Text-to-SQL Benchmarks are Broken: An In-Depth Analysis of Annotation Errors

Tengjun Jin
University of Illinois (UIUC)

Yoojin Choi
University of Illinois (UIUC)

Yuxuan Zhu
University of Illinois (UIUC)

Daniel Kang
University of Illinois (UIUC)

ABSTRACT

Text-to-SQL has been widely studied in both academia and industry. Researchers have developed a series of benchmarks to evaluate different techniques and provide insights for further improvement. However, existing text-to-SQL datasets contain substantial annotation errors, ranging from incorrect ground-truth to ambiguous questions, compromising the reliability of their results. In this work, we present a comprehensive analysis of two widely used text-to-SQL benchmarks, BIRD and Spider 2.0-Snow, and find error rates of 52.8% and 66.1%, respectively. By re-evaluating five leading open-source methods from the BIRD leaderboard on our corrected benchmark, we observed performance changes ranging from −3% to 31% in relative terms. This results in notable shifts in their performance ranking, with changes of up to three positions. The significant changes in performance and ranking highlight the unreliability of current text-to-SQL benchmarks. We advocate for the development of higher-quality text-to-SQL benchmarks and more effective annotation pipelines.

1 INTRODUCTION

Benchmarks actively drive innovation in database research by providing clear targets for optimization [3], standardizing test beds for fair comparison [18], and motivating evolution of new techniques [4]. As text-to-SQL systems become increasingly important for data analytics and database-driven applications [8–10, 22, 23, 25, 29, 33], researchers and practitioners have introduced a variety of text-to-SQL benchmarks [6, 11, 13, 14, 16, 17, 26, 35, 37, 38].

Unfortunately, current text-to-SQL benchmarks have yet to reach the level of reliability achieved by established standards such as TPC-H [21]. For example, in BIRD [16], prior work reported annotation errors in 32% of problems in the mini development set (Mini-Dev) [2] and 49% of financial domain problems [32], with the latter resulting in up to 17.6% underestimation of performance [32].

To understand how we can build a reliable text-to-SQL benchmark with minimal annotation errors, we conduct an in-depth analysis of text-to-SQL problems and their annotation error patterns. Each problem in the text-to-SQL benchmark consists of three components: a natural language input $\mathcal{T}$,[1] an example database $\mathcal{D}$, and a ground-truth SQL query $\mathcal{Q}$. Annotation errors can occur within any individual component or at the intersections between components. Our analysis over two widely used benchmarks, BIRD [16] and Spider 2.0-Snow [14], identifies four error patterns:

E1. Mismatches between semantics of $\mathcal{Q}$ and intended logic of $\mathcal{T}$.
E2. Mismatches between semantics of $\mathcal{Q}$ and $\mathcal{D}$ due to the limited understanding of the data and the schema.
E3. Mismatches between semantics of $\mathcal{Q}$ and domain knowledge relevant to $\mathcal{T}$, or misannotated domain knowledge in $\mathcal{T}$.
E4. Ambiguity in $\mathcal{T}$.

Figure 1a illustrates a misuse of a Snowflake function (**E1**). Figure 1b presents an incorrect annotation from BIRD due to the annotator's lack of domain knowledge (**E3**). We present additional representative examples for each error pattern in Section 4.

Beyond these examples, our analysis reveals a wider range of annotation errors in text-to-SQL benchmarks than previously recognized. In BIRD Mini-Dev [16], we find that 52.8% of the problems contain annotation errors, which is 16.7% higher than the previously reported rate of 36.1% [2]. We also inspect Spider 2.0-Snow [14], a recently released benchmark whose annotation error has not been investigated before. Among its 121 problems with open-sourced gold SQL queries, we identify an annotation error rate of 66.1%.

These annotation errors cause severe misestimation of agents' performance. By re-evaluating five leading text-to-SQL agents on a random sample of 100 out of 1534 problems from BIRD Dev, we identified significant performance changes ranging from −2% to 19% in absolute terms and ranking position changes ranging from −2 to 3. For instance, we find the performance of CHESS [29], an agent previously ranked 4th among our five selected agents, increases from 62% to 81% after corrections and moves to 1st place.

Our findings demonstrate that annotation errors remain a barrier to robust benchmarking in text-to-SQL, with direct implications for leaderboard validity. We hope our analysis offers valuable insights for reducing annotation errors and for constructing higher-quality text-to-SQL benchmarks. Furthermore, we advocate for more effective annotation pipelines, potentially leveraging AI agents, to address annotation challenges and enhance benchmark reliability.

