Assignment 1 – Setareh Babajani

In this report, I will briefly describe what I did during each section. At the end, I will compare and analyze my results. I will upload all the files needed on Canvas, but you can also find them here.

Query Selection:

In this part, first I loaded the bird_min_dev dataset. Then chose three different databases (“student_club”, “debit_card_specializing” and “european_football_2”). As it said we should use different difficulty and length for queries, I added a new column (query length) to the data file and by considering that, I took 5 queries per each difficulty per each database (so at the end I had 45 queries). The details of them can be find at “selected_45_queries_assign1.csv”.

Query Generation:

In this section, I used the open-weight model called “Qwen2.5-Coder-7B-Instruct” due to the limitation of the colab GPU and its good performance for text generation tasks. For producing the prompt, there are two notes:

- I made a dictionary in which keys are the databases and the values are the schema related to them. I got this schema from the BIRD-Benchmark. This will help the model in producing the SQL queries. Then, I give the schema and evidence (which is in the dev file they provided) to the model.
- I tried two different prompts (both of them are at the code file). For the first one the prompt was general so the resulted queries were not good and there were some rows without any

produced SQL query! So, I changed it by covering the problems and made it a little complicated. You can find the details in the code.

Finally, my model made SQL query for each question. You can find them at "queries_with_generated_sql.csv".

Evaluation:

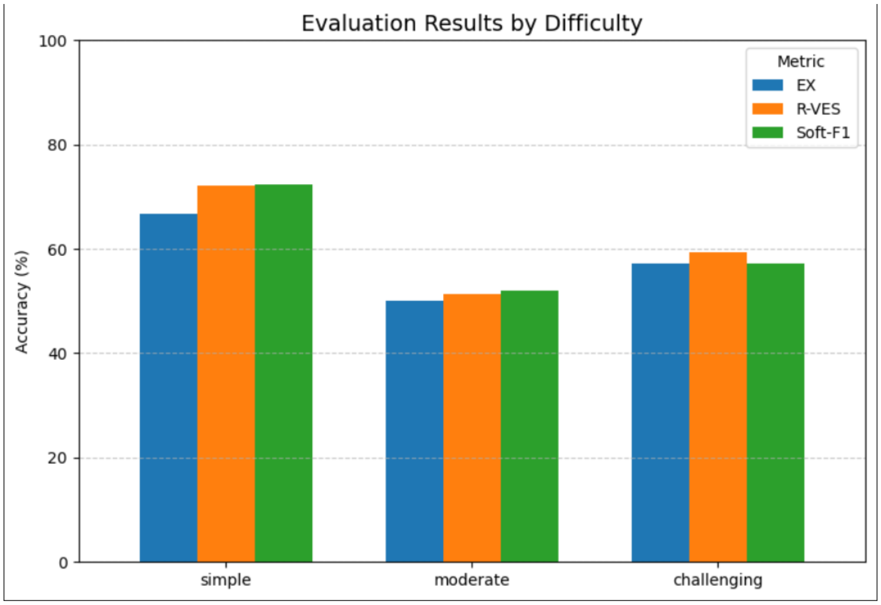
In this section, I used the three metrics told at the mini-bird benchmark (EX, R-VES and Soft-F1). I prepared the files and directories needed for running the "run_evaluation.sh" in which all of the metrics ran automatically. You can find the result of the assertion in "my_predictions_SQLite.txt" too.

```
start calculate EX
                     simple               moderate             challenging          total
count                15                   16                   14                   45
======================================    EX    ====================================
EX                   66.67                50.00                57.14                57.78
===========================================================================================
Finished EX evaluation for mini dev set

start calculate R-VES
                     simple               moderate             challenging          total
count                15                   16                   14                   45
======================================    R-VES    =================================
R-VES                72.17                51.28                59.44                60.78
===========================================================================================
Finished R-VES evaluation for mini dev set

start calculate Soft-F1
                     simple               moderate             challenging          total
count                15                   16                   14                   45
======================================    Soft-F1    ===============================
Soft-F1              72.25                52.08                57.14                60.38
===========================================================================================
Finished Soft-F1 evaluation for mini dev set
```

