NAACL 2024

KAIST

The 6th Clinical Natural Language Processing Workshop at NAACL 2024

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling

June 21, 2024

Sangryul Kim, Donghee Han, Sehyun Kim

sangryul@kaist.ac.kr

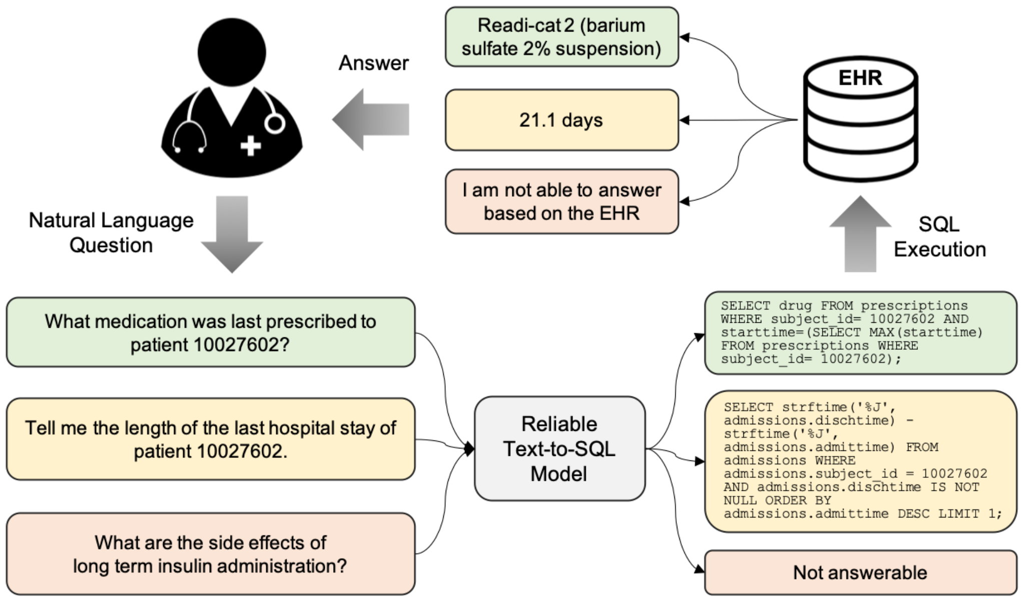
00 Contents

01

Task Description

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling

Overview of Task


Answer
Readi-cat 2 (barium sulfate 2% suspension)
21.1 days
I am not able to answer based on the EHR
EHR
SQL Execution
Natural Language Question
What medication was last prescribed to patient 10027602?
Tell me the length of the last hospital stay of patient 10027602.
What are the side effects of long term insulin administration?
Reliable Text-to-SQL Model
Not answerable

```
SELECT drug FROM prescriptions
WHERE subject_id= 10027602 AND
starttime=(SELECT MAX(starttime)
FROM prescriptions WHERE
subject_id= 10027602);
```

```
SELECT strftime('%J',
admissions.dischtime) -
strftime('%J',
admissions.admittime) FROM
admissions WHERE
admissions.subject_id = 10027602
AND admissions.dischtime IS NOT
NULL ORDER BY
admissions.admittime DESC LIMIT 1;
```

- ✓ EHRs store detailed medical histories; a proposed system uses natural language to SQL for easy querying.
- ✓ Shared task aims to build a reliable EHR question-answering system, ensuring accuracy and avoiding incorrect answers to prevent penalties.

01

Task Description

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling

KAIST

Overview of Metric

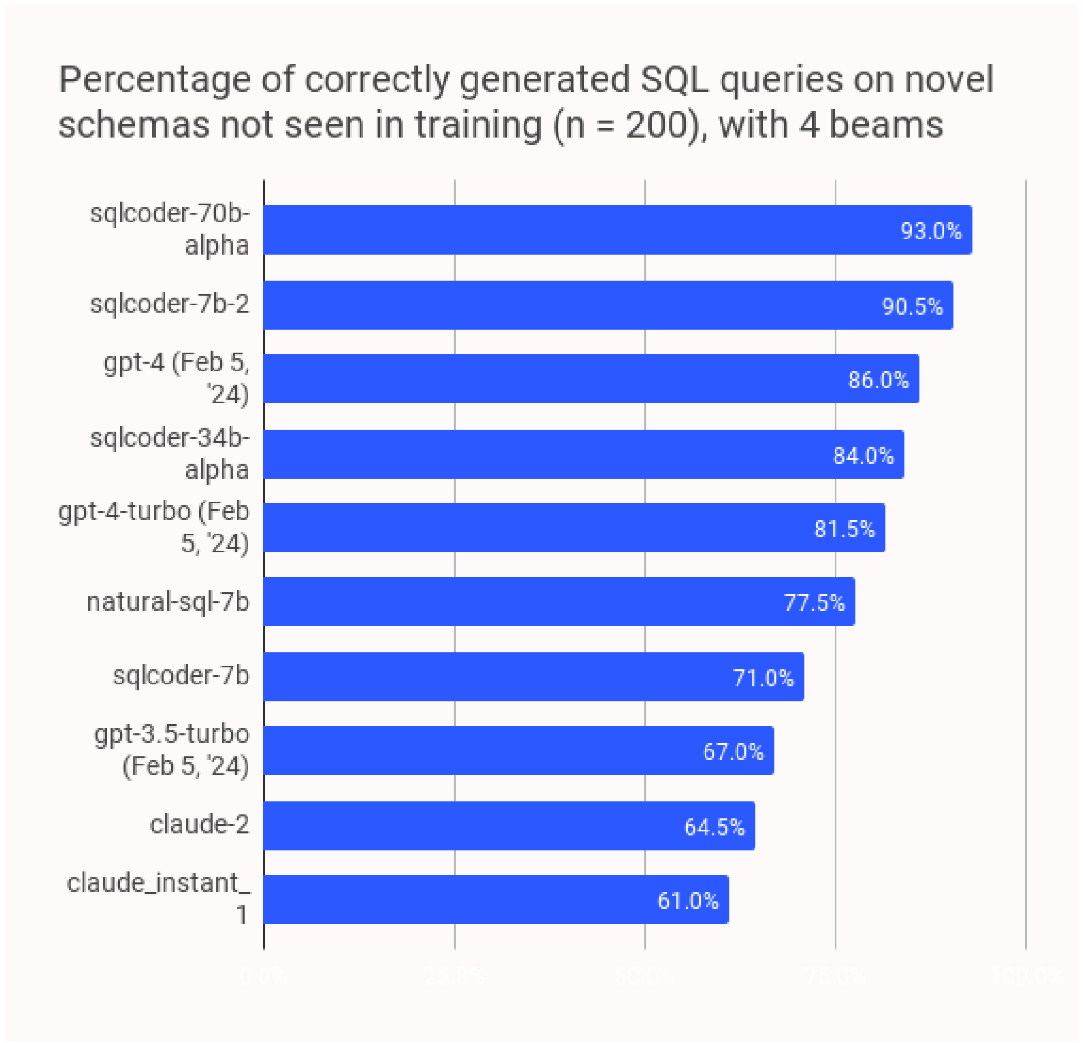
$$\phi_c(x) = \begin{cases} 1 & \text{if } x \in \mathcal{Q}_{ans};\, g(x) = 1;\, Acc(x) = 1 \\ 0 & \text{if } x \in \mathcal{Q}_{ans};\, g(x) = 0, \\ -c & \text{if } x \in \mathcal{Q}_{ans};\, g(x) = 1;\, Acc(x) = 0 \\ -c & \text{if } x \in \mathcal{Q}_{una};\, g(x) = 1, \\ 1 & \text{if } x \in \mathcal{Q}_{una};\, g(x) = 0. \end{cases}$$

