BRING TEXT-TO-SQL TO BI PRODUCTION IN LARGE ENTERPRISE

Big Data Platform Department | PCG Tencent 腾讯
Jun 2024

TARGET AUDIENCE

The intended audience:

- data platform product managers, product strategists,
- data engineers, data scientists, and ML/AI practitioners

The audience will benefit:

- First-hand experience from Tencent's large-scale practice adapting the latest LLM to BI.
- Combining the latest open source and commercial LLM, RAG pattern/framework.
- Build the next-generation BI capabilities with a natural language interface.

ABSTRACT

1. Text-to-SQL capability is available with **fine-tuning** and **prompt engineering** under the hood.
2. Attempt to bridge the gap between POC Text-to-SQL tools and querying with natural language for users in Tencent.

3. We chose **DeepSeekCoder-33B as the foundational model for fine-tuning using LORA**.
4. We developed **a training instruction set** with two primary goals:
    - 1. query pattern and syntax coverage.
    - 2. representing how business context is referenced, especially for multi-table queries.
5. Optimizing the performance and cost-effectiveness of the inference process.

6. Our Model is evaluated and compared with GPT3.5/4 using BIRD and a set of complex real-life queries from Tencent.
7. The model **performs better (i.e., more accurately)** than GPT4 **in cases requiring table joins**, though not as capable of dealing with ambiguous expressions of query intentions.

CONTENT OUTLINE

[Why]-[How]-[Use Case]

1. [Why] Fine-tuning LLMs to improve Text-to-SQL

1.1 The Reason We Choose to Fine-tune?

1.2 Benchmark Testing & Model Selection

2. [How] Fine-tuning Processes

2.1 Data Preparation

2.2 Fine-tuning Process

2.3 Benchmark Testing

3. [Use Case]

3.1 Tencent Video & NBA News

3.2 RAG & Query Rewriting

3.3 BI Demo

1.[WHY] FINE-TUNE LLMS TO IMPROVE TEXT-TO-SQL

1.1 THE REASON WHY TO FINE-TUNE

Data Privacy, Cost, and Performance

1. GPT4/3.5 performs well, but Tencent needs tighter control on **data privacy** and **data security**.

2. The response time of GPT4/3.5 is long, and **the APIs are not cheap**; we are looking for **cost-effective models that respond faster**.

3. The **training dataset** for the open source model is **relatively basic**. As for specific domain knowledge and business jargon/pattern, in-context learning does not always provide the best performance.

4. The SQL generation quality of open models and GPT are **not optimized** for **predicate push down** and **column pruning**, which could not generate high performance SQL clauses when querying big data.

1.1 THE REASON WHY TO FINE-TUNE

The Objective And Steps Of Fine-tuning

1. Objective

- Fine-tune a *more general, vertical* Text-to-SQL model that can serve *multiple business use cases*.
- To solve problems with *high complexity in SQL and Query (hard/extra hard)*
- Train users/business patterns in specific Data Lake / Data Warehouse, and achieve best practice in *memorization* and *generalization.*
- Learn general knowledge of Query-to-SQL and business knowledge/jargon through context-learning.
- Generate stable, correct, high-performing SQL clauses, which is quite different from chat models.

2. Steps to Fine-tune

- Selection of the Foundation Model
- Selection of Training Dataset
- Data Cleaning
- Instruct Tuning
- Benchmark Testing and Evaluation

1.2 BENCHMARK TEST & MODEL SELECTION

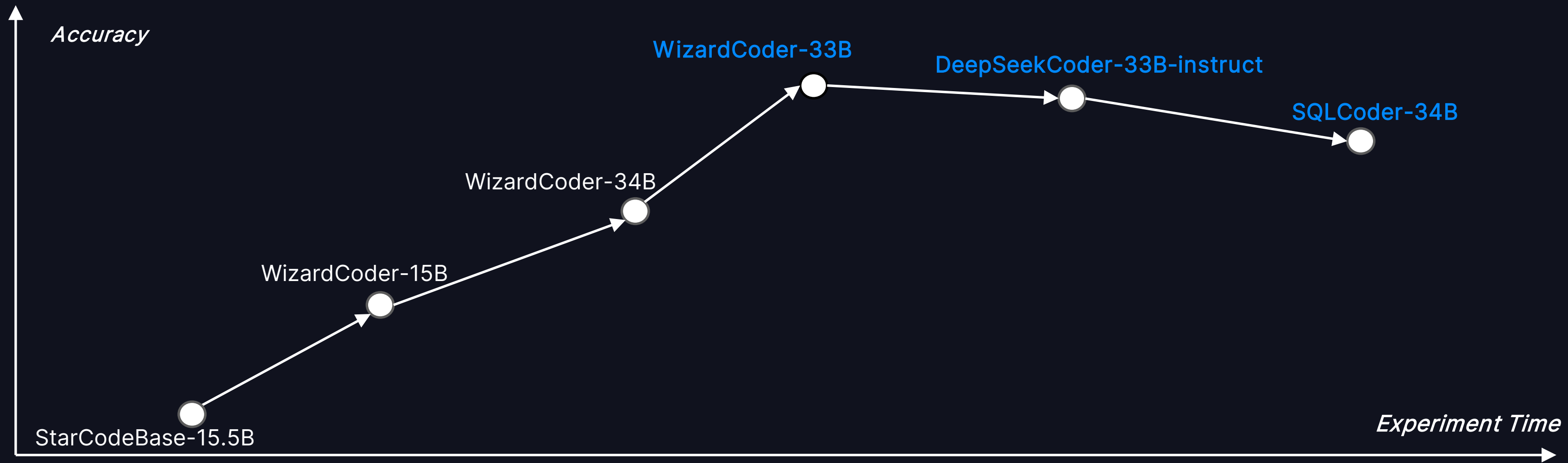
Foundation Model Benchmark

*We used **BIRD Dataset & Tencent Dataset** to evaluate model performance, focused on **Accuracy**.* We chose higher scored models as our foundational model.

