香港大學自然語言處理實驗室
Natural Language Processing Group, The University of Hong Kong

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows

Fangyu Lei*, **Jixuan Chen***
The University of Hong Kong
25/04/2025

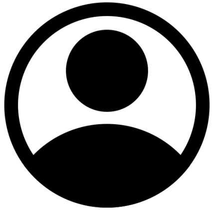
XLANG Lab
https://xlang.ai

Agenda

- Background and Motivations
- Spider 2.0
  - Features of Task
  - Annotation Pipeline
  - Statistics of Examples
- Spider 2.0 Experiments
- Spider 2.0 Analysis

Background and Motivations

- Automated Code Generation



Users

↓

Data Analysis
Business Intelligence

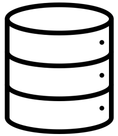
Coding

```
WITH ds AS (
    SELECT
        {{ dbt.date_trunc('day', 'date_day') }} AS
date_day
    FROM {{ ref('int_salesforce__date_spine') }}
),

            •••     [ + 100 lines omitted]

SELECT
            •••
LEFT JOIN task
    ON ds.date_day = task.activity_date
LEFT JOIN event on ds.date_day =
salesforce_event.activity_date
LEFT JOIN opportunities_created
ON ds.date_day = opportunities_created.created
LEFT JOIN opportunities_closed
ON ds.date_day = opportunities_closed.close_date
```

SQLs / Python

Execution Feedback

Database



Background and Motivations

- Text-to-SQL

- LLMs continue to set new records on various text-to-SQL leaderboards

- Text-to-SQL still attracts a lot of attention from academia and industry

- Does it mean text-to-SQL, data code generation, or **more broadly data tasks** have been **resolved**?
- Of course, NO!

Spider 1.0

Yale Semantic Parsing and Text-to-SQL Challenge

Rank	Model	Test
1 Nov 2, 2023	MiniSeek *Anonymous* Code and paper coming soon	**91.2**
1 Aug 20, 2023	DAIL-SQL + GPT-4 + Self-Consistency *Alibaba Group* (Gao and Wang et al.,'2023) code	**86.6**
2 Aug 9, 2023	DAIL-SQL + GPT-4 *Alibaba Group* (Gao and Wang et al.,'2023) code	**86.2**
3 October 17, 2023	DPG-SQL + GPT-4 + Self-Correction *Anonymous* Code and paper coming soon	85.6

Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task

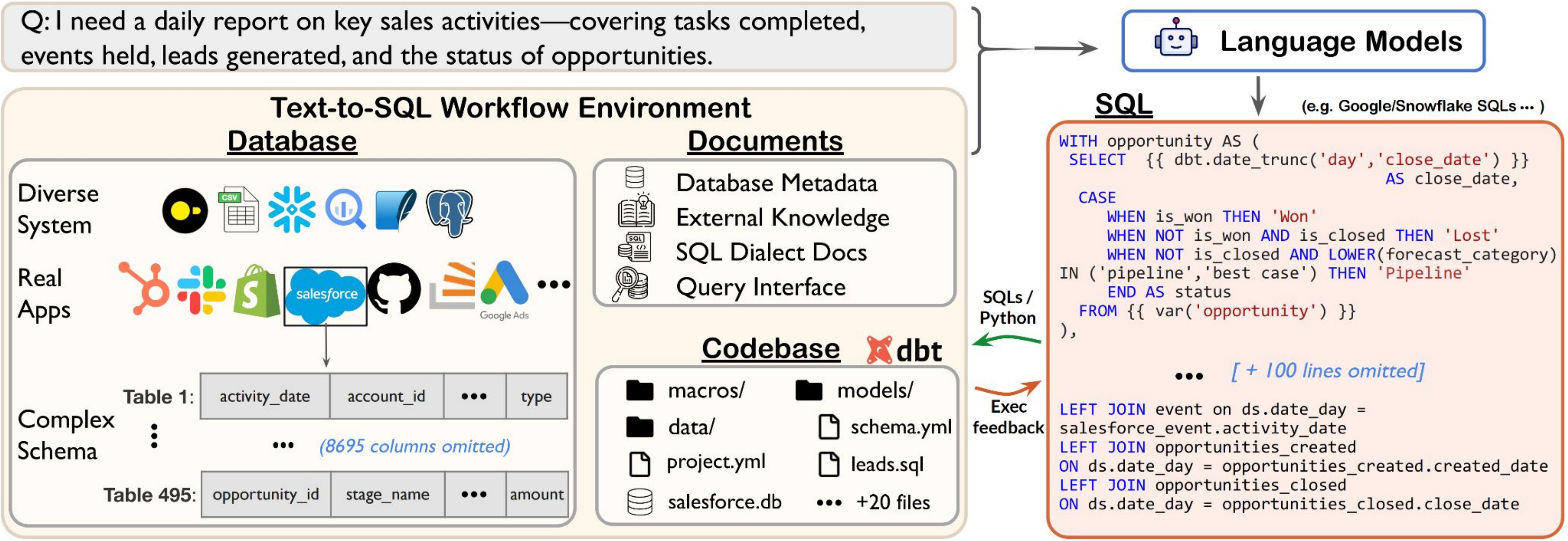
Background and Motivations

- Previous benchmarks **v.s.** Real industrial use
    - Non-industrial databases **v.s.** Diverse database systems, large-scale schemas
    - Simplistic SQL and questions **v.s.** Multiple complex SQL dialects, and functions
    - Instruction+schema -> SQL **v.s.** Project codebases, external knowledge, and various contexts to write SQL across multiple steps

Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task

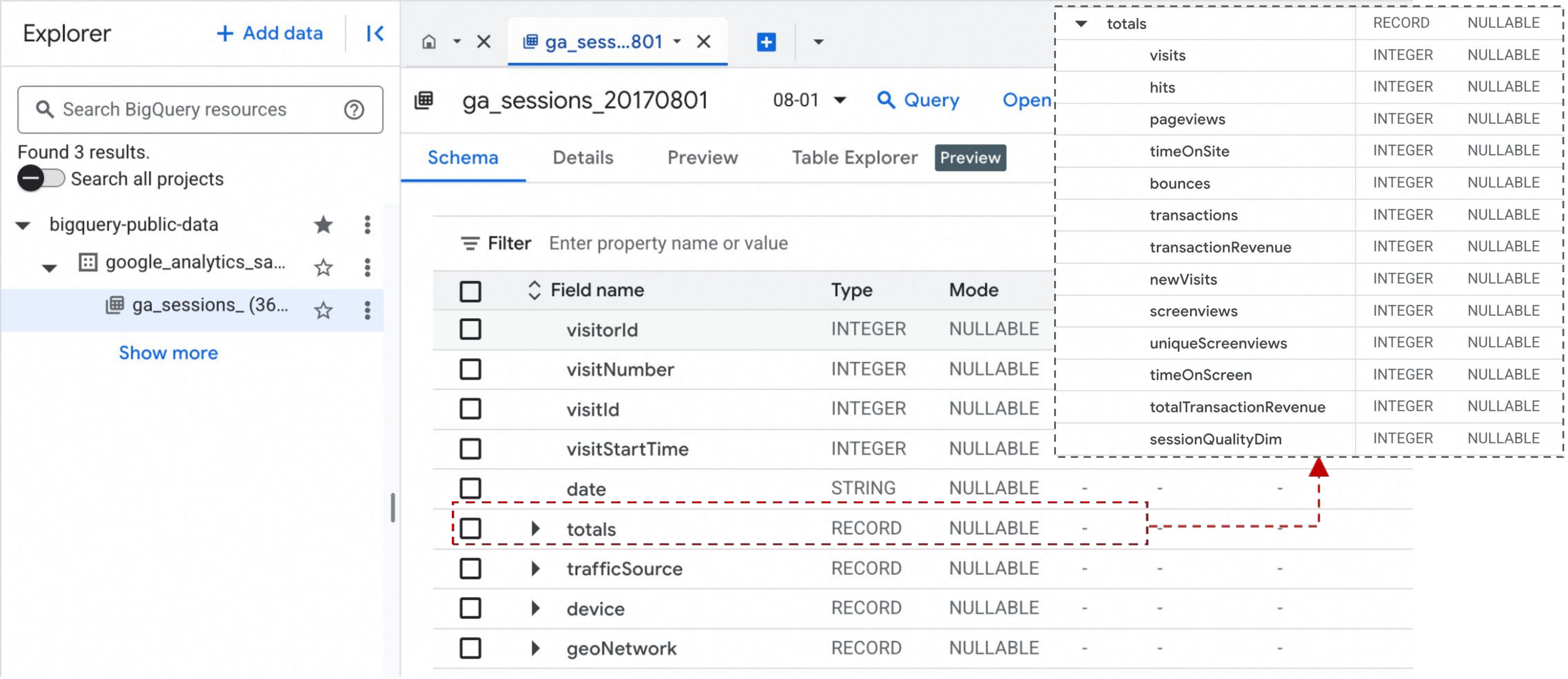
Features of Spider 2.0

- 🍊 Real complex cloud DBs (3000+ cols)
- 🍋 Multi-dialect SQL complexity (e.g., BigQuery, Snowflake)
- 🍎 Agentic coding workflows



Features of Spider 2.0

- Complex Schema of Cloud DBs



Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Features of Spider 2.0

- Complex SQLs with External Knowledge

What is PDP type? **Refer to External docs.**

Question:
Please find out what percentage of the page views on January 2, 2021, were for **PDP type pages**.

