Fine-Tuning LLaMA2 with LoRA — Documentation

Overview

This document demonstrates fine-tuning of LLaMA2 using LoRA (Low-Rank Adaptation) and pushing the trained model to Hugging Face Hub.

- Base Model: meta-llama/Llama-2
- Fine-tuning Approach: LoRA (parameter-efficient fine-tuning)
- Evaluation Metric: Loss & Perplexity
- Deployment: Hugging Face Hub

1. What is Fine-Tuning?

Fine-tuning adapts a pre-trained model (like LLaMA2) to a specific dataset or task by updating only selected parameters.

With LoRA, instead of training all billions of parameters, we update only a small set of low-rank matrices (A and B).

2. LoRA Intuition

Normally, each Transformer layer has a large weight matrix W (d × d).

LoRA approximates the update ∆W as:

∆W = A × B where A (d × r), B (r × d)

- r = rank (small value, e.g., 8–16)
- Only A and B are trained → saves GPU memory & time.

Benefit: Billions of parameters reduced to a few million trainable ones.

3. What is Perplexity?

Perplexity (PPL) is the standard metric for language models.

It measures how well the model predicts text.

PPL = exp(cross-entropy loss)

- Low PPL → Model is less confused → Better performance.

- High PPL → Model is more confused → Poor performance.

Example: If model's PPL = 1.5 → it has ~1.5 choices of confusion on average.

4. Training Logs

During training, logs show:

Step 500: loss=2.31, eval_loss=2.05, perplexity=7.8

Step 1000: loss=1.95, eval_loss=1.80, perplexity=6.0

- loss ↓ and perplexity ↓ → model is learning.

- If they flatten → training has converged.

5. Monitoring on Hugging Face Hub

When pushed to Hugging Face, training metrics (loss, perplexity, epochs) appear in the Training Metrics tab.

6. Evaluation Script

Once the model is pushed, you can evaluate it:

```
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer

model_id = "username/your-finetuned-model"
tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(model_id)

text = "The capital of France is"
inputs = tokenizer(text, return_tensors="pt")

with torch.no_grad():
outputs = model(**inputs, labels=inputs["input_ids"])
loss = outputs.loss
perplexity = torch.exp(loss)

print("Test Loss:", loss.item())
print("Perplexity:", perplexity.item())
```

7. Comparing with Base Model

- Run evaluation on base model (e.g., meta-llama/Llama-2)
- Run on fine-tuned model
- Compare Loss & Perplexity → Lower = Better

8. Qualitative Check

You can also test via text generation:

```
from transformers import pipeline

pipe = pipeline("text-generation", model=model, tokenizer=tokenizer)
output = pipe("Explain quantum computing in simple terms:", max_new_tokens=100)

print(output[0]["generated_text"])
```

Summary

- Fine-tuning with LoRA updates only low-rank matrices.
- Perplexity tells how much the model improved.
- Training logs + Hugging Face metrics show progress.
- Always compare base vs fine-tuned model.