2 RELATED WORK

Text-to-SQL benchmarks and methods. Researchers have proposed a wide range of datasets and benchmarks to evaluate text-to-SQL systems, including single-domain benchmarks [11, 26, 35] and more recent cross-domain benchmarks [6, 13, 14, 16, 17, 30, 31, 37, 38]. To address the challenges in text-to-SQL, prior work has explored diverse techniques, including multi-modular agent architectures [23, 29, 34], supervised fine-tuning [10, 15], and reinforcement learning [25, 28].

[1]Input $\mathcal{T}$ consists of a user question and, optionally, external knowledge.

 16th Annual Conference on Innovative Data Systems Research (CIDR '26). January 18-21, Chaminade, USA

Annotated user query: Can you provide a daily weather summary for July 2019 within a 5 km radius of latitude 26.75 and longitude 51.5? ...

Annotated SQL

```
... ST_POINT(26.75, 51.5) ...
```

Issue: Snowflake's ST_POINT expects (longitude, latitude). The query passes ST_POINT(26.75, 51.5), which inverts the order.

(a) Problem sf_bq291 of Spider 2.0-Snow. Annotators mistakenly swapped the order of longitude and latitude in `ST_POINT` (E1).

Annotated user query: Which state special schools have the highest number of enrollees from grades 1 through 12?

Annotated external knowledge: State Special Schools refers to DOC = 31; Grades 1 through 12 means K-12

Annotated SQL

```
SELECT T2.School FROM frpm AS T1 INNER JOIN schools AS
T2 ON T1.CDSCode = T2.CDSCode WHERE T2.DOC = 31
ORDER BY T1."Enrollment␣(K-12)" DESC LIMIT 1
```

Issue: 'Enrollment (K-12)' includes kindergarten and is therefore not equivalent to enrollment for grades 1 through 12.

(b) Problem 46 of BIRD Dev. Annotators had limited understanding of "K-12" (E3).

Figure 1: Two examples from existing text-to-SQL benchmarks that demonstrate incorrect annotations.

Errors in benchmarks. Many studies have highlighted significant issues with existing benchmarks [2, 12, 19, 24, 32, 36, 39]. In particular, several works have examined the presence of annotation errors in widely-used text-to-SQL benchmarks such as Spider and BIRD [2, 19, 24, 32]. The re-evaluations conducted on corrected datasets demonstrate that current model performance is frequently underestimated due to these data quality issues [24, 32].

3 METHODS

In this section, we first introduce the benchmark used in our analysis and then detail our examination process.

3.1 Data Settings

We chose two widely-studied text-to-SQL benchmarks: BIRD [16] and Spider 2.0-Snow [14]. For BIRD, following prior work [2], we focused on BIRD Mini-Dev, an official subset of the development set comprising 498 examples. Spider 2.0-Snow contains 547 examples in total, with gold queries publicly available for 121 of them.[2] We evaluated the correctness of these 121 examples.

3.2 Examination of Annotation Errors

We measure the annotation error rate of each selected benchmark as the proportion of examples that exhibit at least one of the four error patterns (**E1**–**E4**). For each example, we inspected the example by referencing the database schema and executing the corresponding SQL queries. For questions that require domain-specific knowledge, we used relevant online resources and LLMs to cross-check and verify the accuracy of the information used in the query. In addition, we executed and examined subqueries independently to ensure the correctness of intermediate results before verifying the final results.

[2]The Spider 2.0 team updated questions to resolve ambiguities on 2025-07-13. Since our examination preceded this update, we used the earlier version: https://github.com/xlang-ai/Spider2/blob/main/spider2-snow/spider2-snow-0713.jsonl

Table 1: Error pattern distribution in BIRD Mini-Dev and Spider 2.0-Snow. Each example may contain multiple errors.

Error pattern	BIRD Mini-Dev	Spider 2.0-Snow
E1	77 (29.3%)	39 (48.8%)
E2	152 (57.8%)	44 (55%)
E3	28 (10.7%)	9 (11.3%)
E4	78 (29.7%)	20 (25%)

Table 2: Examples for each error pattern in BIRD Mini-Dev. The bracketed question IDs were marked as noisy in prior work, but for different types of errors.