- It is very important to reduce the -c point.
- First, create a Text2SQL model with high accuracy for answerable questions.
- Second, Distinguish well between answerable and unanswerable questions, which is equivalent to catching hallucinations in Text2SQL.

02

Backgrounds

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling



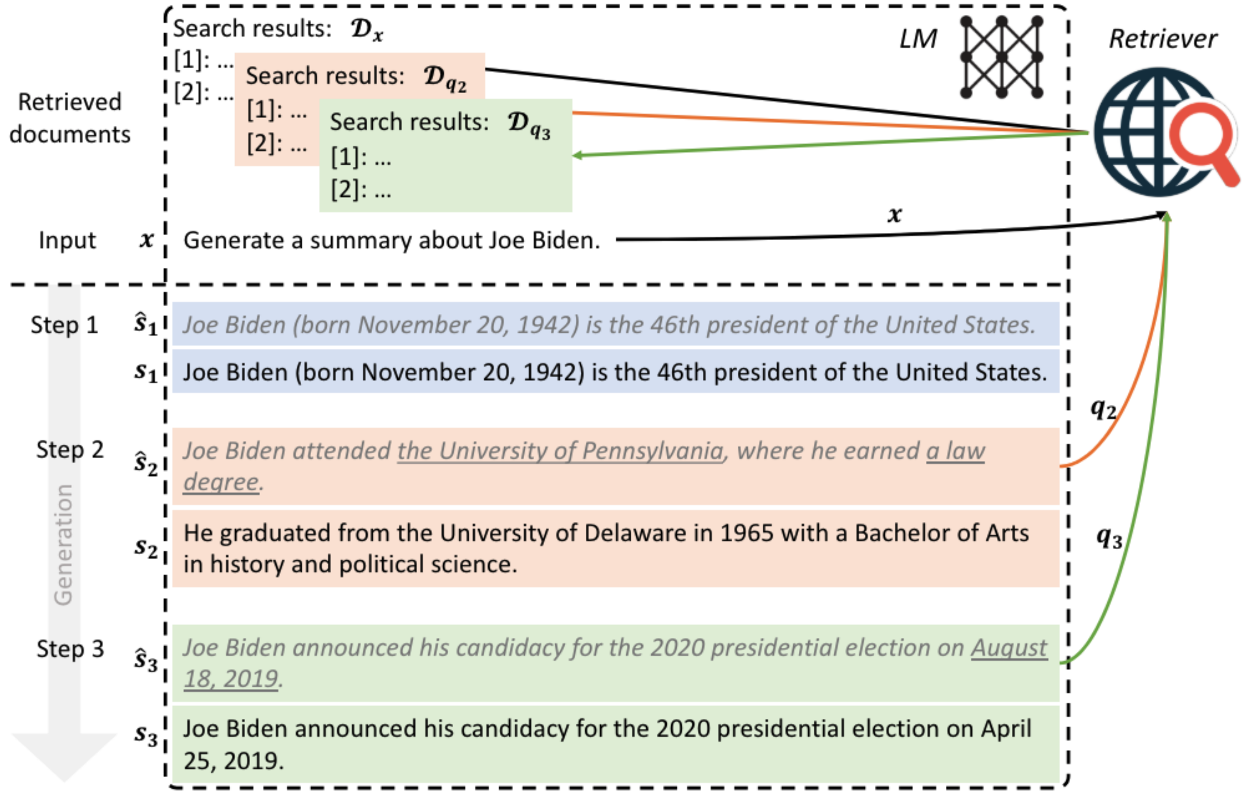
- As the performance of Large Language Models (LLMs) has improved recently, it seems beneficial to fine-tune using the most well-trained LLM available.
- For cost-effective and efficient experimentation, we fine-tuned our project's data using the GPT-3.5-Turbo model.

