Text-to-SQL Evaluation Toolkit

Oktie Hassanzadeh
IBM
hassanzadeh@us.ibm.com

Yotam Perlitz
IBM
Y.Perlitz@ibm.com

Nhan Pham
IBM
nhp@ibm.com

Tanvi Kaple
IBM
Tanvi.Kaple@ibm.com

Karolina Źróbek
IBM
karolina.zrobek@ibm.com

Long Vu
IBM
lhvu@us.ibm.com

Michael Glass
IBM
mrglass@us.ibm.com

Dharmashankar Subramanian
IBM
dharmash@us.ibm.com

Mohammadreza Pourreza
University of Alberta
pourreza@ualberta.ca

Davood Rafiei
University of Alberta
drafiei@ualberta.ca

ABSTRACT

Text-to-SQL systems translate natural language questions into executable SQL queries, enabling intuitive access to structured data. While recent large language models have substantially improved generation quality, evaluating these systems remains a complex challenge: SQL semantics are subtle, multiple valid query formulations exist for the same question, and execution-based metrics are implemented inconsistently across the community. We demonstrate Text2SQL-Eval, an open-source, modular framework for rigorous evaluation of text-to-SQL systems. The toolkit provides a comprehensive suite of over twelve metrics, spanning execution accuracy, SQL syntactic equivalence, and LLM-as-judge scoring, together with integrated pipelines for inference, SQL execution against real databases, SQL profiling, and detailed error analysis. A web-based dashboard enables interactive exploration of benchmark results, cross-pipeline comparison, and per-record drill-down with live re-evaluation. The demonstration walks attendees through evaluating and comparing text-to-SQL pipelines on both established public benchmarks and new enterprise benchmarks. Attendees will learn to diagnose failure patterns and use LLM-as-judge to assess predictions where traditional metrics fall short.

PVLDB Reference Format:
Oktie Hassanzadeh, Yotam Perlitz, Nhan Pham, Tanvi Kaple, Karolina Źróbek, Long Vu, Michael Glass, Dharmashankar Subramanian, Mohammadreza Pourreza, and Davood Rafiei. Text-to-SQL Evaluation Toolkit. PVLDB, 19(12): 4582 - 4585, 2026.
doi:10.14778/3827998.3828071

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/IBM/text2sql-eval-toolkit.

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828071

1 INTRODUCTION

Text-to-SQL systems translate natural language questions into executable SQL queries, enabling non-technical users to query relational databases without writing SQL [6]. Recent advances in large language models (LLMs) have dramatically improved generation quality on benchmarks such as Spider [13] and BIRD [8], with state-of-the-art systems approaching human-level accuracy on some datasets [3, 9].

However, *evaluating* these systems remains a complex and often underappreciated challenge, for several reasons. First, **SQL semantics are subtle**: minor syntactic differences (e.g., column ordering, aliasing, use of `DISTINCT`) can be semantically equivalent or divergent, and simple string matching fails to capture this nuance [12]. Second, **execution-based metrics are inconsistent**: implementations differ in how they handle column sets, row ordering, and empty results, leading to fragmented comparisons [16]. Third, **multiple valid SQL formulations** exist for the same question; a query for "a list of customers" might correctly return IDs, names, or both, yet most frameworks support only a single ground-truth query. Finally, there is **no widely adopted, unified implementation** for text-to-SQL evaluation, forcing researchers to rely on ad hoc scripts that are difficult to reproduce and compare.

We present Text2SQL-Eval (the Text-to-SQL Evaluation Toolkit), an open-source, modular framework that addresses these challenges end-to-end and provides:

(1) A **comprehensive evaluation engine** with over twelve metrics spanning execution accuracy variants, SQL syntactic equivalence, and LLM-as-judge scoring, with support for multiple ground-truth queries per question.
(2) **Integrated pipelines** for LLM-based SQL inference (standard and agentic), execution against real databases (SQLite, PostgreSQL, MySQL, DB2), SQL profiling, and error analysis.
(3) An **interactive web dashboard** for browsing benchmarks, comparing pipelines, performing cross-pipeline error analysis, and running live per-record evaluations.

(4) A **standardized, reproducible implementation** released as open source and validated across multiple established benchmarks, providing a unified foundation for text-to-SQL evaluation in the research community.