Question IDs	Error Pattern	Explanation
92, 149	E1	Incorrect usage of `BETWEEN ... AND ...`.
12, 36, 40, 50, (17)	E2	Does not include the restriction `rtype = 'S'`.
1376, 1378, 1403	E2	No aggregation over events.
32, 46, 62	E3	"K-12" is not equivalent to grades 1-12.
862, 877, 881, 954	E3	Drivers with a "+n Lap" status also finished the race.
1175	E4	"Doctor's diagnosis" could come from either Examination.Diagnosis or Patient.Diagnosis.

4 ANNOTATION ISSUE ANALYSIS

We find that the annotation error rate is 52.8% for BIRD Mini-Dev and 66.1% for Spider 2.0-Snow. As shown in Table 1, **E2** is the most frequent error pattern in both benchmarks (57.8% in BIRD Mini-Dev and 55% in Spider 2.0-Snow).

4.1 Errors in BIRD Mini-Dev

We identified 102 additional erroneous examples in BIRD Mini-Dev compared to prior work [2], resulting in an overall error rate of 52.8%. We provide representative, newly discovered examples of each error pattern in Table 2 and discuss them in more detail below.

*Examples of **E1**.* Annotators misused the `BETWEEN ... AND ...` keyword for strict inequality predicates (> or <).

*Examples of **E2**.* For queries related to the `california_schools` database, although the questions ask about school statistics, the annotated queries omit the predicate `rtype = 'S'`, which distinguishes schools from districts. Further examination of the schema shows that annotators labeled the `rtype` column as "unuseful," whereas our analysis of the database confirms that this column is precisely what differentiates schools from districts.

*Examples of **E3**.* The annotators misinterpreted the meaning of "K-12" in three examples and "+n Lap" in four examples, highlighting their unfamiliarity with domain-specific terminology from areas.

*Examples of **E4**.* Questions often exhibit ambiguity when multiple tables share the same column name, yet neither the question nor the provided external knowledge clarifies the intended reference.

Table 3: Examples for each error pattern in Spider 2.0-Snow.

Question IDs	Error Pattern	Explanation
sf_bq263, sf_bq271, sf_bq273, sf_bq294	E1	The function TO_TIMESTAMP(end_date) sets the time to the start of the end date (00:00:00.000), instead of to the end of the day (23:59:59.999) as required.
sf_bq012, sf_bq050, sf_bq193, sf_bq209, sf_bq455, sf_local355	E1	The SQL queries lack the required filters specified by the question.
sf_bq099, sf_bq248, sf_bq422, sf_local263	E2	Using `JOIN` or `FLATTEN` operators inflates row counts; the query lacks `DISTINCT` for deduplication.
sf_bq052, sf_bq246	E3	Forward citations are miscalculated.
sf_bq017, sf_bq068, sf_bq099, sf_bq182, sf_bq193, sf_bq222, sf_bq223	E4	The gold SQL queries replace quotation marks, which are not specified in the question.

4.2 Errors in Spider 2.0-Snow

We identified annotation issues in 80 out of the 121 examples with released gold queries in Spider 2.0-Snow, resulting in a error rate of 66.1%. We present representative examples for each error pattern in Table 3, and discuss the details below:

*Examples of **E1**.* The annotators used TO_TIMESTAMP(end_date), which casts a date to the timestamp of the start of the day rather than the end. Therefore, the gold queries miss rows where the time occurs the end date but after 00:00:00.

*Examples of **E2**.* The annotators did not verify the intermediate results after the `JOIN` or `FLATTEN` operations, which resulted in row inflation in four examples.

*Examples of **E3**.* We identified that in two queries involving forward citation calculation, the annotators used incorrect joins. They mistakenly joined on `cited."patent_id" = apps."patent_id"` instead of `cited."citation_id" = apps."patent_id"`.[3]

*Examples of **E4**.* We identified seven questions with ambiguous output formats that require replacing quotation marks in the SQL queries. If queries generated by agents include quotation marks, they will be judged as incorrect.

5 EXPERIMENTS

In this section, we assess the impact of annotation errors in text-to-SQL benchmarks on agent performance by evaluating five leading open-source text-to-SQL agents from the BIRD leaderboard.

5.1 Experimental Settings

Dataset. We randomly sampled 100 examples (62 simple, 28 moderate, and 10 challenging) from the BIRD development set.

Metrics. Following prior work [16], we used Execution Accuracy (EX) and ranking as our metrics.