[1] https://huggingface.co/defog/sqlcoder-7b-2

02

Backgrounds

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling


Search results: $\mathcal{D}_x$
[1]: ...
[2]: ...
Search results: $\mathcal{D}_{q_2}$
[1]: ...
[2]: ...
Search results: $\mathcal{D}_{q_3}$
[1]: ...
[2]: ...
LM
Retriever
Retrieved documents
Input x
Generate a summary about Joe Biden.
x
Step 1
$\hat{s}_1$ Joe Biden (born November 20, 1942) is the 46th president of the United States.
s_1 Joe Biden (born November 20, 1942) is the 46th president of the United States.
Step 2
$\hat{s}_2$ Joe Biden attended the University of Pennsylvania, where he earned a law degree.
s_2 He graduated from the University of Delaware in 1965 with a Bachelor of Arts in history and political science.
Step 3
$\hat{s}_3$ Joe Biden announced his candidacy for the 2020 presidential election on August 18, 2019.
s_3 Joe Biden announced his candidacy for the 2020 presidential election on April 25, 2019.
q_2
q_3
Generation

$$y_t = \begin{cases} \hat{s}_t & \text{if all tokens of } \hat{s}_t \text{ have probs} \geq \theta \\ s_t = \mathrm{LM}([\mathcal{D}_{q_t}, x, y_{<t}]) & \text{otherwise} \end{cases}$$

- In the FLARE paper on retrieval-augmented generation, if the probability of any generated token falls below a certain threshold (θ), retrieval is triggered based on that token.

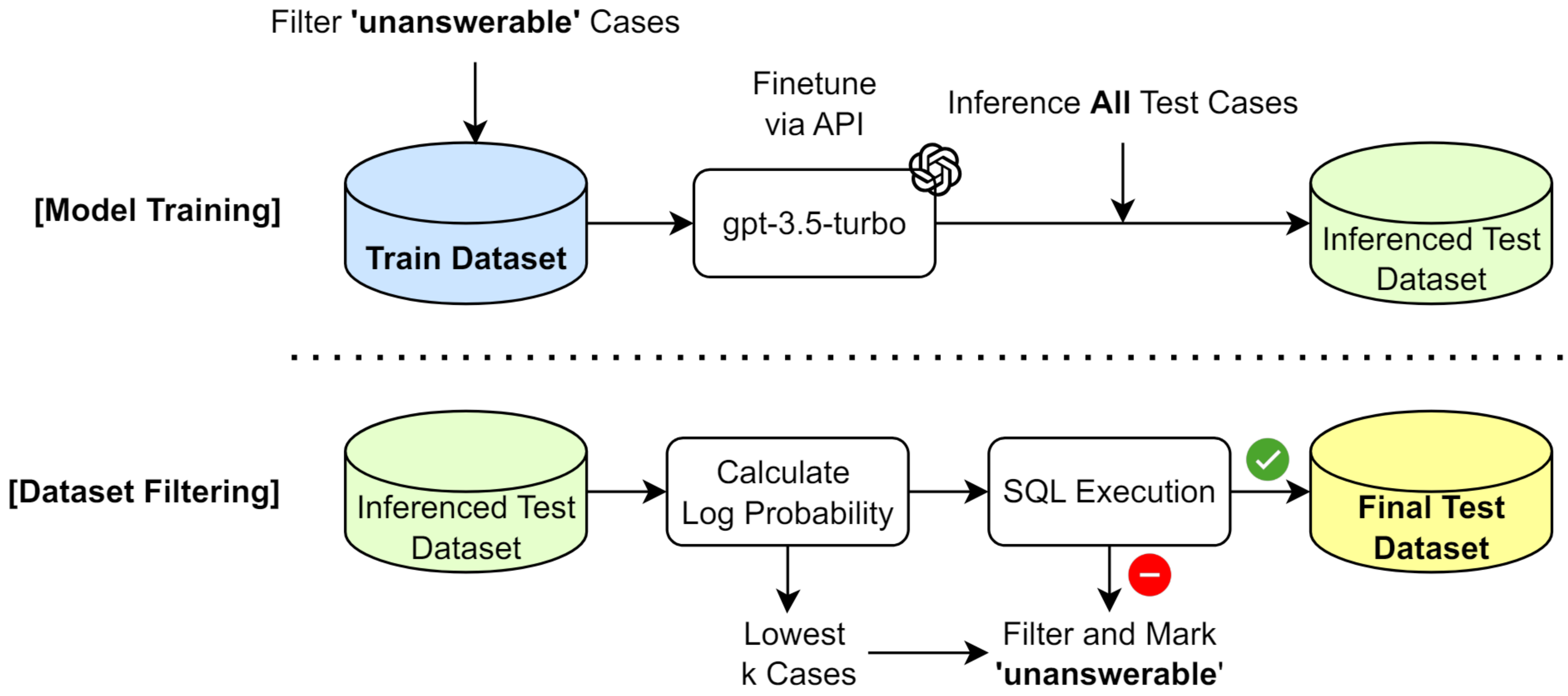
- This means that the log probability of a token can measure the model's uncertainty and be used to reduce hallucinations