Analysis:

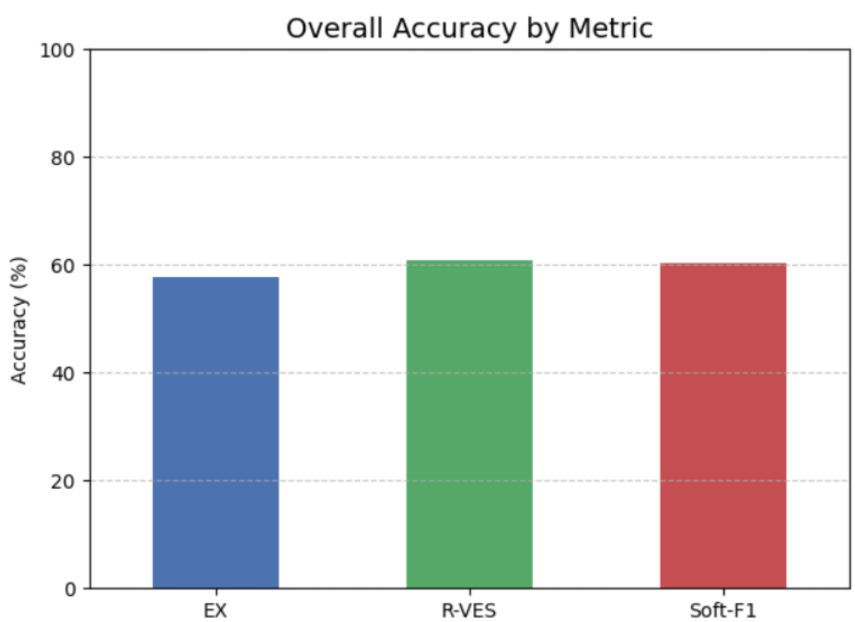
Evaluation Results by Difficulty:

You can find that based on the difficulty, all the metrics had better accuracies at simple queries (which is logical). But what is a little strange is that the model's performance was better on challenging queries in comparison to moderate ones!

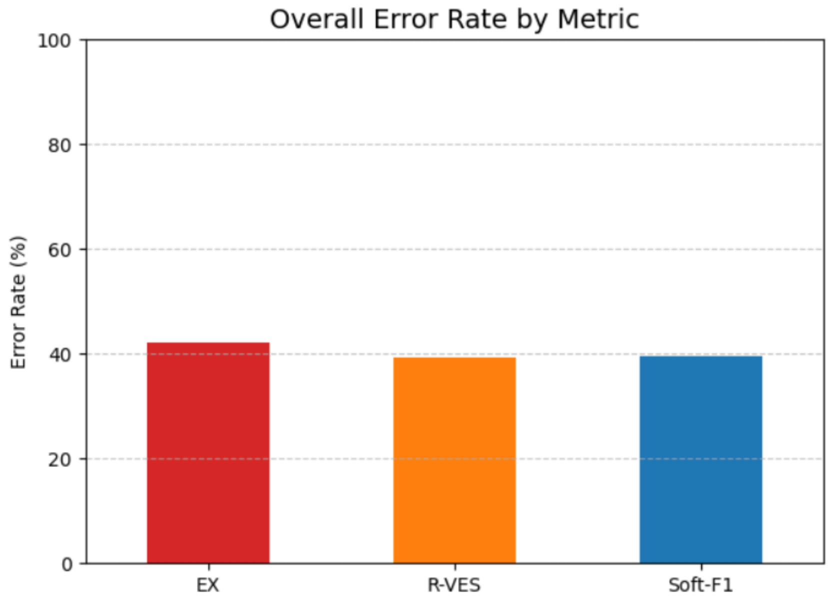


Overall Accuracy:

R-VES and Soft-F1 had better accuracies than EX.