The toolkit is released under the Apache 2.0 license and is available as a Python package on PyPI and on GitHub.

2 BACKGROUND AND RELATED WORK

Text-to-SQL benchmarks. Spider [13] introduced cross-database evaluation with complex queries over 200 databases, and remains a widely used benchmark. BIRD [8] emphasizes large, realistic databases with noisy values and external knowledge. Beaver [1] targets enterprise scenarios with MySQL databases, Archer [14] focuses on arithmetic, commonsense, and hypothetical reasoning, and Spider 2.0 [7] extends evaluation to enterprise workflows.

Evaluation methodology. Early work relied on exact string match of SQL queries, which proved too restrictive. Execution accuracy, comparing the result sets of predicted and ground-truth queries, became the dominant metric [13], but implementations vary in how they handle exact vs. subset column matches and empty results. Zhong et al. [16] proposed test-suite accuracy, executing queries against multiple database instances to reduce false positives. Despite these advances, researchers still rely on benchmark-specific scripts (e.g., Spider's `eval.py`, BIRD's evaluation code) that implement a single metric and cannot be easily extended; other pipelines remain unpublished or tightly coupled to specific frameworks. To our knowledge, no prior work provides a standalone, extensible toolkit that unifies multiple metric families, supports diverse database engines, and offers both programmatic and interactive interfaces.

LLM-as-judge. The idea of using a language model to evaluate outputs has gained traction in NLP [15]. In text-to-SQL, LLM-as-judge can assess whether a predicted query is semantically correct even when it differs from the ground truth in structure or in the columns returned, offering a complementary signal to execution-based metrics.

Text2SQL-Eval differs from prior work by providing not a new benchmark or model, but a *unified evaluation framework* that integrates multiple metric families, supports diverse benchmarks and database engines, and offers both programmatic APIs and an interactive dashboard for exploratory analysis.

3 SYSTEM OVERVIEW

Figure 1 shows the toolkit architecture. The system is organized as a set of modular components connected through a shared data format (JSON prediction files), allowing users to adopt individual components or the full pipeline.

3.1 Benchmark Data Catalog

The toolkit ships with pre-configured benchmark definitions for BIRD Mini-Dev [8] (SQLite and PostgreSQL), Spider Dev and Spider Realistic [13], Beaver [1] (MySQL), and Archer [14] (SQLite). Each benchmark is defined by a JSON configuration specifying question data, schemas, prediction files, and database connection parameters. Adding a new benchmark requires only a JSON entry; no code changes are needed. The catalog supports SQLite (local files), PostgreSQL MySQL, and DB2 databases, with extensibility for additional engines.

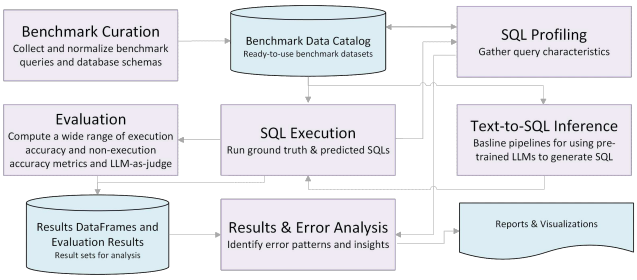

Figure 1: Architecture of the toolkit showing the modular pipeline from benchmark curation through evaluation and error analysis.

3.2 Text-to-SQL Inference

Two inference pipelines are provided as baselines. The *standard pipeline* performs single-shot LLM inference: it constructs a prompt from the question, schema, and optional evidence, sends it to an LLM, and extracts the predicted SQL. The *agentic pipeline* extends this with multi-attempt generation, where the agent executes candidate SQL, observes errors, and iteratively refines the prediction. Both pipelines run asynchronously with configurable concurrency, support multiple LLM providers, and record inference metadata (token usage, latency) for downstream analysis. The built-in pipelines are reference baselines: any text-to-SQL system can be evaluated by supplying predictions in the toolkit's JSON format (question id, predicted SQL, optional metadata).

3.3 SQL Execution Engine

The execution component runs both ground-truth and predicted SQL against benchmark databases, capturing result DataFrames with concurrent execution, per-query timeouts, and database-specific dialect handling. For benchmarks with multiple ground truths, all are executed. A *logic SQL* mode substitutes the ground-truth `SELECT` clause into the predicted query, enabling evaluation of query logic independent of column selection.

