Matthew Poncini
J.B. Speed School of Engineering
CSE-590-52-4255: Special Topics - GenAI
20 July 2025

CSE 590 GenAI Homework Assignment 2

Introduction

The purpose of this assignment was to gain hands-on experience fine-tuning a large language model (LLM) using parameter-efficient techniques on a real-world dataset. The overall goal was to apply practical LLM fine-tuning techniques on a specific downstream task using modern tools and efficient training methods. This assignment reinforced core skills in LLM training, including prompt design, efficient model loading, and evaluation of generative performance.

Dataset and Model Selection

Dataset: gbharti/finance-alpaca (Hugging Face)
Samples Used: 1000 for training, 300 for testing
Model: meta-llama/Llama-2-7b-hf (quantized 4-bit for efficient fine-tuning)

Evaluation Metric

For this assignment, I used ROUGE as my evaluation metric.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a family of metrics used to evaluate the quality of machine-generated text by comparing it to human-written reference text. It is ideal for tasks like summarization and text generation; which is why I selected to use it as my metric. ROUGE specifically evaluates n-gram overlap between the generated output and the reference.

I compared the pretrained and trained models ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-Lsum scores. ROUGE-1 measures overlap of unigrams. ROUGE-2 measures overlap of bigrams. ROUGE-L measures the longest common subsequence. ROUGE-Lsum applies ROUGE-L to multi-sentence texts.

In general, higher ROUGE scores mean the generated responses more closely matched the reference outputs in phrasing and structure.

Methodology Pipeline

- Loading and preparing the dataset: from the gbharti/finance-alpaca dataset, 1000 examples were selected for training and a separate 300 for evaluation. The examples were then formatted to match a prompt structure similar to the Alpaca format.
- Tokenization: The tokenizer used, AutoTokenizer, was loaded with the LLaMA-2 model. This ensured compatibility between the input and tokenization logic. Tokenized sequences were padded to the same length, converted to PyTorch tensors, and transferred to the GPU.

- Pretrained Model: the pretrained LLaMA-2-7B model was evaluated using the 300-sized sample test set to establish a performance baseline. The evaluation used ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-Lsum to measure the overlap between predicted and reference text.
- Fine-Tuning LLaMA-2-7B:
  - Efficiency: used a combination of 4-bit quantization with Low-Rank Adaptation (LoRA) called QLoRA. This methodology allowed fine-tuning to take place with minimum resources. Instead of training all model parameters, LoRA inserts additional trainable layers into the existing model architecture. It does not replace or modify the original model weights. These adapter layers were added in parallel to specific attention components: q_proj and v_proj.
  - Trained LLaMA-2-7B on the 1000-sized sample test set.
  - The fine-tuned LLaMA-2 model with LoRA adapters was saved and reloaded using the AutoPeftModelForCausalLM class.
- Trained Model: prompts were tokenized and passed to the model, just as in the pretrained phase. The same 300 test prompts used during pretrained evaluation were used to measure post-training performance. ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-Lsum were computed.

Metric Results

Pretrained Model ROUGE Evaluation:	**Trained Model ROUGE Evaluation:**	**Reference ROUGE Evaluation:**
rouge1: 0.1205	rouge1: 0.2519	rouge1: 0.2127 → 0.4426
rouge2: 0.0203	rouge2: 0.0400	rouge2: 0.0488 → 0.1772
rougeL: 0.0802	rougeL: 0.1371	rougeL: 0.1575 → 0.3563
rougeLsum: 0.0818	rougeLsum: 0.1526	

These results clearly show that fine-tuning with QLoRA improved your model's performance across all ROUGE metrics. This shows that training LLaMA-2-7B to a domain of financial advice enabled the model to provide more relevant and accurate responses.

Reference Result for Similar Task

AI practitioner Cristiano Aguilar conducted a comparable fine-tuning task using the armanc/scientific_papers dataset, leveraging 20,000 training samples and 6,000 test examples. He fine-tuned the LLaMA-3.2-1B-Instruct model using LoRA-based parameter-efficient tuning, inserting adapter layers into the q_proj and v_proj attention projections. The entire training process was completed on a single NVIDIA A6000 GPU, demonstrating both efficiency and scalability.

https://clearintelligence.substack.com/p/fine-tuning-llama-llm-with-lora-a

Aguilar’s work focused on summarization of scientific content while my project had a structured Q&A. Both of our projects focused on projection adapters, but our training was more memory-efficient by utilizing quantization alongside LoRA. Aguilar’s training lasted about 28 hours, while my model trained in less than one hour. Aguilar did have higher ROUGE scores in both the base and fine-tuned models, but the training in my project was significantly more efficient and saw a similar percent increase in metrics.