Model	Accuracy-BIRD Dataset(No Retrieval)	Accuracy-In Tencent Dataset
ChatGLM-6b	2.4%	4%
ChatGLM2-6b	2.4%	4%
Belle-13B-ext	0%	0%
StarCodeBase-15.5B	2.1%	2%
WizardCoder-15B	14.9%	10%
WizardCoder-34B	24.3%	18%
WizardCoder-33B	39.2%	32%
DeepSeekCoder-33B-instruct	35%	34%
SQLCoder-34B	17.4%	28%

1.2 BENCHMARK TEST & MODEL SELECTION

Choose Foundation Models With Good Performance On Code Generation



StarCodeBase-15.5B--> WizardCoder-15B--> WizardCoder-34B --> WizardCoder-33B --> DeepSeekCoder-33B-instruct --> SQLCoder-34B

- With the development of LLMs in 2023, various specialized models emerged--they are different from general base models because of the pre-training dataset and the instruction set.
- We compared the performance of general models (ChatGLM, Baichuan, Belle), which perform well in Chinese, and specialized open-sourced models (StarCoder, WizardCoder).
- Initially, we used Belle and StarCoder as our base model. We picked WizardCoder, DeepSeekCoder, and SQLCoder as our potential choices when they were available.

2.[HOW] FINE-TUNING PROCESSES

2.1 DATA PREPARATION

Challenge: Open-source Dataset & Tencent Dataset

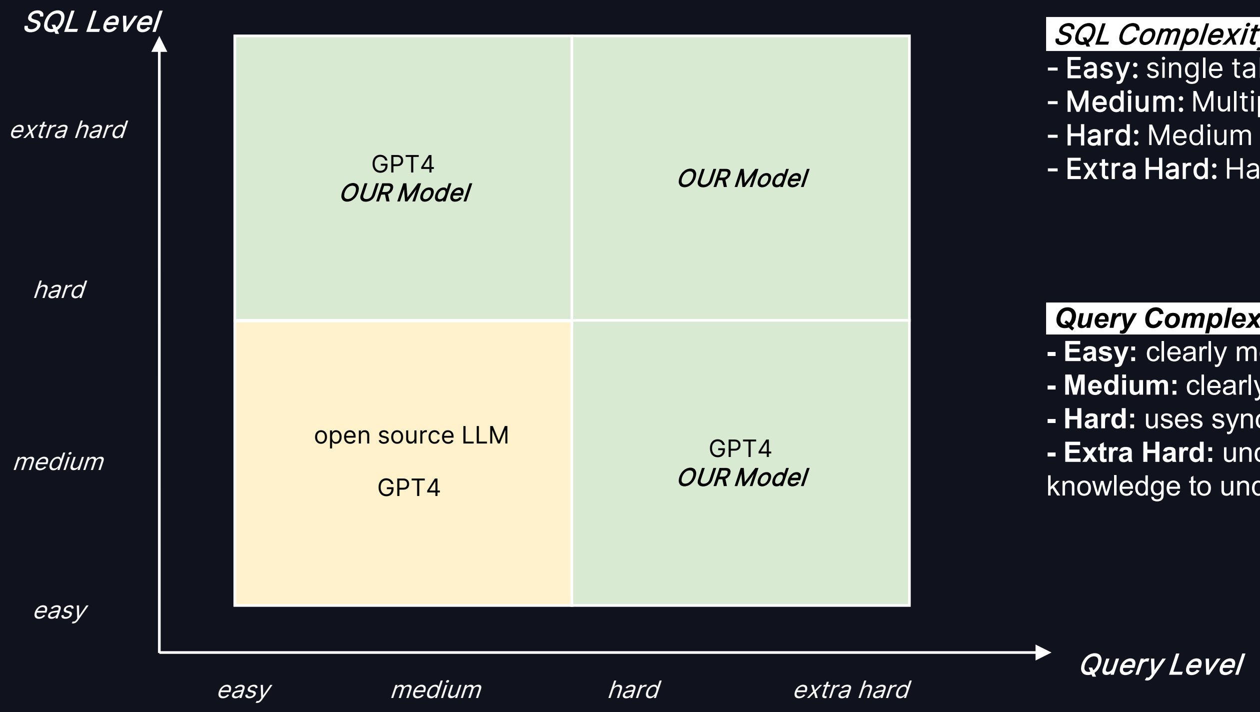
- The quality of training data is a crucial factor in fine-tuning.
- The common datasets from the open community (BIRD, Spider, Chase, DuSQL) **are in English, which requires translation.**
- Considering the dataset quality and actual usage from reality, we use Spider (Chinese) & Tencent datasets as training data.

- The following compares the CSpider (Spider Chinese) dataset and the Tencent dataset.

Characteristic	Spider(Chinese) Dataset	Tencent Dataset
table fields	few	**many**
table size	small	**large**
table count	few	**many**
table quality	good	some are good, some are poor
calculation methods	few	many
SQL quality	some are good, some are poor	some are good, some are poor

2.1 DATA PREPARATION

Challenge: Complexity Level Of SQL & Query



SQL Complexity:
- **Easy:** single table + filtering
- **Medium:** Multiple Easy + join
- **Hard:** Medium + aggregate & group
- **Extra Hard:** Hard + subquery

Query Complexity :
- **Easy:** clearly mentions fields
- **Medium:** clearly mention the value of fields
- **Hard:** uses synonyms of the fields
- **Extra Hard:** unclear, needs specific business/domain knowledge to understand