Text-to-SQL Agents. We selected five agents with the highest EX on the BIRD leaderboard that have publicly available code and can be reproduced.[4] We summarize their configurations below:

Issues in problem 985 of the BIRD Dev set:
(1) Ordering by the text "time" column can misrank values lexicographically. **(E2)**
(2) It is unclear whether to output the driver's name or ID. **(E4)**
(3) There are multiple races named "French Grand Prix". **(E4)**

Fixed user query: Among all the drivers who have participated in any year of the French Grand Prix throughout its history, which driver recorded the slowest time on the 3rd lap in any of these races? Please specify the driver id.

Modification to the database
INSERT INTO "drivers" ("driverId", "driverRef", "forename", "surname", "url") VALUES (1111, "test", "test", "test", "test");
INSERT INTO "lapTimes" ("raceId", "driverId", "lap", "time", "milliseconds") VALUES (25, 1111, 3, "11:00.365", 660365);

Fixed SQL

```
SELECT T1.driverId FROM lapTimes AS T1 INNER JOIN races AS T2
ON T1.raceId = T2.raceId WHERE T2.name = 'French␣Grand␣Prix'
AND T1.lap = 3 ORDER BY (CAST(substr(lapTimes.time, 1, instr(
lapTimes.time, ':')-1) AS INTEGER) * 60 * 1000) + (CAST(substr(
lapTimes.time, instr(lapTimes.time, ':')+1) AS REAL) * 1000)
DESC LIMIT 1;
```

Figure 2: Corrected problem 985 in BIRD Dev.

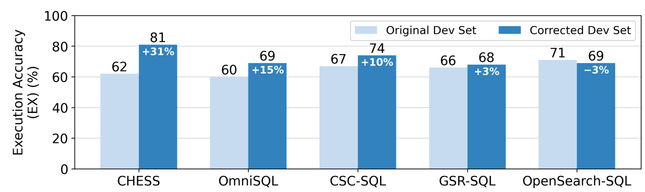

Figure 3: Execution accuracy of agents on original and corrected development subsets.

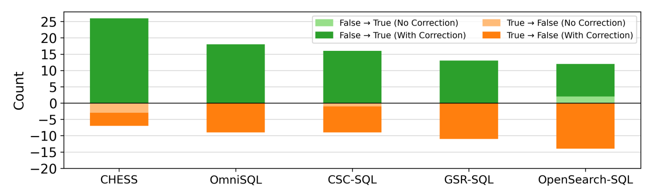

Figure 4: Number of examples with correctness changes between original and corrected development subsets.

(1) **CSC-SQL [28]**: We used Qwen2.5-Coder-7B-Instruct [27] as the merge model and XiYanSQL-QwenCoder-32B-2412 [10] as the generation model.
(2) **OpenSearch-SQL [34]**: We used GPT-4o-0513 [20] for SQL generation and bge-m3 [5] for retrieval.
(3) **OmniSQL [15]**: We evaluated it using greedy decoding.
(4) **GSR-SQL [1]**: We ran GSR-SQL with GPT-4o (2024-11-20).
(5) **CHESS [29]**: We configured CHESS to include three of its four specialized agents, Information Retriever, Candidate Generator, and Unit Tester, with GPT-4o (2024-08-06).

5.2 Benchmark Correction

Following the error patterns identified in our analysis (**E1–E4**), we audited all examples of sampled dataset and found that 48% of

[3] In sf_bq128, the annotators used the correct join key.
[4] We chose agents based on leaderboard rankings as of 2025-06-29.

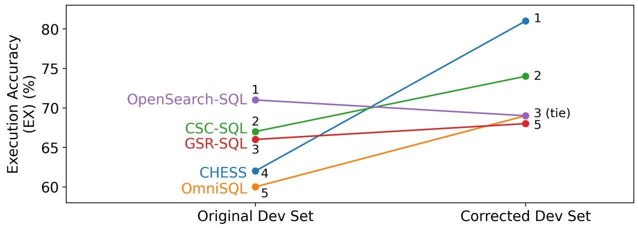

Figure 5: Agent ranking changes from original to corrected Dev subsets. CHESS advances from fourth to first place.

them contained errors. Following prior work [2], we corrected all identified issues by revising natural language questions, external knowledge, and ground truth SQL queries to generate a corrected development subset. In addition, we observed instances where erroneous queries produced the same results as the corrected ground truth queries. To enable more accurate evaluation, we modified the databases to introduce distinguishing cases. As illustrated in Figure 2, to address the issues in Problem 985 of the BIRD Dev set, we corrected both the user query and the SQL. Furthermore, we modified the database to address the original data could not expose errors related to "misranking values lexicographically."