10 Representative Samples from the Test Sets

1. **Prompt:** Is it ok to just report to 1 credit bureau instead of all 3

   Base Model Response: The law allows you to report to one or all 3. There is no requirement to report to all 3.

   Fine-Tuned Model Response: I think it is a good idea to get your credit report from each of the three credit bureaus, to make sure you are not being reported inaccurately. I do not think you have to report to all three, though. I think you can just report to Experian, and then I believe the other two will get your information from Experian. I am not sure if that is true, but I think it is. I do know that you have to report to at least one of the three. I think you have to report to Trans Union, but I am not sure about that. I think it is a

   ChatGPT-4 Response: Yes, it's okay to report to just one credit bureau—there’s no legal requirement to report to all three (Experian, Equifax, TransUnion). However, doing so can lead to inconsistencies in credit reports, which might affect how lenders view a consumer’s creditworthiness. Reporting to all three ensures broader visibility and more accurate credit scoring across the board. Many lenders and credit-building services aim to report to all major bureaus for this reason. But due to cost or partnership limitations, some choose to report to only one or two.

2. **Prompt:** How to diversify IRA portfolio given fund minimum investments and IRA contribution limits?

   Base Model Response: IRA portfolio is diversified by selecting a combination of funds that has a minimum investment of $500 and a maximum investment of $5,000.

   Fine-Tuned Model Response: A diversified portfolio will be hard to achieve using IRA's, but it is not impossible. IRA's have a few limitations. They have annual contribution limits and there are some tax issues involved with some investments.

ChatGPT-4 Response: To diversify an IRA with fund minimums and contribution limits, focus on low-minimum, broad-based ETFs or mutual funds (e.g., target-date or balanced funds). Use a broker that offers no-minimum or fractional share investing. Prioritize asset allocation (stocks, bonds, international) and rebalance over time. Start with a core holding and diversify further as contributions grow.

3.

Prompt: Choosing which ESPP stocks to sell?

Base Model Response: -

Fine-Tuned Model Response: I've been in a similar situation. I'm not sure how the taxes would work, but it might make sense to sell some of the ESPP shares and reinvest the proceeds in your retirement account. This would reduce your tax burden, and it would help you reach your retirement goals more quickly. As a bonus, you'd have a diversified portfolio, and you'd have a larger base of shares to sell at retirement.

ChatGPT-4 Response: When choosing which ESPP stocks to sell, prioritize shares with the highest gains and shortest holding periods to reduce risk. Consider selling shares that have met qualifying disposition rules (held >2 years from offer date and >1 year from purchase) for favorable tax treatment. Evaluate company outlook, portfolio concentration, and financial goals. Avoid overexposure to your employer's stock.

4.

Prompt: How to evaluate an annuity

Base Model Response:
$15,000 $15,000 = 15000/100 $15,000

Fine-Tuned Model Response: Here's a good way to think about annuities: Suppose you buy an annuity for $100,000, and the company agrees to pay you $500 a month for the rest of your life. Now, suppose you can invest that $100,000 in a portfolio that earns 10% a year. Which is the better deal? Well, in the first year, you'll get $5,000 from the annuity, and you'll earn $5,000 in interest on your $10

ChatGPT-4 Response: To evaluate an annuity, assess the type (fixed, variable, indexed), fees, guaranteed returns, payout options, and insurer strength. Compare projected income vs. other investments. Check surrender periods and penalties. Understand tax implications and inflation protection. Match features to your retirement goals and risk tolerance.

5. **Prompt:** Good book-keeping software?