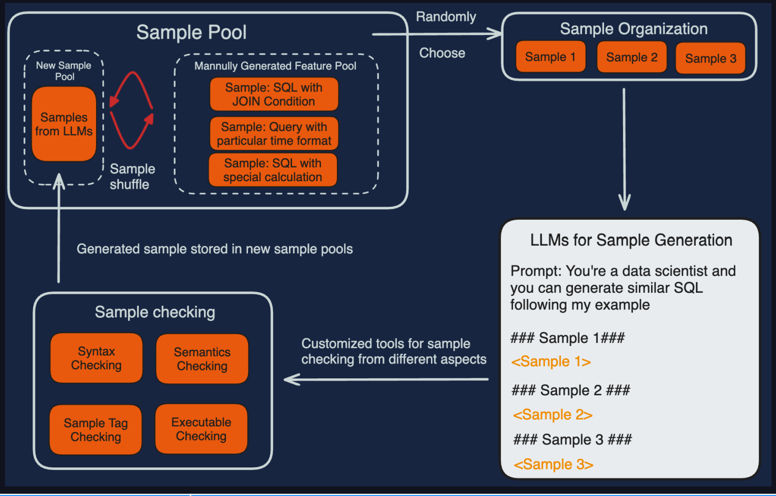
2.1 DATA PREPARATION

Prepare The Training Data - Part 1

1. Self-instruction: Align Training Data with Actual Data Distribution

The training sample needs to be comprehensive enough to cover SQL at various difficulty levels. Tencent samples and CSpider samples do not meet these requirements. To improve it, we randomly select manually written, high-quality instructions as seeds and format them into few-shot templates.

This allows LLMs to generate samples with specified features. The new samples generated by the LLM will be added to the sample pool, serving as an alternative set for the next round of model selection.



2. SQL Syntax Distribution

Training data should cover comprehensive SQL. Public datasets cover well on basic SQL such as querying, filtering, aggregating, and sorting.

We need to generate more samples containing join conditions and subqueries to improve the coverage of advanced SQL.

Dataset	Complexity Coverage
Open-Sourced Dataset	**Basic DDL: CREATE TABLE / ALTER TABLE**
	Simple filtering and aggregation: WHERE / GROUP BY
	Sorting and Limiting
Generated Dataset	**Complex filtering and aggregation**
	Join condition / Subquery

2.1 DATA PREPARATION

Prepare The Training Data - Part 2

1. Enhancing Query Coverage by Query Rewriting

Asking the same question in various ways can improve the robustness of Text-To-SQL performance. We use the open-sourced LLM to rewrite the original query to diversify the sample. The rewriting need to conform to the actual user query norms, and the rewritten query combined with table information will become the new training samples.

2. Covering More Complexity Levels

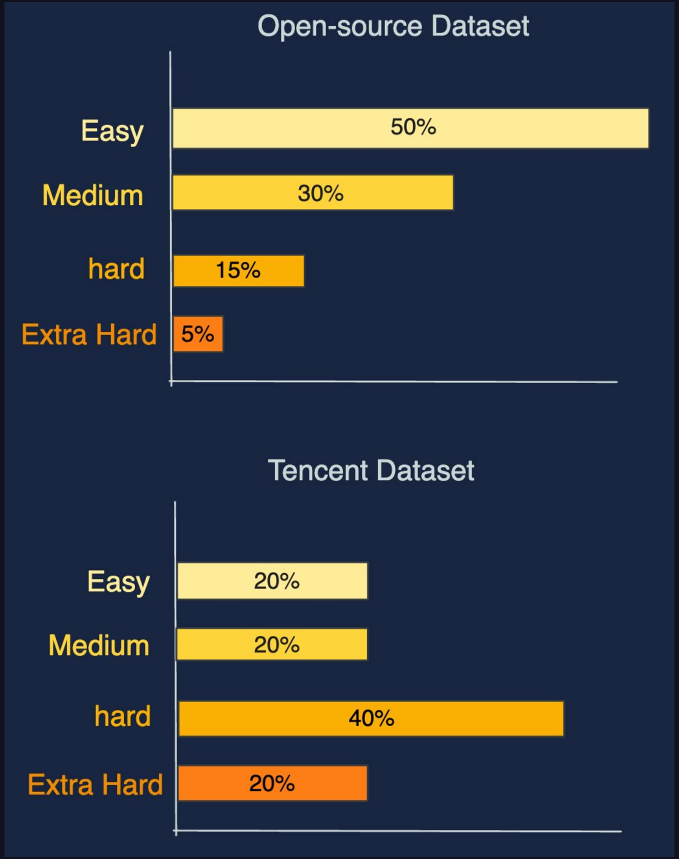
The queries and SQL in public datasets are relatively simple. The right figure shows the difficulty level distribution of samples in public datasets. In real business scenarios, there are many hard and extra-hard questions, and GPT-4 has an outstanding performance in solving them. To improve our model, we particularly extract samples covering hard and extra-hard levels from the Tencent dataset as the training set (right-bottom figure). This allows us to solve more practical problems.

3. Fixing Bad Cases in Foundation Models

Foundation Models contain bad cases, which could influence the effectiveness of fine-tuning. To address the case, we use enhanced samples to correct the foundation model and achieve better performance.

4. Including Real Users' Habits and Usage Patterns

We include samples from real users' habit--conventions in specific business, time representations, metrics calculation--to improve models' memory and performance.



2.1 DATA PREPARATION

Prepare The Training Data - Part 3

Instruct-Tuning Template

```
### Instruction:
<user query>

### Input:
TableName: <Table Name> ,
TableFields: <Table Field Schema>

### Response:
```

Tag	Description
User query	**User's questions in natural language, including hints** *(relationships between fields/time & data format...etc.)*
Table Name	**<table_name>**
Table Field Schema	**The collection of fields in a table, representing as follows:** **<field_original_name>|<field_name>|<field_type>** **<field_original_name>:** ***original field name in the table*** **<field_name>:** ***display name of the field*** **<field_type>:** ***unified field type***

2.1 DATA PREPARATION

Prepare The Training Data - Part 4

Instruct-Tuning Template

- Business users often configure a dictionary (enum) for certain dimensional fields, and they usually do not explicitly specify these field names when querying.

- To solve the problem, we input such field enums as complementary dataset and insert them in the tuning instruct. By in-context learning from the instruct dataset, models could auto-fill what is implied by users from the field enums.

