**Fine-Tuning Gemma-2B for Text-to-SQL Generation**

**1. Project Framing and Objective**

**Goal:** Train a Large Language Model (LLM) to convert natural language questions about database schemas into valid SQL queries, using parameter-efficient fine-tuning within resource constraints (Using Google Colab).

**Model:** Gemma-2B—an open-source, performant LLM well-suited for custom tasks with modest hardware.

**Task:** Text-to-SQL is challenging because it requires schema interpretation, precise SQL syntax, and robust generalization.

**2. Environment & Dependencies**

**Key Libraries**

* **Transformers (Hugging Face):** For model and tokenizer loading, generation.
* **trl (Transformer Reinforcement Learning):** SFTTrainer for streamlined supervised fine-tuning.
* **PEFT (Parameter Efficient Fine-Tuning):** LoRA/QLoRA for low-memory adapter training.
* **BitsAndBytes:** 4-bit quantization of the model, enabling training/evaluation on ordinary GPUs.
* **Comprehensive Evaluation Suite:** Includes evaluate, rouge\_scorer, bert\_score, detoxify, textstat, and sentence\_transformers.

**Best Practice:** All code and dependencies are managed within one notebook for smooth reproducibility and grading.

**3. Dataset Preparation**

* **Source:** Synthetic Text-to-SQL dataset (Hugging Face)
* **Structure:**
  + Input: Natural language question + database schema context
  + Output: Target SQL query

**Processing Steps:**

* **Preprocessing:**
  + Extract question from schema context.
  + Format prompt using a template (Gemma chat-style: user\n...model\n).
  + Example prompt structure:

user  
You are a powerful text-to-SQL model...  
### CONTEXT  
[database schema]  
### QUESTION  
[user's question]  
model

* + **Tokenization:** Left padding, max\_length=512, pad tokens' labels set to -100 (ignored for loss).
* **Data Split:**
  + 90% for training, 10% for validation. Ensures reliable evaluation of generalization during training.

**4. Model Setup & Fine-Tuning**

**Key Configurations**

* **Model Loading:**  
  Load Gemma-2B (with tokenizer), using QLoRA 4-bit quantization for low memory footprint.
* **LoRA Adapter Config:**
  + lora\_alpha=16 (scaling)
  + r=64 (adapter rank)
  + lora\_dropout=0.1 (regularization)
  + Bias terms not adapted for efficiency.
* **Training Arguments:**
  + Batch size per device = 4 (memory constraint)
  + Optimizer: paged\_adamw\_32bit (efficient for quantized weights)
  + Learning rate ≈ 2e-4 (effective for adapter-based fine-tuning)
  + Max epochs: 1 (sufficient for synthetic dataset, limits overfitting)
  + Gradient clipping and scheduler ensure stable, reproducible training.

**Saving:** Only LoRA adapter weights (very small file), plus the tokenizer, for efficient checkpointing.

**5. Memory Management and Monitoring**

* Explicitly track device memory before and after training.
* Print stats like reserved memory, percentage used, and LoRA footprint.
* Ensures training fits within hardware limits and offers best practices for future projects.

**6. Model Evaluation**

**Generation**

* **Deterministic generation:** No sampling, max tokens set to 128, no-repeat n-gram penalty to reduce repetition.

**Metrics (per project requirements)**

|  |  |  |
| --- | --- | --- |
| Metric | Description | Insights from Your Output |
| BLEU | n-gram match with reference SQL | 0.23: moderate structural accuracy |
| ROUGE-1/2/L | Keyword/phrase/sequence similarity | 0.54/0.32/0.49: good with keywords/phrases |
| METEOR | Synonym/word-order match | 0.44: robust to minor SQL variations |
| GLEU | Token-level general n-gram measure | 0.23: moderate |
| Repetition Rate | Ratio of repeated to unique tokens | 0.11: low, indicates varied outputs |
| Flesch Reading Ease | Text ease (mainly for NLP, SQL likely low) | 26.24: expected low |
| CoSIM | Cosine similarity of sentence embeddings | 0.75: strong semantic match |
| BERTScore (F1) | Token-level contextual match | 0.38: reasonable for SQL outputs |
| Toxicity | Toxic content detection (should be near zero) | 0.00: all outputs non-toxic |
| Novelty | 1 - avg token overlap with reference | 0.42: moderate balance between copying and creativity |
| Diversity (distinct-2) | Bigram diversity across outputs | 0.99: very high; strong variety |

**Result Table Presentation:** All metrics displayed clearly for comparison between full fine-tuning and parameter-efficient methods.

