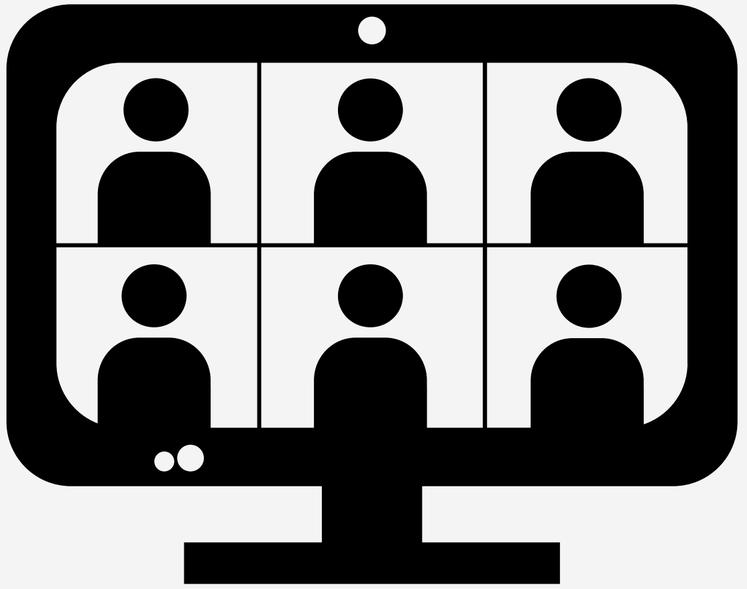
HPE Project Presentation

Text-2-SQL Code Generation And Execution

Mentor: Mr. Karthikeyan Soundararajan Sir

Agenda Of Meeting



Objective

Use machine Learning models to generate SQL code from Natural Language

Goal # 1

Create a model that accurately translates natural language queries into SQL statements.

Goal # 2

Integrating Large Language Models (LLMs) and Generative AI into a Text-to-SQL machine learning project

Goal # 3

- Integrating schema retrieval into the Text-2-SQL pipeline.
- Optimization and Deployement.

Fine Tuning

1. Pretrained Model

- Start with a base model like GPT, LLaMA, Gemma, etc.
- Pretrained on massive corpora with general language knowledge.

2. Task-Specific Objective

- Fine-tune for a specific downstream task:
- e.g., Text-to-SQL, Summarization, Classification, etc.

3. Data Formatting

- Format prompts and responses in instruction-response pairs.
- For Text-to-SQL:
- Input = Natural Language Query
- Output = SQL Query
- Use the model's tokenizer.
- Truncate, pad, and encode inputs/outputs into token IDs.

4. Quantization & Optimization

- Apply 4-bit quantization (NF4) to save GPU memory.
- Use BitsAndBytesConfig for efficient loading.

5. Parameter-Efficient Fine-Tuning

- Apply LoRA or QLoRA: train only small adapter layers.
- Reduces trainable parameters from billions to millions.

6. Training

- Use small batch sizes (Colab-friendly: 1–2).
- Apply gradient accumulation and mixed precision (fp16).
- Use the fine-tuned model to generate predictions.
- Use greedy or beam decoding to convert prompts to SQL.

Quantization Aware Techniques

- Post-Training Quantization (PTQ)
- Quantization Aware Training (QAT)
- BitsAndBytes 4-bit Quantization (NF4)
- Double Quantization
- Quantized LoRA (QLoRA)
- PEFT (Parameter-Efficient Fine-Tuning)

LoRA / QLoRA

1. What is LoRA? (Low-Rank Adaptation)

- LoRA introduces low-rank matrices into the model's attention layers.
- Instead of updating the full model, it only updates a small set of added trainable parameters.

🔹 How it works:

- Replaces weight update:
- $W \approx W + \Delta W$
- Where $\Delta W = A \times B$
- (A and B are low-rank matrices, e.g., rank r=8)
- Original weights stay frozen (not updated).

2. What is QLoRA? (Quantized LoRA)

- QLoRA = LoRA + 4-bit Quantization
- Designed to fine-tune very large LLMs (e.g., 4B, 7B, 13B) on limited GPUs (e.g., T4, A100).

🔹 Key Features:

- Freezes base model (loaded in 4-bit using bitsandbytes).
- Applies LoRA adapters to small parts of the model.
- Greatly reduces VRAM usage — <8 GB for models like LLaMA 7B.

🔹 Uses:

- Efficient fine-tuning in Google Colab, Kaggle, etc.
- Works well with Hugging Face’s transformers + peft.

Data Processing And Tokenization

- **Goal**: Prepare input-output pairs in a prompt-response format suitable for language model training.
- **Instruction Format:** Structure each training sample as:
- **Dataset Formatting:** Use a custom function (e.g. format_example) to combine question + DB ID into a single string.
- **Tokenizer Role:** Converts text to input_ids using the model's vocabulary (e.g., AutoTokenizer from Hugging Face).
- **Truncation & Padding:**
  - truncation=True: Clips overly long inputs
  - padding="max_length": Ensures all inputs are the same length
- **Tokenizing Both Input & Output:** Tokenize instruction part as input Tokenize SQL query as the label (target output)
- **Label Alignment:** Ensure model_inputs["labels"] = labels["input_ids"] to align training targets.
- **Efficient Mapping:** Use Hugging Face .map(batched=True) to apply the tokenization across the dataset efficiently.
- **Memory Optimization**: Use reduced max_length (e.g., 384 for input, 128 for output) for compatibility with Colab/Kaggle GPUs.
- **Remove Original Columns:** Pass remove_columns=train_data.column_names to reduce dataset size after tokenization.

