checklist for the LLM to diagnose and resolve specific points of confusion.

Three Progressive Interaction Models for Evaluation

The paper proposes and evaluates three feedback mechanisms, which form the basis of our reproduction:

M1

Method 1 (Baseline): Simple Feedback

The user provides a simple 'correct/incorrect' signal. The LLM then uses the history of incorrect queries to self-correct.

M2

Method 2 (Sphinteract): Clarification Questions (CQs)

The LLM uses the SRA paradigm to generate targeted, multiple-choice questions to resolve specific ambiguities.

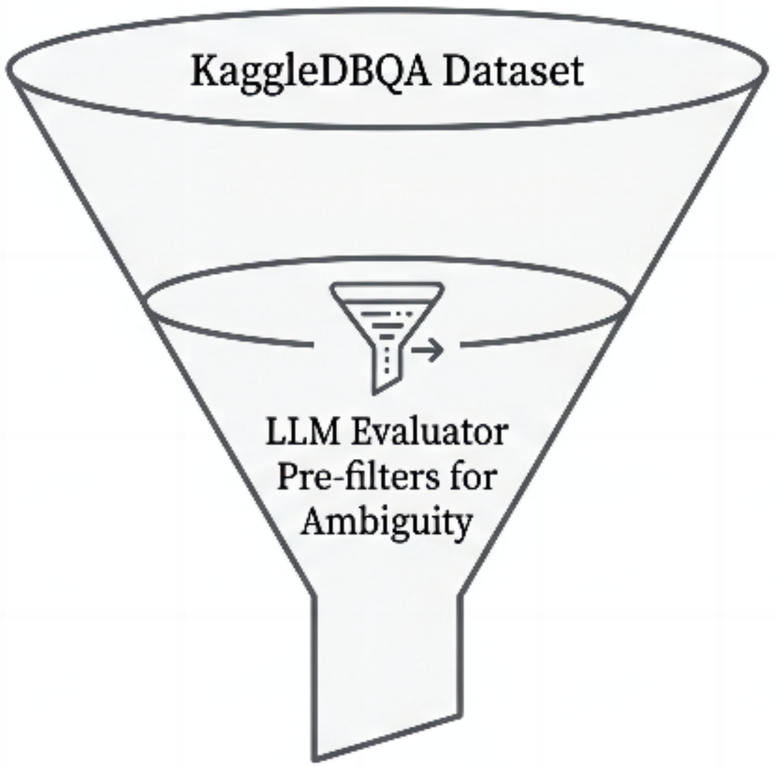
M3

Method 3 (Sphinteract+ES): CQs with Early Stopping

An enhancement where the LLM can decide to stop asking questions if it determines no significant ambiguity remains, aiming to optimize interaction cost.

Our Reproduction Approach: A Focused Stress Test

Objective: To reproduce the three interaction models (**M1**, **M2**, **M3**) and validate their performance on the KaggleDBQA dataset.



Key Methodological Choice: Instead of random sampling, we curated a test set of **30 questions pre-filtered and identified as 'ambiguous'** by an LLM evaluator.

Rationale: This creates a challenging benchmark specifically designed to stress-test the framework's core value proposition—resolving ambiguity. It provides a clearer signal of the interactive system's true value.

Code Availability: Our complete implementation is publicly available on GitHub.
https://github.com/Tongshanzhi/Sphinteract_reproduce.git

Engineering for Robust and Reproducible LLM Experiments

The reproduction required overcoming several practical engineering challenges to ensure reliable results:

Challenge 1: API Instability & Rate Limiting - Solution: Implemented robust error handling with exponential backoff and automatic fallback to alternative models (e.g., `gpt-4o-mini`, `gpt-4o`) to ensure experiment completion.

Challenge 2: Efficient Ambiguous Sample Selection - Solution: Developed a parallelized batch processor to check samples for ambiguity, with an early-stopping mechanism to halt processing once the target of 30 samples was found.

Challenge 3: Nuanced Performance Analysis - Solution: Implemented distinct status tags (`Initial Correct`, `Interactive Correct`, `Syntax Fix Correct`) to differentiate how each query was successfully resolved.

A Controlled Environment for Comparing Methods

Dataset

30 ambiguous questions selected from the KaggleDBQA benchmark.

Evaluation Settings

Both Zero-shot and 3-shot SQL generation.