Base Model Response: I would say that there is no such thing as good book-keeping software. I'm sure that if you're a small business owner, you have a pretty good idea of how you like to keep track of your finances. So, if you have a system that you're comfortable

Fine-Tuned Model Response:
If you're going to be paying your taxes in Canada, you should use QuickBooks. It's the standard for the industry and very simple to use. It's also the one that is most compatible with the Canada Revenue Agency. You can use it to generate reports that will be accepted by the CRA. If you're looking for a free solution, then the best one I've found is called GnuCash. It's pretty simple to use, and it will do most of what you want. It's not as compatible with the CRA, but it's still

ChatGPT-4 Response: Good bookkeeping software options include QuickBooks (best all-around), Xero (great for collaboration), FreshBooks (ideal for freelancers), Wave (free and user-friendly), and Zoho Books (affordable with strong features). Choose based on your business size, budget, and accounting needs.

6. **Prompt:** What questions should I ask a mortgage broker when refinancing a condo in Wisconsin?

Base Model Response:
* What are the current interest rates?
* What are the current closing costs?
* What are the current fees?
* What are the current restrictions?
* What are the current property taxes?
* What are the current condo fees?
* What is the current con

Fine-Tuned Model Response: I'm not a broker, but I would suggest you ask: 1. What are your fees? 2. How much will my monthly payment be? 3. What is the APR? 4. What is the term? 5. What are the penalties for paying it off early? 6. How much will I save? 7. Do you know the ins and outs of the condo association? 8. What is your experience with condo financing? 9. What is your experience with the condo association? 10. What is the

ChatGPT-4 Response: Ask your mortgage broker:

1. What interest rate and APR can you offer?
2. What are all fees and closing costs?
3. Will PMI apply, and can it be removed?
4. Is my condo eligible for full or limited review?
5. How does the HOA affect loan approval?
6. What's the break-even point?
7. How long is the rate lock?
8. What documents are required?
9. How long will closing take?
10. Do you work with multiple lenders?

7. **Prompt**: What is the most effective saving money method?

Base Model Response:
1. Spend less than you earn.
2. Save more than you earn.
3. Invest more than you earn.
4. Invest less than you earn.

Fine-Tuned Model Response:
Response: A few things you might try:
1. Save your change

If you don't spend your change, you'll have a nice pile of money.

2. Use a debit card

If you use a debit card instead of a credit card, you'll spend less.

3. Use a budget

If you don't know where your money is going, you won't have any left.

4. Don't buy anything you don't need

If

ChatGPT-4 Response: The most effective saving method is paying yourself first—automatically transfer a set portion of income (e.g., 20%) to savings before spending. Combine this with the 50/30/20 budget rule (50% needs, 30% wants, 20% savings), track expenses, and use high-yield savings accounts. Automate savings and set clear goals for best results.

8. **Prompt:** What are the financial advantages of living in Switzerland?

Base Model Response: Living in Switzerland is very expensive, but there are many advantages.

The cost of living in Switzerland is very high.

There are many advantages to living in Switzerland, such as the high standard of living and the high quality of education.

Fine-Tuned Model Response: You can make a lot of money in Switzerland. You can also lose a lot of money in Switzerland.

As a general rule of thumb, Switzerland is not a great place to live if you are planning on retiring on a limited income. It is a great place to live if you are planning on making a lot of money.

Comment: I've been in Switzerland for 10 years now. It's true, you can make a lot of money here. But it's

ChatGPT-4 Response: Financial advantages of living in Switzerland include high salaries, low unemployment, strong currency (Swiss franc), and a stable economy. The country offers a favorable tax structure in some cantons, excellent healthcare, and a strong pension system. Swiss residents benefit from reliable public services and infrastructure, plus access to global banking and investment options.

9. **Prompt:** What is a negotiable security and how are they related to derivatives?

Base Model Response: A negotiable security is a type of financial instrument that can be transferred from one party to another. Negotiable securities include stocks, bonds, and other financial instruments that are traded on the open market.

Derivatives, on the other hand, are financial instruments that derive

Fine-Tuned Model Response: The term negotiable security is a technical term that is used in the context of securities. A negotiable security is a security that can be sold, bought, and transferred without restriction. There are two types of negotiable securities: (1) registered securities and (2) bearer securities. In the case of a registered security, the owner of the security is recorded on the books of the issuer of the security. In the case of a bearer security, the owner of the security is not recorded on the books of the issuer of the security. Bearer securities are usually

ChatGPT-4 Response: A negotiable security is a transferable financial instrument, like stocks or bonds, that can be bought or sold in the market. It gives legal ownership and can be easily traded. Derivatives are financial contracts whose value is based on underlying assets, often including negotiable securities. For example, stock options or futures are derivatives tied to the value of negotiable securities.