Inferencing

1.Load Fine-Tuned Model

Load your LoRA or QLoRA fine-tuned model from disk or Hugging Face Hub.

2.Prepare Inference Prompt

Match the training format (prompt-style).

3. Tokenize Input

Tokenize with truncation and tensor return.

4. Generate Prediction

Use .generate() with parameters for length and sampling

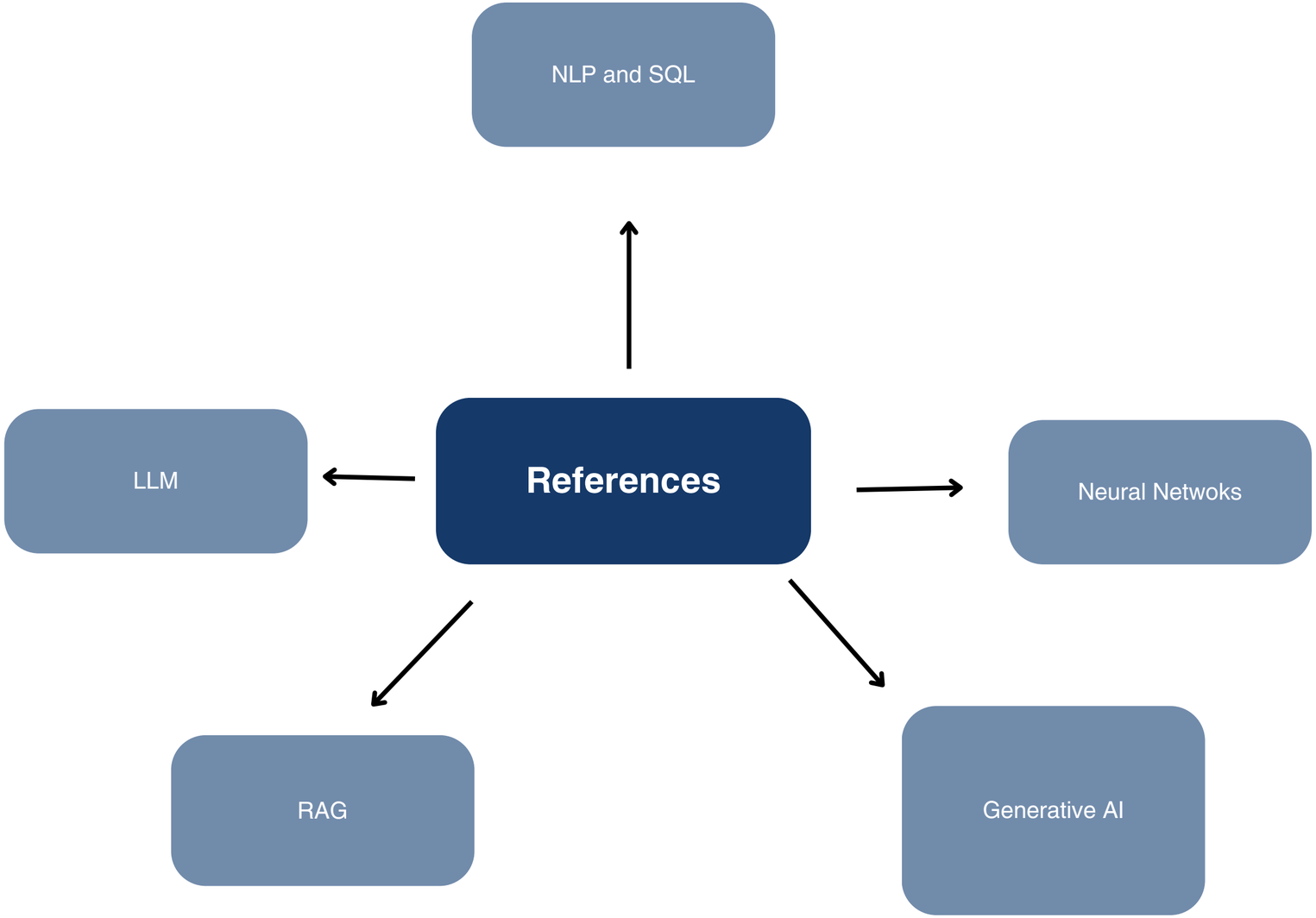
5.Decode Output

Convert generated token IDs back to text.

5.Post-Processing

Extract only the SQL part if needed

REFERENCES


NLP and SQL
LLM
References
Neural Netwoks
RAG
Generative AI

THANK YOU

Team Members:

- Raunak Jain
- Anshul Suwalka
- Vidhi Sharma
- Vishesh Jain
- Priyanshi Sharma