[2] Active Retrieval Augmented Generation, EMNLP 2023

Our Methods

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling

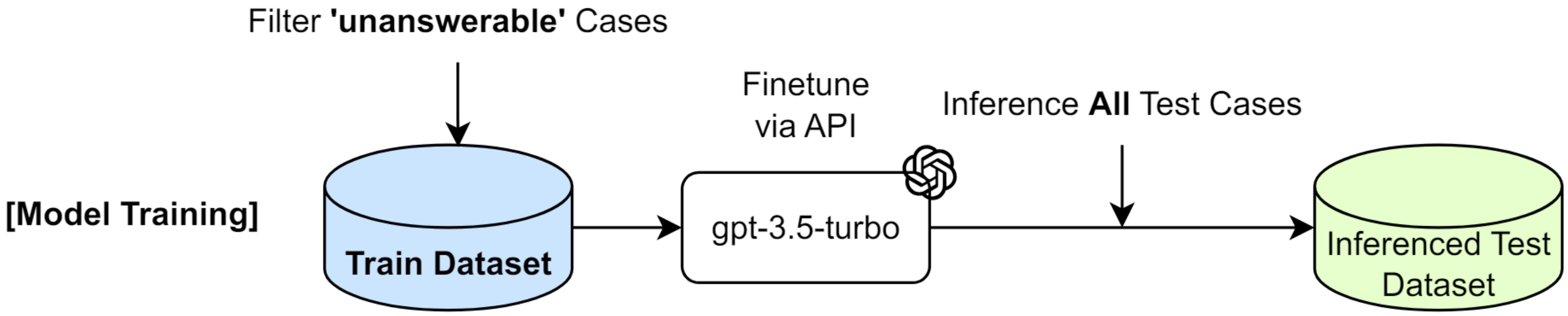
Our Overall Method



Our Methods

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling

Our Overall Method

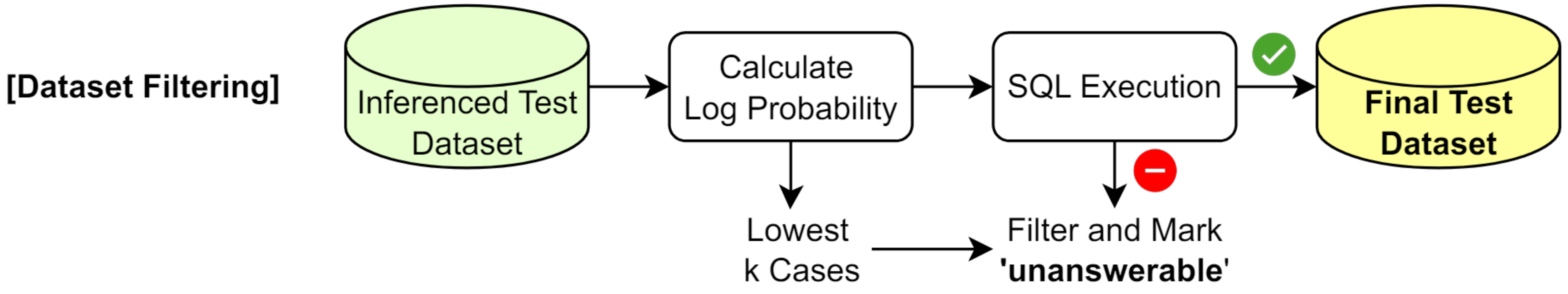


- ✓ We first fine-tuned a Text2SQL model to generate SQL from given questions, removing unanswerable questions during training.
- ✓ This process aims to maximize Text2SQL accuracy to accurately answer answerable questions in the test dataset.

