



TACO: A Benchmark for Open-Domain Text-to-SQL with Ambiguous and Cross-Database Queries

Chao Deng
Renmin University, China
dengc@ruc.edu.cn

Ju Fan*
Renmin University, China
fanj@ruc.edu.cn

Yuyu Luo
HKUST (Guangzhou)
yuyuluo@hkust-gz.edu.cn

Qinliang Xue
Renmin University, China
xueql@ruc.edu.cn

Meihao Fan
Renmin University, China
fmh1art@ruc.edu.cn

Yuxin Zhang
Renmin University, China
yuxin.zhang@ruc.edu.cn

Min Zhang
Renmin University, China
Beijing Big Data Centre
zmbbdc@gmail.com

Xiaofeng Jia*
Beijing Big Data Centre
jiaxiaofeng119@gmail.com

Jing Zhang
Renmin University, China
zhang-jing@ruc.edu.cn

Xiaoyong Du
Renmin University, China
duyong@ruc.edu.cn

ABSTRACT

Text-to-SQL aims to translate natural language questions into executable SQL queries over structured databases. Existing benchmarks mainly focus on closed-domain settings with predefined database schemas and well-specified questions, but they fall short in addressing the challenges of open-domain scenarios, such as ambiguous questions, unspecified databases, and cross-database querying. To bridge this gap, we introduce TACO, a benchmark for open-domain Text-to-SQL with Ambiguous and Cross-database queries. TACO consists of 1,500 real-world Text-to-SQL examples based on a smart city data service and 13,000 high-quality synthetic examples generated based on large-scale open data portals, covering diverse domains such as transportation, healthcare, and finance. To construct the synthetic examples, we develop an effective data synthesis pipeline that preserves the complexity of real-world queries. To demonstrate the utility of TACO, we introduce a baseline TACO-SQL composed of question rewriting, table linking, and query planning, to illustrate the challenges posed by TACO and to better understand the limitations of existing Text-to-SQL approaches. Extensive experiments on TACO using a variety of recent Text-to-SQL approaches show that, while TACO-SQL achieves the best results, a significant gap still remains between the existing approaches and human-written SQL. These findings highlight the difficulty of open-domain Text-to-SQL and position TACO as a valuable benchmark to drive future research.

PVLDB Reference Format:

Chao Deng, Ju Fan, Yuyu Luo, Qinliang Xue, Meihao Fan, Yuxin Zhang, Min Zhang, Xiaofeng Jia, Jing Zhang, and Xiaoyong Du. TACO: A Benchmark for Open-Domain Text-to-SQL with Ambiguous and Cross-Database Queries. PVLDB, 19(6): 1386 - 1399, 2026. doi:10.14778/3797919.3797942

*Ju Fan and Xiaofeng Jia are the corresponding authors.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 19, No. 6 ISSN 2150-8097. doi:10.14778/3797919.3797942

PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at <https://github.com/Akanezora0/TACO-Benchmark>.

1 INTRODUCTION

Text-to-SQL, which translates natural language (NL) questions into SQL queries, provides a user-friendly interface for non-technical users to access structured data, supporting applications in business intelligence and data-driven decision-making [9, 18, 32, 49]. With the rapid advancement of large language models (LLMs), many LLM-powered approaches have emerged (see the recent survey [25]), achieving strong performance on standard Text-to-SQL benchmarks [18, 25, 32].

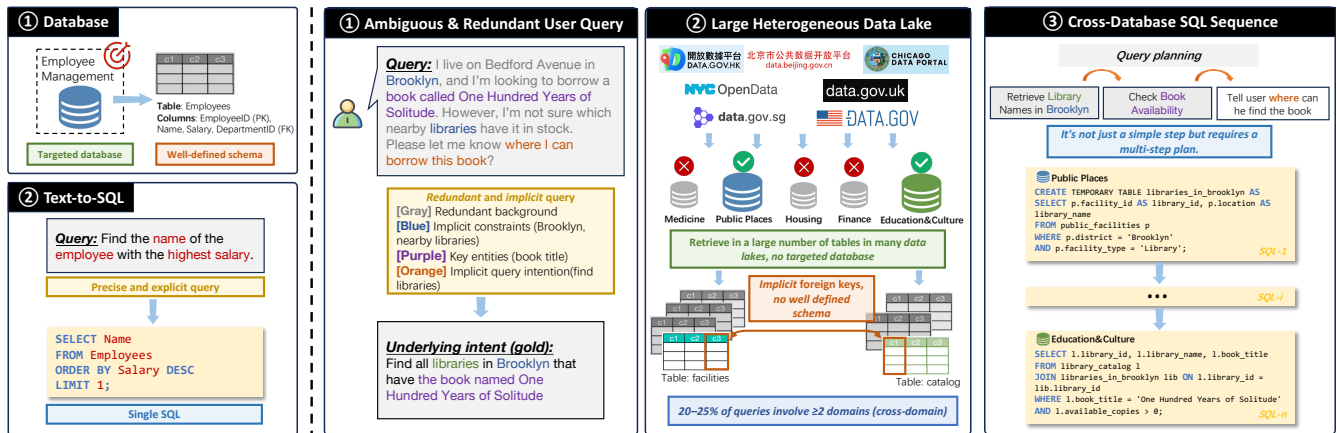
Traditional Focus: Closed-Domain Text-to-SQL. Most prior work focuses on the *closed-domain* setting, as shown in Figure 1(a), where the target database is predefined and the schema is fully observable. Users pose relatively precise questions, and the task reduces to translating an NL question into a single SQL query over a known schema. For example, the question “Find the name of the employee with the highest salary” can be translated into an SQL query given the schema of an Employee Management database.

New Challenges: Open-Domain Text-to-SQL. In many real deployments, data is organized as large *data lakes*, such as smart-city data services [4] and open government data portals [1, 2], where information spans numerous heterogeneous databases. In these settings, the closed-domain assumptions of Text-to-SQL often do not hold, and three challenges arise

(1) **Ambiguous NL questions.** Users often have limited knowledge of the underlying data and issue vague or redundant questions with unclear intent or incomplete constraints [3, 7, 16, 35, 38].

(2) **Unspecified target databases.** NL Questions rarely specify the relevant databases or tables; the system must retrieve candidate tables from a large, heterogeneous data lake [8, 27, 55].

(3) **Cross-database querying.** Answering a single NL question may require combining data from multiple databases with weak or implicit relationships, requiring multi-step query planning and result integration [15, 50].



(a) Closed-domain setting with a single database and explicit queries.

(b) Open-domain setting in TACO with ambiguous queries, large data lakes, and cross-database SQL sequences.

Figure 1: Comparison of closed-domain and open-domain Text-to-SQL scenarios.

Example 1.1. Figure 1(b) illustrates an open-domain scenario over an open data portal such as NYC Open Data¹. A user may ask: “Could you tell me which libraries around Brooklyn have a copy of *One Hundred Years of Solitude*?” The question is informally phrased and omits schema details, and relevant information may reside in separate databases (e.g., Public Places and Education & Culture). Thus, answering the question requires a sequence of SQL queries whose intermediate results must be joined across databases. This combination of ambiguity, database discovery, and cross-database reasoning extends beyond traditional Text-to-SQL benchmarks.

Limitations of Existing Benchmarks. Several benchmarks have advanced Text-to-SQL research, including WikiSQL [57], Spider [49], BIRD [23], Spider 2.0 [20], and NL2SQL-BUGs [26]. These benchmarks typically assume a single target database (or a small set of databases) with fully observable schemas and relatively concise NL questions, and thus primarily evaluate closed-domain settings. Recent diagnostic datasets (e.g., SNAILS [27], Ambrosia [35]) consider schema naming and ambiguity, but still operate under the same closed-domain assumption. Closer to our setting, union-style and interaction benchmarks such as Bird-Union/Spider-Union [37], BIRD-Interact [17], and SWE-SQL (Bird-Critic) [24] address challenges of schema scale, multi-turn clarification, or SQL debugging. However, they generally rely on a unified schema or reuse existing benchmarks, and do not model data-lake table retrieval and cross-database execution over heterogeneous datasets. Therefore, a benchmark that jointly evaluates ambiguous questions, large-scale table retrieval, and cross-database reasoning remains missing.

Our Open-Domain Text-to-SQL Benchmark: TACO. To address this gap, we introduce TACO, a benchmark for open-domain Text-to-SQL with Ambiguous and Cross-database queries. TACO evaluates three capabilities of Text-to-SQL approaches: (i) resolving ambiguity and redundancy in user NL questions, (ii) retrieving relevant tables from large heterogeneous data lakes, and (iii) planning and generating SQL pipelines that may span multiple databases.

¹<https://opendata.cityofnewyork.us/>

TACO consists of two complementary datasets, each consisting of NL questions paired with gold SQL (or short SQL sequences):

Type-1 (TACO-SmartCity): Real Examples from Smart City Services. We collect 1,500 NL questions from a Beijing smart city data service, where data from 31 government departments are integrated into a shared data lake. Domain experts decompose each question, annotate executable SQLs, and validate results via cross-execution checking. The resulting benchmark subset covers 117 tables across 31 databases, with an average of 1.19 databases per query, and includes many cases with vague intent, implicit constraints, and cross-database reasoning.

Type-2 (TACO-OpenData with TACO-Beijing and TACO-US): Synthetic Examples from Open Data Portals. Real-world data is limited in scale and constrained by privacy. To enable large-scale, fully public evaluation, we develop a data synthesis pipeline that preserves structural patterns of real SQLs, populates content using public schema, and generates NL questions via Chain-of-Thought prompting. Applying this pipeline to the Beijing Municipal Open Data Platform [1] and the U.S. Government’s Open Data Portal [2] produces 13,000 bilingual examples across 52 databases and 13,004 tables, forming the TACO-Beijing and TACO-US datasets.

The real and synthetic datasets form a unified testbed for open-domain Text-to-SQL. As shown in Table 1, TACO matches or exceeds classic benchmarks in the number of databases and tables, and explicitly targets large, heterogeneous data lakes. Furthermore, compared with Spider and BIRD, TACO contains substantially longer NL questions and SQL queries (Figure 2), indicating higher levels of ambiguity, redundancy, and structural complexity.

