12 SEPTEMBER 2024

Mdigital

Local LLMs & RAG

with Ollama and Postgres

AGENDA

Local large language models

1. Why use local LLMs?
2. How to use Ollama
3. Local Vector DB
4. Put it together: RAG
5. If we have time: Fine-tuning

Mdigital

Apologies

I’m a web developer…

Mdigital

Local LLM advantages

- ✓ You can keep your data private
- ✓ You don’t need to use paid APIs
- ✓ Efficient - limit power consumption
- ✓ Learning and experimentation

Ollama

Easy to install

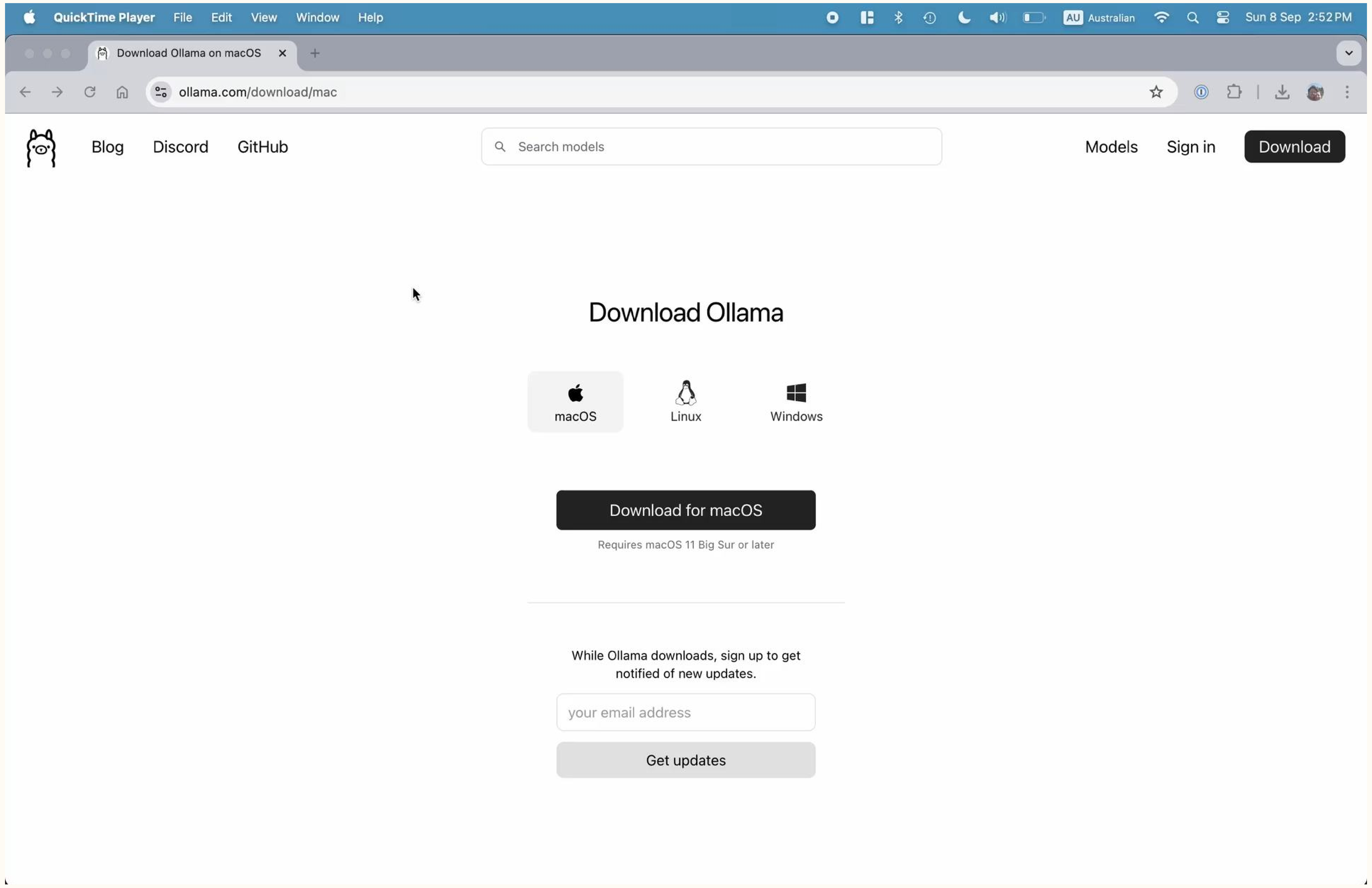
Access various models

Chat completions

Embeddings

REST API (OpenAI compatible)

Support LoRA for fine tuned models



Download Ollama

Requires macOS 11 Big Sur or later

While Ollama downloads, sign up to get notified of new updates.