Methods Compared

- **M1**: Baseline (Simple Feedback / Self-Correction)
- **M2**: Sphinteract (Clarification Questions)
- **M3**: Sphinteract + Early Stopping

Core Metrics

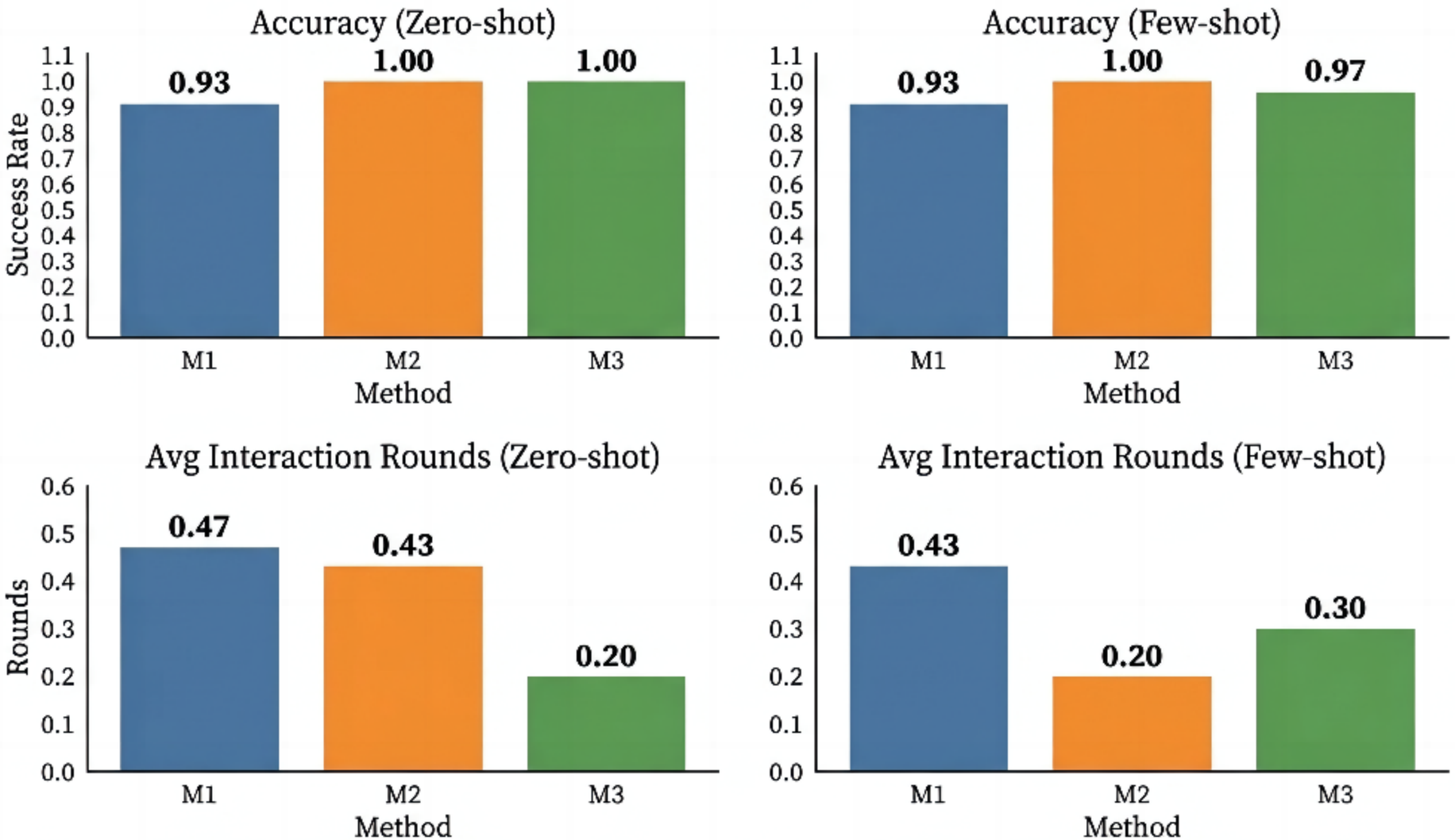
Execution Accuracy: The percentage of generated SQL queries that produce the correct result when executed.
Average Interaction Rounds: The mean number of user interactions required per question.

Feedback Simulation

An oracle using GPT-4o answered clarification questions based on the ground-truth SQL, a standard practice in interactive systems research.

