FINCH: Financial Intelligence using Natural language for Contextualized SQL Handling

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Text-to-SQL, the task of translating natural language questions into SQL queries, has long been a central challenge in NLP. While progress has been significant, applying it to the financial domain remains especially difficult due to complex schema, domain-specific terminology, and high stakes of error. Despite this, there is no dedicated large-scale financial dataset to advance research, creating a critical gap. To address this, we introduce a curated financial dataset comprising 292 tables and 85,638 natural language–SQL pairs, enabling both fine-tuning and rigorous evaluation. Building on this resource, we benchmark reasoning models and language models of varying scales, providing a systematic analysis of their strengths and limitations in financial Text-to-SQL tasks. Finally, we propose a finance-oriented evaluation metric that captures nuances overlooked by existing measures, offering a more faithful assessment of model performance.

1 Introduction & Motivation

Translating human questions into SQL has long been studied, gaining momentum in the neural era with Seq2SQL (2017) [1], which introduced reinforcement learning for executable queries and surpassed rule-based systems. SQLNet (2017) [2] followed with sketch-based decoding, eliminating reinforcement learning. Spider (2018) [3] then provided the first large-scale, complex, cross-domain dataset that reshaped evaluation. Its successors—CoSQL [4] extended challenges to conversational and context-dependent tasks. RAT-SQL (2020) [5] advanced schema linking with relation-aware transformers, while PICARD (2021) [6] introduced constrained decoding for large models. More recently, BIRD (2023) [7] scaled Text-to-SQL benchmarking to 12k text-to-sql pairs. Yet, progress has been largely driven by domains such as encyclopedic knowledge, QA, dialogue, and healthcare—prioritizing generalization across diverse schemas, but not the specialized requirements of finance.

In parallel, executable reasoning over financial data has begun to mature. FinQA [8] enabled numerical reasoning with gold program-of-operations supervision. TAT-QA [9] blended text and tables, essential for bridging structured and unstructured content. ConvFinQA [10] extended this to conversational reasoning, simulating analyst workflows. FinSQL [11] introduced the BULL dataset, targeting schema-level issues in finance, while BookSQL [12] contributed 78,433 NL–SQL pairs in accounting, exposing persistent challenges for general-purpose and large LLMs. However, most efforts have centered on document-based QA or hybrid settings, with limited attention to direct querying over financial SQL databases. As a result, the field still lacks a comprehensive benchmark that captures the precision, terminology, and complexity of real-world financial systems. In this work, we advance Text-to-SQL for finance through three contributions: **Large-scale financial dataset curation:** We consolidate BIRD [7], Spider [3], FinSQL [11], and BookSQL [12] into a unified benchmark dataset (FINCH), normalizing all queries for SQLite. The dataset spans over 33 databases across retail, banking, loans, insurance, sales, marketing, e-commerce, funds, stocks, and accounting. It comprises 292 tables, 2233 columns, 177 relations, and 85,638 NL–SQL pairs—significantly expanding finance-specific coverage for evaluation and fine-tuning. **Benchmarking across models:** We evaluate diverse models: large-scale LLMs (Qwen3-235B-A22B[1]), medium- and small-scale ones (GPT-OSS-120B[2], GPT-OSS-20B[3], Qwen3-8B[4]), and reasoning-centric systems (Phi-4-mini-reasoning[5], Arctic-Text2SQL-R1-7B[6]). Surprisingly, GPT-OSS-120B outperforms even larger LLMs like Qwen3-235B-A22B. Arctic-Text2SQL-R1-7B, though smaller, ranks third, highlighting the effectiveness of domain-specific fine-tuning. Results reveal that carefully adapted reasoning models can rival or surpass general-purpose LLMs in finance. **Finance-specific evaluation metric:** We design a metric integrating component matching and execution accuracy with adaptive weighting and tolerance thresholds. This reduces undue penalties for minor floating-point mismatches while emphasizing structural correctness—such as columns, tables, and conditions. Evaluations show that this metric better reflects financial requirements and yields a more faithful assessment of model performance.

2 FINCH: Dataset Description

To address the lack of large-scale, finance-specific Text-to-SQL benchmarks, we constructed the **FINCH** dataset by consolidating four major open-source resources: *BIRD*[3], *Spider*[3], *BULL*[11], and *BookSQL*[12] . Since *Spider* and *BIRD* are not inherently aligned with financial applications, we systematically filtered and retained only domains relevant to finance such as sales, retail, card transactions, banking, loans, insurance, and e-commerce. We took only training and validation samples of BookSQL as the text-to-sql pair was available only for these partition.

