Text-to-SQL Code Generation vs Text Generation Models

Saurav Kumar Singh[1], Shreya Surendra Choure[2], and Upasana Chaudhari[1]

[1]saurav2@pdx.edu, upasana@pdx.edu
[2]schoure@pdx.edu

1 Project Overview

Our project investigates the performance enhancements in translating natural language questions into SQL queries through Parameter-Efficient Supervised Fine-Tuning (PEFT) on models such as Code Bison, GPT-3.5, and Gemma. By incorporating both prompt engineering and fine-tuning techniques, the initiative aims to refine these models' ability to understand and generate accurate SQL queries based on a given schema or the context of a table.

2 Main Goals

Investigate the improved performance of code and text generation models following parameter efficient supervised fine tuning on task of translating natural language questions into SQL queries, considering the provision of a given schema or context.
What problem task(s) do you address?

1. Closed models perform after fine tuning latest LLMs.
2. Lack of text-to-sql research on latest models like Gemma.
3. Generating visualization directly from Natural language question.

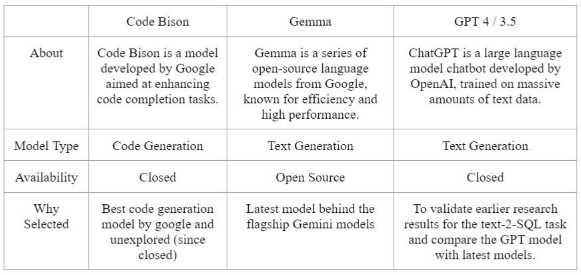
3 Dataset Description

This dataset is built from the two famous datasets: WikiSQL and Spider. Comprising 78,577 instances that include natural language queries, corresponding SQL CREATE TABLE statements, and SQL queries that address the questions within the context provided by the CREATE statements. Designed with text-to-SQL language models in mind, this dataset aims to mitigate the frequent misattribution of column and table names that often occurs when models are trained on text-to-SQL datasets. For fine tuning our large language models we have used 55K records from the train dataset and 1K records as test dataset. The training data was later transformed into the model required fine tuning format.https://huggingface.co/datasets/b-mc2/sql-create-context

Data Sample:
Question:
What is the country with the album best of and weeks on chart is less than 5?
Context:
```
CREATE TABLE table_name_53
(country VARCHAR, album VARCHAR,
weeks_on_chart VARCHAR)
```
Answer:
```
SELECT country FROM table_name_53
WHERE album = "best of" AND
weeks_on_chart < 5
```

4 Methodology

4.1 Models Tested through



	Code Bison	Gemma	GPT 4 / 3.5
About	Code Bison is a model developed by Google aimed at enhancing code completion tasks.	Gemma is a series of open-source language models from Google, known for efficiency and high performance.	ChatGPT is a large language model chatbot developed by OpenAI, trained on massive amounts of text data.
Model Type	Code Generation	Text Generation	Text Generation
Availability	Closed	Open Source	Closed
Why Selected	Best code generation model by google and unexplored (since closed)	Latest model behind the flagship Gemini models	To validate earlier research results for the text-2-SQL task and compare the GPT model with latest models.

Figure 1: Model Tested Through

4.2 Fine tuning proposed approach for Open Source

Parameter-Efficient Fine Tuning offers a more resource-conscious approach to model refinement, demanding fewer computational and storage resources. It enhances model performance, especially in understanding and responding to language-based tasks, and shows superior adaptability and generalization compared to conventional fine-tuning techniques.
Low-Rank Adaptation, or LoRA refines the fine-tuning process for large language models by targeting text generation tasks, leading to cost savings and faster response times. It also mitigates storage issues without degrading the quality of the model's outputs.
Quantized Low-Rank Adaptation or QLoRA, takes the principles of LoRA further by implementing quantization. This advanced method decreases resource requirements even more, yet it either preserves or improves the model's performance on various tasks.