Our Methods

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling

Our Overall Method

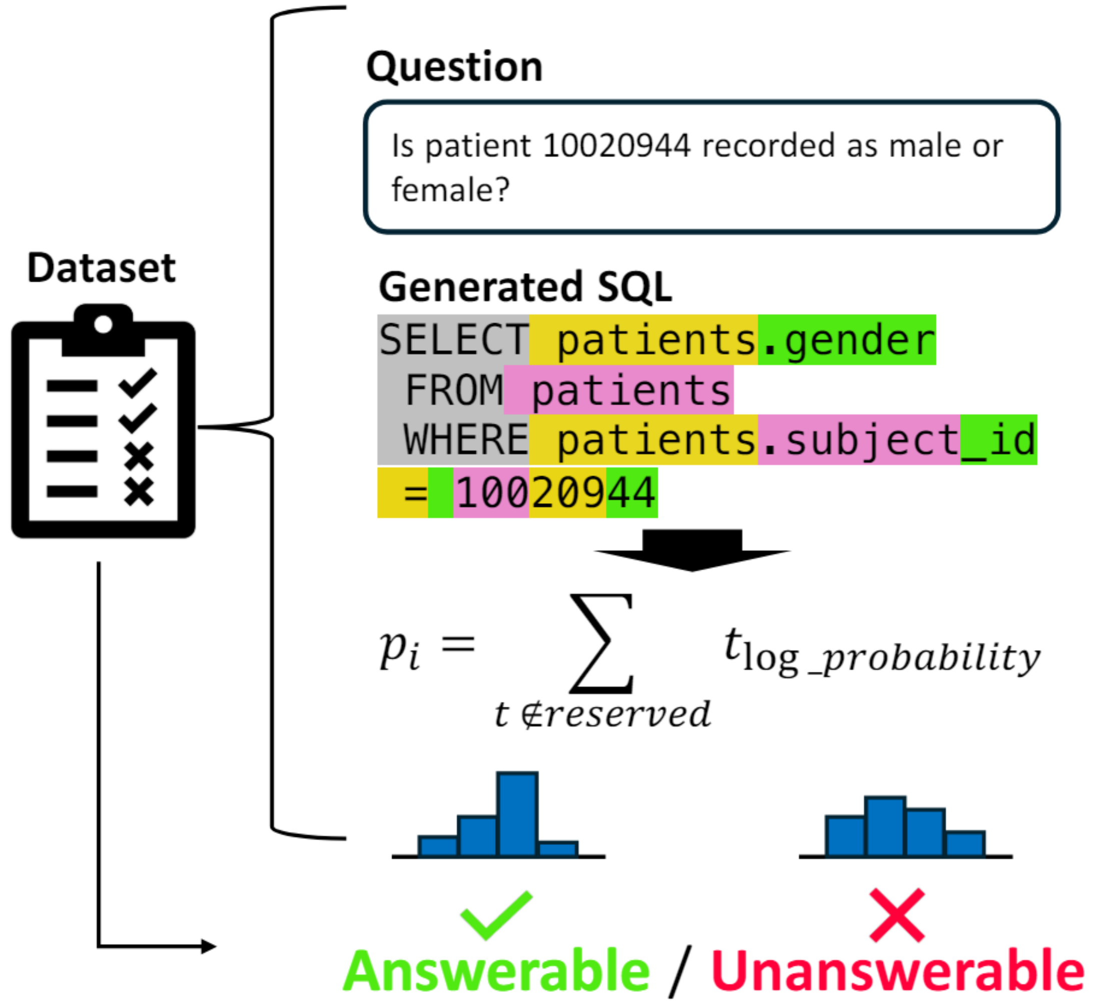


- ✓ We propose a two-step filtering method here to distinguish unanswerable SQL queries that were incorrectly generated due to hallucination.
- ✓ The first is a filtering method called "**ProbGate**" which uses log probability, and the second involves filtering through SQL Execution to verify any syntax errors in the generated SQL.

Our Methods

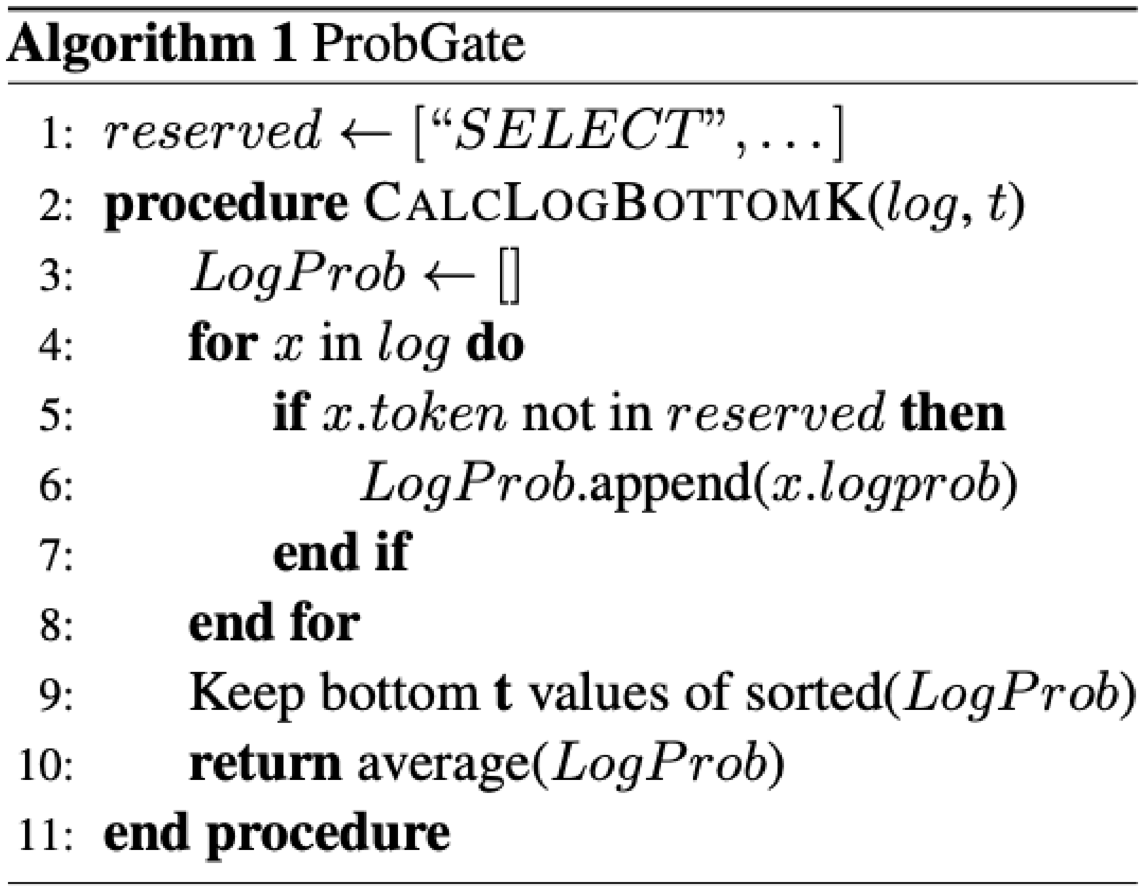
ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling

ProbGate


Dataset
Question
Is patient 10020944 recorded as male or female?
Generated SQL
SELECT patients.gender FROM patients WHERE patients.subject_id = 10020944
$p_i = \sum_{t \notin reserved} t_{\log\_probability}$
Answerable / Unanswerable

- For a given question, we can calculate the log probability of each token when generating SQL.
- We average the log probability values for tokens (up to 't' tokens), excluding reserved words like SELECT, FROM, etc.
- This averaged value is considered representative of the log probability for one test dataset. We compare it against other datasets, considering the lowest 'k' datasets as unanswerable.