Dataset	#Size	#DB	#T/DB	ORDER BY	GROUP BY
Spider[3]	10,181	200	5.1	1,335	1,491
BIRD[7]	12,751	95	7.3	2,576	881
BULL[11]	4,966	3	26	638	431
BookSQL[12]	78,433	1	7	12,392	15,849
FINCH	**85,638**	**33**	**8.85**	**13,529**	**16,762**

Table 1: Comparison of FINCH with existing Text-to-SQL datasets. Columns indicate dataset size (#Size), number of databases (#DB), table to DB ratio and occurrence counts of key SQL constructs.

The final version of FINCH consists of **33 databases**, encompassing **292 tables**, **2,233 columns**, and **177 relations**, with a total of **85,638 NL–SQL pairs**. In terms of difficulty distribution, FINCH includes 9418 easy, 37422 medium, and 38798 hard examples, there were 7035 instances that were originally unlabeled were annotated following the schema complexity definition of [12]. This design ensures coverage across diverse financial operations, schema complexities, and SQL constructs.

Unlike *Spider*[3] and *BIRD*[7], which emphasize cross-domain generalization, FINCH is tailored for finance, making it uniquely positioned to evaluate both the accuracy and robustness of Text-to-SQL models in high-stakes, domain-specific contexts. Compared to *BookSQL*[12] , which is limited to a single accounting database, FINCH spans multiple financial domains with significantly richer schema variety. Furthermore, FINCH provides extensive use of SQL constructs such as `ORDER BY`, `GROUP BY`, and nested queries, making it particularly suited for evaluating complex reasoning in financial settings.

In summary, FINCH is the large-scale, finance-specific Text-to-SQL dataset that combines breadth (multiple domains), depth (rich schema complexity), and difficulty (non-trivial SQL constructs). We believe FINCH will serve as a cornerstone resource for evaluating and fine-tuning both large language models and reasoning models in financial applications.

[1] https://huggingface.co/Qwen/Qwen3-235B-A22B
[2] https://huggingface.co/openai/gpt-oss-120b
[3] https://huggingface.co/openai/gpt-oss-20b
[4] https://huggingface.co/Qwen/Qwen3-8B
[5] https://huggingface.co/microsoft/Phi-4-mini-flash-reasoning
[6] https://huggingface.co/Snowflake/Arctic-Text2SQL-R1-7B

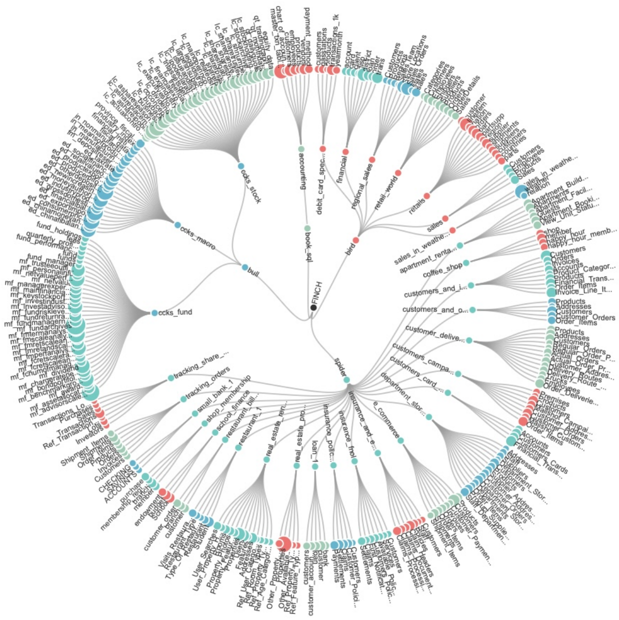

Figure 1: FINCH dataset showing the combination of different databases and tables.

3 Experiment Setup and Evaluation Metrics

We benchmark reasoning-optimized models to test their transferability to financial Text-to-SQL. Our evaluation spans large and medium-scale models (Qwen3-235B-A22B, GPT-OSS-120B), smaller compute-efficient ones (Qwen3-8B, GPT-OSS-20B), and reasoning baselines (Phi-4-mini-reasoning, Arctic-Text2SQL-R1-7B). This design provides **scaling contrast** to examine whether SQL fidelity improves with size, **family diversity** to mitigate architecture-specific biases, and **reasoning alignment** to assess whether enhanced chain-of-thought or tool-use capabilities aid schema grounding, operator selection, and compositional joins in finance. We employ a uniform one-shot prompting protocol: each example provides the natural-language question and database schema, and models must generate a single valid SQL query.