**7. Interpreting Results & Project Strengths**

* **Accuracy vs. Diversity:** Metrics like BLEU/ROUGE/METEOR assess SQL correctness, while Diversity/Novelty tracks output variability—a crucial trade-off for generation tasks.
* **Resource Optimization:** QLoRA with LoRA adapters enables viable training even on <32GB GPUs.
* **Generalization:** Validation split and semantic metrics (CoSIM, BERTScore) ensure that the model learns SQL structure beyond memorization.
* **Reusability:** Modular code means you can swap:
  + Another dataset (any text generation task)
  + A different model (Llama, Mistral, GPT-Neo, etc.)
  + Alternative fine-tuning method (Prefix-Tuning, Full Finetune)

**8. Best Practices for Future Projects**

1. **Dataset Choice & Prompt Design**
   * Tailor prompt templates to model expectations (user/model dichotomy for chat models)
   * Schema/context clarity is vital for text-to-SQL tasks
2. **Model Selection**
   * Evaluate hardware/memory before choosing model size and quantization scheme
   * Prefer parameter-efficient methods for large LLMs on modest hardware
3. **Training Strategy**
   * Keep batch sizes and sequence lengths within memory capacity
   * Use reproducible splits; document hyperparameters meticulously
   * Monitor resource usage, especially for experiments in Colab/Kaggle
4. **Comprehensive Evaluation**
   * Use the full suite of metrics covering syntactic, semantic, and non-text aspects (like toxicity, diversity)
   * Present results in clear tables for easy comparison
5. **Modularity & Reproducibility**
   * Save only necessary model artifacts (adapters/tokenizer)
   * Keep all logic in one notebook/script for hassle-free reruns
6. **Explainability**
   * Annotate code for each section (setup, processing, training, evaluation)
   * Be able to explain why each library/method/component is chosen
   * Practice talking through each step (as done in this document)

**9. Final Reference Table: Section-by-Section Explanation**

|  |  |  |  |
| --- | --- | --- | --- |
| Section | What It Does | Why It's Needed | Generalization/Future Use |
| Setup & Dependencies | Loads libraries, ensures reproducibility | Required for any LLM project | Swap models/libraries as needed |
| Dataset Prep | Formats prompts, tokenizes, splits data | Feeds data correctly to model | Adapt for new datasets/tasks |
| Model Config | Quantizes & configures model and adapters | Lowers memory, enables scalable fine-tuning | Switch to other model types |
| Training Loop | Runs SFT with LoRA/QLoRA adapters | Efficient supervised LLM fine-tuning | Plug in different optimizers/strategies |
| Memory Monitoring | Tracks resource usage | Prevents OOM errors, optimizes efficiency | Apply for any hardware-constrained setting |
| Evaluation Processing | Generates/predicts, runs 13+ metrics | Deep model assessment, shows trade-offs | Use/extend metrics for any output type |
| Results & Analysis | Summarizes, interprets strengths/weaknesses | Informs decisions, shows model fit | Extend for new tasks/problems |

**10. How to Present and Apply to New Projects**

* **Start with Objective**: What is your generation or translation target?
* **Explain Model/Method Choice**: Why this model, why this fine-tuning style?
* **Walk Through Data and Prompt Design**: Show how input is converted and why that matters.
* **Show Training Strategy**: Discuss adapter config, quantization, batch size and optimizer.
* **Highlight Evaluation**: Use metric tables, show trade-offs (accuracy, diversity, toxicity, efficiency).
* **Finish with Takeaways**: What worked, what could be improved, how to generalize for future work.

**This guide is your go-to resource for presenting LLM finetuning projects and building new ones—enabling clear explanations, robust experimentation, and modular project replication.**