4.3 Code Bison Fine tuning workflow

The flowchart describes the process for fine-tuning the Code Bison model using Vertex AI. Initially, the dataset is formatted for Vertex AI and loaded into a cloud bucket. Next, the fine-tuning pipeline is established. After tuning, the model's performance is evaluated using queries, and evaluation metrics are generated. The outcome is a fine-tuned model endpoint ready for deployment.

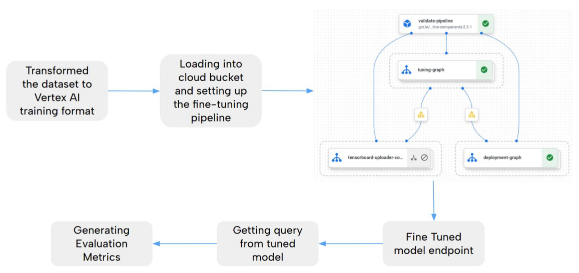

Transformed the dataset to Vertex AI training format
Loading into cloud bucket and setting up the fine-tuning pipeline
validate-pipeline
tuning-graph
tensorboard-uploader-co...
deployment-graph
Fine Tuned model endpoint
Getting query from tuned model
Generating Evaluation Metrics

Figure 2: Code Bison Model Fine Tuning Work Flow

4.4 GPT 3.5 turbo Fine tuning workflow

The image presents a fine-tuning workflow for GPT-3.5 Turbo: For GPT-3.5 Turbo fine-tuning, we used 10,000 records for fine-tuning and 1,000 records for testing. We couldn't use 55,000 records because it was too expensive to train on all the records.Data is first prepared and validated. A fine-tuning job is then created, followed by the model's validation. Finally, the model undergoes testing and evaluation.

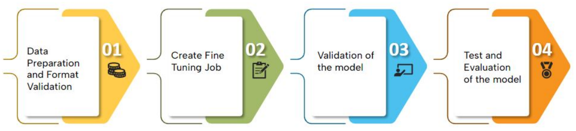

Data Preparation and Format Validation
01
Create Fine Tuning Job
02
Validation of the model
03
Test and Evaluation of the model
04

Figure 3: GPT-3.5 turbo Fine Tuning Work Flow

4.5 Gemma Fine tuning workflow

The workflow diagram for fine-tuning Gemma begins with dataset transformation, followed by setting up training hyperparameters. Then, configurations for LoRA and QLoRA are applied. This leads to the merging of the base and fine-tuned model, culminating in a supervised fine-tuning phase, after which the model's performance is evaluated.

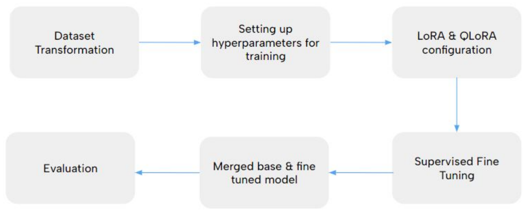

Dataset Transformation
Setting up hyperparameters for training
LoRA & QLoRA configuration
Supervised Fine Tuning
Merged base & fine tuned model
Evaluation

Figure 4: Gemma Model Fine Tuning Work Flow

5 Result

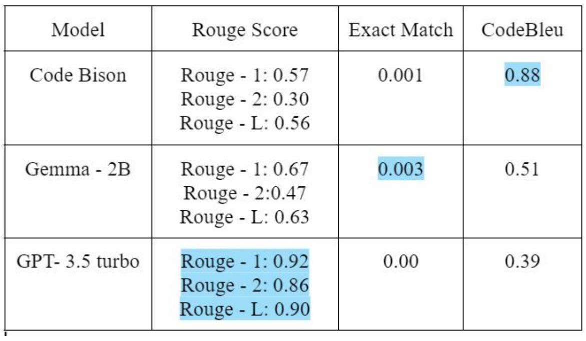
5.1 Performance - Prompt Engineering

ROUGE Scores (measuring text similarity):
Code Bison: Shows the lowest ROUGE scores across the board, indicating less similarity to reference summaries or texts.
Gemma - 2B: Demonstrates moderate improvement over Code Bison with better scores, particularly in ROUGE-2, which means it's better at capturing two-word phrases.
GPT-3.5 Turbo: Has the highest ROUGE scores, suggesting it's most effective at producing text closely matching the reference materials.