3.4 Evaluation Engine

The evaluation engine is the core of the toolkit. Given the result DataFrames from the execution engine, it computes the metrics listed in Table 1.

Execution accuracy variants. The baseline execution accuracy requires an exact match between predicted and ground-truth result sets, respecting `ORDER BY` semantics. *Non-empty* execution accuracy restricts evaluation to non-empty results, avoiding trivially correct empty matches. *Subset non-empty* further relaxes the comparison to allow missing or extra columns, accommodating questions with ambiguous column requirements (e.g., "list customers" could validly return IDs, names, or both). The BIRD variant follows BIRD's set-based comparison rules for leaderboard compatibility. Unlike the widely used Spider evaluation script, which attempts to parse SQL to identify columns but fails on many complex queries, our toolkit

Table 1: Evaluation metrics provided by the toolkit.

Metric	Description
Execution-based	
Execution Accuracy (EX)	Exact result set match
Non-Empty EX	Match when both results non-empty
Subset Non-Empty EX	Match allowing extra/missing columns
BIRD EX	Set-based match per BIRD rules
SQL syntactic equivalence	
SQL Exact Match (EM)	Exact string match
SQLGlot EM	AST-based equivalence via SQLGlot [10]
SQLGlot Optimized EM	Equivalence after query optimization
SQLParse EM	Token-based equivalence
Syntactic EM	Any of the above holds
Parsability	
SQLGlot Parsable	Predicted SQL parses with SQLGlot
SQLParse Parsable	Predicted SQL parses with sqlparse
LLM-based	
LLM-as-Judge Score	LLM assessment of correctness (0/0.5/1)
Efficiency	
Inference Time	LLM inference latency (ms)
Execution Time	SQL execution time (ms)
Token Usage	Prompt, completion, and total tokens

compares result sets directly and supports established benchmarks so that users can compare against prior published work.

SQL syntactic equivalence. The toolkit computes multiple forms of SQL equivalence using SQLGlot [10] (AST-based, with and without optimization) and sqlparse (token-level), useful for understanding whether models generate structurally similar queries even when execution results differ.

LLM-as-judge. When execution-based metrics are inconclusive (e.g., a prediction returns a correct subset of columns) or no ground truth is available, the toolkit invokes an LLM judge that receives the question, SQL pairs, and result DataFrames, returning a score (1/0.5/0) with an explanation. The judge configuration (model, prompt, parameters) is editable via YAML files in the dashboard; to reduce cost, the judge is skipped when subset execution accuracy already confirms a match.

Multiple ground truths. Each benchmark question can have multiple valid ground-truth SQL queries; the toolkit evaluates against all and reports the maximum score. This matters because structurally different queries can be equally valid: rows and columns may be transposed, different operators may yield slightly different results due to rounding, or joins may follow alternative paths. We show examples where relying on a single ground truth unfairly penalizes correct predictions.

Efficiency and resource tracking. The toolkit records per-record inference latency, SQL execution time, and token usage (prompt, completion, and total tokens) when available from the prediction metadata. These efficiency metrics are displayed alongside correctness metrics in the dashboard and included in aggregated summaries.

Summary and aggregation. Per-record metrics are aggregated into per-pipeline summaries with averages and standard deviations. Failed predictions (execution errors, timeouts, inference failures) are counted as incorrect, ensuring metrics reflect real-world reliability.

3.5 SQL Profiling and Error Analysis

The profiling component uses SQLGlot to extract structural features (tables, joins, aggregations, subqueries, nesting depth) and assigns difficulty categories, enabling stratified evaluation by query complexity. The error analysis module identifies failure patterns, generates per-category breakdowns, and exports detailed reports with examples of failed predictions (including prompts, SQL pairs, and result tables) to Markdown files for offline review.

4 INTERACTIVE DASHBOARD

The toolkit includes an optional web dashboard (Figure 2) providing a graphical interface for all major evaluation workflows.

Benchmark overview. The home screen (Figure 2a) displays benchmark tiles showing name, database type, number of records, and evaluated pipelines. Selecting a tile opens per-pipeline metrics in a sortable, paginated table, optionally stratified by query difficulty category.