Instruct Sample

(The training corpus is in Chinese corpus, and the prompt template is shown as it is)

Instruction:
找出每个可以容纳100多名学生的宿舍的设施数量，**join条件为dormid和id**
(find out all amenities that can host more than 100 students, ***join dormid with id***)

Input:
表: dorm
表字段:
Dormid(宿舍编号)**: bigint,**
dorm_name:宿舍名称:string,
student_capacity:学生容量:bigint

表: has_amenity,
表字段:
Id(编号)**:bigint,**
amenid:设施编号:bigint

Response:
SELECT COUNT(*), t1.dormid FROM (SELECT dormid FROM dorm WHERE student_capacity > 100) AS t1 JOIN (SELECT id FROM has_amenity) AS t2 ON t1.dormid = t2.id GROUP BY t1.dormid;

2.1 DATA PREPARATION

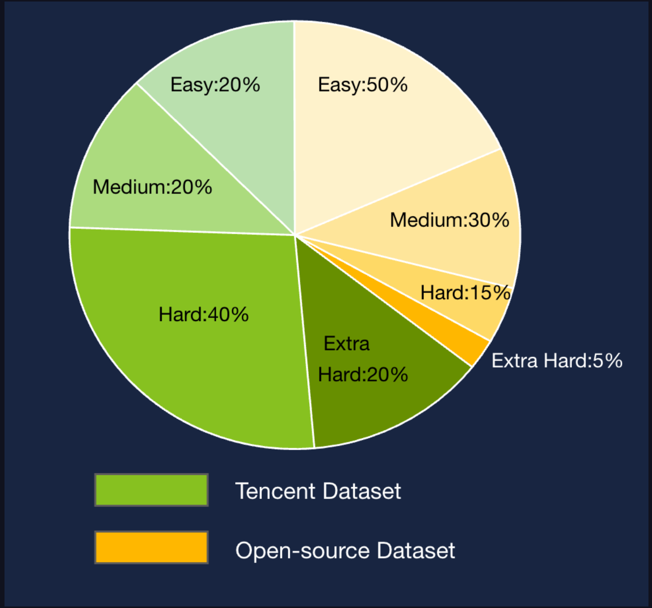
Data Cleaning And The Final Dataset

- ### Challenge of Dirty Data

1. Neither open dataset nor Tencent dataset are perfectly reliable
2. Datasets generated by GPT are not 100% correct

- ### Examples of Dirty Data

1. Unalignment of table/field names between Query and SQL, upper/lower case, missing fields
2. Fields used in Select and Group never appeared in the Prompt
3. SQL section used but never defined
4. Clearly requested for predicate push down, but SQL generated does not contain optimization
6. Fields defined do not align with what is used in the SQL
7. Rewriting queries to increase the variety, but these queries are actually not the same
8. Querying multiple tables, which could be queried in a single table
9. Can't tell the attributes between tables, e.g., used attributes in Table A for Table B
10. Time range, e.g. , last week had 7 days, but the data sample only has 6 days
11. Missing filtering and joining; Adding unnecessary filtering
12. Bad cases on special fields caused by base models require fixing.



We prepared a total of 16K+ refined samples for model training. Sample distribution as above.

2.2 FINE-TUNING PROCESSES

Hot Take-aways For Fine-tuning

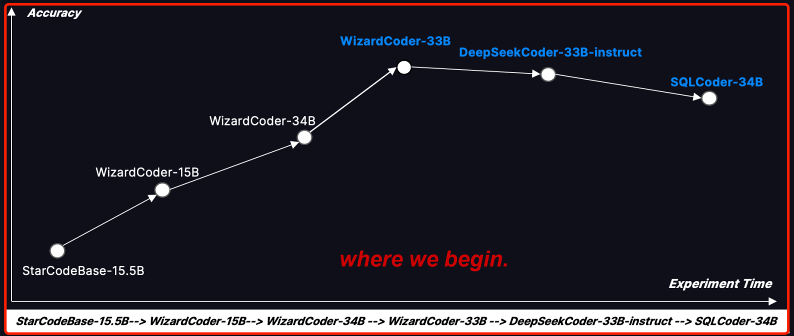
Dataset → LLM Model (SFT) → Fine-tuned Model

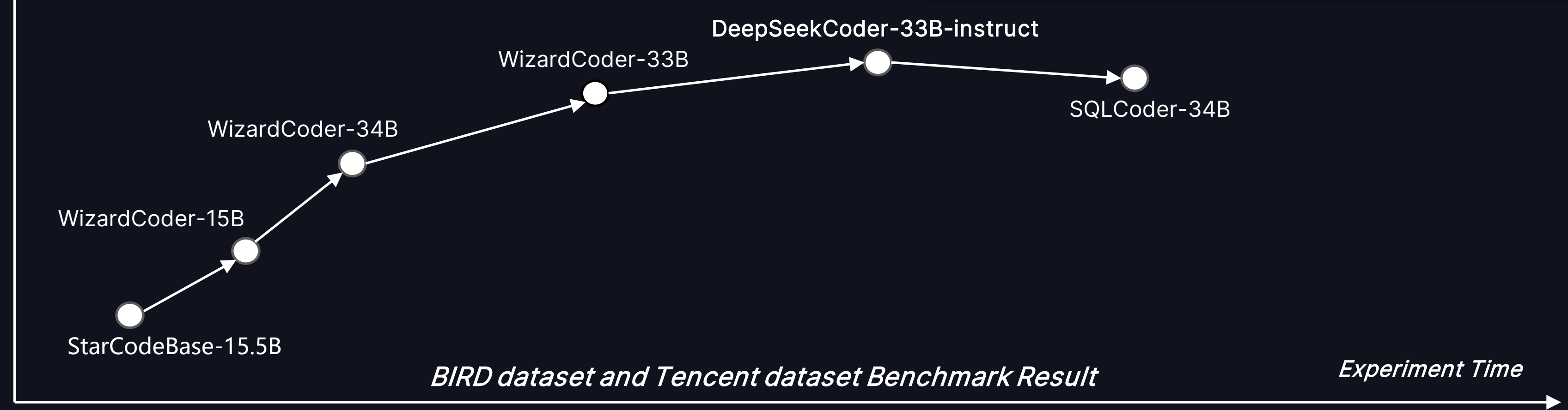
Training	*Inference*
• finetune_type=LoRA • LoRA Model Name=q_proj,v_proj,k_proj,o_proj • LoRA dim=16 • max token length=4096 • optimizer=AdamW • learning_rate=0.0001 • epoch=3	To increase the stability of returned results, config the params as follows: • temperature=0 • top_p=0 • top_k=0 • repetition_penalty=1 • length_penalty=1 • num_beams=1

1. Model with more params: Full-parameter tuning works better on larger sample size, LORA works better on smaller sample size.
2. Model with less params: Full-parameter tuning works better.
3. Full-parameter tuning might cause a huge problem on "forgetting", but LORA might save it.
4. Compared to sample size, sample quality is more important. A high-quality sample with a small size can achieve good performance.