Interactive Methods Achieve Near-Perfect Accuracy on Ambiguous Queries

Performance Overview



Analysis: Accuracy Gains vs. Interaction Efficiency

Finding 1 (Accuracy)

In the zero-shot setting, both M2 (Sphinteract) and M3 (Sphinteract+ES) achieve **100% accuracy** on this challenging ambiguous dataset, demonstrating a powerful capability to resolve errors. The M1 baseline stalls at 93.3%.

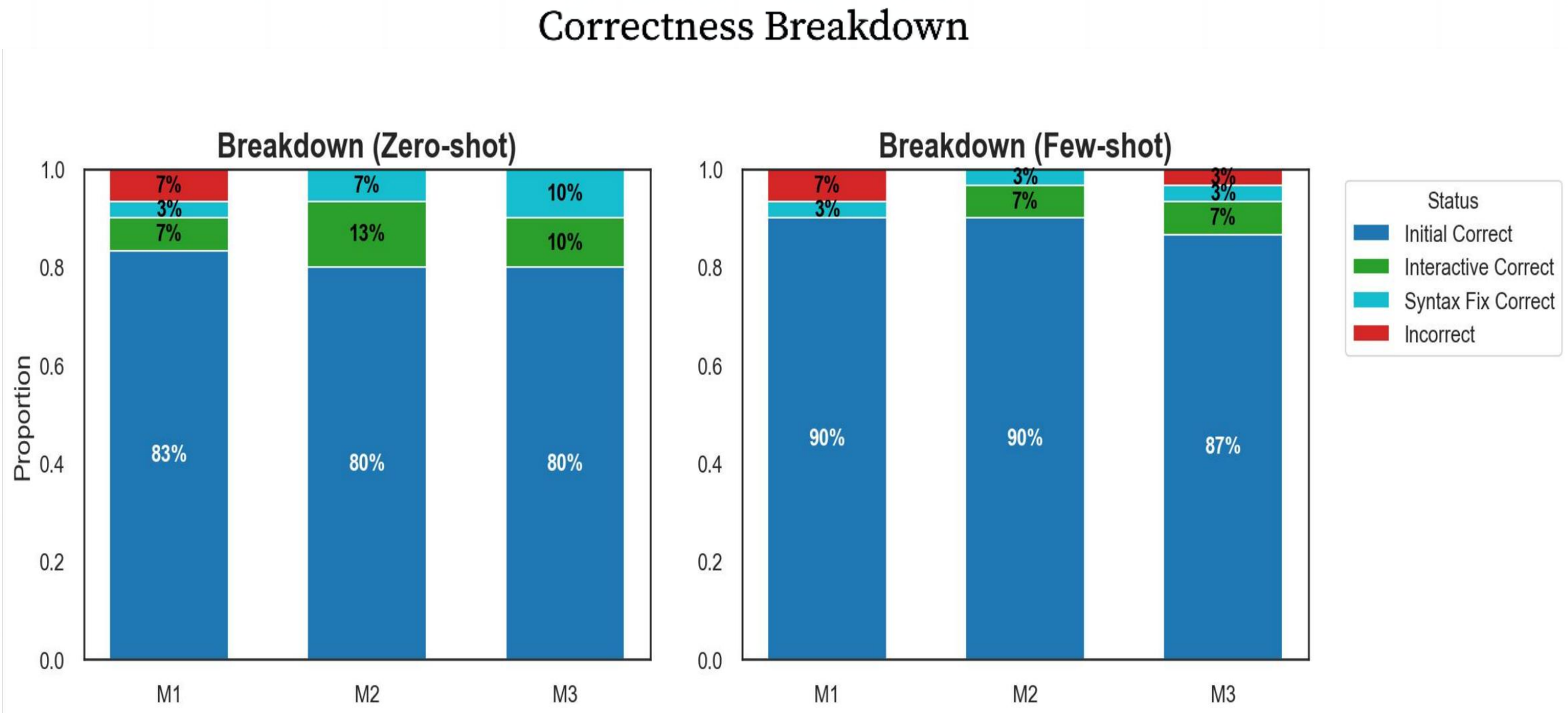
Finding 2 (Efficiency)

Few-shot examples primarily improve efficiency, not peak accuracy. For M2, 3-shot prompting **cut the average interaction rounds in half** (from 0.43 to 0.20).

Finding 3 (Optimal Trade-off)

M3 (Early Stopping) provides the best overall performance in the zero-shot setting, matching M2's perfect accuracy with the **lowest interaction cost** (0.20 rounds).