ProbGate



Algorithm 1 ProbGate

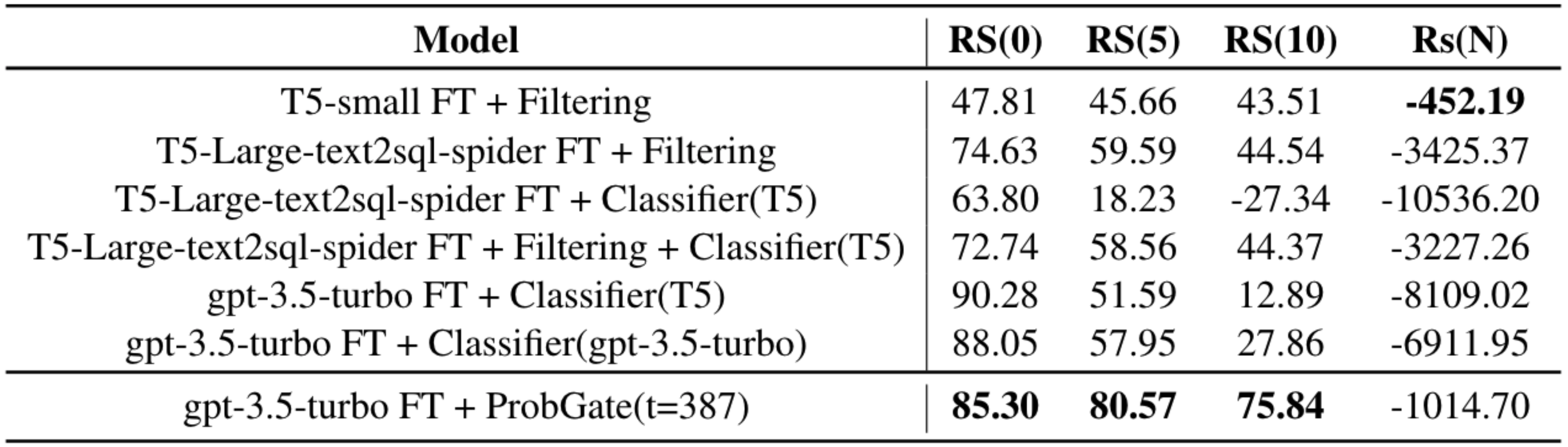
```
$reserved \leftarrow [\text{"}SELECT\text{"}, \ldots]$
procedure CALCLOGBOTTOMK($log$, $t$)
    $LogProb \leftarrow []$
    for $x$ in $log$ do
        if $x.token$ not in $reserved$ then
            $LogProb$.append($x.logprob$)
        end if
    end for
    Keep bottom t values of sorted($LogProb$)
    return average($LogProb$)
end procedure
```

- Our primary focus here was finding the appropriate values for the parameters '**t**' and '**k**'.
- We assumed the optimal value of '**k**' to be 425, based on the final outcomes of submissions labeled as '**null**' and the performance of the fine-tuned model.
- Similarly, through hyperparameter optimization, we determined the value of '**t**' to be 10.

04

Results and Analysis

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling



Model	RS(0)	RS(5)	RS(10)	Rs(N)
T5-small FT + Filtering	47.81	45.66	43.51	**-452.19**
T5-Large-text2sql-spider FT + Filtering	74.63	59.59	44.54	-3425.37
T5-Large-text2sql-spider FT + Classifier(T5)	63.80	18.23	-27.34	-10536.20
T5-Large-text2sql-spider FT + Filtering + Classifier(T5)	72.74	58.56	44.37	-3227.26
gpt-3.5-turbo FT + Classifier(T5)	90.28	51.59	12.89	-8109.02
gpt-3.5-turbo FT + Classifier(gpt-3.5-turbo)	88.05	57.95	27.86	-6911.95
gpt-3.5-turbo FT + ProbGate(t=387)	**85.30**	**80.57**	**75.84**	-1014.70

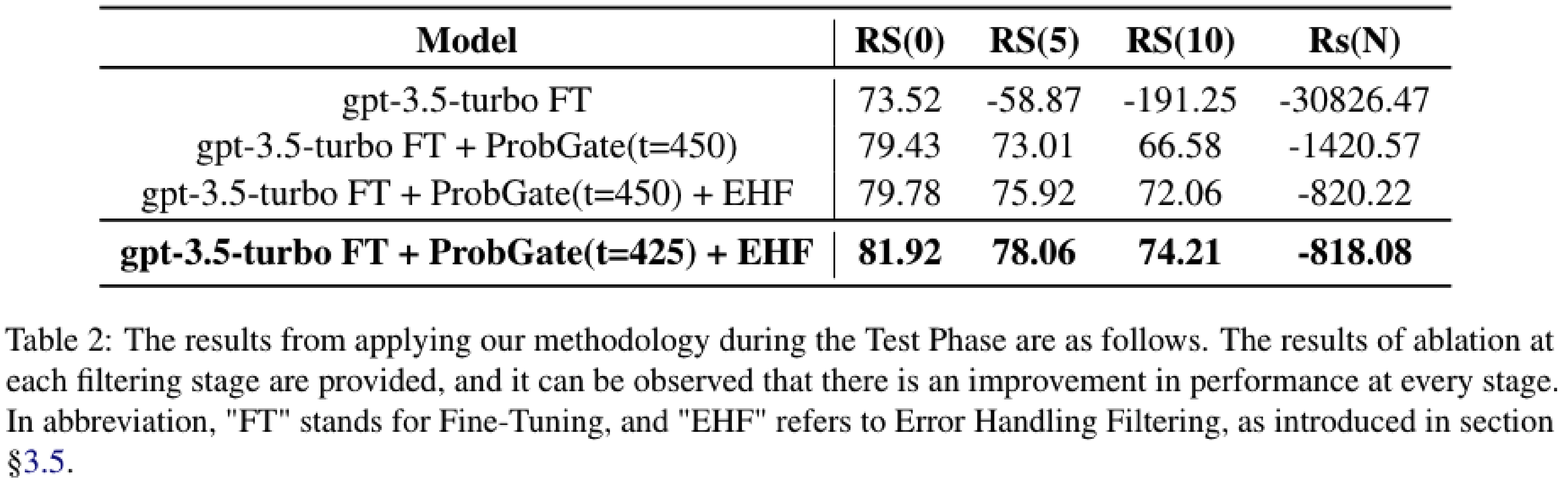
Table 1: Model Selection and Ablation Study in Dev Phase dataset. In the case of the T5-Large model, it is a model that was first fine-tuned using the Spider dataset, which is one of the Text2SQL datasets, and then subsequently trained on the EHRSQL dataset. In abbreviation, "FT" stands for Fine-Tuning.