10. Prompt: How can I stop a merchant from charging a credit card processing fee?

Base Model Response: The merchant has the right to charge the fee they want to charge, but you can also try to negotiate a lower fee with them.

Disclaimer:

The information provided in this answer is not legal advice, but rather an answer to a general question. Please consult a lawyer

Fine-Tuned Model Response: Just pay the merchant directly, not with a credit card. You can pay the merchant with a check, cash, or by setting up an account with the merchant.

Comment: I think that's a very good answer. I don't know how to edit it to make it a community wiki answer though.

Answer: You can ask the merchant to bill you, instead of billing yourself.

For example, if you have an account with a merchant, you can ask them to charge your credit card for the amount of the bill, instead of using your credit card to pay the bill.

ChatGPT-4 Response: You generally can't force a merchant to stop charging credit card processing fees, but you can try these steps:

1. Ask politely if they'll waive the fee—some may do so for loyal customers.
2. Pay with cash or debit to avoid the fee.
3. Report excessive fees to your credit card issuer or network (e.g., Visa), especially if they violate rules.
4. Check local laws—some states restrict or ban credit card surcharges.
5. Shop elsewhere if the fee is unreasonable.

Merchants are allowed to charge fees in most states but must disclose them clearly.

Analysis

The side-by-side comparison of outputs from the base and fine-tuned models reveals several important qualitative trends related to relevance, coherence, hallucination, and response completeness. The base model gave the most undesired responses, most noticeably in prompts 3 and 4. Responses like this in the base model were more prevalent since it lacked domain-specific grounding in finance-related topics. When the base model did provide a reasonable response, it seemed like it was very direct. The fine-tuned model tended to have more humanistic responses. The tone of each LLM's response got progressively more humanistic as they went down, with ChatGPT giving the most humanistic responses.

The fine-tuned model did also show signs of hallucination and had its response cut off. This truncation occurred because the model's generate() function was set with a max_new_tokens limit of 64 or 128, depending on the evaluation phase. If the response exceeded that token length, it was forcibly stopped. This was a trade-off to keep evaluation efficient and avoid overlong outputs in limited-resource environments. However, ChatGPT's max_new_tokens was set to 128 and it did not have a cut-off response. The fine-tuned model attempted longer, more conversational replies, so cutoffs were more prominent post-training. Prompts 8 and 10 highlighted behavioral overfitting in the fine-tuned model. It was impersonating a commenter. This is a sign that the training over-fitted and evidence that the training data contained some more humanistic dialogue.

All in all, the fine-tuning model did improve performance. The increased ROUGE scores are reflected in the ten samples shown. The trade-off between verbosity and factual control was a challenge in tuning. In this case, the gains in user-aligned communication outweigh the drawbacks. In future projects that use very little training data, I will keep in mind that the quality of the training data is very important. This project demonstrated the practical benefits of fine-tuning a large language model using parameter-efficient techniques such as QLoRA. By adapting the LLaMA-2-7B model to a domain-specific financial Q&A dataset, we observed measurable improvements in both quantitative performance and qualitative output.

Works Cited

Bharti, G. (2023). Finance Alpaca Dataset. Hugging Face.
https://huggingface.co/datasets/gbharti/finance-alpaca

Aguilar, C. (2024). Fine-Tuning LLaMA LLM with LoRA: A Simple Walkthrough. Clear Intelligence Substack.
https://clearintelligence.substack.com/p/fine-tuning-llama-llm-with-lora-a

Dettmers, T. et al. (2023). QLoRA: Efficient Fine-Tuning of Quantized LLMs. arXiv preprint.
https://arxiv.org/abs/2305.14314

Hugging Face Documentation.

