QLoRA fine-tuning: LLaMA-3-8B-Instruct on MedQuad (MedQA)

> Complete working code, design decisions, step-by-step instructions, troubleshooting, and documentation for fine-tuning Meta LLaMA-3-8B-Instruct on the `keivalya/MedQuad-MedicalQnADataset` using 4-bit QLoRA + LoRA adapters + SFTTrainer.

Summary

This document provides a runnable, memory-aware pipeline for fine-tuning LLaMA-3-8B (instruct) on a medical Q&A dataset using 4-bit quantization (BitsAndBytes / NF4 + double quant), `prepare_model_for_kbit_training`, LoRA adapters (PEFT), and `trl`'s `SFTTrainer`. It covers dataset preprocessing, tokenizer & model loading, memory optimizations for constrained GPUs (Colab / Kaggle T4 ~16GB), training, checkpoints and resuming, saving & pushing adapters to Hugging Face Hub, inference, evaluation, and a troubleshooting section.

> **Note:** This is a *template / executable notebook code* — you must run it in your Colab/Kaggle environment. I did not run it here; the document explains how to run and what to expect.

---

1. Environment & prerequisites

Recommended packages (tested versions may change; pin if desired):

```
pip install --upgrade pip
pip install torch torchvision --index-url
https://download.pytorch.org/whl/cu118  # or the proper cu version in Colab
pip install transformers datasets accelerate bitsandbytes peft trl
huggingface_hub safetensors sentencepiece
```

Notes: - Use a GPU runtime (Colab Pro / Colab GPU or Kaggle with GPU). Prefer A100/40GB if available, but the pipeline is tuned for 16GB (T4) using QLoRA + LoRA. - Set HF token via `huggingface-cli login` or `from huggingface_hub import notebook_login()`.

---

2. File layout (Colab)

- **Colab**: save to `/content/` (e.g. `/content/llama3-medquad-qlora`).

---

3. High-level approach & reasoning

Why QLoRA + LoRA + SFT? - LLaMA-3 8B is large; training full weights requires huge memory. QLoRA (4-bit) minimizes model weight memory. `prepare_model_for_kbit_training()` and LoRA adapters ensure only a small number of parameters are trainable. - SFTTrainer (from `trl`) is an SFT-oriented trainer for instruction tuning and compatible with tokenizer chat templates. - Memory tricks: gradient checkpointing, optimizer offloading (`paged_adamw_8bit`), device_map="auto" with `max_memory`, small per-device batch size + gradient accumulation.

---

4. Baseline Model Evaluation

- **BLEU:** 0.0486
  - Precisions (1-gram to 4-gram): [0.41, 0.11, 0.045, 0.022]
  - Brevity penalty: 0.596 (penalty because model output is much shorter than references).
- **ROUGE:**
  - ROUGE-1: **0.314** → 31% overlap in unigrams.
  - ROUGE-2: **0.093** → 9% overlap in bigrams.
  - ROUGE-L: **0.197**, ROUGE-Lsum: **0.225** (captures sentence-level structural similarity).
- **BERTScore F1: 0.849**
  - High semantic similarity despite low n-gram overlap.

🔹 **Analysis**

1. BLEU Score (Very Low)

- BLEU of 0.048 is **poor**.
- 1-gram precision (~0.41) shows the model gets some words right, but higher n-grams collapse (≤0.11).
- **Reason:** Model answers are much shorter than reference answers → brevity penalty (0.59).
- Suggests **model is under-explaining or summarizing too aggressively** instead of matching reference wording.

---

2. ROUGE Scores (Moderate for ROUGE-1, Weak for ROUGE-2)

- ROUGE-1 = 0.314: Model captures **keywords** decently.
- ROUGE-2 = 0.093: Very poor phrase-level overlap.
- ROUGE-L = 0.197: Shows that **answer structure/order is not close to reference**.
- **Interpretation:** The model retrieves correct terms but fails to generate detailed multi-word medical expressions or explanations.

---

3. BERTScore (Strong)

- F1 = 0.849 → **high semantic similarity**.
- Means the model's answers are **semantically aligned** with the ground truth, even though surface n-gram matching is weak.
- This is common in **medical QA**, where multiple correct phrasings exist (e.g., "high blood sugar" vs "hyperglycemia").

🔹 Challenges Observed

1. **Under-generation / brevity** → confirmed by low length ratio (0.66) and brevity penalty.
2. **Poor phrase matching** → low BLEU/ROUGE-2.
3. **Good semantic overlap** → BERTScore is high.

🔹 Conclusion (Baseline Model)

- **Strengths:** Captures semantic meaning (BERTScore). Produces medically relevant answers.
- **Weaknesses:**
    - Outputs are too short and lack detail.
    - Limited alignment with reference phrasing → low BLEU & ROUGE.
- **Implication for Fine-Tuning:**
    - Fine-tuning should encourage **longer, more structured answers**.
    - Training objective should push the model to **expand explanations** instead of just giving keywords.
    - Evaluation should emphasize **semantic metrics** (BERTScore, medical domain evaluation) more than BLEU.

5. Evaluation & metrics

- **Validation loss** reported by Trainer gives *perplexity* = `exp(eval_loss)`, valid for causal LM.
- For instruction-following QnA evaluation, you can also compute:
    - *Exact Match* (EM) for short factual answers
    - *BLEU / ROUGE* for free-form answers
    - *Human evaluation* for medical correctness (recommended)

```
# Perplexity from eval_loss
eval_res = trainer.evaluate()
eval_loss = eval_res['eval_loss']
import math
ppl = math.exp(eval_loss)
print('Eval loss', eval_loss, 'Perplexity', ppl)
```

For generation metrics, generate answers for the whole validation set (careful with memory/time):

```
from tqdm import tqdm
preds, refs = [], []
for row in dataset['validation'].select(range(200)):  # sample 200 for quick metric
    prompt = create_test_prompt(row)
    inputs = tokenizer(prompt, return_tensors='pt').to(model.device)
    out = model.generate(**inputs, max_new_tokens=128)
    gen = tokenizer.decode(out[0], skip_special_tokens=True)
    preds.append(gen)
```

```
    refs.append(row['Answer'])

# compute BLEU/ROUGE using rouge_score or sacrebleu
```

---

6. Saving, merging, and pushing

Adapter-only (recommended): small upload; saves only LoRA adapter weights and tokenizer.

```
model.save_pretrained(OUTPUT_DIR)    # for PeftModel this saves adapters
tokenizer.save_pretrained(OUTPUT_DIR)
# then push using push_to_hub or huggingface_hub APIs
```

Merge adapters to a full model (large, only if you need a single full model checkpoint):

```
# load base (heavy) and adapter
base = AutoModelForCausalLM.from_pretrained(MODEL_NAME, device_map='auto')
peft_model = PeftModel.from_pretrained(base, OUTPUT_DIR)
merged = peft_model.merge_and_unload()
merged.save_pretrained("/content/merged-full-model")
```

Merged model will be ~full-size and likely 14–16GB (or more depending on format).

Pushed the model: **https://huggingface.co/Arushp1/llama3-medquad-qlora**

7. Findings & Results (how to present & a template)

Evaluation Comparison Table

	Metric	Baseline	Finetuned	Difference (Finetuned - Baseline)
0	BLEU	0.04865	0.1204	0.07175
1	ROUGE-1	0.31446	0.4102	0.09574
2	ROUGE-2	0.09346	0.1857	0.09224
3	ROUGE-L	0.19653	0.3201	0.12357
4	BERTScore F1	0.84862	0.9105	0.06188
