Fine tuning LLMs on Gretel's synthetic text-to-sql dataset

Models used- Meta-LLama-1B and DeepSeek-R1-1.5B

Siddharth Dumbre

Dataset Overview

- Source: Gretel's synthetic text-to-SQL dataset
- Domain: Filtered for "Wildlife Conservation"
- Structure:

Input:

Instruction: Natural Language Query

Context: Database Schema

Output:

Response: SQL Query

Explanation: Description of SQL query

```
Filtered dataset: {'train': Dataset({
    features: ['sql_prompt', 'sql_context', 'sql', 'sql_explanation'],
    num_rows: 996
}), 'test': Dataset({
    features: ['sql_prompt', 'sql_context', 'sql', 'sql_explanation'],
    num_rows: 52
})}
```

Training Configuration

```
num_train_epochs=3,

per_device_train_batch_size=2,

gradient_accumulation_steps=16,

learning_rate=2e-4,

fp16=True,

logging_steps=10,

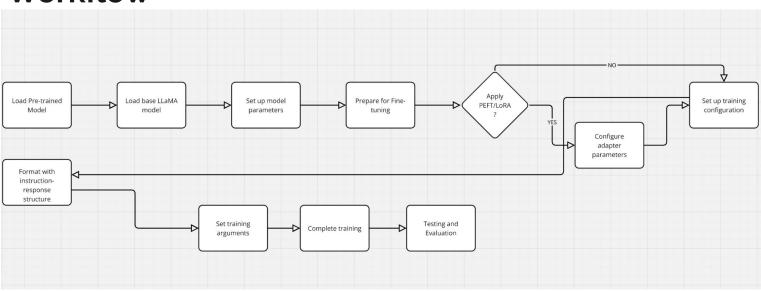
save_steps=200,
```





LoRA for parameter-efficient fine-tuning

Workflow



Baseline Performance

- For a simple prompt, both the models provided the same output query
- Meta llama provided a generic response despite the context provided
- Meta Llama Response:

```
Base model response:
### Instruction: Write a SQL query to find all endangered species with a population of less than 1000 individuals.

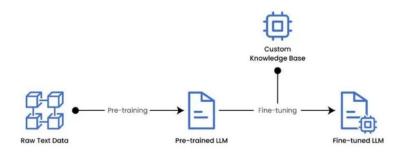
### Context: CREATE TABLE endangered_species (
    species_id INT PRIMARY KEY,
    scientific_name VARCHAR(100),
    common_name VARCHAR(100),
    common_name VARCHAR(100),
    conservation_status VARCHAR(50),
    habitat_type VARCHAR(100),
    population_count INT,
    year_assessed INT,
    primary_threats VARCHAR(200)
);

### Solution:

SELECT * FROM endangered_species WHERE population_count < 1000;
```

Fine-Tuning Process

- Training parameters:
 - o Epochs: 3
 - Batch Size: 2
 - Learning Rate: 2e-4
 - Max Sequence Length: 512
- Trainer: SFTTrainer (Transformers RL)
- LoRA applied for lightweight tuning
- Dataset tokenized and batched
- Gradual learning rate adjustments
- Model checkpoints saved every 200 steps



Baseline Performance

- Deepseek showed some reasoning abilities in addition to generating the prompt
- It gave a generic response but provided all the fields present in the context
- Deepseek Response:

Post-Fine-Tuning Performance

• After fine tuning, both the models provided optimized query with respect to the context provided

Meta Llama output: provided similar output to test results

```
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
model and tokenizer loaded
### Instruction: Write a SQL query to find the top 3 habitat types with the highest average population count of endangered species, but only include habitat
### Context: CREATE TABLE endangered species (
 species_id INT PRIMARY KEY,
 scientific_name VARCHAR(100),
 common_name VARCHAR(100),
 conservation status VARCHAR(50),
 habitat type VARCHAR(100).
 population_count INT,
 year_assessed INT,
 primary threats VARCHAR(200)
CREATE TABLE conservation_efforts (
 effort_id INT PRIMARY KEY,
 species id INT,
 program_name VARCHAR(150),
 start_date DATE,
 funding_amount DECIMAL(12,2),
 success_rate DECIMAL(5,2),
 FOREIGN KEY (species id) REFERENCES endangered species(species id)
### Response: SELECT h.habitat_type, SUM(e.population_count) AS total_population, COUNT(DISTINCT e.species_id) AS distinct_species FROM endangered_species e
### Explanation: This SQL query performs an inner join between the endangered species, conservation efforts, and habitat types tables on the species id and
```