Evaluation traditionally relies on four metrics. **Exact Matching (EM)**[1] checks query string identity, **Execution Accuracy (EX)**[1] judges output equivalence, **Component Matching (CM)**[3] assesses clause-level correctness, and **Valid Efficiency Score (VES)**[7] rewards efficient yet correct queries. While useful, these metrics misalign with financial needs: EM and EX penalize cosmetic differences or trivial rounding errors; CM treats all clauses equally, ignoring the higher weight of WHERE, JOIN, GROUP BY, HAVING, and AGG; and VES penalizes necessary financial complexity. To address this, we propose FINCH, a clause-sensitive metric integrating execution accuracy with tolerance. FINCH emphasizes material correctness, balances structure with execution, and offers a more faithful evaluation of financial SQL performance.

3.1 Proposed Metric (FINCH Score):

Let the gold SQL be q^* and the model SQL be $\hat{q}$.

1) Component-wise Score (Structure/Semantics) Select components K (e.g., SELECT, WHERE, GROUP BY, HAVING, ORDER BY, JOIN, AGG, LIMIT, SUBQUERY). For each $k \in K$, compute similarity $s_k(\hat{q}, q^*) \in [0,1]$ (e.g., exact/set/token F1). With weights $w_k \geq 0$, $\sum w_k = 1$, define

$$S(\hat{q}, q^*) = \sum_{k \in K} w_k \, s_k(\hat{q}, q^*).$$

In finance, heavier weights go to WHERE, JOIN, GROUP BY, HAVING, AGG, as they encode business logic.

2) Execution Accuracy (with Tolerance) Let $r_{\hat{q}}, r_{q^*}$ be results. Execution similarity is

$$e(\hat{q}, q^*) = \begin{cases} 1, & \dfrac{|r_{\hat{q}} - r_{q^*}|}{\max(1, |r_{q^*}|)} \leq \tau, \\ 0, & \text{otherwise}, \end{cases}$$

where τ (e.g., 10^{-4} or 0.01%) provides tolerance for materiality.

3) Combined Score. Structure and execution are integrated via

$$\text{FINCH}(\hat{q}, q^*) = S(\hat{q}, q^*)^{\beta} \cdot \big(\delta + (1-\delta)e(\hat{q}, q^*)\big),$$

with $\beta \geq 1$ emphasizing structure, $\delta \in [0, 1)$ controlling penalty for execution failure. Strict: $\delta = 0$; finance-friendly: $\delta \in [0.2, 0.5]$.

4 Results & Analysis

Evaluation on the FINCH dataset employed standard metrics—*Exact Matching (EM)*[1], *Execution Accuracy (EX)*[1], *Component Matching (CM)*[3], *Strict Accuracy*[3]- the combination of exact matching and execution accuracy, and the proposed *FINCH Score*. Consolidated results are shown in Table 2. Overall, **GPT-OSS-120B achieves the strongest performance**, outperforming all models across most metrics. Interestingly, **Arctic-Text2SQL-R1-7B ranks third**, despite its modest scale, underscoring the value of domain-specific finetuning for aligning schema, SQL structure, and natural language. These findings highlight that while scale helps, reasoning-centric finetuning can rival much larger models. Moreover, comparing **Strict Accuracy** with the **FINCH score** illustrates FINCH's sensitivity: it credits partially correct queries that traditional metrics mark as failures, particularly by weighting core clauses such as `SELECT`, `WHERE`, and `JOIN` over others.

Accuracy Metric	**Qwen3-8B**	**Arctic-Text2SQL-R1-7B**	**Phi-4-mini-reasoning**	**GPT-OSS-20B**	**GPT-OSS-120B**	**Qwen3-235B-A22B**
Exact Matching	0.50	0.60	0.00	0.30	1.80	0.70
Execution Accuracy	0.80	2.30	0.20	7.50	27.80	2.50
Component Matching	3.50	3.70	1.00	5.20	16.60	2.80
Strict Accuracy (EM+EX)	0.10	0.20	0.00	0.30	1.70	0.20
FINCH Score	**1.20%**	**1.50%**	**0.40**	**3.00**	**11.60**	**1.20**

Table 2: Model performance comparison across evaluation metrics (accuracy scores in %).