Gold SQL:

```
WITH base_table AS (
-- pulls relevant columns from relevant dates
SELECT
    event_name,
    event_date,
    ...
  FROM
    `bigquery-public-data.ga4_obfuscated_sample_ecommerce.events_*`
  WHERE
    _table_suffix = '20210102'
AND event_name IN ('page_view')
),
unnested_events AS (
-- unnests event parameters to get to relevant keys and values
  SELECT
    ...
 MAX(CASE WHEN c.key = 'page_title' THEN c.value.string_value END) AS page_title,
    MAX(CASE WHEN c.key = 'page_location' THEN c.value.string_value END) AS page_location
  FROM
    base_table,
    UNNEST (event_params) c
  GROUP BY 1,2,3
),
unnested_events_categorised AS (
-- categorizing Page Titles into PDPs and PLPs
SELECT
  *,
  CASE WHEN
    ARRAY_LENGTH(SPLIT(page_location, '/')) >= 5
AND
CONTAINS_SUBSTR(ARRAY_REVERSE(SPLIT(page_location, '/'))[SAFE_OFFSET(0)], '+') AND
   (LOWER(SPLIT(page_location, '/')[SAFE_OFFSET(4)]) IN
('accessories','apparel','brands',...,'wearables')
   OR LOWER(SPLIT(page_location, '/')[SAFE_OFFSET(3)]) IN
('accessories','apparel','brands',...,'wearables'))
THEN 'PDP'
  WHEN
NOT(CONTAINS_SUBSTR(ARRAY_REVERSE(SPLIT(page_location, '/'))[SAFE_OFFSET(0)], '+')) AND
(LOWER(SPLIT(page_location, '/')[SAFE_OFFSET(4)])
IN
('accessories','apparel','brands',...,'wearables')
OR LOWER(SPLIT(page_location, '/')[SAFE_OFFSET(3)])
IN
('accessories','apparel','brands',...,'wearables')
THEN 'PLP'
ELSE page_title
END AS page_title_adjusted
  FROM
    unnested_events
)
SELECT (SELECT COUNT(*) FROM
unnested_events_categorised WHERE
page_title_adjusted='PDP') / (SELECT COUNT(*) FROM
unnested_events_categorised)*100;
```

External Knowledge:

Refined Page Classification Criteria

Overview

To enhance our understanding of user engagement on our e-commerce platform, we differentiate between two types of pages based on the URL structure: ***Product Listing Pages (PLPs) and Product Detail Pages (PDPs).*** These classifications are crucial for analyzing user behavior and improving site navigation efficiency.

Product Listing Pages (PLPs)

PLPs are identified by specific characteristics in the URL:

•**The first segment doesn't contain a '+' sign**, ensuring these are not detail views.

•The fourth or fifth segment must contain one of the following category names, indicating a broader category or collection page rather than a specific product focus:

- Accessories
- Apparel
- Brands
- Campus Collection
- Drinkware
- Electronics
- Google Redesign
- Lifestyle
- Nest
- New 2015 Logo
- Notebooks & Journals
- Office
- Shop by Brand
- Small Goods
- Stationery
- Wearables

Product Detail Pages (PDPs)

PDPs, which focus on individual products, are marked by:

•The URL must **be divided into at least five segments.**

•**The presence of a '+' sign in the first segment**, a common marker for detailed product pages.

•**The fourth or fifth segment** must also include **one of the specified category names**, ensuring that the detail being viewed pertains to one of the recognized product categories:

- Accessories
- …
- Wearables

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Features of Spider 2.0

- Complex SQLs
  - Different DB System and Dialect
  - Advanced Function and Operator

Google Big Query | snowflake | PostgreSQL | DuckDB | SQLite

database="BigQuery", function="ST_INTERSECTS", category="geography-functions"

ST_INTERSECTS

ST_INTERSECTS(geography_1, geography_2)

Description

Returns ` TRUE ` if the point set intersection of ` geography_1 ` and ` geography_2 ` is non-empty. Thus, this function returns ` TRUE ` if there is at least one point that appears in both input ` GEOGRAPHY ` s.

If ` ST_INTERSECTS ` returns ` TRUE ` , it implies that ` ST_DISJOINT ` returns ` FALSE ` .

Return type

` BOOL `

database="SQLite", function="group_concat(X,Y)", category="aggregate-functions"

Function: group_concat(X,Y)

Usage: group_concat(X) group_concat(X,Y) string_agg(X,Y)

Descritpion: The group_concat() function returns a string which is the concatenation of all non-NULL values of X. If parameter Y is present then it is used as the separator between instances of X. A comma (",") is used as the separator if Y is omitted.

The string_agg(X,Y) function is an alias for group_concat(X,Y). String_agg() is compatible with PostgreSQL and SQL-Server and group_concat() is compatible with MySQL.

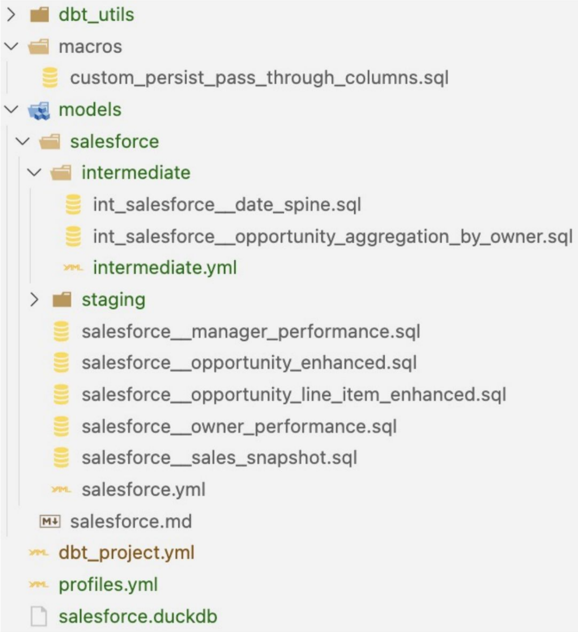
The order of the concatenated elements is arbitrary unless an ORDER BY argument is included immediately after the last parameter.