Remarks. TACO adopts a *pure SQL-based execution pipeline*, ensuring that the benchmark evaluates Text-to-SQL reasoning rather than external scripting. Specifically, each open-domain NL question is mapped to a *logical SQL pipeline* composed of multiple SQL statements, where each statement is executed on its corresponding database. Intermediate results are combined *within the database engine* using standard SQL constructs (e.g., JOIN, UNION, nested subqueries), without relying on Python or shell-level orchestration.

Table 1: Comparison of TACO with representative Text-to-SQL benchmarks, including recent ambiguity-, interaction-, and union-style benchmarks.

Datasets		#-Examples	#-DBs	#-Tables	#-Tables/DB	Redundancy	Table linking	Cross Table	Cross DB
WikiSQL [57]		80,654	26,521	26,521	1.0	×	×	×	×
Spider1.0 [49]		10,181	200	1,020	5.1	×	×	✓	×
KaggleDBQA [19]		272	8	18	2.25	×	×	✓	×
BIRD [23]		12,751	95	694	7.3	×	×	✓	×
Spider2.0-snow [20]		547	152	8,000	52.63	×	×	✓	×
Bird-Critic [24]		–	95	694	7.3	✓	✓	✓	×
BIRD-Interact [17]		6,500	95	694	7.3	✓	✓	✓	×
SpiderUnion [37]		13,000	600	5,000	8.3	×	×	✓	×
BirdUnion [37]		22,000	200	3,000	15.0	×	✓	✓	×
Abacus-SQL [44]		–	–	–	–	×	–	✓	✓
TACO	TACO-SmartCity	1,500	31	113	3.77	✓	✓	✓	✓
	TACO-Beijing	7,000	28	3,010	107.50	✓	✓	✓	✓
	TACO-US	6,000	24	9,994	416.42	✓	✓	✓	✓

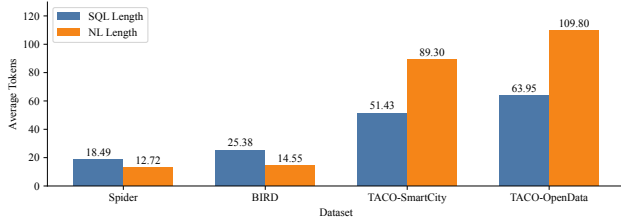


Figure 2: Average tokens in SQL and NL across benchmarks.

Our Open-Domain Text-to-SQL Baseline: TACO-SQL. In order to demonstrate the utility of TACO and analyze existing Text-to-SQL approaches, we introduce TACO-SQL, a modular LLM-based framework for open-domain Text-to-SQL. It consists of three components: (1) *Question Rewriting*, which reformulates ambiguous NL questions to clarify user intent; (2) *Table Linking*, which retrieves relevant tables from large heterogeneous data lakes; and (3) *Query Planning and Generation*, which decomposes complex queries into multi-step plans and produces executable SQL pipelines. Note that the primary contribution of this work is the TACO benchmark. TACO-SQL only serves as a representative baseline that instantiates these components and reveals where the existing models succeed or fail under open-domain conditions.

We conduct an extensive empirical evaluation on TACO, comparing a diverse set of Text-to-SQL approaches, including base LLMs (e.g., GPT series, Llama 3, DeepSeek), LLM-based systems (e.g., DIN-SQL [31], MAC-SQL [39]), supervised fine-tuned models (e.g., CodeS [22], Qwen2.5-Coder), and hybrid strategies (e.g., CHES [36], ZeroNL2SQL [10]). The results show that TACO-SQL consistently improves over the LLM-based baselines, yet a substantial gap remains relative to human experts, indicating the difficulty of open-domain Text-to-SQL.

Contributions. Our main contributions are as follows:

(1) We formalize the open-domain Text-to-SQL problem (Section 2) and introduce TACO, a benchmark designed to evaluate ambiguity resolution, data-lake table retrieval, and cross-database SQL reasoning (Section 3).

(2) We propose TACO-SQL, a LLM-based framework that integrates question rewriting, table linking, and query planning, serving as a practical baseline for open-domain Text-to-SQL (Section 4).

(3) We conduct experiments on TACO using a diverse set of Text-to-SQL methods (Section 5). The results show a substantial performance gap between the approaches and gold SQL, indicating that TACO poses a challenging benchmark for advancing open-domain Text-to-SQL research.

2 PRELIMINARIES

This section first formalizes the open-domain Text-to-SQL problem and its benchmark, and then reviews related work.

2.1 Problem Formulation

Open-Domain Text-to-SQL. The open-domain Text-to-SQL task aims to map a natural language (NL) question Q to one or more SQL queries $S = \{S_1, \dots, S_k\}$, defined as $S = g(Q, \mathcal{D} \mid \Phi)$, where g denotes a model with parameters Φ , and $\mathcal{D} = \{DB_1, \dots, DB_n\}$ is a collection of heterogeneous databases. Each database DB_i contains tables T and columns C , i.e., $DB_i = \langle T, C \rangle$. The generated SQL queries may form an ordered sequence: each S_i specifies a target database and may consume intermediate results produced by earlier steps, yielding a logical execution pipeline whose final result is the query answer.

Open-domain queries typically exhibit: (1) ambiguous or redundant phrasing; (2) missing or implicit database/table references; and (3) information distributed across multiple heterogeneous databases. Figure 1(b) illustrates such a setting over open-data portals.

Types of Supported Queries: In TACO, queries are categorized by the origin of the tables they involve. The current version of TACO supports three types of Text-to-SQL queries: (1) single-table selection and (2) multi-table join queries within a single database, and (3) cross-database queries involving tables from multiple databases. For cross-database queries, intermediate results are combined *purely through SQL* (e.g., JOIN, UNION, nested subqueries) within the database engine, without Python or shell-level scripting.

Open-Domain Text-to-SQL Benchmark. We formalize an open-domain benchmark as $(\mathcal{D}, \{Q, S\})$, where \mathcal{D} is a collection of heterogeneous databases and each example consists of an NL question and its corresponding SQL queries. Note that TACO focuses on single-turn Text-to-SQL: each example contains one NL question and one executable SQL query (or SQL sequence) that runs directly on \mathcal{D} . The benchmark does not include multi-turn dialogue, agentic data-analysis pipelines (e.g., Python/UDF workflows), or non-SQL tasks such as data cleaning or schema repair. Although some queries may require schema- or value-level background knowledge, all answers are obtained strictly through SQL execution over the underlying databases.

2.2 Related Work

Text-to-SQL. LLM-based approaches have substantially advanced Text-to-SQL through in-context learning, instruction tuning, and code generation. Base models (e.g., GPT-4/4o, Llama3) support zero-shot prompting, while methods such as DIN-SQL [31] and MAC-SQL [39] decompose complex queries via multi-step prompting. CHASE-SQL [30] adopts multi-agent collaboration, and fine-tuned large models, such as CodeS [22] and Qwen2.5-Coder further improve SQL generation. Hybrid systems including CHESS [36] and ZeroNL2SQL [10], combine LLM reasoning with schema linking and self-refinement [21, 56]. System-level pipelines for open-domain Text-to-SQL integrate retrieval, rewriting, and debugging [44], and similar agentic workflows have been explored for related data tasks such as table question answering [11] and end-to-end data analysis [54]. Despite strong performance in closed-domain settings, these methods do not explicitly address ambiguity, unspecified schemas, and cross-database planning, motivating the need for an open-domain benchmark [28].

Benchmarks for Text-to-SQL. Early benchmarks such as *WikiSQL* [57] evaluate simple single-table queries. *Spider* [49] and *BIRD* introduce complex multi-table queries across diverse databases, and *Spider2.0* [20] further extends coverage to enterprise-level SQL patterns. Diagnostic benchmarks like *Dr.Spider* [5] and *NL2SQL-BUGs* [26] evaluate dataset quality and semantic robustness. Other benchmarks target specific subproblems: *SNAILS* [27] focuses on schema linking, *Ambrosia* [35] and Wang et al. [38] study ambiguity and unanswerable questions, and *LogicalBeam* [3] explores program search but does not consider open-domain table retrieval.

Closer to our setting, *TailorSQL* [37] introduces *Spider-Union* and *Bird-Union*, which merge schemas from *Spider*/*BIRD* into large unified databases to evaluate join-path prediction and table selection. However, each query still operate within a single logical database, and SQL queries are derived from existing benchmarks rather than real user inputs. *BIRD-INTERACT* [17] evaluates Text-to-SQL in multi-turn conversational and agentic settings by augmenting *BIRD* with hierarchical knowledge bases and user simulators. *SWE-SQL* (also known as *BIRD-CRITIC*) [24] provides a diagnostic benchmark for debugging user-written SQL rather than mapping NL to SQL.

While these benchmarks expand coverage of robustness, ambiguity, interaction, and workload variation, they do not model the open-domain setting (e.g., municipal or open-data platforms), where NL questions may be ambiguous, the target database is unspecified, and relevant information can span multiple heterogeneous

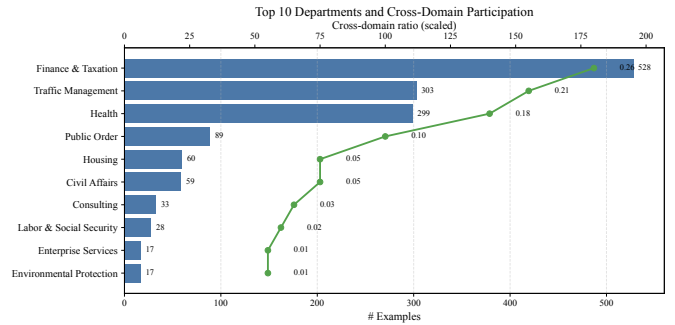


Figure 3: Question distribution by department in the Smart City Data Service dataset (top-10 departments). Bars show the number of queries involving each department, and the line plot shows the proportion of cross-database queries (i.e., queries involving this department and at least one additional department/database).

databases. TACO addresses this setting by grounding the benchmark in real user queries and by explicitly evaluating data-lake table retrieval and cross-database SQL reasoning.

