SQL-of-Thought Replication Report

1. Project Summary & Replication Content

1.1 Paper Background

This project is the supporting code for a research paper focusing on natural language-to-SQL conversion. Aiming at the pain points of existing methods—incapable of correcting syntactically valid but logically incorrect SQL queries and high error rates caused by unstructured reasoning—the paper proposes the SQL-of-Thought multi-agent framework. Its core is to assign specialized tasks to multiple agents, including schema matching, problem decomposition, query plan generation, SQL writing, and targeted error correction. Additionally, a SQL error taxonomy combined with full-process Chain-of-Thought (CoT) reasoning is designed, making error correction evidence-based and reasoning step-by-step.

1.2 Replication Objectives

In accordance with the project requirements, the error correction module was selected for replication to reproduce the ablation experiment of enabling/disabling the error correction loop in the paper (max-critic-attempts 3 vs 0). To simplify the project process, a small subset of the Spider dataset (20 samples) was used to verify the effectiveness of the error correction module.

2. Environment Configuration Description

2.1 Environmental Dependencies

Python Version: 3.10
Core Dependencies:
openai: For invoking the OpenAI GPT-4o-mini model API
json: For data parsing and result storage
pandas: For result statistics and analysis
sqlite3: For interacting with the Spider database
argparse: For command-line parameter parsing
typing: For code type annotation

2.2 Model and API Key Configuration

Base Model: OpenAI GPT-4o-mini
API Configuration: Set OPENAI_API_KEY and OPENAI_BASE_URL via environment variables or in the utils.py file to ensure normal model invocation.

2.3 Data Preparation

Dataset: Subset of the Spider development set (20 samples)

Supplementary Note: The dataset download link provided in the original paper is invalid. The Spider dataset used in this experiment is the complete version re-downloaded from the official channel, with 20 samples extracted as the evaluation subset.
Database Files: SQLite database files matching the Spider dataset, with paths adaptively configured for the local environment in utils.py.

2.4 Computing Resources

This experiment mainly relies on the OpenAI API for large model inference, with all heavy computing tasks completed on OpenAI's cloud server. Local computing resources are only used for data preprocessing, request scheduling, and result statistics, with no special GPU requirements. The experiment can be run on a regular PC/server with a standard CPU and memory ≥ 8GB, which is sufficient to meet the evaluation requirements of the 20-sample subset.

3. Replication Objectives & Metric Definitions

3.1 Replication Objectives

Replicate the paper's conclusion that the error correction module improves the quality of SQL generation, and compare the performance differences between the enabled and disabled states of the error correction function.

3.2 Core Metrics

1. Exact Match (EM): The proportion of generated SQL queries that are completely consistent with the ground truth in string form.

2. Valid SQL: The proportion of generated SQL queries with legal syntax that can be parsed by the database.

3. Execution Accuracy (EA): The proportion of generated SQL queries whose execution results are consistent with the ground truth.

Core Evaluation Metric: Execution Accuracy (EA)

4. Result Comparison: Reproduced Data vs. Paper Data

Metric	Reproduced Data (Error Correction Disabled)	Paper Data (Error Correction Disabled)	Reproduced Data (Error Correction Enabled)	Paper Data (Error Correction Enabled)
Execution Accuracy	85% 85% 90% 95%	85%	90%	95%
Valid SQL	95%	94-99%	100%	94-99%

Exact Match	60%	Not Provided	60%	Not Provided

4.1 Core Conclusion & Result Explanation

For the core metric of Execution Accuracy, the replication successfully verified the core conclusion of the paper. Enabling error correction improved the execution accuracy by 5 percentage points in the reproduced experiment, while the paper reported a 10-percentage-point improvement. The slightly lower overall performance in the replication is attributed to the use of a smaller sample set (20 samples) instead of the 100-sample set/the complete database in the original paper. As for the EM metric, the paper clearly states in the Evaluation Metrics section that EM is not an accurate evaluation indicator—since a single natural language question may correspond to multiple correct SQL queries, and LLMs may simplify variable assignment in subqueries. Therefore, only Execution Accuracy (EA) was used as the core evaluation indicator, with no EM results reported.

5. Modifications & Post-Modification Results

5.1 Modifications

To supplement the cost-benefit dimension not covered in the paper, three new metrics were added: Average Error Correction Rounds, Total LLM API Calls, and Total Time Consumption. The definitions are as follows:

1. Average Error Correction Rounds: The average number of self-correction attempts for each sample during SQL generation. Calculation: Total number of error correction attempts for all samples ÷ Total number of samples.

2. Total LLM API Calls: The total number of large model API invocations throughout the experiment. Calculation: Initial SQL generation (1 call per sample) + API calls in all error correction rounds.

3. Total Time Consumption: The total running time from the start of the experiment to the completion of processing all samples. Calculation: Experiment end time - Experiment start time.

5.2 Post-Modification Results

	Error Correction Disabled	Error Correction Enabled
Average Error Correction Rounds	0.0	0.5
Total LLM API Calls	80	100
Total Time Consumption (s)	155.3	188.6

5.3 Significance of the Results

1. Reflecting the Quality of Initial SQL Queries
The average error correction rounds is only 0.5, indicating that the initial SQL queries generated by the model are of high quality, and most samples can be used directly without correction. This verifies the effectiveness of the SQL-of-Thought framework in the initial generation stage.

2. Measuring the Cost-Effectiveness of the Error Correction Module
A 5% improvement in execution accuracy was achieved with only a 25% increase in LLM API calls and a 21.5% increase in time consumption. This demonstrates that the error correction module has high marginal benefit, realizing a significant performance improvement at a low cost.

6. Debug Log: Key Obstacles & Solutions

6.1 Import Error Fix

Problem: NameError: name 'AutoTokenizer' is not defined
Solution: Add relevant imports for HuggingFace Transformers (AutoTokenizer, AutoModelForCausalLM, torch)

6.2 Dependency Management Optimization

Problem: ValueError: Using a device_map requires accelerate (even when no local model is used)
Solution: Modify local model loading to lazy loading to avoid triggering dependency checks during import.
Problem: ModuleNotFoundError: No module named 'anthropic' / transformers
Solution: Set optional dependencies to on-demand imports, which will not affect the basic functions of the project if not installed.

6.3 Path Issue Fix

Problem: FileNotFoundError: ../../nl2sql_bugs.json
Solution: Use a stable path relative to the repository root directory and add file existence checks.
Problem: sqlite3.OperationalError: unable to open database file
Solution: Uniquely use the _spider_db_path() function to dynamically locate the database path.

6.4 API Configuration Optimization

Problem: openai.AuthenticationError: 401 - Incorrect API key provided
Solution: Support the configuration of OPENAI_BASE_URL and OPENAI_API_KEY to achieve compatibility with multiple OpenAI-compatible services.
Supported Backends: Official OpenAI, Alibaba Cloud DashScope, DeepSeek, etc.

7. Conclusion: Reproducibility Analysis

In this replication study, the improvement effect of the error correction module on the Valid SQL

generation rate and Execution Accuracy is fully reproducible, and the analysis framework of the newly added cost metrics in this study can be directly reused for relevant research. However, the experimental results of the original paper based on the high-performance model Claude Opus 3 and the conclusions drawn from the complete dataset cannot be fully reproduced. The main reasons are the objective differences in model capabilities and the different scales of the datasets used in the experiments, which make it difficult to reach the performance ceiling of the original paper in the replication process.