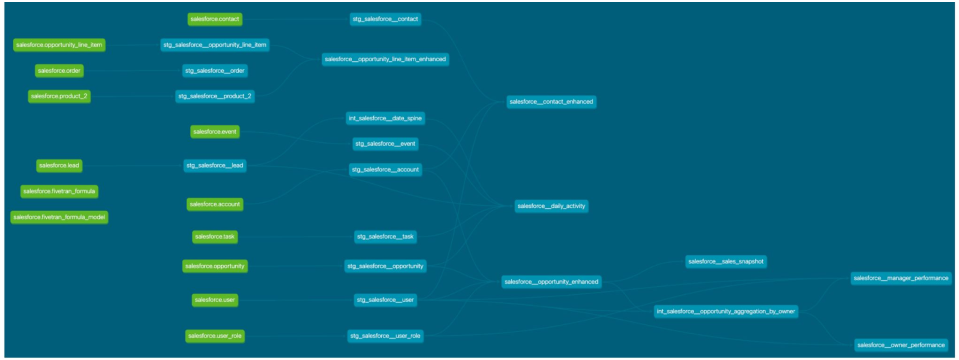
Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Features of Spider 2.0

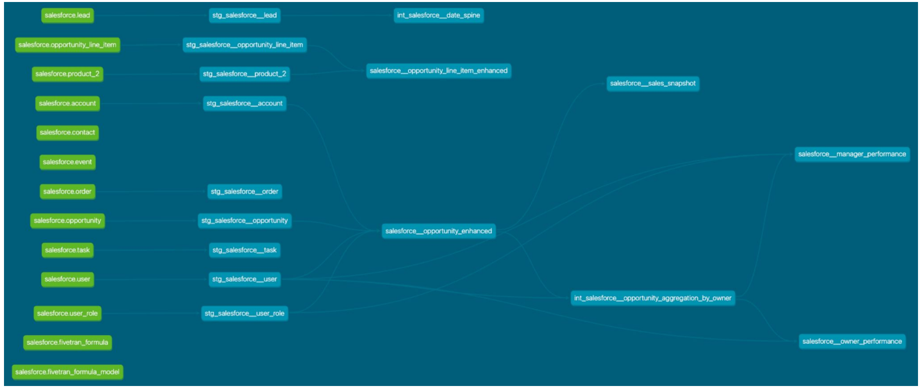
- DBT Project

Write multiple SQL files to complete DBT project





DAG Completion

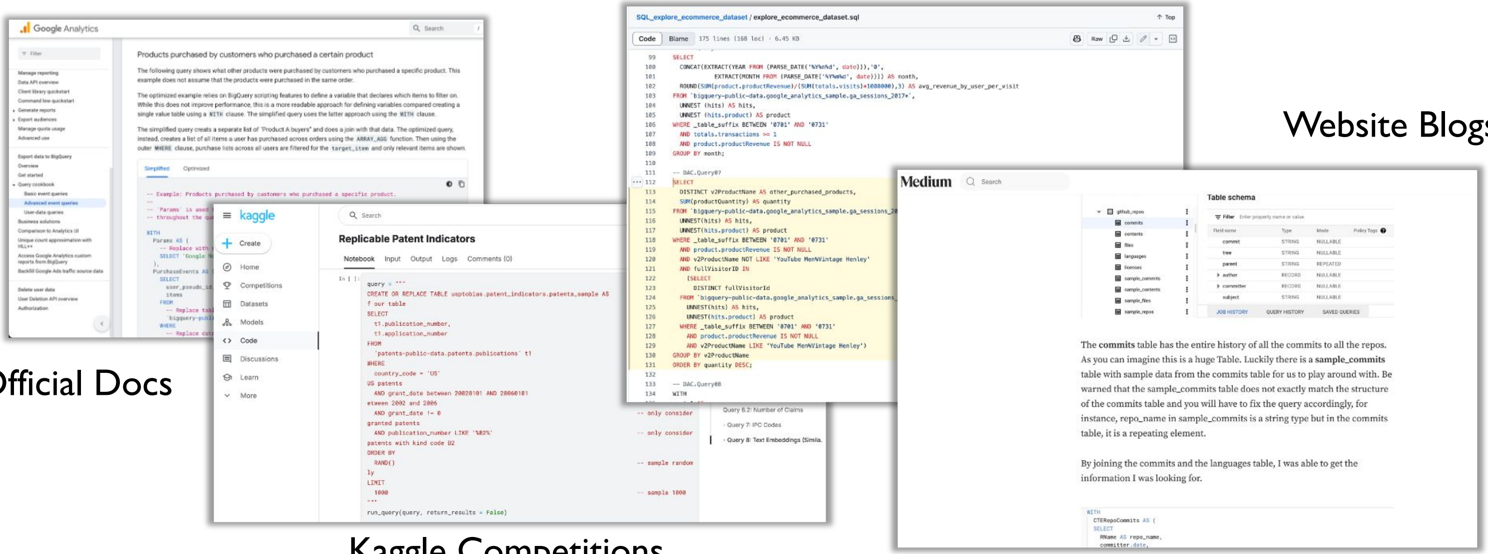


Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Annotation Pipeline

Database Collection → SQL Query Gathering → **SQL Rewriting** →

Context Setup → Natural Language Instruction Writing → Evaluation Script



Annotation Pipeline

Database Collection → SQL Query Gathering → **SQL Rewriting** → Context Setup → Natural Language Instruction Writing → Evaluation Script

```
|--- README.md      # The task description
|--- query.py       # The query interface
|--- BASIC_SQLs     # SQL examples of google analytics
|--- bigquery_credential.json   # Bigquery credentials
|--- 202012.csv     # The predefined answer file,
|--- 202101.csv     # Answer format of data in January 2021
`--- 202011.csv     # Answer format of data in November 2022
```

Original SQL

```sql
SELECT
 start_station_name, end_station_name, avg_bike_duration, avg_taxi_duration, avg_taxi_fare
FROM (
SELECT start_station_name, end_station_name,
 ROUND(start_station_latitude, 3) AS ss_lat, ROUND(start_station_longitude, 3) AS ss_long,
 ROUND(end_station_latitude, 3) AS es_lat,ROUND(end_station_longitude, 3) AS es_long,
 COUNT(*) AS bike_trips
FROM`bigquery-public-data.new_york.citibike_trips`
WHERE start_station_name != end_station_name