3 THE TACO BENCHMARK

To advance research on open-domain Text-to-SQL, we introduce TACO, a benchmark for Text-to-SQL with Ambiguous and CrQs-database queries. TACO consists of two complementary parts: (1) TACO-SmartCity, comprising 1,500 real-world NL-SQL examples from a municipal data service; and (2) TACO-OpenData (Beijing/US), containing 13,000 synthetic examples constructed from large-scale open-data portals. Each example pairs an NL question with an executable SQL query (or an SQL sequence) that can be directly executed over the underlying heterogeneous databases. We describe the curation of real examples in Section 3.1, the synthesis pipeline in Section 3.2, and benchmark statistics in Section 3.3.

3.1 Curating Real Text-to-SQL Examples

To support evaluation on real-world workloads in open-domain settings, we curate real NL-SQL pairs from a Smart City Data Service. The environment integrates databases from multiple departments, yet users typically pose questions without specifying the target database or table, leading to ambiguity, implicit schema reference, and cross-database reasoning requirements. Note that all TACO-SmartCity examples are fully anonymized, with no personal identifiers retained; only the NL-SQL pairs are used for benchmarking.

The Smart City Data Service in Beijing integrates data from 31 government departments. User questions, collected from online platforms, hotlines, and offline service centers, often mention domain entities (e.g., district names) but rarely specify the target database, making ambiguity and schema discovery central challenges. Figure 3 shows the distribution of queries across departments.

Example 3.1 (Real-World Data: Ambiguous, Cross-Department Query). A user reports an auto repair shop in an industrial zone that is “causing pollution and appears to operate without proper licenses”, and requests to “check environmental monitoring data for

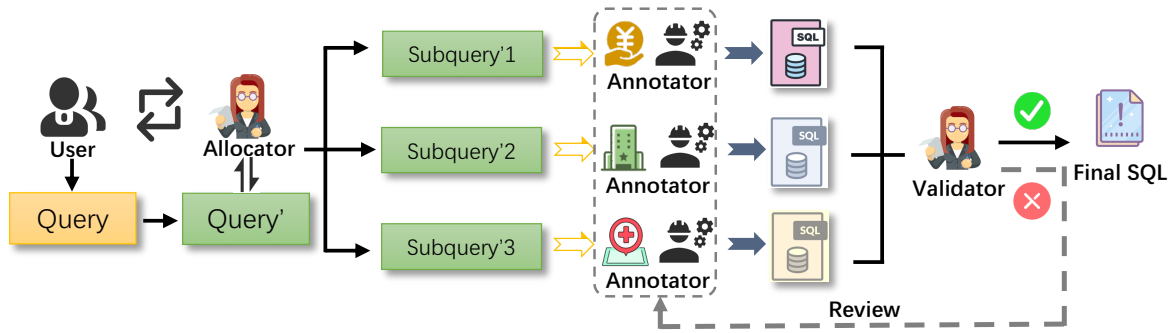


Figure 4: Curation workflow for real Text-to-SQL examples.

businesses in this area to verify compliance with emission standards”. The question references environmental monitoring and business registration but does not specify the relevant databases, tables, or precise locations. The ground-truth SQL is:

```
SELECT monitortime, so2, no3
FROM environmental_bureau.enterprise_emission
WHERE corporationcode IN (
  SELECT ent_name
  FROM market_supervision.business_registration_info
  WHERE business_address LIKE '%Industrial Zone%'
);
```

Curation Workflow. Our curation pipeline (Figure 4) follows a multi-stage, expert-driven process.

(1) *Query Decomposition.* A centralized allocator reviews each user question and manually decomposes complex queries into simpler subqueries while preserving the original semantics.

(2) *Expert Annotation.* Each subquery is assigned to domain experts who author the corresponding executable SQL. No LLMs or automatic rewriting components are involved; all SQL annotations in TACO-SmartCity are written entirely by human experts.

(3) *Validation and Integration.* A separate validation team executes each SQL query on the databases and verifies the correctness. For multi-step queries, intermediate results are combined to produce the final answer. All result integration is performed using *SQL-only* operations (e.g., JOIN, UNION, and nested subqueries) within the database engine, without Python or shell-level orchestration. Each NL-SQL pair is accepted only if: (i) the SQL executes successfully and returns a correct result verified by experts; and (ii) the allocator confirms that the combined result answers the original user question. From this multi-stage pipeline, we select 1,500 high-quality examples whose SQL queries correctly answer the user NL questions, forming the real-world dataset of TACO.

Remarks. Although all SmartCity queries are anonymized, access follows standard practices for restricted datasets. Specifically, qualified researchers may either *apply for controlled access* via a data-use agreement or *submit their models to us for evaluation*. Submitted models are executed in an isolated environment, and only aggregated metrics are returned.

3.2 Synthesizing Text-to-SQL Examples

To protect privacy while retaining real-world structure, we construct a synthetic dataset from publicly available open-data portals, including the Beijing Municipal Open Data Platform [1] and the U.S. Government’s Open Data Portal [2]. These portals cover domains such as public services, healthcare, and transportation. We collect raw files (CSV, XLSX, JSON), parse and clean them, and convert them into relational databases, resulting in 52 databases with 13,004 tables. Example 3.2 illustrates a representative cross-database query in TACO-OpenData, where education and health datasets from different portals are joined via a shared geographic key.

Example 3.2 (Synthetic Data: Cross-Database JOIN over Open Data Portals). The query asks: “Join student performance data with community health statistics to examine the relationship between public health indicators and academic achievement across districts”. Answering this query requires combining education and health datasets from different open-data portals via a shared geographic key.

The corresponding SQL is:

```
SELECT
  e."DistrictName",
  e."NumberofStudentsInNumerator",
  h."Community Area",
  h."Birth Rate",
  h."Mortality Rate"
FROM "State of Washington"."cte_perkins_concentrators_2022_cohort" e
JOIN "City of Chicago"."public_health_statistics__selected_public_health_indicators_by_chicago_community_area__historical" h
  ON e."DistrictName" = h."Community Area"
WHERE e."schoolyear" = '2022';
```

However, these portals provide only raw datasets (databases, tables, and schemas), but neither executable SQL workloads nor paired NL questions. As a result, they cannot be directly used for open-domain Text-to-SQL evaluation. We therefore develop a three-step synthesis pipeline (Figure 5) to generate NL-SQL pairs (Q, S): it preserves structural patterns of real SQL queries, populates query content using the public schemas, and generates corresponding NL questions via chain-of-thought promptings.

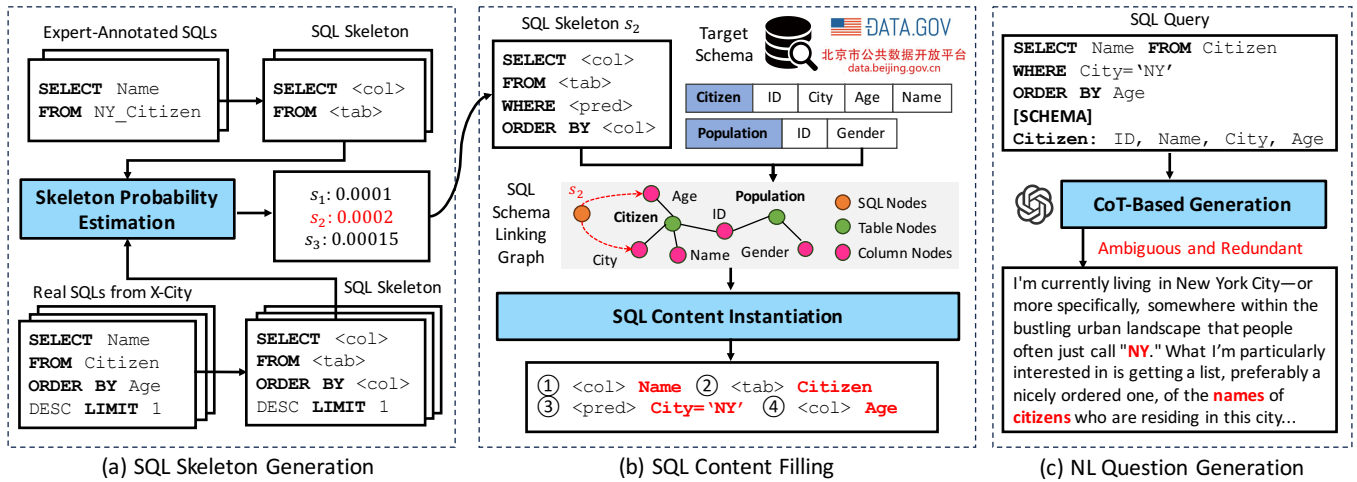


Figure 5: Our three-step data synthesis pipeline for generating realistic NL-SQL (Q, S) pairs.

Step 1: SQL Skeleton Generation. We introduce *SQL skeletons*, i.e., abstract templates that capture high-level structural patterns of SQL queries. To approximate real-world usage, we estimate a *skeleton distribution* from real queries and a small set of expert-designed SQL queries, and sample from this distribution to obtain representative skeletons (see Section 3.2.1).

Step 2: SQL Content Filling. We instantiate each skeleton based on concrete schemas. Databases are represented as a *schema graph*, and a linking algorithm maps skeleton components to specific tables, columns, and predicates, ensuring semantic consistency and enabling cross-database composition (see Section 3.2.2).

Step 3: NL Question Generation. We generate NL questions using an LLM-based Chain-of-Thought (CoT) procedure conditioned on schema context and SQL semantics, which can include ambiguity and redundancy observed in user inputs (see Section 3.2.3).

Using this pipeline, we construct a bilingual synthetic dataset with 13,000 Text-to-SQL examples across 52 databases and 13,004 tables: 7,000 examples from the Beijing platform [1] and 6,000 from the U.S. portal [2]. These examples cover the four query categories introduced in Section 2 (single-DB lookups, multi-table joins, data-lake table retrieval, and cross-database compositions), allowing TACO-OpenData to reflect the open-domain query patterns observed in TACO-SmartCity.