Breakdown of Correctness: Initial Success vs. Interactive Recovery

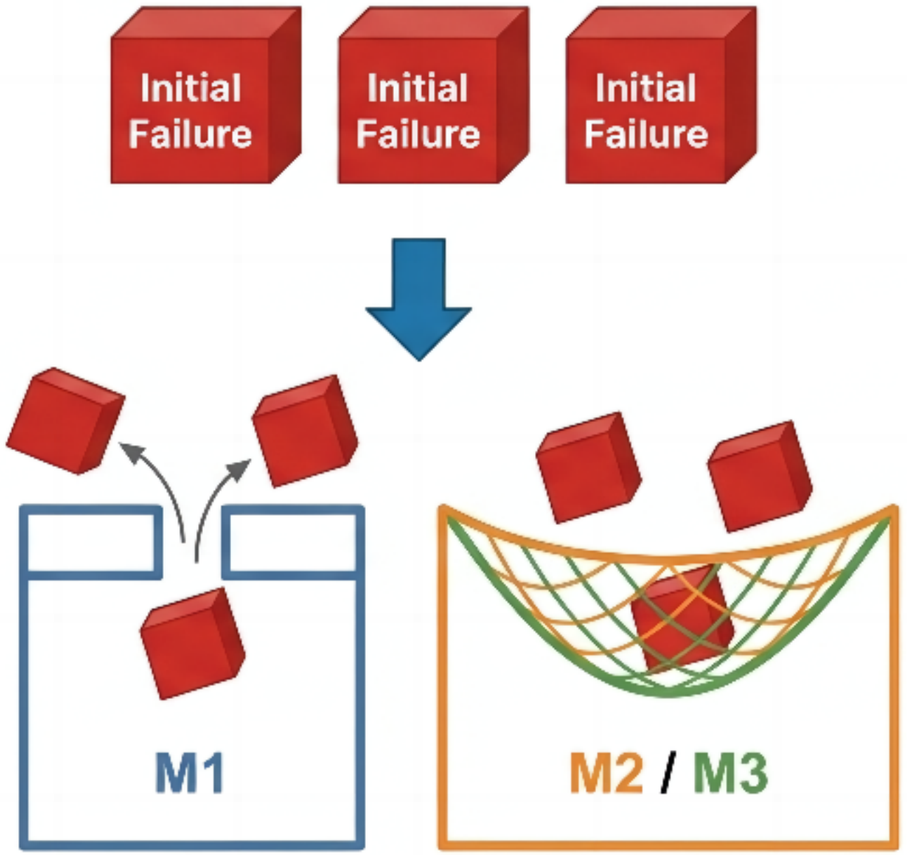


Sphinteract's True Strength is Recovering from Initial Failures

Observation 1: All methods achieve a high rate of correctness on the first attempt (`Initial Correct`), indicating a strong base LLM performance. For example, in zero-shot, M2 and M3 are initially correct for 24/30 samples.

Observation 2: The key differentiator is error recovery. In the zero-shot setting, M2 and M3 **successfully correct 100% of the initially failed queries** through interaction or syntax fixes.

Observation 3: In contrast, the M1 baseline fails to correct 2 out of 30 queries in both zero-shot and few-shot settings, leaving them as final errors.



Conclusion: The interactive mechanism is a highly reliable safety net for the most difficult, ambiguous cases where the model would otherwise fail.

Discussion: Validating the Paper's Claims and Gaining Practical Insights

CONFIRMATION

Our results 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 signal of an interactive system's value compared to random sampling,** which may be dominated by simpler, unambiguous queries.

ENGINEERING TAKEAWAY

Reproducing state-of-the-art LLM research is a significant engineering challenge. Success requires robust solutions for **API instability, efficient data processing,** and **precise metric tracking** to ensure ensure the reliability of experimental results.

Conclusion: Interaction is Key for Robust NL2SQL

Summary

The Sphinteract framework, particularly its SRA paradigm, provides a robust and effective methodology to align LLM-generated SQL with true user intent. Our reproduction on a challenging, challenging, ambiguity-focused dataset confirms its ability to achieve exceptional accuracy by systematically resolving ambiguities where non-interactive methods fail.

Future Work (from Zhao et al.)

- Improving the User Experience (UX) of interaction beyond multiple-choice questions.
- Learning user preferences over time to personalize clarification questions.
- Exploring asynchronous system architectures to improve performance and reduce user wait times.

Thank You

WU Mengfei
LIU Zichun
ZHONG Zhiyuan
SONG Jiafeng

Code Repository: https://github.com/Tongshanzhi/Sphinteract_reproduce.git