GROUP BY start_station_name, end_station_name, ss_lat, ss_long, es_lat, es_long
ORDER BY bike_trips DESC LIMIT 100 ) a
JOIN (     SELECT
 ROUND(pickup_latitude, 3) AS pu_lat, ROUND(pickup_longitude, 3) AS pu_long,
 ROUND(dropoff_latitude, 3) AS do_lat, ROUND(dropoff_longitude, 3) AS do_long,
 COUNT(*) AS taxi_trips
FROM`bigquery-public-data.new_york.tlc_yellow_trips_2016`
GROUP BY pu_lat, pu_long, do_lat, do_long)b
ON
 a.ss_lat=b.pu_lat AND a.es_lat=b.do_lat AND a.ss_long=b.pu_long AND a.es_long=b.do_long
ORDER BY bike_trips DESC LIMIT 20;
```

Question

For the top 20 Citi Bike routes in 2016, which route is faster than yellow taxis and among those, which one has the longest average bike duration? Please provide the start station name of this route. The coordinates are rounded to three decimals.

Reference Plan

1. Focus on 2016 data to determine the top 20 most popular bike routes based on start and end stations, noting their latitude and longitude. 2. Calculate the average bike duration and count the number of bike trips for each route. 3. Extract the average duration for corresponding taxi routes using the same latitude and longitude for start and end points. 4. Calculate the average taxi duration for the matching routes. 5. Filter the results to find the bike route where the average bike duration is shorter than the average taxi duration. 6. Order the results by average bike duration in descending order and limit the output to one record.

Gold SQL (After rewriting)