2.3 BENCHMARK TESTING

Fine-tuning Performance



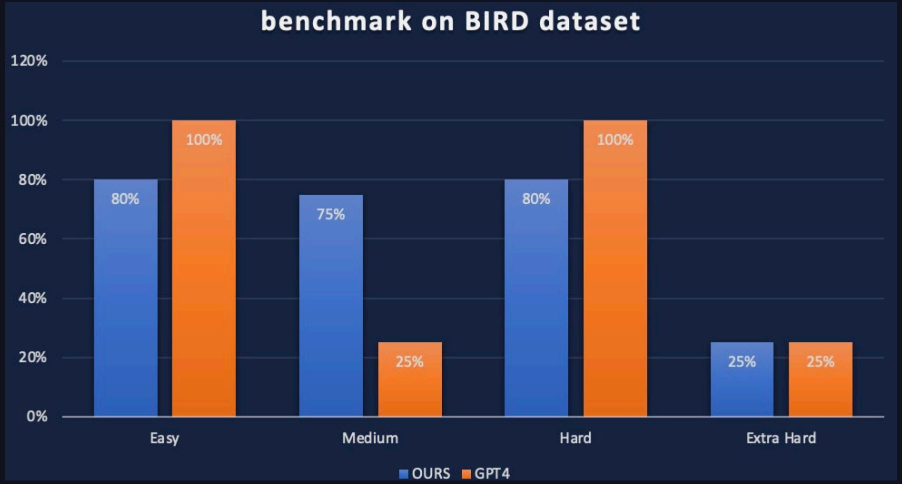


DeepSeekCoder performed better than WizardCoder-33B after finetuning. DeepSeekCoder showed better "comprehension" than other models.

2.3 BENCHMARK TESTING

Our Fine-tuned Model vs. GPT-4 On Benchmark

Model	BIRD Dataset Accuracy(No RAG Process)	Tencent Dataset Accuracy
DeepSeekCoder-33B(not fine-tuned)	35%	34%
GPT-4	45%	52%
Our Model (fine-tuned DeepSeekCoder-33B)	45%	54%



1. The fine-tuned model has the same accuracy as GPT-4 on the BIRD dataset.

2. Fine-tuned model is more accurate than GPT-4 on Tencent dataset.

3. The fine-tuned model has a better result than GPT-4 on medium-level complexity.

2.3 BENCHMARK TESTING

Our Fine-tuned Model vs. GPT-4 On Benchmark

SQL Complexity	Prompt	GPT-4	Our Model
Predicate Push Down & Column Pruning	### Instruction: 查询2016年2月内运送重量最大的订单的城市名称和人口数量 ### Input: 表: text2dsl_stdtest.shipping_shipment表字段: auto_id:auto_id:bigint,ship_id:运输编号:bigint,cust_id:客户编号:bigint,weight:重量:double,truck_id:卡车编号:bigint,driver_id:司机编号:bigint,city_id:城市编号:bigint,ship_date:发货日期:string||表: text2dsl_stdtest.shipping_city表字段: auto_id:auto_id:bigint,city_id:城市编号:bigint,city_name:城市名称:string,population:人口数量:bigint ### Response:	SELECT c.city_name, c.population FROM text2dsl_stdtest.shipping_shipment s JOIN text2dsl_stdtest.shipping_city c ON s.city_id = c.city_id WHERE s.ship_date BETWEEN '2016-02-01' AND '2016-02-29' ORDER BY s.weight DESC LIMIT 1; No predicate push down No column pruning. Problems in readability and SQL performance	SELECT t1.city_name, t1.population FROM (SELECT city_id, city_name, population FROM text2dsl_stdtest.shipping_city) AS t1 JOIN (SELECT city_id, weight FROM text2dsl_stdtest.shipping_shipment WHERE ship_date >= '2016-02-01' AND ship_date <= '2016-02-29' ORDER BY weight DESC LIMIT 1) AS t2 ON t1.city_id = t2.city_id used predicate push down column pruning
In-context Learning	### Instruction: 查询2023年8月DataInsight的工单数量，按照工单接单人聚合，返回前2条数据 ### Input: 表: bds_analysis.t_incident_detailed, 表字段: incident_label:工单问题标签:string,incident_id:工单id:bigint,status:工单状态:string,create_by:工单创建人:string,create_at:工单创建时间:string,ack_by:工单接单人:string,ack_at:工单接单时间:string,resolve_by:工单结单人:string,resolve_at:工单结单时间:string,assign_by:工单响应人/指派人:string,incident_class1:工单一级分类(Datainsight, Datatalk, Datamanager):string ### Response:	SELECT ack_by, COUNT(*) as count FROM bds_analysis.t_incident_detailed WHERE incident_class1 = 'DataInsight' AND create_at >= '2023-08-01' AND create_at < '2023-09-01' GROUP BY ack_by ORDER BY count DESC LIMIT 2;	SELECT ack_by, COUNT(*) AS order_count FROM bds_analysis.t_incident_detailed WHERE incident_class1 = 'DataInsight' AND create_at >= '20230801' AND create_at <= '20230831' GROUP BY ack_by ORDER BY order_count DESC LIMIT 2

3. [USE CASE] TEXT-TO-SQL IN PRODUCTION WITH QUERY REWRITING

3.1 TENCENT VIDEO & NBA NEWS

Business Needs For Text-to-SQL

Tencent Video & Tencent News

1. **Tencent is the exclusive agency for NBA online**, and need for querying score/stats are high. Tencent video has similar needs for data-driven operations.
2. By using natural language to query sports data, will lower the hurdle for sports content creators increase the scale.