- PEFT (https://huggingface.co/docs/peft)
- Transformers Library (https://huggingface.co/docs/transformers)
- TRL (https://huggingface.co/docs/trl)

OpenAI. (2025). ChatGPT (July 20 version) [Large language model].
https://chat.openai.com

Appendix

Link to Colab Notebook:
https://colab.research.google.com/drive/1ABQ7aq_XlfGzVX5u1ZbKU1UDkVS67s-7?usp=sharing

Code:

```
### Step 0: Environment Setup and Library Installation

helps prevent memory fragmentation issues when loading large models or
training with quantized weights.
"""

import os
os.environ["PYTORCH_CUDA_ALLOC_CONF"] = "expandable_segments:True"

"""Install Libraries"""

!pip install peft

!pip install trl

!pip install rouge_score

!pip install evaluate

!pip install bitsandbytes

!pip install -U bitsandbytes

!pip install -U transformers

"""Checks if GPU can be ran"""

import torch

torch.cuda.is_available()

"""Log-in to HuggingFace"""

HF_TOKEN = "insert token here"
!huggingface-cli login --token $HF_TOKEN
```

```
"""### Step 1: Choose a downstream task and select a suitable dataset for
fine-tuning. From the dataset, sample 1,000 examples for training and a
separate, non-overlapping 300 examples for testing.

Chosen dataset for your fine-tuning task: finance-alpaca

https://huggingface.co/datasets/gbharti/finance-alpaca
"""

dataset = "gbharti/finance-alpaca"

from datasets import load_dataset

dataset = load_dataset("gbharti/finance-alpaca")

"""Make dataset viewable"""

import pandas as pd

df = dataset["train"].to_pandas().iloc[:1300]

df

"""Select 300 test sample from dataset"""

test_sample = df.iloc[1000:1300]

"""Select 1,000 training samples from dataset"""

train_sample = df.iloc[0:1000]

"""### Step 2: Select a pre-trained language model to work with.

Chosen pre-trained language model: meta-llama/Llama-2-7b-hf
"""

model_name = "meta-llama/Llama-2-7b-hf"

"""sets up a configuration for loading a quantized version of a large
language model using the BitsAndBytes library from Hugging Face"""

from transformers import BitsAndBytesConfig
```

```
 bnb_config = BitsAndBytesConfig(
  load_in_4bit=True,
  bnb_4bit_use_double_quant=True,
  bnb_4bit_quant_type="nf4",
  bnb_4bit_compute_dtype=torch.bfloat16
)

""" loads and configures the tokenizer associated with meta-llama/Llama-2-
7b-hf"""

from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained(model_name)
tokenizer.padding_side = "left"
tokenizer.pad_token = tokenizer.eos_token

""" loads a quantized large language model onto a GPU using Hugging Face
Transformers and BitsAndBytes, ready for fine-tuning or inference, while
minimizing memory usage."""

from transformers import AutoModelForCausalLM
from transformers import BitsAndBytesConfig

model = AutoModelForCausalLM.from_pretrained(
  model_name,
  quantization_config=bnb_config,
  device_map="auto",
  trust_remote_code=True,
  use_cache=True
)

"""Low-Rank Adaptation configuration, fine-tunes only a small number of
parameters while freezing the rest of the large model. This significantly
reduces memory and compute requirements."""

from peft import LoraConfig

lora_config = LoraConfig(
  r=16,
  lora_alpha=16,
  target_modules=["q_proj", "v_proj"],
  lora_dropout=0.05,
  bias="none",
  task_type="CAUSAL_LM"
)
```

```
 """### Step 3: Evaluate the pre-trained model on the selected dataset to
establish a baseline using an appropriate metric.

Evaluate Model Function
"""

import pandas as pd
from evaluate import load
from tqdm import tqdm
import torch

# makes the generated outputs clean for ROUGE evaluation and display.
def extract_response(text):
  if "### Response:" in text:
    return text.split("### Response:")[-1].strip()
  return text.strip()

# Evaluates a pretrained model on a test set using a generation + ROUGE
scoring loop.
def evaluate_model(model, tokenizer, prompts, references, batch_size=1,
max_new_tokens=64):
  pretrained_metric = load("rouge")
  predictions = []
  records = []

  for i in tqdm(range(0, len(prompts), batch_size)):
    batch_prompts = prompts[i:i+batch_size]