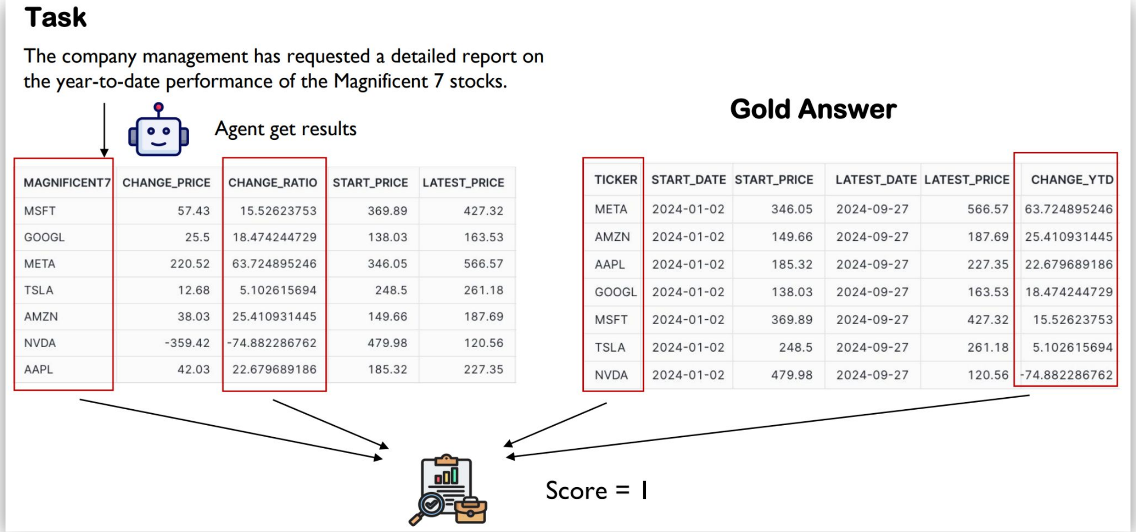
```sql
WITH top20route AS (
SELECT
 start_station_name, end_station_name, avg_bike_duration, avg_taxi_duration
FROM (
SELECT start_station_name, end_station_name,
 ROUND(start_station_latitude, 3) AS ss_lat, ROUND(start_station_longitude, 3) AS ss_long,
 ROUND(end_station_latitude, 3) AS es_lat, ROUND(end_station_longitude, 3) AS es_long,
 AVG(tripduration) AS avg_bike_duration,  COUNT(*) AS bike_trips
FROM
`bigquery-public-data.new_york.citibike_trips`
WHERE
 EXTRACT(YEAR from starttime) = 2015 AND start_station_name != end_station_name
GROUP BY  start_station_name, end_station_name, ss_lat, ss_long, es_lat, es_long
ORDER BY  bike_trips DESC LIMIT 20
) a
JOIN (
SELECT
 ROUND(pickup_latitude, 3) AS pu_lat,  ROUND(pickup_longitude, 3) AS pu_long,
 ROUND(dropoff_latitude, 3) AS do_lat,  ROUND(dropoff_longitude, 3) AS do_long,
 AVG(UNIX_SECONDS(dropoff_datetime)-UNIX_SECONDS(pickup_datetime)) AS avg_taxi_duration,
 COUNT(*) AS taxi_trips
FROM
`bigquery-public-data.new_york.tlc_yellow_trips_2015`
GROUP BY
 pu_lat, pu_long, do_lat, do_long
) b
ON
 a.ss_lat = b.pu_lat AND  a.es_lat = b.do_lat AND
 a.ss_long = b.pu_long AND  a.es_long = b.do_long
)
SELECT start_station_name FROM top20route
WHERE avg_bike_duration < avg_taxi_duration
ORDER BY avg_bike_duration DESC LIMIT 1
```

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Annotation Pipeline

Database Collection → SQL Query Gathering → **SQL Rewriting** → Context Setup → Natural Language Instruction Writing → Evaluation Script

- **Execution-based focused evaluation**



MAGNIFICENT7	CHANGE_PRICE	CHANGE_RATIO	START_PRICE	LATEST_PRICE
MSFT	57.43	15.52623753	369.89	427.32
GOOGL	25.5	18.474244729	138.03	163.53
META	220.52	63.724895246	346.05	566.57
TSLA	12.68	5.102615694	248.5	261.18
AMZN	38.03	25.410931445	149.66	187.69
NVDA	-359.42	-74.882286762	479.98	120.56
AAPL	42.03	22.679689186	185.32	227.35

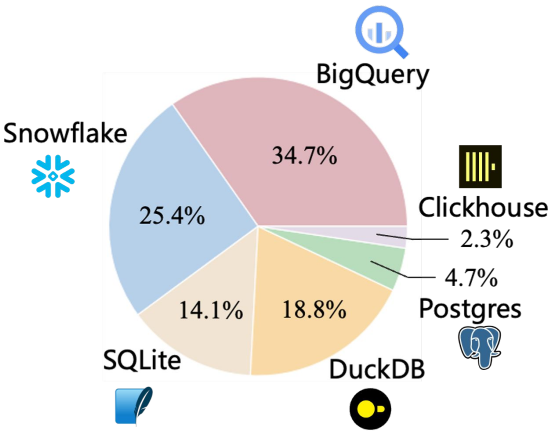
TICKER	START_DATE	START_PRICE	LATEST_DATE	LATEST_PRICE	CHANGE_YTD
META	2024-01-02	346.05	2024-09-27	566.57	63.724895246
AMZN	2024-01-02	149.66	2024-09-27	187.69	25.410931445
AAPL	2024-01-02	185.32	2024-09-27	227.35	22.679689186
GOOGL	2024-01-02	138.03	2024-09-27	163.53	18.474244729
MSFT	2024-01-02	369.89	2024-09-27	427.32	15.52623753
TSLA	2024-01-02	248.5	2024-09-27	261.18	5.102615694
NVDA	2024-01-02	479.98	2024-09-27	120.56	-74.882286762

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Statistics of Spider 2.0

1. Multiple database types and systems
2. Industrial applications such as Google Analytics and Salesforce
3. Large-scale schemas (averaging 812 columns)
4. Complex schema structures (nested schemas, partitioned tables, etc.)