3.2.1 SQL Skeleton Generation. We model the structural patterns of real SQL queries using **SQL skeletons** [14]. A skeleton represents the query structure using only SQL commands and operators, with placeholders such as [tab], [col], and [val] for schema- and value-specific content.

Example 3.3. Figure 5(a) illustrates skeleton extraction: replacing all identifiers in a real SQL query with placeholders yields an SQL skeleton, e.g., “SELECT [col] FROM [tab] ORDER BY [col] DESC LIMIT 1”.

We extract SQL skeletons from two sources: (1) existing Text-to-SQL queries from Beijing, capturing generic open-domain patterns

such as ambiguous questions, unspecified databases, and cross-database querying, and (2) a small set of expert-annotated SQL queries over the open-data schemas, encoding domain-specific logic from the Beijing and U.S. portals.

To model the *distribution* of SQL structures, we follow prior work [45, 47] and decompose each SQL query into an abstract syntax tree (AST) represented as a sequence of context-free grammar (CFG) rules. We estimate rule probabilities from frequency counts and combine rules from the two sources via a weighted mixture, yielding a global distribution over SQL structures.

Skeletons are generated top-down by sampling CFG rules from this distribution. This ensures syntactic validity and aligns generated skeletons with prevalent query patterns in the target environment. Finally, we select a diverse subset of skeletons: high-probability skeletons cover frequent patterns, while lower-probability ones capture realistic long-tail structures [33, 46]. Section 3.3 reports the number of distinct skeletons observed in the final dataset.

3.2.2 SQL Content Filling. Given the generated skeletons, we instantiate them into executable SQL queries over the target schemas (Figure 5(b)). We represent each schema as an *SQL-Schema Linking Graph (SSLG)*, whose nodes correspond to tables and columns and whose edges encode relationships such as primary-foreign key links [12, 40]. The filling procedure consists of three stages:

(1) *Candidate Selection via SSLG.* We construct the SSLG over the database schema. The graph encodes valid structural connections and is used to propose candidates when instantiating clauses such as FROM and JOIN.

(2) *Incremental Filling of Query Components.* Guided by the skeleton (e.g., SELECT, FROM, WHERE, GROUP BY), we instantiate missing elements step by step. The SSLG constrains choices to semantically valid tables, columns, and join paths, ensuring that generated queries comply with schema constraints and foreign-key structure.

(3) *LLM-Guided Predicate Generation.* To produce realistic predicates and values, we employ an LLM conditioned on the skeleton and SSLG context [33, 41]. For example, when instantiating a WHERE

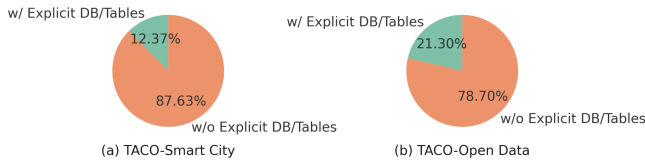


Figure 6: Proportion of NL questions with explicit DB/table mentions in TACO-SmartCity and TACO-OpenData.

clause, the model generates filters (e.g., date ranges, regions, or thresholds) consistent with column types. The resulting SQL queries are executed to verify syntactic validity and non-empty outputs; invalid queries will be discarded or repaired in subsequent iterations.

3.2.3 NL Question Generation. Given a synthesized SQL query, this step generates an NL question that is consistent with the SQL semantics while exhibiting ambiguity and redundancy typical of open-domain inputs. We use an LLM-based Chain-of-Thought (CoT) procedure with three phases:

(1) *SQL Semantic Interpretation.* The model parses the SQL structure (e.g., SELECT targets, JOIN conditions, WHERE filters) together with schema metadata (table/column descriptions and relationships) to derive a natural-language description of the query semantics.

(2) *Query Intent Inference.* The model maps the derived semantics to plausible user intents (e.g., monitoring service coverage, comparing districts, tracking trends).

(3) *Scenario-Conditioned Query Generation.* The model generates NL questions by conditioning on the inferred intent and schema context, adding contextual details (e.g., time, location, user role) and allowing colloquial phrasing and implicit constraints, while preserving semantic equivalence to the SQL.

3.2.4 Quality Control of the Synthesis Process. Quality control is embedded in the synthesis pipeline. During NL generation, we use CoT prompts that explicitly ask the LLM to verify whether the generated query is consistent with the SQL semantics and schema context, and low-confidence outputs would be regenerated. In addition, a lightweight LLM is used as a consistency checker: for each candidate NL-SQL pair, it scores semantic alignment with the associated schema, and only high-scoring pairs are retained. We further conduct a stratified human evaluation on sampled synthetic examples (stratified by SQL complexity) to verify semantic correctness, fluency, and consistency. As reported in Table 2, the synthetic NL-SQL pairs obtain average scores of 4.57/5 for semantic correctness and 4.61/5 for fluency, with strong inter-annotator agreement, providing evidence for the reliability of the synthesized data.

3.3 Dataset Statistics of TACO

This section provides a quantitative characterization of TACO, focusing on (i) SQL/NL complexity, (ii) implicit schema references, (iii) cross-database usage, and (iv) structural diversity. Beyond reporting dataset scale, these analyses characterize the sources of difficulty in the benchmark and compare the synthetic TACO-OpenData with the real TACO-SmartCity subset, showing that the synthetic dataset preserves similar query patterns and structural properties.

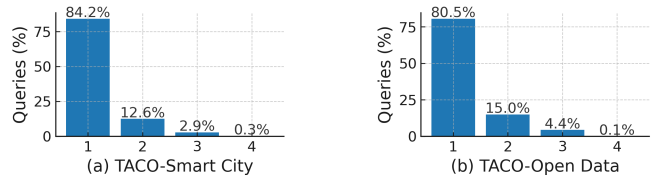


Figure 7: Number of databases involved in each query.

Table 2: Human evaluation of synthetic NL-SQL pairs.

Dataset	#Samples	Correct	Minor Fix	Incorrect
TACO-OpenData-Beijing	120	82%	13%	5%
TACO-OpenData-US	120	79%	15%	6%
Overall	240	80.5%	14%	5.5%

Dataset	Fluency (1-5)	Realism (1-5)	Agreement κ
TACO-OpenData-Beijing	4.3	4.1	0.78
TACO-OpenData-US	4.2	4.0	0.75
Overall	4.25	4.05	0.77

3.3.1 Statistics of TACO-SmartCity. The real-world subset contains 113 tables and 1,500 user queries from 31 municipal departments. Key statistics are summarized below.

(1) *NL and SQL Length.* Figure 2 reports average token counts across benchmarks. TACO-SmartCity queries are significantly longer (SQL: 51.43 tokens; NL: 89.30 tokens) than *Spider* and *BIRD*, indicating increased linguistic redundancy and more complex query intents.

(2) *Implicit Schema References.* As shown in Figure 6, 87.63% of NL questions do *not* explicitly mention any database or table, highlighting the need for table retrieval and schema linking.

(3) *Cross-Database Querying.* Figure 7 shows that about 20% of queries involve multiple databases. Many require multi-step pipelines with intermediate result reuse, increasing complexity relative to closed-domain benchmarks.

3.3.2 Statistics of TACO-OpenData. We evaluate whether TACO-OpenData exhibits characteristics comparable to TACO-SmartCity. The synthetic data contains 13,000 NL-SQL pairs across 52 databases with 13,004 tables.

(1) *Token Length and Redundancy.* Figure 2 shows that SQL and NL token-length distributions of the synthetic data are similar to those of the real queries, indicating comparable levels of redundancy.

(2) *Implicit Schema References.* As shown in Figure 6, 78-87% of TACO-OpenData queries do not explicitly mention a database or table, aligning with the TACO-SmartCity distribution.

(3) *Cross-Database Query Patterns.* Figure 7 shows that the synthetic data includes frequent cross-database querying behaviors, with many queries combining information from 2-4 databases, consistent with the real subset.

(4) *SQL Structural Complexity.* Figure ?? reports that synthetic SQL queries in TACO-OpenData are comparable to the real dataset in TACO-SmartCity in length, number of joins, and subquery depth, following heavy-tailed distributions. The prevalence of long queries,

multi-hop joins, and nested subqueries exceeds that of benchmarks such as Spider and BIRD.

(5) *Diversity of SQL Skeletons.* We count distinct SQL skeletons and find 2,317 unique skeletons in the synthetic dataset, suggesting that the workload is not limited to a small set of repeated structures.

Summary. TACO-OpenData closely matches TACO-SmartCity along multiple measurable dimensions while remaining public and reproducible. In particular, the two subsets exhibit similar distributions in SQL complexity, implicit schema linking, cross-database querying, and template diversity, indicating that the synthesis procedure preserves key workload characteristics of open-domain Text-to-SQL.

4 TACO-SQL FOR OPEN-DOMAIN TEXT-TO-SQL

To illustrate the use of TACO and evaluate the strengths and limitations of current approaches, we introduce TACO-SQL, an LLM-based framework for open-domain Text-to-SQL, as shown in Figure 8. TACO-SQL is designed to address key challenges such as query ambiguity, large-scale table linking, and the complexity of cross-database query planning. The framework consists of three key components: (1) *Question Rewriting*, (2) *Table Linking*, and (3) *Query Planning and Generating*, each addressing a specific aspect of open-domain Text-to-SQL.

Remarks. The primary contribution of this paper is the TACO benchmark and its analysis of open-domain Text-to-SQL. TACO-SQL only serves as a reference baseline that exposes representative failure modes and supports modular replacement of components in future work.

4.1 Question Rewriting

In this component, we employ an LLM to rewrite noisy or an ambiguous user NL question into an explicit intent statement while preserving entities and constraints required for downstream table linking. The rewriting step normalizes ambiguous expressions, removes redundant conversational content, and makes implicit conditions explicit (e.g., identifiers, attributes, or target objects).

For instance, the original query “*I need my employee records to finish a report. Please tell me where I can get my employee records. My employee ID is E12345, Thanks.*” is rewritten as “*Find storage locations for employee records with ID E12345*”. The rewritten query explicitly exposes the target entity (employee records) and constraint (ID E12345), enabling more reliable table retrieval in the next stage.