Pipeline comparison. Users select two pipelines and two metrics to generate cross-pipeline confusion matrices, with timing and token-usage deltas; clicking any cell drills into the underlying records.

Error analysis. The analysis view (Figure 2b) supports full-text search, metric-condition filtering (e.g., `EX = 0` and `llm_score = 1`), and cross-pipeline disagreement filters. Selecting a record shows question, SQL pairs with syntax highlighting, result tables, and LLM-as-judge verdicts. Users can edit and re-execute predicted SQL directly from the interface.

Evaluation playground. The playground lets users select a question, view cached predictions, and submit new SQL for live evaluation with the full metric suite, including LLM-as-judge.

LLM judge configuration. A built-in YAML editor allows modifying the LLM-as-judge model, prompt template, and generation parameters without editing files on disk.

The dashboard communicates through a REST API that also supports programmatic access for CI/CD integration.

5 DEMONSTRATION PLAN

The demonstration runs on a laptop with the Text2SQL-Eval dashboard in a web browser (Python 3.11+; internet needed only for LLM-as-judge calls). Pre-computed results for multiple benchmarks and pipelines are loaded, including established public benchmarks and new enterprise benchmarks, with proprietary models such as Gemini-3-flash [4] and open models such as GPT-OSS-120B [11], Llama [2], and Granite [5]; database files are available locally for live SQL execution. The demonstration proceeds through the following scenarios.

Scenario 1: Benchmark exploration. We begin at the dashboard home screen, showing tiles for six pre-loaded benchmarks (Figure 2a). We select BIRD Mini-Dev (500 questions, SQLite) and review pipeline-level metrics. We then switch from an established public benchmark to a new enterprise benchmark to contrast evaluation settings and failure modes.

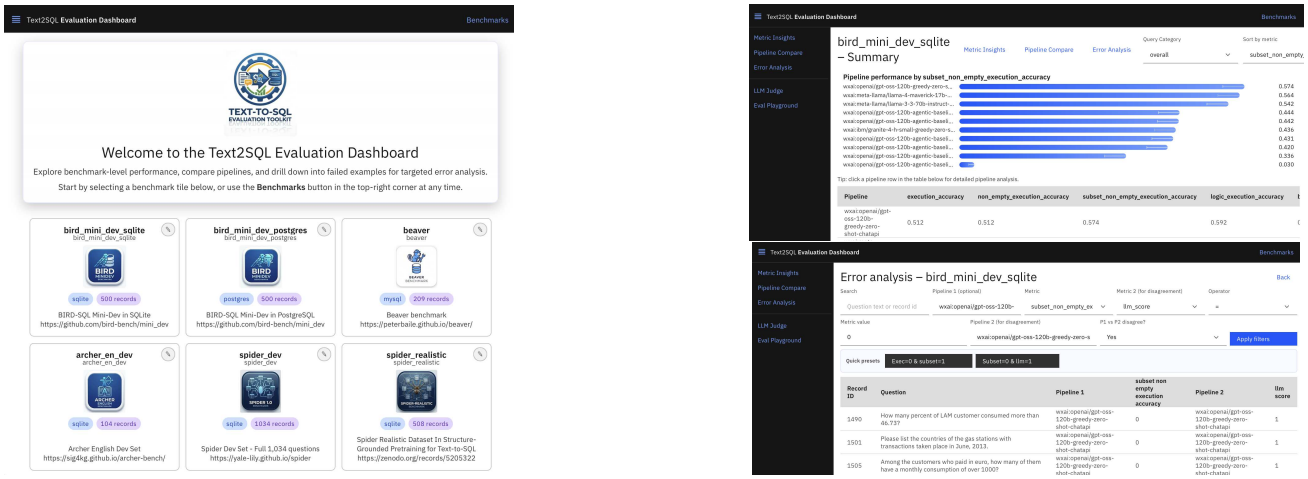

Figure 2: The Text2SQL-Eval dashboard: (a) home screen benchmark overview; (b) summary and error-analysis views.

Scenario 2: Error analysis and pipeline comparison. We compare two pipelines, a standard zero-shot baseline and an agentic pipeline with iterative self-correction, on the BIRD benchmark. The comparison view reveals the confusion matrix: questions where the agentic pipeline recovers from initial failures, and cases where additional attempts introduce regressions. Token-usage and inference-time deltas are also displayed, showing the cost/accuracy trade-off. We inspect specific examples from each quadrant.