Exact Match (measuring identical output):
Code Bison: Has a very low score (0.001), indicating it almost never produces an output that's

exactly the same as the reference.
Gemma - 2B: Improves slightly over Code Bison with a score of 0.003.
GPT-3.5 Turbo: Scores 0.00, which means it did not produce any exact matches in the data provided.

CodeBLEU (measuring code generation quality): Code Bison: The highest score at 0.88, indicating it excels in code generation or understanding, possibly outperforming the others in tasks like source code summarization or translation.
Gemma - 2B: Scores 0.51, which is significantly lower than Code Bison, suggesting it has moderate performance in code-related tasks.
GPT-3.5 Turbo: With the lowest score of 0.39, it might be less suited for code generation tasks compared to the other models.



Model	Rouge Score	Exact Match	CodeBleu
Code Bison	Rouge - 1: 0.57 Rouge - 2: 0.30 Rouge - L: 0.56	0.001	0.88
Gemma - 2B	Rouge - 1: 0.67 Rouge - 2:0.47 Rouge - L: 0.63	0.003	0.51
GPT- 3.5 turbo	Rouge - 1: 0.92 Rouge - 2: 0.86 Rouge - L: 0.90	0.00	0.39

Figure 5: Prompt Engineering Performance

5.2 Performance - Fine Tuning

ROUGE Scores (measuring text similarity): Code Bison: Shows high ROUGE scores (1: 0.95, 2: 0.90, L: 0.94), indicating a very strong similarity to reference texts.
Gemma - 2B: Has slightly lower ROUGE scores (1: 0.93, 2: 0.88, L: 0.92) compared to Code Bison, but still indicates high similarity.
GPT-3.5 Turbo: Achieves the highest ROUGE scores (1: 0.99, 2: 0.97, L: 0.98), suggesting it is exceptionally effective at producing text similar to reference materials.

Exact Match (measuring identical output): Code Bison: Scores relatively high at 0.67, indicating it often produces outputs that are exactly the same as the reference.
Gemma - 2B: Scores a bit lower at 0.59, suggesting it is less likely than Code Bison to produce an exact match.
GPT-3.5 Turbo: Has the highest score of 0.79, showing it is most likely to generate outputs that exactly match the reference.

CodeBLEU (measuring code generation quality): Code Bison: Has a high score at 0.88, indicating strong performance in code-related tasks.
Gemma - 2B: Shows a very similar score at 0.89, which is nearly equivalent to Code Bison, suggesting comparable ability in code tasks.
GPT-3.5 Turbo: With the highest score at 0.93, it is indicated to be the best at code generation or understanding tasks.