- We compared various models, filtering methods, and classifiers in the dev set with our final methodology, but none showed satisfactory performance.
- This methodology has been proven optimal for datasets in this project, where the distribution of train, dev, and test sets significantly differs.

04

Results and Analysis

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling



Model	RS(0)	RS(5)	RS(10)	Rs(N)
gpt-3.5-turbo FT	73.52	-58.87	-191.25	-30826.47
gpt-3.5-turbo FT + ProbGate(t=450)	79.43	73.01	66.58	-1420.57
gpt-3.5-turbo FT + ProbGate(t=450) + EHF	79.78	75.92	72.06	-820.22
gpt-3.5-turbo FT + ProbGate(t=425) + EHF	**81.92**	**78.06**	**74.21**	**-818.08**

Table 2: The results from applying our methodology during the Test Phase are as follows. The results of ablation at each filtering stage are provided, and it can be observed that there is an improvement in performance at every stage. In abbreviation, "FT" stands for Fine-Tuning, and "EHF" refers to Error Handling Filtering, as introduced in section §3.5.

- In our final test set ablation study, we found that the combination of GPT-3.5-Turbo FT, ProbGate (t=425), and Error Handling Filtering yielded the best results.
- Ultimately, the smallest variance in results from rs0 to rs10 means that our approach incurred fewer '-c' penalties compared to other methodologies.

04

Results and Analysis

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling

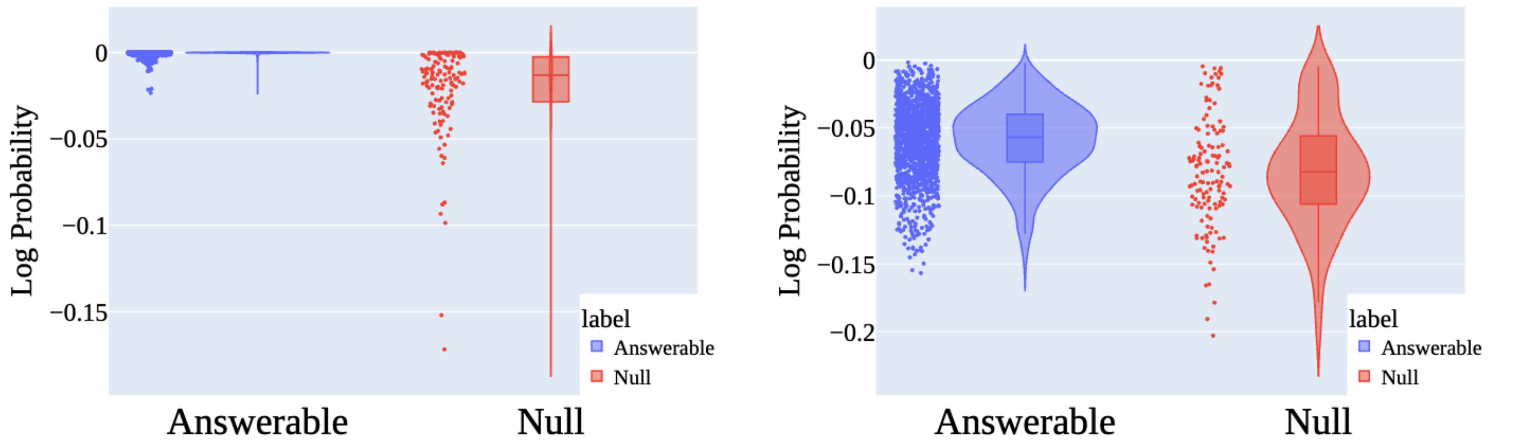


Figure 3: Left - Log Probability Distribution of the Fine-Tuned Model, Right - Log Probability Distribution of the Unfine-Tuned Model

- We compare the log probability distribution between 30% mixed answerable and unanswerable data using a model trained on about 70% answerable data from the train dataset.
- A clear log probability difference between answerable and unanswerable questions is evident, and this distinction is more pronounced in the fine-tuned model.

05

Conclusion, Limitations

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling

KAIST

Conclusion

1 Why Answerable data only finetune with LLM?

- ✓ Using a dataset with minimal noise ensures high accuracy and guarantees stable generation of complex SQL statements.

2 Why ProbGate?

- ✓ The trained model can effectively filter answerable and unanswerable data across different data distributions stably.

3 Why EHF?

- ✓ Grammatical errors can ultimately only be detected by actual execution; SQLs that pass this stage are syntactically correct, executable, and likely to achieve high accuracy.

Conclusion, Limitations

Limitations

1. It was challenging to distinguish between SQL statements for nonsensical questions (truly unanswerable) and those that are executable but query non-existent values in the table.

2. It was difficult to determine the hyperparameter values 't' and 'k' depending on the data. During the competition, we believed that finding them empirically was the best approach.