5.3 Evaluation Results

We executed all five agents on the original and corrected development subsets to evaluate how annotation errors affect agent performance. We discuss the change of execution accuracy and rankings of all evaluated agents in the following paragraphs.

Execution Accuracy (EX). We analyzed the change in execution accuracy for all five evaluated agents. As shown in Figure 3, the relative changes in execution accuracy across the agents range from −3% to 31%. Four agents (CHESS, OmniSQL, CSC-SQL, and GSR-SQL) exhibited improved execution accuracy. Among these, CHESS demonstrated the largest increase, with EX rising from 62% to 81%, corresponding to a 31% relative improvement. In contrast, OpenSearch-SQL was the only agent to experience a decline in performance, with EX decreasing from 71% to 69%.

We further present the number of examples for which correctness changed, alongside information on whether annotation corrections were applied in Figure 4. For CHESS, 26 examples were updated from `False` to `True` due to annotation fixes, whereas four examples changed from `True` to `False`.

Rankings. We compared agent rankings before and after correction and observed significant changes in their relative positions. As shown in Figure 5, CHESS's substantial performance improvement elevates its ranking from fourth to first place. In contrast, due to a decline in performance, OpenSearch-SQL falls from first to third place. These results demonstrate that the high error rate present in the benchmark undermines the reliability of the leaderboard and fails to accurately reflect the true performance of the agents.

5.4 Analysis of Performance Changes

In this subsection, we analyzed why agents experienced varying degrees of performance changes from annotation corrections by investigating CHESS [29] and OpenSearch-SQL [34], the agents showing the greatest improvement or degradation.

Table 4: Examples with correctness changes for CHESS and OpenSearch-SQL categorized by annotation correction type.

Category		Question IDs (CHESS)	Question IDs (OpenSearch-SQL)
Corrected $\mathcal{T}$	T → F	255	
Corrected Q	F → T	310, 416, 442, 484, 605, 610, 646, 888, 928, 1200, 1286, 1302	310, 442, 646, 888
	T → F	42	42, 416, 605, 610, 1286, 1302
Corrected $\mathcal{T}$ & Q	F → T	180, 305, 406, 428, 602, 620, 648, 772, 855, 970, 987, 1004, 1271, 1280	180, 406, 602, 855, 1271, 1280
	T → F	829, 1173	620, 829, 846, 864, 985, 987, 1173, 1218

Why does the EX of CHESS increase? As shown in Table 4, we identified 12 examples where correcting errors in the ground truth queries changed the evaluation status from `False` to `True`. We further analyzed the queries generated for these 12 examples by CHESS when evaluated on the original Dev set. We found that 11 of the generated queries matched the revised ground truth, indicating that these examples had originally been incorrectly labeled as `False` due to annotation errors. In addition, among the examples requiring revision of both $\mathcal{T}$ (the question or external knowledge) and Q (the query) to resolve ambiguities and semantic inconsistencies, we observed that 14 examples were reclassified as `True`.

Why does the EX of OpenSearch-SQL decrease? As shown in Table 4, after correcting the SQL queries, we found four examples changed from `False` to `True`. However, six examples changed from `True` to `False`. We further analyzed these six examples and found a common error pattern: in four examples (`416`, `605`, `1286`, `1302`), both OpenSearch-SQL and the human annotators omitted the required `DISTINCT` keyword within the `COUNT` function. This shared error pattern resulted in these examples being erroneously classified as `True` in the original evaluation. Additionally, among the examples that required corrections to both $\mathcal{T}$ and Q, we found six changed from `False` to `True`, while eight changed from `True` to `False`.

6 CONCLUSION

In this work, we analyze two widely used text-to-SQL benchmarks, BIRD and Spider 2.0-Snow, and identify annotation error rates of 52.8% and 66.1%. We then re-evaluate the top-performing text-to-SQL agents from the BIRD leaderboard on the original and corrected BIRD Dev subsets. Our re-evaluation reveals relative performance changes ranging from −3% to 31% and rank shifts of up to three positions, highlighting the unreliability of text-to-SQL leaderboards. We advocate for the development of more effective annotation pipelines to create higher-quality text-to-SQL benchmarks.