Business Background

1. Users' SQL skills, and familiarity with the BI tool prevent them from generating business insights.
2. The skills required for users to independently conduct data analytics are quite high.
3. If Text-to-SQL reaches a high accuracy, and users could interact with the database directly through natural language, data analysis would be much more available.

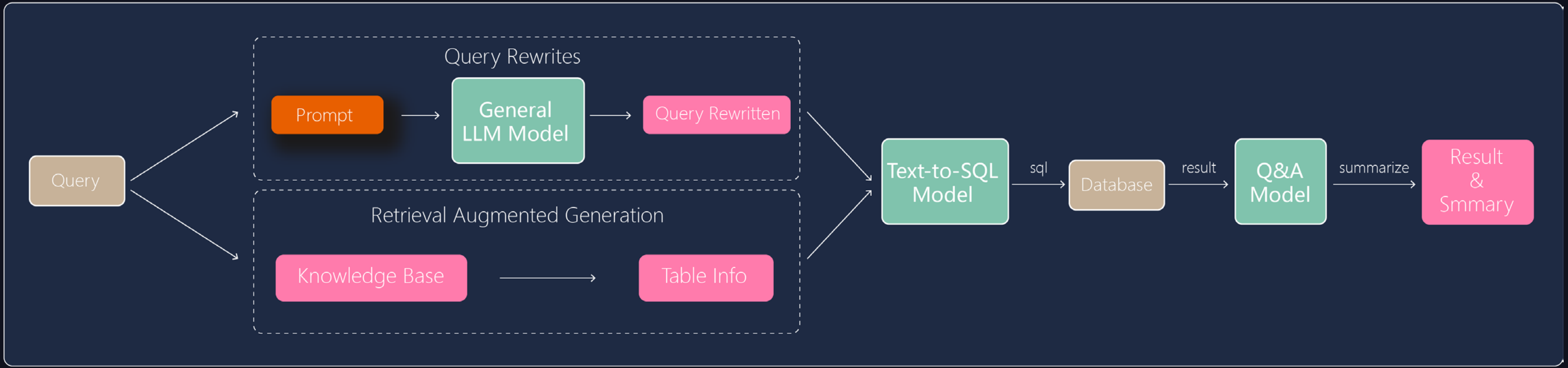
Sample Users' Queries

- **Sports Content Creation:** How has LeBron James performed in the past 10 games?
- **Film and television IP consumption analysis:** Show me the click-through rate of users who have watched film and television show after a (specific) advertisement page.
- **News content consumption analysis:** Consumption views of articles in recent days, categorized by content type.

When evaluating public and Tencent datasets, the fine-tuned model's performance can compete with GPT-4 and even surpass GPT-4 in some difficult problems. However, to use Text-To-SQL online, the model's accuracy needs to reach over 80%. We found that there is a lack of domain knowledge and a need to use context learning to bridge the gap between users' queries and schemas. We use an enhanced data process to address this to optimize the Text-to-SQL performance.

3.2 RAG & QUERY REWRITING

RAG + Query Rewriting: Enhancing Text-to-SQL in production

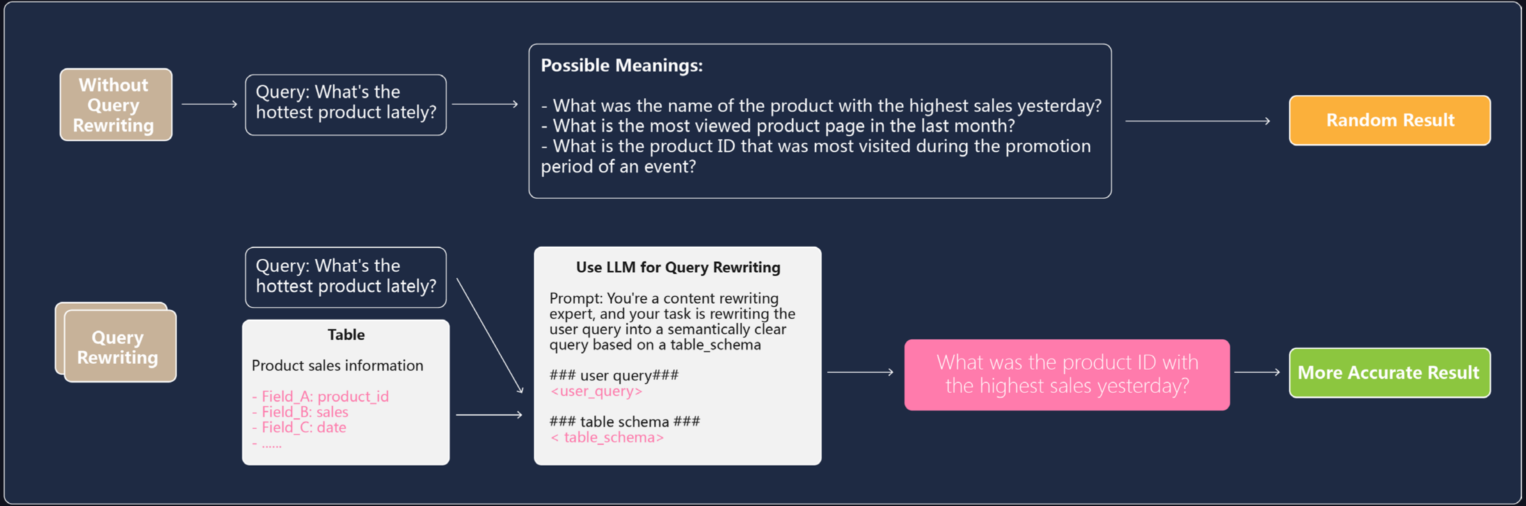


Combine with other techniques like *Query Rewriting.*

1. NBA News contains various professional terms and domain knowledge. For example, in the NBA Finals, the two teams must win four out of seven games to win the championship.
2. To achieve better results, passing the NBA domain knowledge to the Text-to-SQL model through prompts is necessary. Solely relying on the training of the Text-to-SQL model is not enough to solve business problems.
3. To identify which table and data to solve the user's question, we enhance the retrieval through query rewriting & RAG. At last, we run Text-To-SQL on that table.