    batch_refs = references[i:i+batch_size]

    # Tokenizes input prompts with truncation and padding
    inputs = tokenizer(
      batch_prompts,
      return_tensors="pt",
      padding=True,
      truncation=True
    ).to(model.device)

    # produce text responses (ensures no gradients are computed — saving
memory)
    with torch.no_grad():
      outputs = model.generate(
       input_ids=inputs["input_ids"],
       attention_mask=inputs["attention_mask"],
```

```
 do_sample=True,
 temperature=0.7,
 top_p=0.9,
 max_new_tokens=64,
 pad_token_id=tokenizer.eos_token_id
 )

    # Converts model token outputs back into strings.
    batch_preds = tokenizer.batch_decode(outputs, skip_special_tokens=True)
    batch_preds = [extract_response(p) for p in batch_preds]


    # Metric Calculation and Result Collection
    for prompt_text, pred, ref in zip(batch_prompts, batch_preds,
batch_refs):
      pretrained_metric.add(prediction=pred, reference=ref)
      predictions.append(pred)
      records.append({
        "prompt": prompt_text,
        "reference": ref,
        "prediction": pred
      })

  pretrained_results_df = pd.DataFrame(records)
  return pretrained_metric.compute(), predictions, pretrained_results_df

"""Evaluate Execution:"""

prompts = [
  f"### Instruction:\n{row['instruction']}\n\n### Response:"
  for _, row in test_sample.iterrows()
]
references = list(test_sample["output"])

baseline_scores, baseline_outputs, pretrained_results_df = evaluate_model(
  model,
  tokenizer,
  prompts,
  references
)

rint("ROUGE Evaluation:")
for metric, score in baseline_scores.items():
  print(f"{metric}: {score:.4f}")

pretrained_results_df
```

```
 """### Step 4: Fine-tune the model on the chosen dataset.

Memory Cleanup and Re-initialization
"""

import gc
import torch

gc.collect()
torch.cuda.empty_cache()

"""Re-loads the tokenizer and re-downloads the dataset and selects the
first 1,000 samples from the training set for fine-tuning.

(Colab Notebook is restarted)
"""

import os
from transformers import AutoTokenizer
from datasets import load_dataset

model_name = "meta-llama/Llama-2-7b-hf"
tokenizer = AutoTokenizer.from_pretrained(model_name)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "left"

data = load_dataset("gbharti/finance-alpaca")
train_sample = data["train"].select(range(1000))

"""Reconfigure Quantization for Training"""

from transformers import BitsAndBytesConfig

bnb_config = BitsAndBytesConfig(
  load_in_4bit=True,
  bnb_4bit_use_double_quant=True,
  bnb_4bit_quant_type="nf4",
  bnb_4bit_compute_dtype=torch.bfloat16
)


from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained(model_name)
tokenizer.padding_side = "left"
tokenizer.pad_token = tokenizer.eos_token
```

```
from transformers import AutoModelForCausalLM

foundation_model = AutoModelForCausalLM.from_pretrained(
  model_name,
  quantization_config=bnb_config,
  device_map="auto",
  use_cache=False,
  trust_remote_code=True
)

from peft import LoraConfig


lora_config = LoraConfig(
  r=16,
  lora_alpha=16,
  target_modules=["q_proj", "v_proj"],
  lora_dropout=0.05,
  bias="none",
  task_type="CAUSAL_LM"
)

"""Formats each training example into a structured prompt-response pair"""

def build_prompt_completion(example):
  if example["input"]:
    prompt = f"### Instruction:\n{example['instruction']}\n\n###
Input:\n{example['input']}\n\n### Response:"
  else:
    prompt = f"### Instruction:\n{example['instruction']}\n\n### Response:"
  return {
    "prompt": prompt,
    "completion": example["output"]
  }

train_sample = train_sample.map(build_prompt_completion)

"""epares the dataset for training by tokenizing each example's prompt
which includes both instruction and expected response.

Truncates sequences longer than 512 tokens and pads shorter ones to that
length for consistent batch shapes
"""
```

```
def tokenize_fn(example):
 tokenized = tokenizer(
 example["prompt"],
 truncation=True,
 padding="max_length",
 max_length=512
 )
 tokenized["labels"] = tokenized["input_ids"].copy()