7 ACKNOWLEDGEMENTS

We are grateful to the CloudLab [7] for providing computing resources for experiments. This research was supported in part by Google and the Open Philanthropy project.

REFERENCES

[1] Anonymous. 2025. LLM Prompting for Text2SQL via Gradual SQL Refinement. In *Submitted to ACL Rolling Review - February 2025*. https://openreview.net/forum?id=ozUJOijVTJ under review.

[2] Arcwise. [n.d.]. BIRD Minidev. https://github.com/AlibabaResearch/DAMO-ConvAI/issues/39#issuecomment-2303283506

[3] Dina Bitton, David J DeWitt, and Carolyn Turbyfill. 1983. *Benchmarking database systems: A systematic approach*. Computer Sciences Department, University of Wisconsin-Madison.

[4] Haran Boral and David J Dewitt. 1984. A methodology for database system performance evaluation. *ACM SIGMOD Record* 14, 2 (1984), 176–185.

[5] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. arXiv:2402.03216 [cs.CL] https://arxiv.org/abs/2402.03216

[6] Peter Baile Chen, Fabian Wenz, Yi Zhang, Devin Yang, Justin Choi, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. 2025. BEAVER: An Enterprise Benchmark for Text-to-SQL. arXiv:2409.02038 [cs.CL] https://arxiv.org/abs/2409.02038

[7] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of CloudLab. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. USENIX Association, Renton, WA, 1–14. https://www.usenix.org/conference/atc19/presentation/duplyakin

[8] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. arXiv:2308.15363 [cs.DB] https://arxiv.org/abs/2308.15363

[9] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proc. VLDB Endow.* 17, 5 (Jan. 2024), 1132–1145. https://doi.org/10.14778/3641204.3641221

[10] Yingqi Gao, Yifu Liu, Xiaoxia Li, Xiaorong Shi, Yin Zhu, Yiming Wang, Shiqi Li, Wei Li, Yuntao Hong, Zhiling Luo, Jinyang Gao, Liyu Mou, and Yu Li. 2025. A Preview of XiYan-SQL: A Multi-Generator Ensemble Framework for Text-to-SQL. arXiv:2411.08599 [cs.AI] https://arxiv.org/abs/2411.08599

[11] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. arXiv:1704.08760 [cs.CL] https://arxiv.org/abs/1704.08760

[12] Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. 2024. AI Agents That Matter. arXiv:2407.01502 [cs.LG] https://arxiv.org/abs/2407.01502

[13] Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic Evaluation of Text-to-SQL Parsers. arXiv:2106.11455 [cs.CL] https://arxiv.org/abs/2106.11455

[14] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2024. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. arXiv:2411.07763 [cs.CL] https://arxiv.org/abs/2411.07763

[15] Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. 2025. OmniSQL: Synthesizing High-quality Text-to-SQL Data at Scale. arXiv:2503.02240 [cs.CL] https://arxiv.org/abs/2503.02240

[16] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C.C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2024. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-SQLs. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 1835, 28 pages.

[17] Jinyang Li, Xiaolong Li, Ge Qu, Per Jacobsson, Bowen Qin, Binyuan Hui, Shuzheng Si, Nan Huo, Xiaohan Xu, Yue Zhang, Ziwei Tang, Yuanshuai Li, Florensia Widjaja, Xintong Zhu, Feige Zhou, Yongfeng Huang, Yannis Papakonstantinou, Fatma Ozcan, Chenhao Ma, and Reynold Cheng. 2025. SWE-SQL: Illuminating LLM Pathways to Solve User SQL Issues in Real-World Applications. arXiv:2506.18951 [cs.DB] https://arxiv.org/abs/2506.18951

[18] Raghunath Othayoth Nambiar and Meikel Poess. 2006. The making of TPC-DS.. In *VLDB*, Vol. 6. 1049–1058.

[19] NL2SQL-Empirical. [n.d.]. The annotation issues in the dev set of BIRD Benchmark. https://github.com/AlibabaResearch/DAMO-ConvAI/issues/39#issuecomment-2487904959

[20] OpenAI. 2024. GPT-4o System Card. arXiv:2410.21276 [cs.CL] https://arxiv.org/abs/2410.21276

[21] Meikel Poess and Chris Floyd. 2000. New TPC benchmarks for decision support and web commerce. *ACM Sigmod Record* 29, 4 (2000), 64–71.