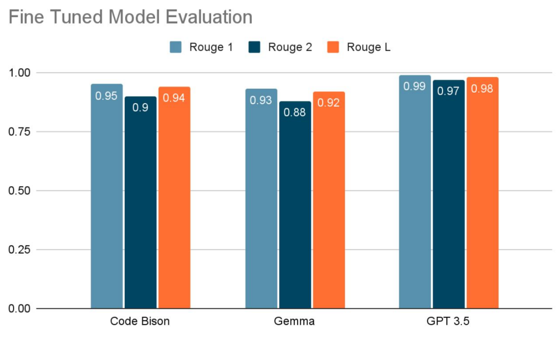


Figure 6: Rouge Scores

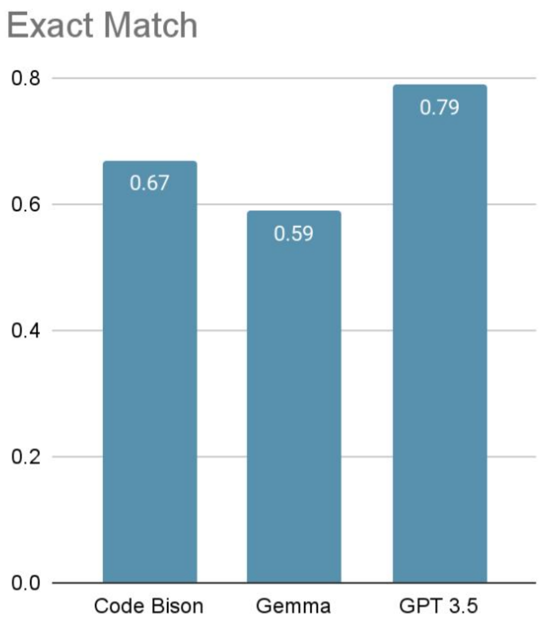


Figure 7: Exact Match

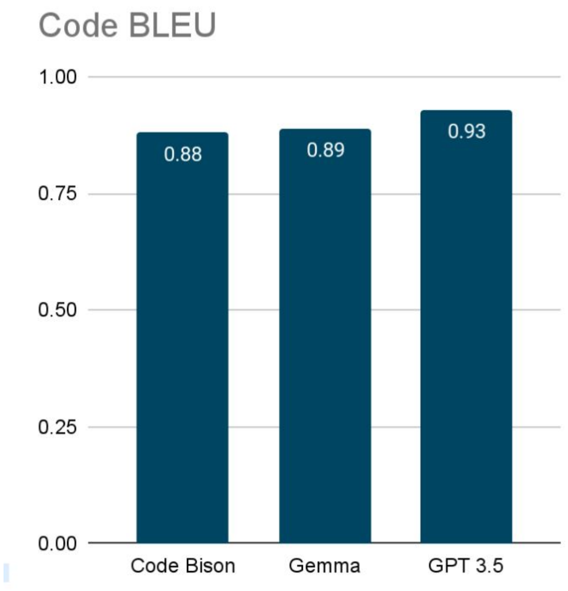
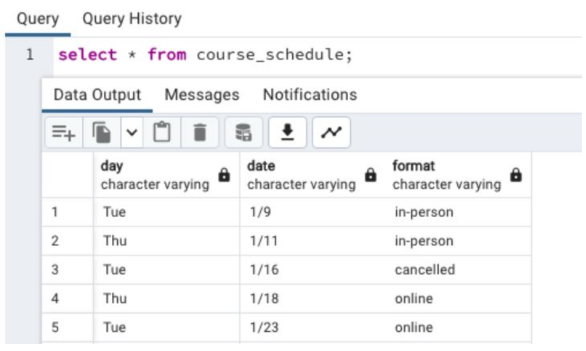


Figure 8: CodeBleu

5.3 Model Experimentation

For demonstration purposes we created a table for how many LLM classes were online and how many of them were offline.



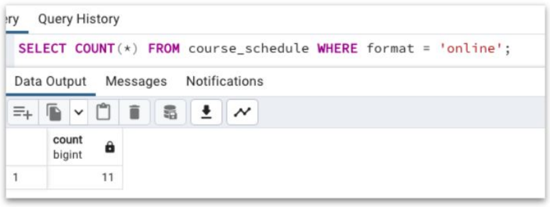
```
select * from course_schedule;
```

	day character varying	date character varying	format character varying
1	Tue	1/9	in-person
2	Thu	1/11	in-person
3	Tue	1/16	cancelled
4	Thu	1/18	online
5	Tue	1/23	online

Figure 9: Sample Data from Class Schedule Table

Question:
Count number of online classes.
Context:
`CREATE TABLE course_schedule (day VARCHAR, format VARCHAR)`
Generated Queries / Answer:
Code Bison & Gemma:
`SELECT COUNT(*) FROM course_schedule WHERE format = "Online"`
GPT - 3.5 turbo:
`SELECT COUNT(*), format FROM course_schedule WHERE day = "online" GROUP BY format`