Statistics	Number (% of Total)
Total Levels (#tokens)	632 (100%)
- Easy (#tokens < 80)	160 (25.32%)
- Medium ($80 \leq$ #tokens < 160)	279 (44.15%)
- Hard (#tokens ≥ 160)	193 (30.54%)
- With Project-level (DBT)	78 (12.34%)
- With Documentation	82 (12.97%)
- With Functions	474 (75.00%)
- With Partition Tables	54 (8.54%)
- With Multiple Schemas	140 (22.15%)
- With Nested Schemas	117 (18.51%)



Dataset	# Test Examples	# Test DB	# Col / DB	# Tok. / SQL	# Func. / SQL	External Knowledge	SQL Dialect	Project Level
WikiSQL (Zhong et al., 2017)	15,878	5,230	6.3	12.2	0.0	✗	✗	✗
Spider 1.0 (Yu et al., 2018)	2,147	40	27.1	18.5	0.0*	✗	✗	✗
KaggleDBQA (Lee et al., 2021)	272	8	23.4	13.8	0.0	✓	✗	✗
SEDE (Hazoom et al., 2021)	857	1	212.0	46.9	1.4	✗	✗	✗
BIRD (Li et al., 2024b)	1,789	15	54.2	30.9	0.4*	✓	✗	✗
Spider 2.0-lite	547	158	803.6	144.5	6.5	✓	✓	✗
Spider 2.0-snow	547	152	**812.1**	**161.8**	6.8	✓	✓	✗
Spider 2.0	632	213	743.5	148.3	**7.1**	✓	✓	✓

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Baseline: Spider-Agent

- Based-on ReAct Framework
- Hosted on Real Docker environment
- Complex File and Database operation

Action	Description
BASH	Executes shell commands, such as checking file information, running code, or executing DBT commands.
CreateFile	Creates a new file with specified content.
EditFile	Edits or overwrites the content of an existing file.
ExecuteSQL	Executes a SQL query on BigQuery/Snowflake, with an option to print or save the results.
GetTables	Retrieves all table names and schemas from a specified BigQuery/Snowflake dataset.
GetTabInfo	Retrieves detailed column information for a specific table in BigQuery/Snowflake.
SampleRows	Samples a specified number of rows from a BigQuery/Snowflake table and saves them as JSON.
FAIL	Agent decides the task is infeasible.
Terminate	Agent decides the task is finished.

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Result: Spider-Agent

- Best model o1/o3-mini on Spider-Agent only solves 23%!!
- Existing LLMs are still far from being expert on real-world text-to-SQL workflow tasks.
- Existing code agents struggle with solving database-related coding tasks.

Model	Spider 2.0-Lite				Spider 2.0-Snow			
	Easy	Medium	Hard	**Overall**	Easy	Medium	Hard	**Overall**
o1-preview	33.59%	23.58%	15.03%	23.22%	39.84%	21.14%	15.61%	**23.77%**
o3-mini	32.03%	26.02%	13.87%	**23.40%**	31.25%	18.29%	11.56%	19.20%
Claude-3.5-Sonnet	26.56%	15.85%	6.94%	15.54%	25.00%	16.26%	7.51%	15.54%
GPT-4o	22.66%	13.41%	5.78%	13.16%	24.22%	11.38%	6.94%	12.98%
DeepSeek-V3	19.53%	6.50%	4.05%	8.78%	20.31%	6.1%	4.05%	8.78%
Qwen2.5-Coder	13.89%	4.17%	3.38%	5.30%	11.72%	4.47%	2.31%	5.48%

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Result: Spider-Agent

- LLM-agent frameworks struggle interpreting databases with nested schema.

Task Subset	% of Total	SR (↑)
w/ Nested Column	18.51%	10.34%
w/o Nested Columns	68.04%	27.38%

- The performance drops when external documents are required.

Task Subset	% of Total	SR (↑)
w/ External Doc	12.97%	11.54%
w/o External Doc	73.58%	26.64%

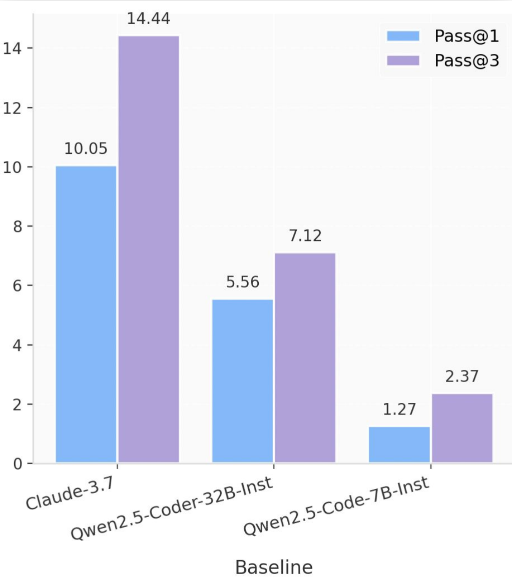
- LLM-agent frameworks struggle performing project-level tasks.