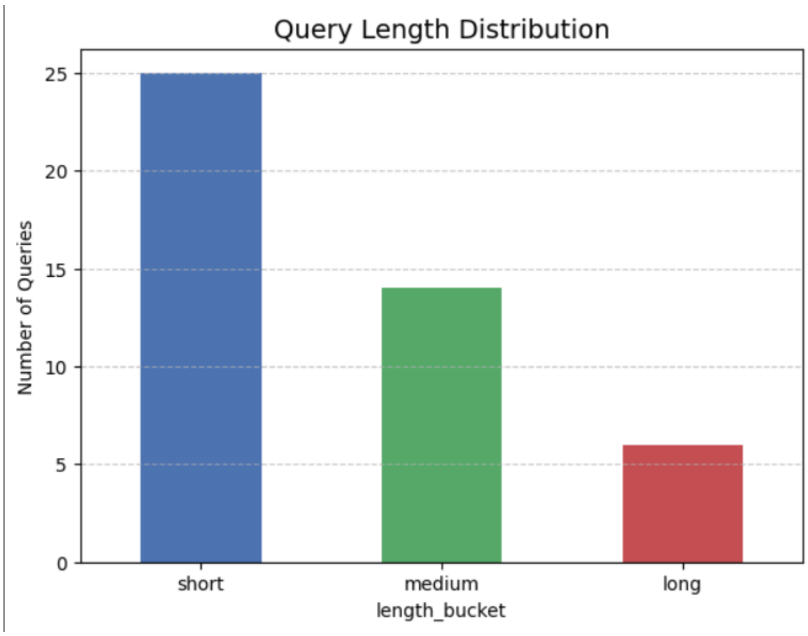


Overall Error-Rate:

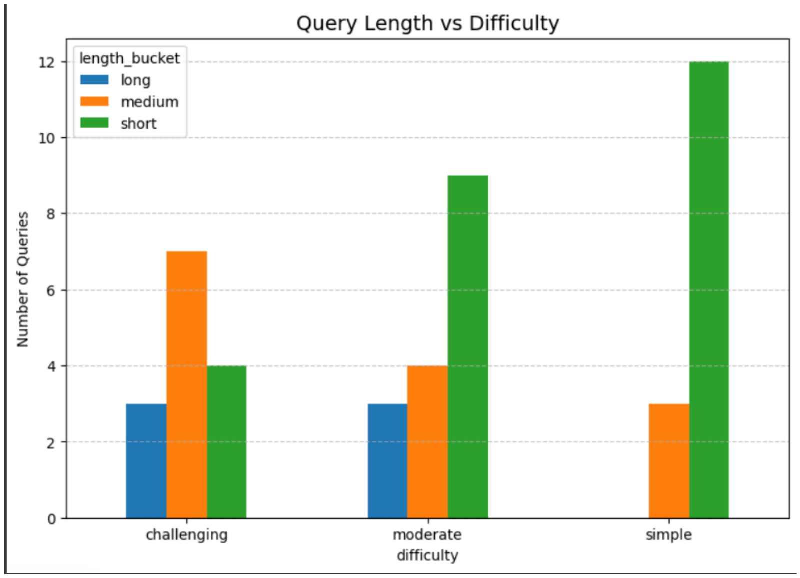


Query Length Distribution:

Here, length smaller than 15 are called short, smaller and equal to 20 are medium and otherwise are long.



Query Length vs Difficulty:



One problem I saw during the generation was that the model did not use explicit JOINs during generation or made different table names, so it resulted in a wrong SQL query and errors during building them. So, I changed the prompt to fix it.

Common sources of failure:

There are some differences in wrong capitalization or table aliases (SQLite treats identifiers as case-insensitive), wrong JOIN structure (the model often joins fewer tables but the ground truth has multi-table joins to connect via foreign keys), and inconsistent functions which cause mismatch (query runs incorrectly in SQLite).

Observed trends:

In simple queries, the model does fairly well. Errors usually were minor naming issues. In moderate queries, there is a big drop! Failures come from multi-join reasoning and date handling. In challenging queries, the model performed slightly better than moderate which suggests the model sometimes handles complex reasoning when schema alignment is clear, but fails on mid-complex joins.

Comparison:

I compared my result to the SOT model of the mini-dev leadership based on the EX metric. I will compare and summarize what I did and what they did.