3.2 RAG & QUERY REWRITING

Rewriting Queries To Locate Data Resource More Accurately



Challenge

- The user's query is usually bounded with business scope, carrying heavy business knowledge.
- The query expression is usually colloquial. Similarly, different users' expressions of the same concept are not always the same and often not clear enough.

Solution

- The input queries need to include the actual intent of users' queries and combine domain knowledge to obtain a precise understanding of their intent.
- LLM has excellent reasoning and generalization capabilities, we can rewrite queries using a general open-sourced LLM with few-shots.

3.2 RAG & QUERY REWRITING

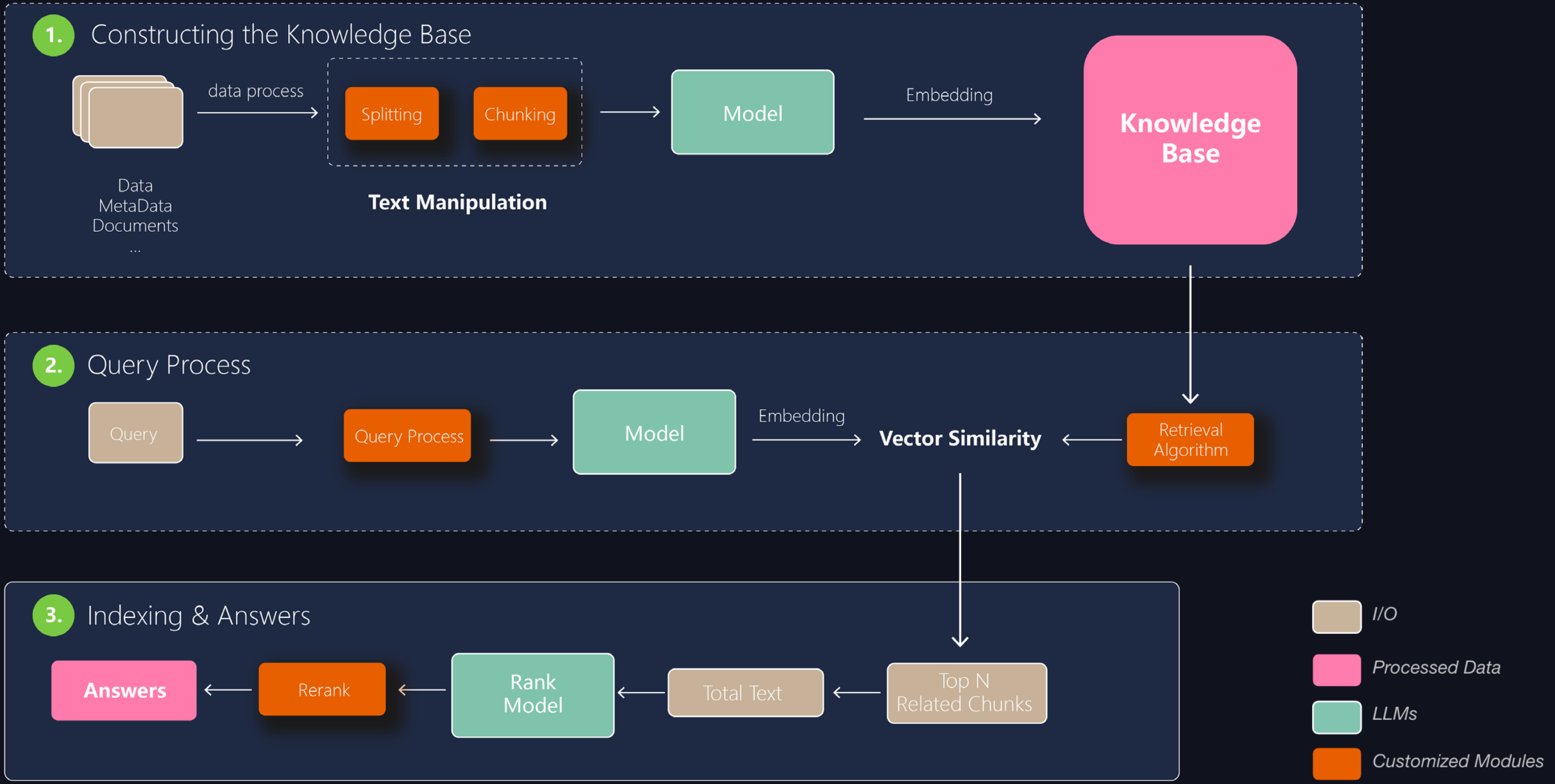
WITH/WITHOUT QUERY REWITES

The query accuracy is only 29.2% **when solely using Text-to-SQL** Service, and it improves to 87.8% when combined with RAG & query rewriting.

Without Query Rewriting	SQL	Correctness	With Query Rewriting	SQL	Correctness
Who won last year's finals?	SELECT team_name FROM team_match_stats WHERE type = 'Finals' AND is_win = 1	False	*Which team won the most games in the 2023 season of the Grand Finals?*	SELECT team_name, COUNT(*) FROM player_match_stats WHERE season_id='2023' AND season_type='Finals' GROUP BY team_name ORDER BY COUNT(*) DESC LIMIT 1	True
Who are the players who have hit 50+ three-pointers this season?	SELECT player_name FROM player_match_stats WHERE season_id = '2023' AND three_point_goals_attempted >= 50	False	*In the 2023 season, which players have a cumulative total of more than 50 three-point shots?*	SELECT player_name, sum(three_point_goals) as total_three_point_goals FROM player_match_stats WHERE season_id = '2023' GROUP BY player_name HAVING total_three_point_goals >= 50	True
What is the Pacers' best season record?	SELECT MAX(score) , season_id FROM team_match_stats GROUP BY season_id	False	*What season did the team Pacers win the most games?*	SELECT season_id FROM team_match_stats WHERE team_name = 'Pacers' AND is_win = 1 GROUP BY season_id ORDER BY count(*) DESC LIMIT 1	True