Scenario 3: Metric comparison and LLM-as-judge. Using the same confusion-matrix view, we compare execution accuracy against subset non-empty execution accuracy. This highlights predictions that score 0 under strict execution accuracy but 1 under the relaxed variant, for example when a query returns the correct rows but includes an extra column or omits one that the ground truth selects. We then filter for records where execution accuracy is 0 but the LLM-as-judge score is 1, identifying predictions that are semantically correct but differ in columns or row orderings. We drill into specific examples, examining the SQL, result tables, and LLM explanation, illustrating the complementary value of multiple metrics for nuanced evaluation.

Scenario 4: Interactive evaluation. Using the evaluation playground, we select a question, view the ground-truth SQL, and manually write an alternative query, running the evaluation live to see how the full metric suite scores it, toggling LLM-as-judge on and off to observe its effect.

Audience interaction. Attendees can browse benchmarks and pipelines, write their own SQL predictions for any question and evaluate them in real time, explore cross-pipeline disagreements, and experiment with LLM-as-judge configurations. Attendees interested in the API can also see a short code example that evaluates a prediction in three lines of Python.

6 CONCLUSION

We present the Text-to-SQL Evaluation Toolkit, an open-source framework for comprehensive and reproducible evaluation of text-to-SQL systems. The toolkit unifies execution-based, syntactic, and LLM-based metrics in a modular architecture that supports multiple benchmarks and database engines, and its interactive dashboard enables researchers and practitioners to compare pipelines, diagnose errors, and iteratively improve their systems. It is publicly available on PyPI and GitHub under the Apache 2.0 license. Future work includes more efficiency metrics (such as BIRD's Valid Execution Score), more complex agentic LLM-based evaluation, support for additional database engines, semantic parsing metrics, and automated benchmark curation tools.

REFERENCES

[1] Peter Baile Chen, Fabian Wenz, Yi Zhang, et al. 2024. BEAVER: An Enterprise Benchmark for Text-to-SQL. *arXiv preprint arXiv:2409.02038* (2024).

[2] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).

[3] Dawei Gao, Haibin Wang, Yaliang Li, et al. 2024. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. In *PVLDB*, Vol. 17. 1132–1145.

[4] Gemini Team, Google. 2026. Gemini 3 Flash. Google DeepMind Technical Report.

[5] IBM Granite Team. 2024. Granite Code Models: A Family of Open Foundation Models for Code Intelligence. *arXiv preprint arXiv:2405.04324* (2024).

[6] Marios Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A Survey on Deep Learning Approaches for Text-to-SQL. *The VLDB Journal* 32, 4 (2023), 905–936.

[7] Fangyu Lei, Jixuan Chen, Yuxiao Ye, et al. 2024. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. *arXiv preprint arXiv:2411.07763* (2024).

[8] Jinyang Li, Binyuan Hui, Ge Qu, et al. 2024. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. In *NeurIPS*, Vol. 36.

[9] Xinyu Liu, Shuyu Shen, Boyan Li, et al. 2024. A Survey of NL2SQL with Large Language Models: Where are we, and where are we going? *arXiv preprint arXiv:2408.05109* (2024).

[10] Toby Mao. 2024. SQLGlot: SQL Transpiler and Optimizer. https://github.com/tobymao/sqlglot Accessed: 2026-07-07.

[11] OpenAI. 2025. gpt-oss-120b & gpt-oss-20b Model Card. *arXiv preprint arXiv:2508.10925* (2025).

[12] Mohammadreza Pourreza and Davood Rafiei. 2023. Evaluating Cross-Domain Text-to-SQL Models and Benchmarks. In *Proc. EMNLP*. 1601–1611.

[13] Tao Yu, Rui Zhang, Kai Yang, et al. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proc. EMNLP*. 3911–3921.

[14] Danna Zheng, Mirella Lapata, and Jeff Z. Pan. 2024. Archer: A Human-Labeled Text-to-SQL Dataset with Arithmetic, Commonsense and Hypothetical Reasoning. In *Proc. EACL*. 94–111.

[15] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. 2024. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *NeurIPS*, Vol. 36.

[16] Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic Evaluation for Text-to-SQL with Distilled Test Suites. In *Proc. EMNLP*. 396–411.