Baseline:

1. Mine: I used Qwen2.5-Coder-7B-Instruct (which is open-source, smaller than GPT-4) using the dataset of 45 examples. I added database schema into the prompt to reduce hallucinations. I forced explicit JOIN usage when needed.
2. Published Model: They used GPT-4 (which is much larger closed-source LLM). They used Two-stage paradigm to reduce hallucinations (Schema Linking and Logical Synthesis).

Results:

1. Mine: EX: 57.78%, R-VES: 60.78%, Soft-F1: 60.38%
2. Published Model:

METHOD	DEV	TEST
w/o knowledge		
Palm-2	18.77	24.71
Codex	25.42	24.86
ChatGPT	24.05	26.77
ChatGPT+COT	25.88	28.95
Claude-2	28.29	34.60
GPT-4	30.90	34.88
TA-SQL+GPT-4	**50.58** (↑ 19.68)	**54.38** (↑ 19.50)
w/ knowledge		
Palm-2	27.38	33.04
Codex	34.35	36.47
ChatGPT	37.22	39.30
ChatGPT+COT	36.64	40.08
Claude-2	42.70	49.02
DIN-SQL+GPT-4 ♣	50.72	55.90
DAIL-SQL+GPT-4 ♣	54.76	56.08
GPT-4	46.35	54.89
TA-SQL+GPT-4	**56.19** (↑ 9.84)	**59.14** (↑ 4.25)

Table 2: Execution Accuracy (EX) (%) on BIRD. ♣ means the model uses self-consistency or re-modification mechanisms. ↑ is an absolute improvement.

However, by using weaker open-weighted models like CodeLlama and DeepSeek, there is a lower performance:

MODEL	SIM.	MOD.	CHALL.	TOTAL
Closed-Source LLM				
GPT4	54.35	34.64	31.70	46.35
+TA-SQL	63.14	48.60	36.11	56.19
GPT4-turbo	59.35	38.92	27.78	50.19
+TA-SQL	60.54	40.86	38.19	52.48
Claude	51.34	30.07	23.24	42.47
+TA-SQL	56.97	39.78	27.78	48.89
ChatGPT	47.60	22.44	18.31	37.22
+TA-SQL	51.57	33.76	25.69	43.74
Open-Source weaker LLM				
DeepSeek	51.68	29.03	18.06	41.66
+TA-SQL	53.41	32.04	19.44	43.74
CodeLlama	34.81	15.48	11.11	26.73
+TA-SQL	37.30	13.33	11.11	27.57

Strengths & Weaknesses:

1. Mine: Explicit schema prompting and JOIN enforcement likely reduced hallucinations in simple and challenging queries. But there is no specialized two-stage process (my improvements rely only on prompt engineering). Also, my results are only on 45-sample subset.
2. Published Model: Their strength is on structured hallucination mitigation. But there is the limitation of the price of the GPT4 and weakness on challenging queries. They also showed the accuracy gap remains very large when using weaker models.

Overall comparison of EX metric between my result on 45 example and their results using weak models on the whole mini-dev datasets:

Model	Simple	Moderate	Challenging	Total
DeepSeek	51.68	29.03	18.06	41.66
DeepSeek+TA-SQL	53.41	32.04	19.44	43.74
CodeLlama	34.81	15.48	11.11	26.73
CodeLlama+TA-SQL	37.30	13.33	11.11	27.57
Qwen2.5-Coder-7B-Instruct	66.67	50.00	57.14	**57.78**

References:

1. Jianqiao Lu et al., *Before Generation, Align it! A Novel and Effective Strategy for Mitigating Hallucinations in Text-to-SQL Generation*, 2024. https://arxiv.org/abs/2405.15307
2. BIRD: A Benchmark for Large-Scale Database Grounded Text-to-SQL Evaluation. https://bird-bench.github.io/
3. BIRD mini-dev Repository. https://github.com/bird-bench/mini_dev

4. Qwen2.5-Coder-7B-Instruct Model. Hugging Face. https://huggingface.co/Qwen/Qwen2.5-Coder-7B-Instruct