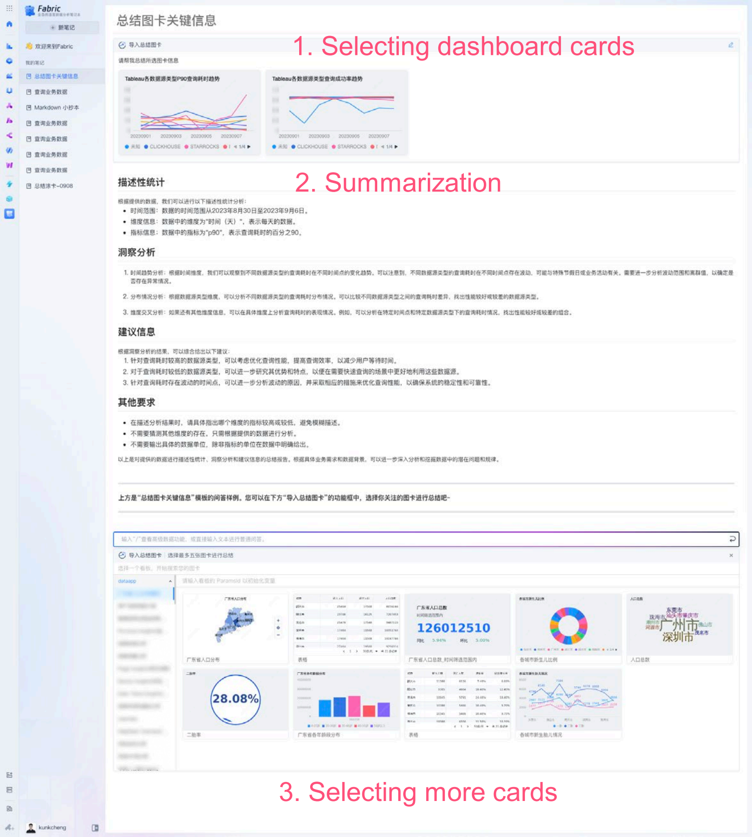
3.2 RAG & QUERY REWRITING

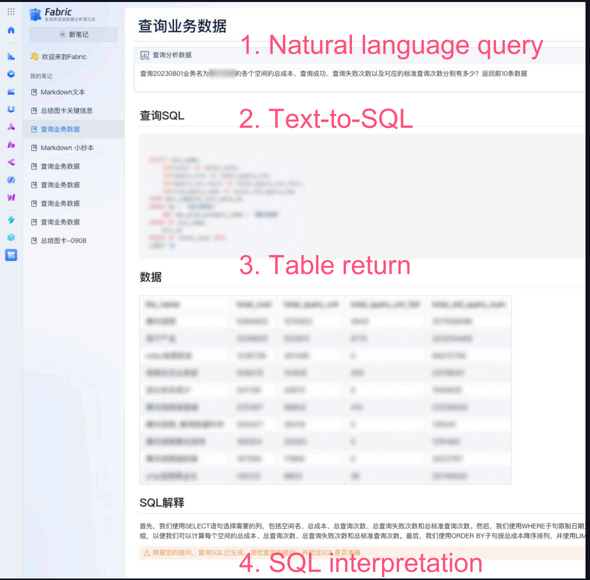
Text-to-SQL In Production



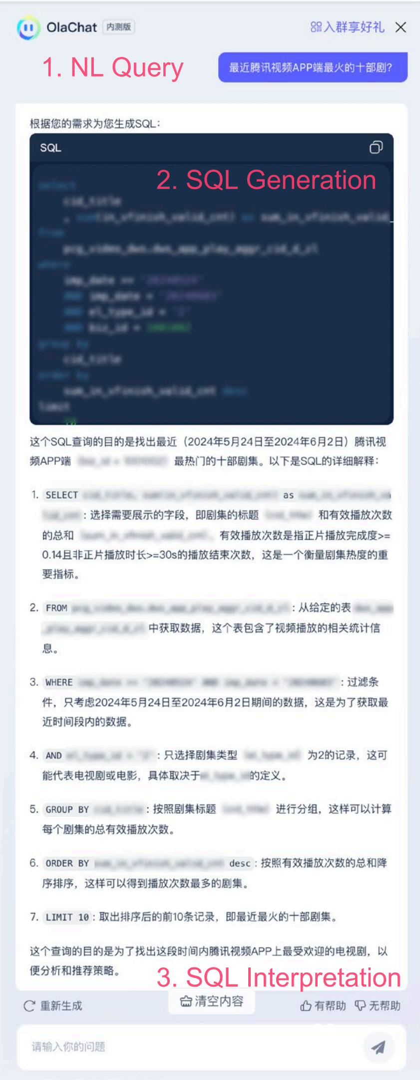
3.3 BI DEMO

Demo: Text Queries & Diagrams Summarization





Text-To-SQL OlaChat



腾讯灯塔

Dashboard Summarization

3.3 BI DEMO

Demo: BI Copilot Enables Natural Language Queries

1. Natural language query
2. Intention Inference: Text-to-SQL
3. Most related tables
4. SQL return
5. Interpretation of SQL

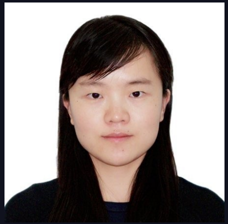
Text-to-SQL

1. Natural language query
2. Intention Inference
3. Text-to-SQL – Execution
4. Card Visualization
5. Hints

Text-to-Viz

OlaChat 腾讯灯塔

THANK YOU FOR LISTENING!
Q&A;)



Hehuan Liu
Senior AI/ML Scientist
(+86) 13910972658
liuhehuan@gmail.com | mermaidliu@tencent.com
www.linkedin.com/in/hehuanliu/

Kun Cheng
Product Manager
(+86) 13679263697 | (+1) 617-599-7131
kchengdusp@outlook.com
www.linkedin.com/in/kun-cheng-3040701b3/