4.2 Table Linking

This component identifies relevant tables from large, heterogeneous databases given the rewritten query, reducing the schema search space for downstream query planning and SQL generation [13, 42, 52, 53]. As illustrated in Figure 8, table linking consists of two stages: an *offline contrastive fine-tuning* phase that aligns query and schema representations, and an *online retrieval* phase that retrieves candidate tables at inference time. In the current implementation, table linking operates at the *whole-query* level: the NL question is treated as a single unit to retrieve the top- k tables from the entire data lake, while finer-grained decomposition is handled downstream in the query planning stage (Section 4.3).

Offline Contrastive Fine-Tuning. In the offline stage, we construct a labeled dataset $\mathcal{D} = \{(q_i, T_i^+)\}_{i=1}^N$, where each query q_i is associated with a set of relevant tables T_i^+ . Then, we adopt a dual-encoder architecture that embeds both the queries and table schema descriptions into a shared semantic space. Specifically, let $f_q(q)$ and $f_t(t)$ denote the query and table encoders, respectively. The similarity between a query q and a table t is computed using cosine similarity:

$$\text{sim}(q, t) = \frac{f_q(q) \cdot f_t(t)}{\|f_q(q)\| \|f_t(t)\|}.$$

To ensure that matching query-table pairs are closely aligned, while non-matching pairs are separated, we optimize the encoders using the InfoNCE loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp(\text{sim}(q_i, t_i^+)/\tau)}{\exp(\text{sim}(q_i, t_i^+)/\tau) + \sum_{t \in \mathcal{N}_i} \exp(\text{sim}(q_i, t)/\tau)},$$

where τ is a temperature parameter and \mathcal{N}_i denotes a set of negative tables for query q_i . The objective increases the similarity of positive query-table pairs relative to negatives, encouraging alignment between natural-language queries and schema descriptions [43, 51].

Online Retrieval. In the online stage, the fine-tuned dual-encoder is employed to process incoming user queries. For a query q , we compute its embedding $f_q(q)$, while the embeddings of candidate tables $\{f_t(t_j)\}_{j=1}^M$ are pre-computed and indexed for efficiency. We then compute cosine similarity $\text{sim}(q, t_j)$ between the query and each table, and rank the candidate tables. The top- k relevant tables are selected as:

$$\mathcal{T}^* = \{t_j \mid \text{rank}(\text{sim}(q, t_j)) \leq k\}.$$

To improve recall under heterogeneous naming and phrasing, we apply two augmentation strategies [29]. *Query variants* are generated via synonym substitution and paraphrasing, and *schema augmentation* expands abbreviations and enriches table descriptions [6, 34, 48]. Section 5 reports an ablation comparing whole-query linking with a subquery-aware variant and a hybrid strategy, measured by gold-table recall and subquery coverage.

4.3 Query Planning and Generating

Queries in open-domain Text-to-SQL may span multiple databases and contain implicit dependencies across sub-queries. To handle such complexity, we transform the rewritten question q , the retrieved tables \mathcal{T}^* and associated schema text \mathcal{D}^* (e.g., table descriptions) into a structured execution plan \mathcal{P}^* used for SQL generation.

Structured Execution Plan. We construct a prompt containing q , descriptions of the tables in \mathcal{T}^* , and key schema information, and provide the prompt to an LLM to produce an execution plan:

$$\mathcal{P}^* = f_{\text{plan}}(q, \mathcal{T}^*, \mathcal{D}^*),$$

where $f_{\text{plan}}(\cdot)$ denotes the planning function implemented via LLM prompting. The output \mathcal{P}^* is an ordered list of sub-queries $\{p_1, p_2, \dots, p_n\}$, each of which is associated with its corresponding table(s) and operation, specifying the execution order. This prompting-based strategy leverages the LLM’s reasoning capabilities to infer sub-query dependencies, eliminating the need for manually crafted dependency graphs. The resulting plan ensures

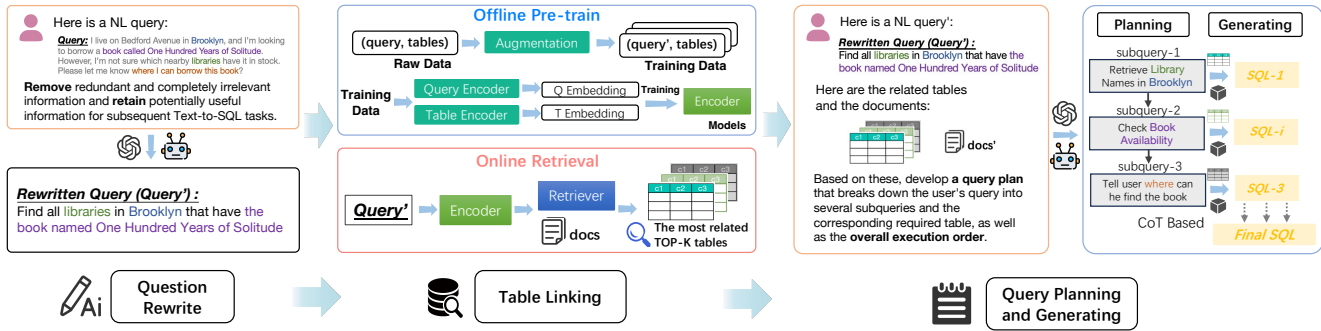


Figure 8: An overview of the TACO-SQL framework for open-domain Text-to-SQL.

that inter-table operations are logically and contextually sequenced. Each subquery in \mathcal{P}^* corresponds to one SQL statement, and the resulting plan forms an *SQL-only* pipeline whose intermediate results are combined within the database engine using standard SQL operators, consistent with the problem formulation in Section 2.1.

SQL Generation via Step-by-Step Reasoning. Unlike direct NL-to-SQL generation, we leverage a step-by-step method to generate intermediate steps that guide the final SQL generation. Given the execution plan \mathcal{P}^* , relevant tables \mathcal{T}^* , and schema \mathcal{D}^* , we decompose the NL question into several subqueries as a sequence of reasoning steps $C = \{c_1, c_2, \dots, c_n\}$, along with corresponding table descriptions and key schema details. Then, we apply an existing Text-to-SQL model to generate an SQL query:

$$S_{\text{draft}} = g(C, \mathcal{P}^*, \mathcal{T}^*, \mathcal{D}^*),$$

where $g(\cdot)$ denotes the SQL generation model.

Iterative Refinement and Error Correction. We employ an iterative refinement procedure driven by automatic error detection and feedback. Specifically, intermediate and final SQL queries are executed on the underlying databases, and execution errors are returned to the LLM to revise the query. All refinement and result computation are performed entirely within the database engine using SQL, without external scripts. The iteration terminates once an executable SQL query is produced, which is then evaluated against the gold SQL in TACO.

5 EXPERIMENTS

In this section, we evaluate the TACO benchmark and the proposed TACO-SQL pipeline through three key sets of experiments.

- **Exp-1** evaluates existing Text-to-SQL approaches on TACO without additional preprocessing, measuring performance under the open-domain setting.
- **Exp-2** analyzes how query-level factors, such as NL length, schema size, and cross-database complexity, affect performance.
- **Exp-3** examines the contribution of each component in TACO-SQL through ablation studies, e.g., *Question Rewriting*, *Table Linking*, and *Query Planning*.

5.1 Experimental Setup

This section first presents the experimental setup to evaluate Text-to-SQL approaches on TACO.

Baselines. We evaluate a diverse set of Text-to-SQL approaches under the open-domain setting.

(1) *Base LLMs.* General-purpose LLMs such as GPT-4, GPT-4o, GPT-o1, Llama-3, and DeepSeek-v3 are evaluated in the zero-shot setting. This setting measures model performance when applied directly to large and heterogeneous schemas without task-specific training.

(2) *LLM-Based Methods.* Approaches such as *DIN-SQL* [31] and *MAC-SQL* [39] rely on structured prompting or decomposition strategies for reasoning over complex queries.

(3) *SFT-Based Methods.* Models including *CodeS-33B* [22] and *Qwen2.5-Coder-32B* are fine-tuned on SQL and code corpora. This setting evaluates the effect of supervised fine-tuning when the target database is not specified.

(4) *Hybrid Methods.* Systems such as *CHESS* [36] and *ZeroNL2SQL* [10] combine LLM reasoning with schema linking or iterative refinement, representing practical strategies for Text-to-SQL.

Method Configurations. All models are evaluated under a unified and reproducible setup, including consistent prompting templates, schema-visibility settings, and decoding configurations (subject to model-specific context limits). As TACO is released as an evaluation-only benchmark, all examples are treated as a held-out test set. To support reproducibility, we provide prompt templates, configuration files, and evaluation scripts in our GitHub repository.

Schema Visibility Settings. Models receive the raw user query together with the full schema collection (database–table–column hierarchy), reflecting the open-domain setting and exposing models to large heterogeneous schema spaces. In some cases, context-length limits lead to partial truncation and affect performance. Moreover, to ablate the impact of table retrieval, Exp-3 additionally reports results where models are provided with the gold relevant tables, so that downstream SQL reasoning can be evaluated independently of the table-linking stage.

Evaluation Metric. We adopt Execution Accuracy (EX) [49], following prior benchmarks such as BIRD [23]. EX measures whether the result returned by the predicted SQL matches the result of the ground-truth SQL.

5.2 Exp-1: Baseline Performance on TACO

This experiment evaluates how existing Text-to-SQL approaches, which are largely developed for *closed-domain* settings, perform

Table 3: Baseline Performance on the TACO benchmark, reported as Execution Accuracy (EX) in %. The best results in each dataset are shown in bold. A hyphen (“-”) indicates that the dataset cannot be evaluated by closed-source models due to privacy restrictions, and therefore no result is reported.