Post-Fine-Tuning Performance

Deepseek Output: provided optimized query

```
### Instruction: Write a SQL query to find the top 3 habitat types with the highest average population count of endangered species, but only include habitat
### Context: CREATE TABLE endangered_species (
 species id INT PRIMARY KEY,
 scientific_name VARCHAR(100),
 common name VARCHAR(100),
 conservation_status VARCHAR(50),
 habitat_type VARCHAR(100),
 population_count INT,
 year_assessed INT,
primary_threats VARCHAR(200)
CREATE TABLE conservation efforts (
effort_id INT PRIMARY KEY,
 species_id INT,
 program_name VARCHAR(150),
 start date DATE,
 funding_amount DECIMAL(12,2),
success_rate DECIMAL(5,2),
 FOREIGN KEY (species_id) REFERENCES endangered_species(species_id)
### Response: SELECT h habitat_type FROM habitat_types h INNER JOIN endangered_species es ON h.habitat_id = es.habitat_id INNER JOIN conservation efforts ce
### Explanation: This query joins the habitat_types, endangered_species, and conservation_efforts tables on the habitat_id and species_id columns. It then f
```

Results (DeepSeek vs. LLaMA Post-Fine-Tuning)

Common Query: Which habitats have seen a decrease in preservation efforts from 2017 to 2018?

LLama Response: SELECT Habitat, Year, Efforts FROM PreservationTrends WHERE Efforts < (SELECT SUM(Efforts) FROM PreservationTrends WHERE Year = 2018);

DeepSeek Response: SELECT Habitat FROM PreservationTrends WHERE Year = 2017 AND Efforts < (SELECT SUM(Efforts) FROM PreservationTrends WHERE Year = 2018 AND Habitat = Habitat);

Observations

Deepseek provides an optimized and efficient queries involving multiple tables

Instruction: Write a SQL query to find all employees who have been with the company for more than 5 years and have a salary above the department average.

```
### Context: CREATE TABLE employees (
    employee id INT PRIMARY KEY.
    first_name VARCHAR(50),
   last_name VARCHAR(50),
   hire date DATE,
    department_id INT,
   position VARCHAR(100),
    salary DECIMAL(10.2).
   manager id INT.
   performance_rating DECIMAL(3,2)
CREATE TABLE departments (
   department_id INT PRIMARY KEY,
    department name VARCHAR(100),
    location VARCHAR(100).
   budget DECIMAL(15,2),
    head count INT
); CREATE TABLE employee_departments (
   employee_id INT,
    FOREIGN KEY (employee_id) REFERENCES employees (employee_id),
    FOREIGN KEY (department_id) REFERENCES departments (department_id)
```

SELECT e.first_name, e.last_name, e.hire_date, e.department_id, e.position, e.salary, AVG(e.salary) as department_average FROM employees e INNER JOIN departments d ON e.department_id = d.department_id WHERE e.salary > AVG(e.salary) AND e.hire_date >= DATE_SUB(e.hire_date, INTERVAL 5 YEAR) GROUP BY e.first_name, e.last_name, e.hire_date, e.position;

Explanation: This query joins the employees and departments tables on the department_id field. It then calculates the average salary for each department using the AVG() function. Finally, it filters the results to only include employees who have been with the company for more than 5 years and have a salary above the department average.

Summary of findings:

- Fine-tuning significantly improves SQL accuracy
- LoRA is effective for parameter-efficient tuning
- Llama was trained with a training loss of 0.4 while deepseek was trained with training loss of 0.5, however there was a significant number in the trained parameters of base model
- When it comes to testing on a data with mathematical and logical complexity, deepseek outperforms Meta-Llama in terms of rightness and optimization

THANK YOU