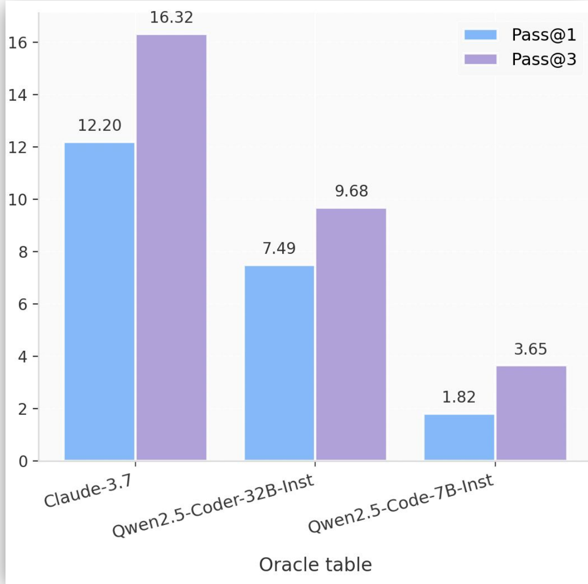
Task Subset	% of Total	SR (↑)
w/ DBT Project	12.34%	12.82%
w/o DBT Project	87.65%	23.22%

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Baseline: Non-Agent Baseline Experiments

- We use heuristic rules to do schema linking and apply end-to-end methods.





* We provide the ground-truth tables for spider2-lite and spider2-snow to help quick benchmarking and analysis.

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Leaderboard (UP to 2025-04-22)

Spider 2.0-Snow | Spider 2.0-lite | Spider 2.0

Spider 2.0-Snow is a self-contained text-to-SQL task that includes well-prepared database metadata and documentation, includes **547** examples, all hosted on *Snowflake*, which offers participants free quotas. If you want to test performance on *a single SQL dialect*, don't hesitate to use **Spider 2.0-Snow**.

Rank	Method	Score
1 Jan 28, 2025	ReFoRCE + o1-preview *Hao AI Lab x Snowflake* [Deng et al. '25]	**31.26**
2 Mar 8, 2025	Spider-Agent + Claude-3.7-Sonnet-20250219	**24.50**
3 Mar 16, 2025	Spider-Agent + Claude-3.7-Sonnet-20250219-Thinking	**24.31**
4 Nov 30, 2024	Spider-Agent + o1-preview	**23.58**
5 Feb 11, 2025	Spider-Agent + o1-2024-12-17	**23.21**
6 Feb 1, 2025	Spider-Agent + o3-mini-2025-01-31	**19.20**
7 Mar 7, 2025	Spider-Agent + Claude-3.5-Sonnet-20241022 *AWS ProServe*	**19.01**
8 Feb 10, 2025	Spider-Agent + Claude-3.5-Sonnet-20241022	**15.54**
9 Mar 13, 2025	Spider-Agent + Gemini-2.0-Pro	**13.89**
10 Nov 30, 2024	Spider-Agent + GPT-4o-2024-11-20	**12.98**

Spider 2.0-Snow | **Spider 2.0-lite** | Spider 2.0

Spider 2.0-Lite is a self-contained text-to-SQL task that includes well-prepared database metadata and documentation. This setup enables a text-in, text-out approach, facilitating faster development and evaluation. Spider 2.0-lite, which has **547** examples, is designed to handle queries for *BigQuery*, *Snowflake*, and *SQLite* databases.

Rank	Method	Score
1 Jan 28, 2025	ReFoRCE + o1-preview *Hao AI Lab x Snowflake* [Deng et al. '25]	**30.35**
2 Mar 16, 2025	Spider-Agent + Claude-3.7-Sonnet-20250219-Thinking	**28.52**
3 Mar 8, 2025	Spider-Agent + Claude-3.7-Sonnet-20250219	**25.41**
4 Mar 28, 2025	LinkAlign + DeepSeek-V3 [Wang et al. '25]	**24.86**
5 Feb 10, 2025	Spider-Agent + o3-mini-2025-01-31	**23.40**
6 Nov 30, 2024	Spider-Agent + o1-preview	**23.03**
7 Mar 10, 2025	Spider-Agent + DeepSeek-R1	**13.71**
8 Feb 10, 2025	Spider-Agent + GPT-4o-2024-11-20	**13.16**
9 Mar 13, 2025	Spider-Agent + QwQ-32B	**11.33**
10 Dec 31, 2024	Duo *Anonymous*	**8.96**

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

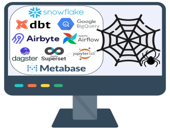
Why Proprietary LLMs Underperform

- Wrong Schema Linking
  - The database schema is too long, making it hard to locate the target table.
- Suboptimal Database Coding Agent
  - Poor bug-fixing and exploration capability
  - Inefficient, costly
- Natural Language Grounding Error
  - When queries have many complex conditions, logical errors often occur.
- SQL Dialects Hallucination
  - Some models struggle to troubleshoot in complex cloud databases.

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Conclusion

- **Complexity:** Diverse SQL dialects, advanced functions, and terabyte-scale data.
- **Agentic task setting:** Models interact with codebases, documentation, and databases dynamically.
- **Goal:** To foster intelligent agents for autonomous data engineering in real-world settings.
- **Limitations:**
  - Annotation is expensive and not scalable
  - Models are unable to handle such complexity



spider2-sql.github.io

https://github.com/xlang-ai/Spider2

Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows (Lei et al., 2024)

Acknowledgements

We thank the following institution for their gift funds supporting our open-source initiatives!

snowflake

Thank you for listening!

XLANG Lab

https://xlang.ai

UWNLP

香港大學自然語言處理實驗室

Natural Language Processing Group, The University of Hong Kong