Baselines	TACO Benchmarks		
	TACO-SmartCity	TACO-Beijing	TACO-US
GPT-4	-	11.50	12.84
GPT-4o	-	12.06	13.16
GPT-o1	-	14.18	15.32
DeepSeek-v3	-	11.30	12.28
Llama3-70B	5.33	6.66	8.16
DIN-SQL (GPT-4o)	-	7.48	8.02
MAC-SQL (GPT-4o)	-	6.90	7.64
CodeS-33B	8.47	9.02	9.96
Qwen2.5-Coder-32B	5.91	6.52	7.28
CHESS	-	10.10	11.84
Zero-NL2SQL	-	7.62	8.26

under the open-domain conditions of TACO. All baselines are evaluated *without* auxiliary preprocessing; models receive only the raw user question and the full collection of database schemas, with no question rewriting, table linking, or planning modules. This configuration measures performance when Text-to-SQL approaches are directly applied to the open-domain setting.

Table 3 reports execution accuracy across a broad range of model families. All models achieve low execution accuracy (6 – 15%). This result indicates that TACO presents challenges even for strong LLMs. The performance gap is consistent with factors such as ambiguous user intent, large schema search spaces, implicit table references, and cross-database reasoning, which are less emphasized in traditional benchmarks.

Among all baselines, GPT-based models (e.g., GPT-4o and GPT-o1) and DeepSeek-v3 achieve better results. For example, GPT-o1 reaches 14–15% on TACO-Beijing and TACO-US. The improvement over other baselines is consistent with the broader pretraining of general-purpose LLMs and their handling of long queries and large schema contexts. SFT-based models such as *CodeS-33B* and *Qwen2.5-Coder-32B* achieve moderate performance (8 – 10%), suggesting that SQL-focused fine-tuning alone does not address open-domain ambiguity or table-retrieval challenges. Task-specific prompting methods such as *DIN-SQL* and *MAC-SQL* perform comparably, but their performance decreases relative to closed-domain benchmarks.

Finding 1. Current Text-to-SQL approaches show limited performance under the open-domain setting. The low execution accuracy demonstrates that TACO reveals limitations in both base LLMs and SQL-tuned models, particularly for queries without explicit table references and those requiring multi-step cross-database reasoning. These observations motivate the use of modular components, such as question rewriting, table linking, and structured planning, to address the challenges of open-domain Text-to-SQL.

5.3 Exp-2: Impact Factors on Open-Domain Text-to-SQL Performance

In this experiment, we analyze how the factors, query length, schema size (number of tables), and cross-database complexity,

affect model performance on TACO. The goal is to analyze performance at a finer granularity than aggregated EX scores and examine which properties of open-domain queries are associated with performance variation, complementing the dataset statistics in Section 3.3. Figures 9, 10 and 11 report execution accuracy bucketed by these factors, and Table 4 lists representative results for selected models. We have the following observations.

Effect of Query Length. Figure 9 shows that EX generally decreases as query length increases. On TACO-Beijing, for example, GPT-o1 achieves around 15% EX for short queries (0-40 tokens), dropping to roughly 13% for long queries (81+ tokens). A similar pattern appears on TACO-US for models such as CodeS-33B and Qwen2.5-Coder-32B. This trend is consistent with longer queries containing more redundancy and implicit conditions, which complicate NL-to-SQL mapping.

Effect of Schema Size (#Tables). Figure 10 reports the effect of schema size on performance. On TACO-Beijing, GPT-o1 achieves over 25% EX when the relevant schema contains 0-25 tables, but the EX drops below 10% when more than 50 tables are present. Similar patterns hold across models and on TACO-US. The results are consistent with the need to identify relevant tables before SQL generation.

Effect of Cross-Database Complexity. Figure 11 shows lower performance on cross-database queries. On TACO-US, GPT-o1 exceeds 20% EX on single-database questions but drops to only a few percent for cross-database ones, with similar gaps across baselines. Cross-database queries require coordinating multiple query steps and maintaining consistency across databases, which increases the complexity of the generation process.

Finding 2. Text-to-SQL performance decreases with longer queries, larger schema sizes, and cross-database complexity. Together with the structural statistics in Section 3.3, these results indicate that TACO captures the key challenges of open-domain Text-to-SQL.

5.4 Exp-3: The TACO-SQL Framework

We evaluate the TACO-SQL framework by measuring the incremental contribution of its components on TACO.

Specifically, we evaluate the framework under five progressively stronger configurations: (1) **Origin**: Direct SQL generation from the raw query with access to the full schema. (2) **QR**: Applying Question Rewriting on top of **Origin**. (3) **TL**: Providing the oracle set of gold relevant tables, ablating the table-retrieval stage and evaluating downstream SQL generation. (4) **QR+TL**: Applying retrieval-based Table Linking after Question Rewriting. (5) **TACO-SQL = QR+TL+QP**: Adding Query Planning to the QR+TL configuration.

Overall Results. Table 4 summarizes results across the three datasets. We have the following observations.

(1) *Effect of Table Linking (QR+TL).* Across models, applying Table Linking after Question Rewriting yields the largest incremental performance gain. For example, GPT-o1 increases from 14.18% (TACO-Beijing, Origin) to 32.74% (QR+TL), more than doubling EX. Similar gains appear for CodeS-33B, CHESS, and Qwen2.5-Coder-32B. These results indicate that providing retrieved tables instead of the full schema could significantly improve performance in the open-domain Text-to-SQL setting.

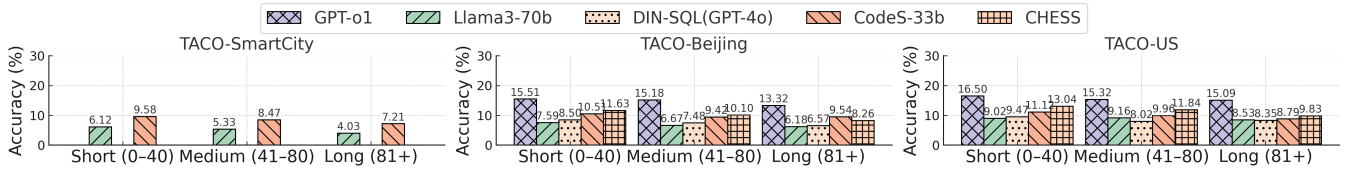


Figure 9: Effect of NL Query Length (Execution accuracy decreases with longer queries).

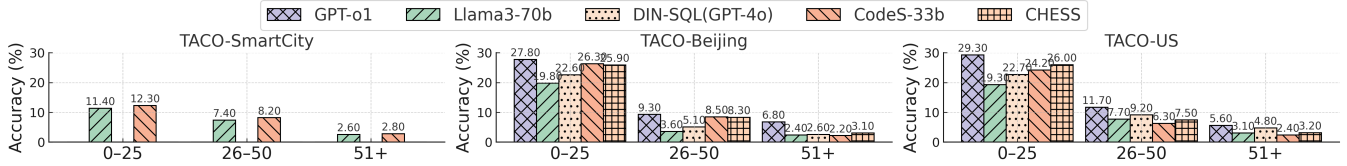


Figure 10: Effect of Schema Size (Execution accuracy decreases as the number of tables increases).

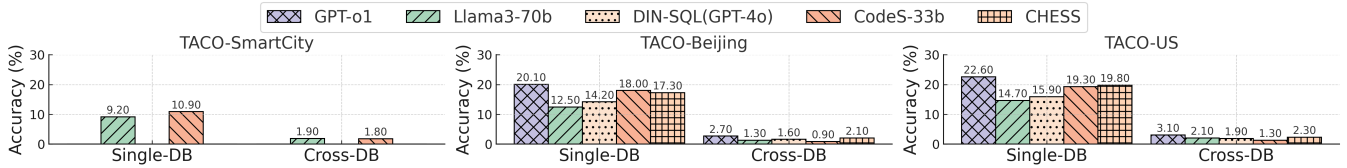


Figure 11: Effect of Cross-Database Complexity (Multi-database queries show lower accuracy than single-database queries).

Table 4: Model Performance on the TACO Benchmark Under Different Settings. Best results for each model are bolded. Origin: raw query with full schemas. QR: Question Rewriting. TL: gold relevant tables (no table retrieval). QR+TL: Question Rewriting with table linking. QR+TL+QP (TACO-SQL): adds Query Planning. A hyphen (“-”) indicates results are unavailable for closed-source models due to privacy restrictions.

Models	TACO-SmartCity					TACO-Beijing					TACO-US				
	Origin	QR	TL	QR+TL	QR+TL+QP (TACO-SQL)	Origin	QR	TL	QR+TL	QR+TL+QP (TACO-SQL)	Origin	QR	TL	QR+TL	QR+TL+QP (TACO-SQL)
GPT-4	-	-	-	-	-	11.50	14.38	21.72	25.52	26.80	12.84	16.92	25.44	28.36	30.04
GPT-4o	-	-	-	-	-	12.06	16.42	26.44	28.60	28.94	13.16	16.28	30.08	31.76	30.04
GPT-o1	-	-	-	-	-	14.18	18.58	29.16	32.74	34.92	15.32	22.06	32.90	33.24	35.18
DeepSeek-v3	-	-	-	-	-	11.30	11.28	19.22	21.74	22.98	12.28	13.08	24.06	25.50	23.66
Llama3-70B	5.33	6.54	14.56	19.53	19.60	6.66	8.53	17.78	21.26	22.73	8.16	9.60	19.40	20.58	19.67
DIN-SQL(GPT-4o)	-	-	-	-	-	7.48	9.92	18.46	21.76	20.52	8.02	11.38	24.12	25.82	26.40
MAC-SQL(GPT-4o)	-	-	-	-	-	6.90	7.88	15.06	18.34	19.20	7.64	7.92	17.88	19.60	19.88
CodeS-33B	8.47	10.12	24.86	26.22	28.18	9.02	13.56	25.94	28.32	28.78	9.96	11.50	27.38	29.18	27.66
Qwen2.5-Coder-32B	5.91	7.65	18.32	22.83	19.79	6.52	8.52	19.74	22.06	20.34	7.28	9.40	15.20	18.14	20.38
CHES	-	-	-	-	-	10.10	10.02	24.94	28.30	30.22	11.84	13.50	27.12	31.44	32.98
Zero-NL2SQL	-	-	-	-	-	7.62	8.50	23.88	26.82	27.46	8.26	9.10	23.04	25.74	25.36

(2) *Effect of Question Rewriting (QR)*. Applying Question Rewriting alone produces modest but consistent improvements. For example, GPT-4 increases from 11.50% to 14.38% on TACO-Beijing, with similar patterns across other models. QR clarifies query intent and normalizes expressions, but does not change the set of candidate tables provided for SQL generation, and therefore yields smaller gains than configurations that include Table Linking.