 return tokenized

import os

output_directory = os.path.join(".", "peft_lab_outputs")

"""Defines the key hyperparameters and behaviors for fine-tuning the model
using Hugging Face's Trainer or SFTTrainer"""

from transformers import TrainingArguments

training_args = TrainingArguments(
  output_dir="./peft_lab_outputs",
  per_device_train_batch_size=1,
  gradient_accumulation_steps=1,
  learning_rate=2e-4,
  num_train_epochs=1,
  logging_steps=1,
  save_strategy="no"
)

"""Creates a data collator that dynamically batches tokenized examples for
training."""

from transformers import DataCollatorForLanguageModeling


data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer,
mlm=False)

"""SFTTrainer is a wrapper around Hugging Face's Trainer that supports
parameter-efficient fine-tuning (PEFT) via LoRA"""

from trl import SFTTrainer
from transformers import DataCollatorForLanguageModeling
```

```
trainer = SFTTrainer(
  model=foundation_model,
  args=training_args,
  train_dataset=train_sample,
  peft_config=lora_config,
  data_collator=data_collator
)

trainer.train()

# Define output directory for saving
output_directory = "./peft_outputs"
peft_model_path = os.path.join(output_directory, "lora_model")

trainer.model.save_pretrained(peft_model_path)
tokenizer.save_pretrained(peft_model_path)

"""### Step 5: Evaluate the fine-tuned model's performance using the same
metric."""

import gc
import torch
gc.collect()
torch.cuda.empty_cache()

import os
os.environ["PYTORCH_CUDA_ALLOC_CONF"] = "expandable_segments:True"

from transformers import BitsAndBytesConfig

bnb_config = BitsAndBytesConfig(
  load_in_4bit=True,
  bnb_4bit_compute_dtype=torch.float16,
  bnb_4bit_quant_type="nf4",
  bnb_4bit_use_double_quant=True,
  llm_int8_enable_fp32_cpu_offload=True
)

peft_model_path = "./peft_outputs/lora_model"

from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained(peft_model_path)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "left"
```

```
bnb_config = BitsAndBytesConfig(
  load_in_4bit=True,
  bnb_4bit_compute_dtype=torch.float16, # valid for GPU
  bnb_4bit_quant_type="nf4",      # use 'nf4' for GPU compatibility
  llm_int8_enable_fp32_cpu_offload=False # must be False when not
offloading
)

from peft import AutoPeftModelForCausalLM
from transformers import AutoTokenizer

# Load model + tokenizer from saved adapter
loaded_model = AutoPeftModelForCausalLM.from_pretrained(
  peft_model_path,
  device_map="auto",
  is_trainable=False,
  quantization_config=bnb_config,
  torch_dtype=torch.float16
)

from datasets import load_dataset

test_sample = load_dataset("gbharti/finance-alpaca",
split="train[1000:1300]")

prompts = [
  f"Instruction: {ex['instruction']}\nInput: {ex['input']}\nResponse:"
  for ex in test_sample
]
references = [ex["output"] for ex in test_sample]

from evaluate import load
from tqdm import tqdm
import pandas as pd

metric = load("rouge") # or use rouge_scorer if preferred
results = []
all_outputs = []

batch_size = 4
loaded_model.eval()
```

```
for i in tqdm(range(0, len(prompts), batch_size)):
  batch_prompts = prompts[i:i+batch_size]
  batch_refs = references[i:i+batch_size]

  inputs = tokenizer(batch_prompts, return_tensors="pt", padding=True,
truncation=True).to(loaded_model.device)

  with torch.no_grad():
    outputs = loaded_model.generate(
      input_ids=inputs["input_ids"],
      attention_mask=inputs["attention_mask"],
      do_sample=True,
      temperature=0.7,
      top_p=0.9,
      max_new_tokens=128,
      pad_token_id=tokenizer.eos_token_id
    )

  decoded_outputs = tokenizer.batch_decode(outputs,
skip_special_tokens=True)
  all_outputs.extend(decoded_outputs)

  for prompt_text, pred, ref in zip(batch_prompts, decoded_outputs,
batch_refs):
    metric.add(prediction=pred, reference=ref)
    results.append({
      "prompt": prompt_text,
      "reference": ref,
      "prediction": pred
    })

# Create final DataFrame
results_df = pd.DataFrame(results)

# Compute ROUGE scores
rouge_scores = metric.compute()

results_df

rouge_scores
```