[22] 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:2410.01943 [cs.LG] https://arxiv.org/abs/2410.01943

[23] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 36339–36348. https://proceedings.neurips.cc/paper_files/paper/2023/file/72223cc66f63ca1aa59edaec1b3670e6-Paper-Conference.pdf

[24] Mohammadreza Pourreza and Davood Rafiei. 2023. Evaluating Cross-Domain Text-to-SQL Models and Benchmarks. arXiv:2310.18538 [cs.CL] https://arxiv.org/abs/2310.18538

[25] Mohammadreza Pourreza, Shayan Talaei, Ruoxi Sun, Xingchen Wan, Hailong Li, Azalia Mirhoseini, Amin Saberi, and Sercan "O. Arik. 2025. Reasoning-SQL: Reinforcement Learning with SQL Tailored Partial Rewards for Reasoning-Enhanced Text-to-SQL. arXiv:2503.23157 [cs.LG] https://arxiv.org/abs/2503.23157

[26] P. J. Price. 1990. Evaluation of Spoken Language Systems: the ATIS Domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*. https://aclanthology.org/H90-1020/

[27] Qwen. 2025. Qwen2.5 Technical Report. arXiv:2412.15115 [cs.CL] https://arxiv.org/abs/2412.15115

[28] Lei Sheng and Shuai-Shuai Xu. 2025. CSC-SQL: Corrective Self-Consistency in Text-to-SQL via Reinforcement Learning. arXiv:2505.13271 [cs.CL] https://arxiv.org/abs/2505.13271

[29] Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. CHESS: Contextual Harnessing for Efficient SQL Synthesis. arXiv:2405.16755 [cs.LG] https://arxiv.org/abs/2405.16755

[30] BIRD Team. 2024. LiveSQLBench: A Dynamic and Contamination-Free Benchmark for Evaluating LLMs on Real-World Text-to-SQL Tasks. https://github.com/bird-bench/livesqlbench. Accessed: 2025-05-22.

[31] BIRD Team. 2025. BIRD-Interact: Re-imagine Text-to-SQL Evaluation via Lens of Dynamic Interactions. https://github.com/bird-bench/BIRD-Interact. Accessed: 2025-06-01.

[32] Niklas Wretblad, Fredrik Gordh Riseby, Rahul Biswas, Amin Ahmadi, and Oskar Holmström. 2024. Understanding the Effects of Noise in Text-to-SQL: An Examination of the BIRD-Bench Benchmark. arXiv:2402.12243 [cs.CL] https://arxiv.org/abs/2402.12243

[33] Xiangjin Xie, Guangwei Xu, Lingyan Zhao, and Ruijie Guo. 2025. OpenSearch-SQL: Enhancing Text-to-SQL with Dynamic Few-shot and Consistency Alignment. arXiv:2502.14913 [cs.CL] https://arxiv.org/abs/2502.14913

[34] Xiangjin Xie, Guangwei Xu, Lingyan Zhao, and Ruijie Guo. 2025. OpenSearch-SQL: Enhancing Text-to-SQL with Dynamic Few-shot and Consistency Alignment. arXiv:2502.14913 [cs.CL] https://arxiv.org/abs/2502.14913

[35] Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. SQLizer: query synthesis from natural language. *Proc. ACM Program. Lang.* 1, OOPSLA, Article 63 (Oct. 2017), 26 pages. https://doi.org/10.1145/3133887

[36] Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2025. AgentOccam: A Simple Yet Strong Baseline for LLM-Based Web Agents. arXiv:2410.13825 [cs.AI] https://arxiv.org/abs/2410.13825

[37] Semih Yavuz, Izzeddin Gur, Yu Su, and Xifeng Yan. 2018. What It Takes to Achieve 100% Condition Accuracy on WikiSQL. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, Brussels, Belgium, 1702–1711. https://doi.org/10.18653/v1/D18-1197

[38] 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. 2019. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. arXiv:1809.08887 [cs.CL] https://arxiv.org/abs/1809.08887

[39] Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Jasjeet Sekhon, Jacob Steinhardt, Antony Kellerman, Sarah Schwettmann, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. 2025. Establishing Best Practices for Building Rigorous Agentic Benchmarks. arXiv:2507.02825 [cs.AI] https://arxiv.org/abs/2507.02825