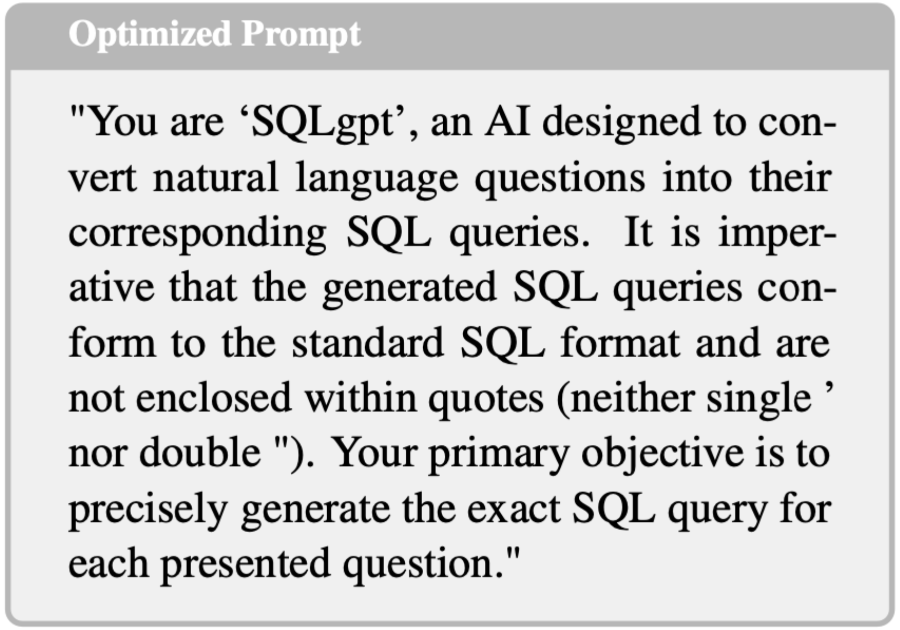
3. Although there were Text2SQL models with performance exceeding gpt-3.5-turbo, we failed to utilize them effectively, and our inability to make good use of table schema information remains a limitation.

06

Appendix

ProbGate at EHRSQL 2024: Enhancing SQL Query Generation Accuracy through Probabilistic Threshold Filtering and Error Handling

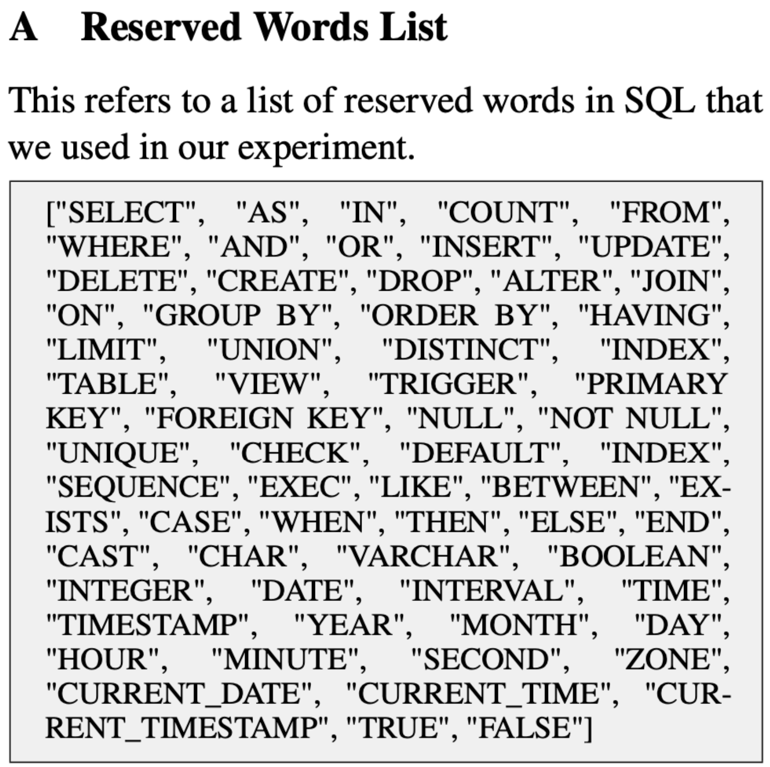
Optimized prompt for Fine-tuning Model



Optimized Prompt

"You are 'SQLgpt', an AI designed to convert natural language questions into their corresponding SQL queries. It is imperative that the generated SQL queries conform to the standard SQL format and are not enclosed within quotes (neither single ' nor double "). Your primary objective is to precisely generate the exact SQL query for each presented question."

Reserved Words



A Reserved Words List

This refers to a list of reserved words in SQL that we used in our experiment.

["SELECT", "AS", "IN", "COUNT", "FROM", "WHERE", "AND", "OR", "INSERT", "UPDATE", "DELETE", "CREATE", "DROP", "ALTER", "JOIN", "ON", "GROUP BY", "ORDER BY", "HAVING", "LIMIT", "UNION", "DISTINCT", "INDEX", "TABLE", "VIEW", "TRIGGER", "PRIMARY KEY", "FOREIGN KEY", "NULL", "NOT NULL", "UNIQUE", "CHECK", "DEFAULT", "INDEX", "SEQUENCE", "EXEC", "LIKE", "BETWEEN", "EXISTS", "CASE", "WHEN", "THEN", "ELSE", "END", "CAST", "CHAR", "VARCHAR", "BOOLEAN", "INTEGER", "DATE", "INTERVAL", "TIME", "TIMESTAMP", "YEAR", "MONTH", "DAY", "HOUR", "MINUTE", "SECOND", "ZONE", "CURRENT_DATE", "CURRENT_TIME", "CURRENT_TIMESTAMP", "TRUE", "FALSE"]

Thank you!

Any Questions?