Clause-level results (Table 3) show persistent weaknesses in `SELECT`, `FROM`, and `WHERE`, with `JOIN` accuracy near zero. Models perform better on peripheral clauses like `LIMIT`, yet semantic grounding remains a challenge.

Model	**SELECT**	**FROM**	**WHERE**	**GROUP BY**	**HAVING**	**ORDER BY**	**LIMIT**
Qwen3-8B	1.60	3.90	0.90	4.80	2.20	1.40	38.20
Arctic-Text2SQL-R1-7B	2.50	3.60	0.70	4.70	1.00	1.30	42.70
Phi-4-mini-reasoning	2.00	2.30	0.40	2.10	1.30	0.40	27.60
GPT-OSS-20B	1.40	6.20	1.50	8.40	3.70	1.50	65.20
GPT-OSS-120B	4.70	27.30	6.90	7.50	6.30	6.30	73.80
Qwen3-235B-A22B	2.00	2.90	0.80	5.40	1.50	1.00	29.80
Average Accuracy	2.37	7.37	1.87	5.48	2.67	1.98	46.55

Table 3: SQL Clause Performance Comparison (accuracy scores in %)

Finally, stratified analysis shows **sharp performance degradation** with query difficulty: GPT-OSS-120B's FINCH score falls from **26.3% (easy)** to **10.5% (medium)** and **4.5% (hard)**. This confirms that current models—regardless of scale—struggle with schema grounding, compositionality, and multi-table reasoning, marking critical challenges for future research.

5 Conclusion & Future Work

In this work, we introduced **FINCH**, the large-scale financial Text-to-SQL benchmark that consolidates multiple open-source resources into a unified, finance-specific dataset comprising 85,638

NL–SQL pairs across 33 databases. Alongside the dataset, we proposed the **FINCH score**, a finance-aware evaluation metric that better captures clause sensitivity, execution tolerance, and domain relevance compared to conventional metrics. Our benchmarking study across diverse large language models and reasoning models demonstrated three key insights: (i) domain-specific fine-tuning, as shown by the Arctic-Text2SQL-R1-7B model, often surpasses the performance of even trillion-scale models, (ii) the majority of model errors concentrate on schema-sensitive clauses such as `SELECT`, `FROM`, and `WHERE`, underscoring persistent challenges in schema grounding, and (iii) current models experience steep accuracy degradation from easy to medium and hard queries, highlighting their limitations in handling compositional and multi-table reasoning. Future work includes multi-modal integration of financial text, tables, and SQL, robust schema linking, and conversational Text-to-SQL for iterative analyst workflows. We envision FINCH and its tailored metric as a foundation for advancing reliable, domain-specific financial Text-to-SQL research.

References

[1] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *ArXiv*, abs/1709.00103, 2017.

[2] Xiaojun Xu, Chang Liu, and Dawn Xiaodong Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *ArXiv*, abs/1711.04436, 2017.

[3] Tao Yu, Rui Zhang, Kai-Chou Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *ArXiv*, abs/1809.08887, 2018.

[4] Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, et al. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *ArXiv*, abs/1909.05378, 2019.

[5] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *Annual Meeting of the Association for Computational Linguistics*, 2019.

[6] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *ArXiv*, abs/2109.05093, 2021.

[7] Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *ArXiv*, abs/2305.03111, 2023.

[8] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, et al. Finqa: A dataset of numerical reasoning over financial data. *ArXiv*, abs/2109.00122, 2021.

[9] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat seng Chua. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[10] Zhiyu Chen, SHIYANG LI, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. Convfinqa: Exploring the chain of numerical reasoning in conversational finance question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2022.

[11] Chao Zhang, Yuren Mao, Yijiang Fan, Yu Mi, Yunjun Gao, Lu Chen, Dongfang Lou, and Jinshu Lin. Finsql: Model-agnostic llms-based text-to-sql framework for financial analysis. *Companion of the 2024 International Conference on Management of Data*, 2024.

[12] Rahul Kumar, Amar Raja Dibbu, Shrutendra Harsola, Vignesh T. Subrahmaniam, and Ashutosh Modi. Booksql: A large scale text-to-sql dataset for accounting domain. *ArXiv*, abs/2406.07860, 2024.