(3) *Evaluation with Gold Tables*. Providing the gold relevant tables improves performance but remains below the QR+TL configuration. For example, GPT-4o achieves 26.44% under TL versus 28.60% under QR+TL on TACO-Beijing. The performance gap suggests that errors

also arise in downstream SQL generation, such as join construction and subquery formulation.

(4) *Effect of Query Planning (QP)*. Adding Query Planning achieves additional improvements, though smaller than those from Table Linking. For example, the EX of GPT-o1 increases from 32.74% (QR+TL) to 34.92% (QR+TL+QP). Moreover, the improvement is larger on cross-database queries. As shown in Table 5, GPT-o1 increases from 10.74% to 16.12% on TACO-Beijing for cross-database queries, and CodeS-33B from 5.28% to 10.10%. This pattern is consistent with settings where multiple query steps must be coordinated across databases. QP mainly affects multi-step SQL composition.

Table 5: Cross-Database Performance With vs. Without Query Planning (QP) (EX %).

Models	TACO-SmartCity		TACO-Beijing		TACO-US	
	w/o QP	w/ QP	w/o QP	w/ QP	w/o QP	w/ QP
GPT-4	-	-	5.52	11.38	6.36	13.94
GPT-4o	-	-	8.60	12.30	8.76	14.06
GPT-o1	-	-	10.74	16.12	12.24	18.54
DeepSeek-v3	-	-	7.74	11.50	7.62	12.68
Llama3-70B	3.53	7.10	4.26	8.38	6.58	9.86
DIN-SQL(GPT-4o)	-	-	7.12	10.04	8.82	13.94
MAC-SQL(GPT-4o)	-	-	6.34	9.76	8.66	10.02
CodeS-33B	5.28	10.10	6.32	9.24	7.18	11.52
Qwen2.5-Coder-32B	4.83	9.47	6.06	8.28	7.14	9.50
CHESS	-	-	8.38	12.60	9.44	13.78
Zero-NL2SQL	-	-	6.82	10.50	7.74	11.12

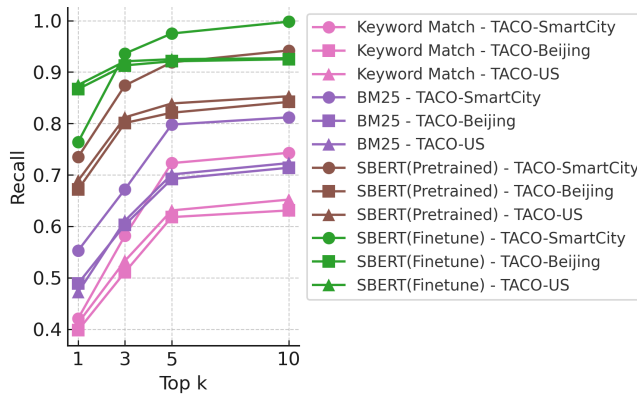


Figure 12: Evaluation on table linking performance under varying top-k (Larger k increases recall but lower precision; k = 5 provides a balanced setting).

Without QP, models tend to express multiple operations in a single query, leading to incorrect joins, missing intermediate filtering, or misplaced aggregation. The structured plan decomposes the task into ordered subqueries, separating table selection, filtering, and aggregation. This is more useful in cross-database queries, where intermediate results should be produced and reused across databases rather than represented as a single flat query.

(5) Impact Across Model Families. General-purpose LLMs such as GPT-o1 and GPT-4o achieve the strongest overall performance. However, fine-tuned models (e.g., CodeS-33B) and hybrid systems (e.g., CHESS) benefit more from the TACO-SQL pipeline, suggesting that SQL-oriented training and structured reasoning components remain particularly helpful under open-domain conditions.

Effect of Table Linking Design. We compare four table-linking methods, keyword matching, BM25, SBERT (pretrained), and SBERT (fine-tuned), using Recall@k as the metric. As shown in Figure 12, increasing k from 1 to 5 improves recall (by 5%–10%), whereas further increases yield only marginal gains. Based on this trade-off between coverage and retrieval noise, we set k = 5 for all subsequent experiments.

We also implement an alternative **subquery-level linking** strategy and evaluate it alongside our default whole-query approach.

Table 6: Whole-Query vs. Subquery Table Linking (200 cross-table/database queries).

Strategy	Recall@k	Subquery Cov.
Whole-query linking	93.4%	91.2%
Sub-query linking	94.2%	92.6%

As shown in Table 6, the subquery-based method achieves performance comparable to whole-query linking, with only marginal differences across evaluation metrics. These results suggest that, under the open-domain conditions of TACO, the full query already provides sufficiently informative signals for table retrieval, and finer-grained decomposition yields limited additional benefit.

Finding 3. The results indicate that solving TACO requires three core capabilities: (i) NL query disambiguation (QR), (ii) table linking over large heterogeneous databases (TL), and (iii) multi-step and cross-database SQL planning (QP). The TACO-SQL framework consistently improves performance across model types, supporting its modular design and highlighting practical directions for future open-domain Text-to-SQL systems.

6 CONCLUSION AND FUTURE WORK

In this paper, we have introduced TACO, a benchmark for evaluating open-domain Text-to-SQL systems. TACO consists of two datasets: TACO-SmartCity, which contains 1,500 real NL-SQL pairs curated through multi-stage expert annotation, and TACO-OpenData, which provides 13,000 synthetic examples generated by a data synthesis pipeline that preserves real-world structural and semantic complexity. We have also presented TACO-SQL, an LLM-based baseline that integrates question rewriting, table linking, and query planning. Our experiments show a substantial performance gap to gold SQL, highlighting the difficulty of open-domain Text-to-SQL and establishing TACO as a challenging benchmark.

For future work, this benchmark raises several broader research directions. Open-domain Text-to-SQL requires evaluating not only final SQL correctness but also intermediate behaviors, including table discovery, ambiguity resolution, and multi-step planning. This motivates evaluation metrics beyond execution accuracy. In addition, real deployments operate over evolving data lakes where schemas and values change over time, and benchmarks that model dynamic environments can better evaluate robustness under schema drift. Finally, practical systems increasingly behave as agentic LLMs that interact with users and databases during problem solving. Handling underspecified intent may require clarification, feedback, and iterative querying. Thus, an interesting direction is to develop evaluation settings that measure interactive and multi-step reasoning rather than single-shot SQL generation.

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 62436010, 62441230, and 62402409) and the Scientific Research Innovation Capability Support Project for Young Faculty (Grant No. SRICSPYF-ZY2025001). We also thank the Beijing Big Data Centre for their collaboration and for providing real-world data and application scenarios.