Correct Answer:11
Our Answer:11



```
SELECT COUNT(*) FROM course_schedule WHERE format = 'online';
```

	count bigint
1	11

Figure 10: Result of the Query

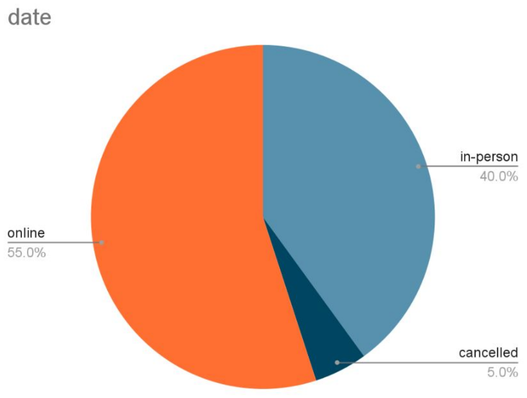


Figure 11: Piechart of the class schedule

6 Interesting insights

- **Superiority of Closed Models**: Closed models generally surpass open-source counterparts in performance metrics.
- **Enhancement Post Fine-Tuning**: Precision markedly improves when models are fine-tuned on specific tasks.
- **Limitation in Exact Match**: Low exact match scores are attributed to the fact that various SQL queries may yield identical results despite differing syntax.
- **High Performance on Simple Queries**: Models demonstrate around 95% accuracy with basic SQL queries.
- **Complex Query Challenges**: Accuracy declines when handling complex queries involving JOINs, AGGREGATIONS, and SUBQUERIES.

- **Contextual Generation Capabilities**: Models can autonomously generate relevant context, which, while not flawless, adds meaningful content.

7 Ethical Considerations

- **Avoiding Misuse**: Ensure that the Text-to-SQL model is not used to generate SQL queries for malicious purposes, such as unauthorized access to databases, data breaches, or cyber attacks.

- **Respecting Privacy**: Do not use the model to extract sensitive information from databases without proper authorization or consent. Respect the privacy and confidentiality of individuals' data at all times.

- **Legal Compliance**: Ensure that the use of the Text-to-SQL model complies with all relevant laws and regulations, including data protection laws (Like GDPR).

8 Future Extensions

- Fine-tune models for specialized domains like finance, healthcare, or legal queries, leading to more accurate and relevant SQL generation.

- Extend the models to work seamlessly with various types of databases, such as NoSQL databases.

- Enhance the models to automatically understand and adapt to different database schemas, reducing the need for manual configuration.

- Invest in more sophisticated NLU (Natural Language Understanding) techniques to handle complex queries, including those with nested conditions, aggregates, joins, and subqueries, more effectively translating natural language to SQL.

- Add capabilities for visualizing query results directly, providing users with more intuitive and insightful ways to interpret and analyze their data.

9 Group Contribution

Option 1: We agree that all group members made a valuable contribution and therefore believe it is fair that each member receive the same grade for the discussion.

References

[1] Hugging Face Datasets: SQL Create Context. https://huggingface.co/datasets/b-mc2/sql-create-context

[2] OpenAI Platform Documentation: Fine-Tuning Guide. https://platform.openai.com/docs/guides/fine-tuning

[3] OpenAI Platform Documentation: Prompt Engineering Guide. https://platform.openai.com/docs/guides/prompt-engineering

[4] Hugging Face Blog: Gemma PEFT. https://huggingface.co/blog/gemma-peft

[5] Hugging Face Documentation: PEFT. https://huggingface.co/docs/peft/en/index

[6] Hugging Face Spaces: Evaluate Metric Exact Match. https://huggingface.co/spaces/evaluate-metric/exact_match

[7] GitHub: CODEBLEU. https://github.com/k4black/codebleu

[8] Google Cloud: Vertex AI for Generative AI models. https://cloud.google.com/vertex-ai/generative-ai/docs/models/tune-code-models

[9] Spider Text-to-SQL Gemma/Keras. https://www.kaggle.com/code/mpwolke/spider-text-to-sql-gemma-keras