REFERENCES

- [1] [n. d.]. Beijing Open Data Portal. <https://data.beijing.gov.cn/>
- [2] [n. d.]. The Home of the U.S. Government's Open Data. <https://data.gov/>
- [3] Adithya Bhaskar, Tushar Tomar, Ashutosh Sathe, and Sunita Sarawagi. 2023. Benchmarking and Improving Text-to-SQL Generation under Ambiguity. arXiv:2310.13659 [cs.CL] <https://arxiv.org/abs/2310.13659>
- [4] Andrés Camero and Enrique Alba. 2019. Smart City and information technology: A review. *Cities* 93 (2019), 84–94. doi:10.1016/j.cities.2019.04.014
- [5] Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xi-ang. 2023. Dr.Spider: A Diagnostic Evaluation Benchmark towards Text-to-SQL Robustness. arXiv:2301.08881 [cs.CL] <https://arxiv.org/abs/2301.08881>
- [6] Hanjie Chen and Yangfeng Ji. 2022. Adversarial Training for Improving Model Robustness? Look at Both Prediction and Interpretation. arXiv:2203.12709 [cs.CL] <https://arxiv.org/abs/2203.12709>
- [7] 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>
- [8] Peter Baile Chen, Yi Zhang, and Dan Roth. 2024. Is Table Retrieval a Solved Problem? Exploring Join-Aware Multi-Table Retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 2687–2699. doi:10.18653/v1/2024.acl-long.148
- [9] Naihao Deng, Yulong Chen, and Yue Zhang. 2022. Recent Advances in Text-to-SQL: A Survey of What We Have and What We Expect. arXiv:2208.10099 [cs.CL] <https://arxiv.org/abs/2208.10099>
- [10] Ju Fan, Zihui Gu, Songyue Zhang, Yuxin Zhang, Zui Chen, Lei Cao, Guoliang Li, Samuel Madden, Xiaoyong Du, and Nan Tang. 2024. Combining Small Language Models and Large Language Models for Zero-Shot NL2SQL. *Proc. VLDB Endow.* 17, 11 (Aug. 2024), 2750–2763. doi:10.14778/3681954.3681960
- [11] Meihao Fan, Ju Fan, Nan Tang, Lei Cao, Guoliang Li, and Xiaoyong Du. 2025. AutoPrep: Natural Language Question-Aware Data Preparation with a Multi-Agent Framework. arXiv:2412.10422 [cs.CL] <https://arxiv.org/abs/2412.10422>
- [12] Yinpei Fu, Songtao Ye, and Hongjie Fan. 2024. Generate Text-to-SQL Queries based on Sketch Filling. *IEEE Access* (2024), 1–1. doi:10.1109/ACCESS.2024.3476927
- [13] Michael Glass, Mustafa Eyceoz, Dharmashankar Subramanian, Gaetano Rossiello, Long Vu, and Alfio Gliozzo. 2025. Extractive Schema Linking for Text-to-SQL. arXiv:2501.17174 [cs.DB] <https://arxiv.org/abs/2501.17174>
- [14] Zihui Gu, Ju Fan, Nan Tang, Lei Cao, Bowen Jia, Sam Madden, and Xiaoyong Du. 2023. Few-shot Text-to-SQL Translation using Structure and Content Prompt Learning. *Proc. ACM Manag. Data* 1, 2 (2023), 147:1–147:28. doi:10.1145/3589292
- [15] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. arXiv:1905.08205 [cs.CL] <https://arxiv.org/abs/1905.08205>
- [16] Moshe Hazoom, Vibhor Malik, and Ben Bogin. 2021. Text-to-SQL in the Wild: A Naturally-Occurring Dataset Based on Stack Exchange Data. arXiv:2106.05006 [cs.CL] <https://arxiv.org/abs/2106.05006>
- [17] Nan Huo, Xiaohan Xu, Jinyang Li, Per Jacobsson, Shipel Lin, Bowen Qin, Binyuan Hui, Xiaolong Li, Ge Qu, Shuzheng Si, Linheng Han, Edward Alexander, Xintong Zhu, Rui Qin, Ruihan Yu, Yiyao Jin, Feige Zhou, Weihao Zhong, Yun Chen, Hongyu Liu, Chenhao Ma, Fatma Özcan, Yannis Papakonstantinou, and Reynold Cheng. 2025. BIRD-INTERACT: Re-imagining Text-to-SQL Evaluation for Large Language Models via Lens of Dynamic Interactions. *arXiv preprint arXiv:2510.05318* (2025).
- [18] Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. Natural language to SQL: where are we today? *Proc. VLDB Endow.* 13, 10 (June 2020), 1737–1750. doi:10.14778/3401960.3401970
- [19] Chia-Hsuan Lee, Aleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic Evaluation of Text-to-SQL Parsers. arXiv:2106.11455 [cs.CL] <https://arxiv.org/abs/2106.11455>
- [20] 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>
- [21] Boyan Li, Jiayi Zhang, Ju Fan, Yanwei Xu, Chong Chen, Nan Tang, and Yuyu Luo. 2025. Alpha-SQL: Zero-Shot Text-to-SQL using Monte Carlo Tree Search. arXiv:2502.17248 [cs.DB] <https://arxiv.org/abs/2502.17248>
- [22] Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiqing Li, and Hong Chen. 2024. CodeS: Towards Building Open-source Language Models for Text-to-SQL. *Proc. ACM Manag. Data* 2, 3, Article 127 (May 2024), 28 pages. doi:10.1145/3654930
- [23] Jinyang Li, Binyuan Hui, Ge Qu, Jiayi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems* 36 (2024).
- [24] Jinyang Li, Xiaolong Li, Ge Qu, Per Jacobsson, Bowen Qin, Binyuan Hui, Shuzheng Si, Nan Huo, Xiaohan Xu, Yue Zhang, et al. 2025. SWE-SQL: Illuminating LLM Pathways to Solve User SQL Issues in Real-World Applications. *arXiv preprint arXiv:2506.18951* (2025).
- [25] Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. 2025. A Survey of NL2SQL with Large Language Models: Where are we, and where are we going? arXiv:2408.05109 [cs.DB] <https://arxiv.org/abs/2408.05109>
- [26] Xinyu Liu, Shuyu Shen, Boyan Li, Nan Tang, and Yuyu Luo. 2025. NL2SQL-BUGS: A Benchmark for Detecting Semantic Errors in NL2SQL Translation. arXiv:2503.11984 [cs.DB] <https://arxiv.org/abs/2503.11984>
- [27] Kyle Luoma and Arun Kumar. 2025. SNAILS: Schema Naming Assessments for Improved LLM-Based SQL Inference. *Proc. ACM Manag. Data* 3, 1, Article 77 (Feb. 2025), 26 pages. doi:10.1145/3709727
- [28] Peixian Ma, Boyan Li, Runzhi Jiang, Ju Fan, Nan Tang, and Yuyu Luo. 2024. A Plug-and-Play Natural Language Rewriter for Natural Language to SQL. arXiv:2412.17068 [cs.DB] <https://arxiv.org/abs/2412.17068>
- [29] Advait Parulekar, Liam Collins, Karthikeyan Shanmugam, Aryan Mokhtari, and Sanjay Shakkottai. 2023. InfoNCE Loss Provably Learns Cluster-Preserving Representations. In *Proceedings of Thirty Sixth Conference on Learning Theory (Proceedings of Machine Learning Research, Vol. 195)*, Gergely Neu and Lorenzo Rosasco (Eds.). PMLR, 1914–1961. <https://proceedings.mlr.press/v195/parulekar23a.html>
- [30] Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeoungh Chung, Shayan Taleai, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Özcan, 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>
- [31] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. arXiv:2304.11015 [cs.CL] <https://arxiv.org/abs/2304.11015>
- [32] Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. 2022. A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions. arXiv:2208.13629 [cs.CL] <https://arxiv.org/abs/2208.13629>
- [33] Zongyue Qin, Chen Luo, Zhengyang Wang, Haoming Jiang, and Yizhou Sun. 2024. Relational Database Augmented Large Language Model. arXiv:2407.15071 [cs.DB] <https://arxiv.org/abs/2407.15071>
- [34] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A. Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. 2021. Data Augmentation Can Improve Robustness. arXiv:2111.05328 [cs.CV] <https://arxiv.org/abs/2111.05328>
- [35] Irina Saparina and Mirella Lapata. 2024. AMBROSIA: A Benchmark for Parsing Ambiguous Questions into Database Queries. arXiv:2406.19073 [cs.CL] <https://arxiv.org/abs/2406.19073>
- [36] Shayan Taleai, 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>
- [37] Kapil Vaidya, Jialin Ding, Sebastian Kosa, David Kernert, Chuan Lei, Xiao Qin, Abhinav Tripathy, Ramesh Balan, Balakrishnan Narayanaswamy, and Tim Kraska. 2025. TailorSQL: An NL2SQL System Tailored to Your Query Workload. *arXiv preprint arXiv:2505.23039* (2025).
- [38] Bing Wang, Yan Gao, Zhoujun Li, and Jian-Guang Lou. 2023. Know What I don't Know: Handling Ambiguous and Unknown Questions for Text-to-SQL. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 5701–5714. doi:10.18653/v1/2023.findings-acl.352
- [39] Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Linzheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2024. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. arXiv:2312.11242 [cs.CL] <https://arxiv.org/abs/2312.11242>
- [40] Chenglong Wang, Alvin Cheung, and Rastislav Bodik. 2017. Synthesizing highly expressive SQL queries from input-output examples. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation (Barcelona, Spain) (PLDI 2017)*. Association for Computing Machinery, New York, NY, USA, 452–466. doi:10.1145/3062341.3062365
- [41] Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-Driven CoT: Exploring Faithful Reasoning in LLMs for Knowledge-intensive Question Answering. arXiv:2308.13259 [cs.CL] <https://arxiv.org/abs/2308.13259>
- [42] Tianshu Wang, Xiaoyang Chen, Hongyu Lin, Xianpei Han, Le Sun, Hao Wang, and Zhenyu Zeng. 2025. DBCopilot: Natural Language Querying over Massive Databases via Schema Routing. (2025).
- [43] Shitao Xiao, Zheng Liu, Weihao Han, Jianjin Zhang, Defu Lian, Yeyun Gong, Qi Chen, Fan Yang, Hao Sun, Yingxia Shao, Denvy Deng, Qi Zhang, and Xing Xie. 2022. Distill-VQ: Learning Retrieval Oriented Vector Quantization By Distilling

- Knowledge from Dense Embeddings. arXiv:2204.00185 [cs.IR] <https://arxiv.org/abs/2204.00185>
- [44] Keyan Xu, Dingzirui Wang, Xuanliang Zhang, Qingfu Zhu, and Wanxiang Che. 2025. Abacus-SQL: A Text-to-SQL System Empowering Cross-Domain and Open-Domain Database Retrieval. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.
- [45] Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. arXiv:1711.04436 [cs.CL] <https://arxiv.org/abs/1711.04436>
- [46] Jiayi Yang, Binyuan Hui, Min Yang, Jian Yang, Junyang Lin, and Chang Zhou. 2024. Synthesizing Text-to-SQL Data from Weak and Strong LLMs. arXiv:2408.03256 [cs.CL] <https://arxiv.org/abs/2408.03256>
- [47] Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task. arXiv:1810.05237 [cs.CL] <https://arxiv.org/abs/1810.05237>
- [48] Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2021. {SC}oRe: Pre-Training for Context Representation in Conversational Semantic Parsing. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=oyZxhRI2RiE>
- [49] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanell 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>
- [50] Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. SParC: Cross-Domain Semantic Parsing in Context. arXiv:1906.02285 [cs.CL] <https://arxiv.org/abs/1906.02285>
- [51] Li Zhang, Shuo Zhang, and Krisztian Balog. 2019. Table2Vec: Neural Word and Entity Embeddings for Table Population and Retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19)*. ACM. doi:10.1145/3331184.3331333
- [52] Shuo Zhang and Krisztian Balog. 2018. Ad Hoc Table Retrieval using Semantic Similarity. In *Proceedings of the 2018 World Wide Web Conference (Lyon, France) (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1553–1562. doi:10.1145/3178876.3186067
- [53] Shuo Zhang and Krisztian Balog. 2020. Web Table Extraction, Retrieval and Augmentation: A Survey. arXiv:2002.00207 [cs.IR] <https://arxiv.org/abs/2002.00207>
- [54] Shaolei Zhang, Ju Fan, Meihao Fan, Guoliang Li, and Xiaoyong Du. 2025. Deep-Analyze: Agentic Large Language Models for Autonomous Data Science. *CoRR* abs/2510.16872 (2025). doi:10.48550/ARXIV.2510.16872 arXiv:2510.16872
- [55] Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2024. MURRE: Multi-Hop Table Retrieval with Removal for Open-Domain Text-to-SQL. arXiv:2402.10666 [cs.CL] <https://arxiv.org/abs/2402.10666>
- [56] Yuxin Zhang, Meihao Fan, Ju Fan, Mingyang Yi, Yuyu Luo, Jian Tan, and Guoliang Li. 2025. Reward-SQL: Boosting Text-to-SQL via Stepwise Reasoning and Process-Supervised Rewards. arXiv:2505.04671 [cs.CL] <https://arxiv.org/abs/2505.04671>
- [57] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. arXiv:1709.00103 [cs.CL] <https://arxiv.org/abs/1709.00103>