Ollama API

```
curl http://localhost:11434/v1/completions -d '{
   "model":"llama3.1",
   "prompt":"What is the meaning of life?"
}'

curl http://localhost:11434/api/embeddings -d '{
  "model": "llama3.1",
  "prompt": "There is a story about eating the birds as you
crossed south georgia island, tell me more"
}'
```

Mdigital

Postgres

Easy to install

Access pgvector

Familiar API

```
docker pull pgvector/pgvector:pg16

docker volume create pgvector-data

docker run --name pgvector-container -e
POSTGRES_PASSWORD=<password> \
 -p 5432:5432 -v pgvector-data:/var/lib/postgresql/data \
 -d pgvector/pgvector:pg16

docker exec -it pgvector-container psql -U postgres
```

Mdigital

Creating a table

```
postgres=# CREATE DATABASE localrag;
CREATE DATABASE

postgres=# \c localrag
You are now connected to database "localrag" as user
"postgres".

localrag=# CREATE EXTENSION vector;
CREATE EXTENSION

localrag=# CREATE TABLE big_items (id bigserial PRIMARY KEY,
chunk text, embedding vector(4096));
CREATE TABLE
```

Mdigital

Putting it all together: demo

Fine tuning the Llama 3.1 model

- ✓ Improves the quality of the response
- ✓ Might be faster in production than rag
- ✓ Helps to control the output of the LLM for certain prompts
- ✓ Can be done using LoRA adapters with ollama

Mdigital

MLX

Framework for training / deploying LLMs

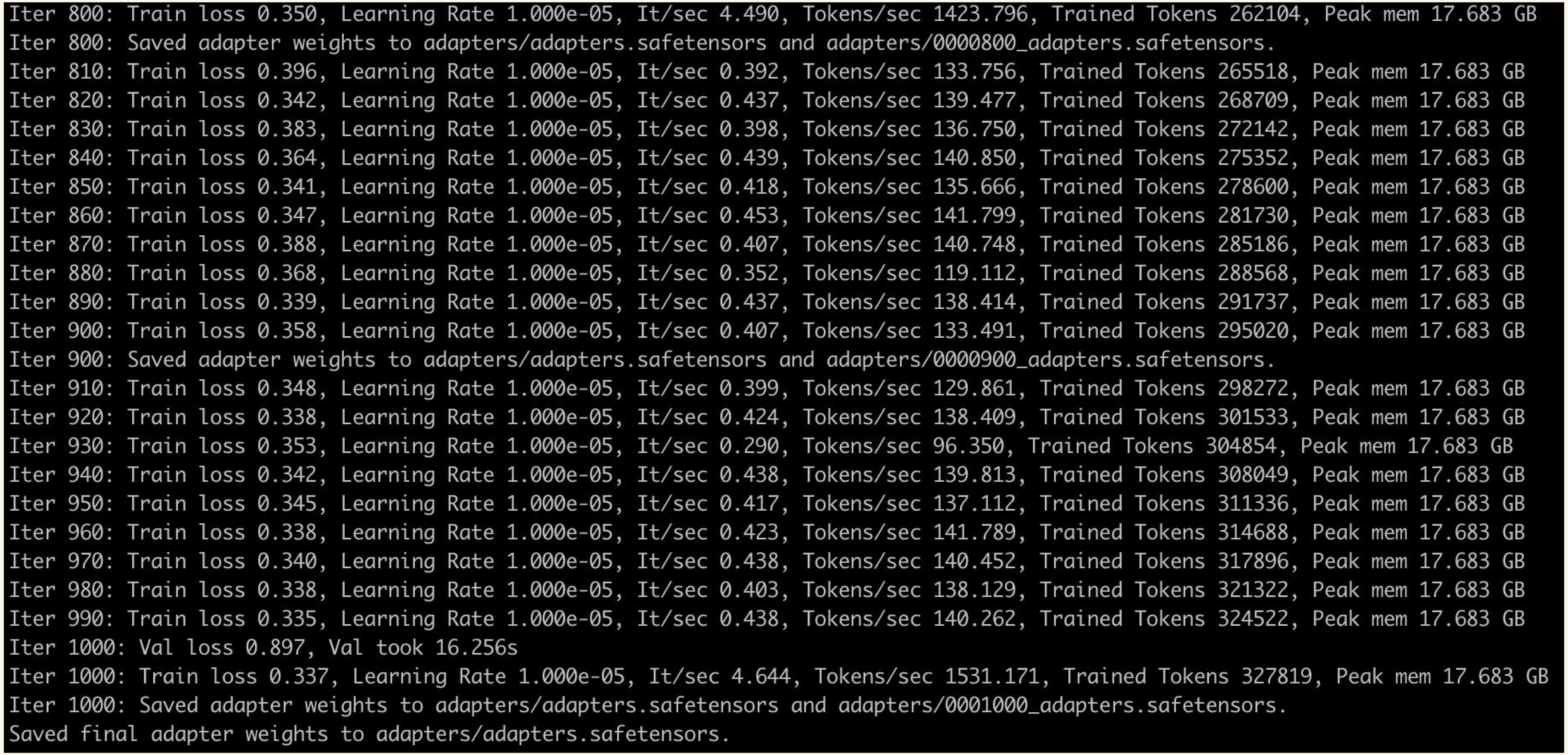
Works on Apple Silicon

Open source from Apple

```
python3 -m venv mlx_env
. ./mlx_env/bin/activate
pip install mlx-lm
pip install "huggingface_hub[cli]"

huggingface-cli login
mlx_lm.lora --train --model meta-llama/Meta-Llama-3.1-8B --
data data --batch-size 2
```

Mdigital



```
Iter 800: Train loss 0.350, Learning Rate 1.000e-05, It/sec 4.490, Tokens/sec 1423.796, Trained Tokens 262104, Peak mem 17.683 GB
Iter 800: Saved adapter weights to adapters/adapters.safetensors and adapters/0000800_adapters.safetensors.
Iter 810: Train loss 0.396, Learning Rate 1.000e-05, It/sec 0.392, Tokens/sec 133.756, Trained Tokens 265518, Peak mem 17.683 GB
Iter 820: Train loss 0.342, Learning Rate 1.000e-05, It/sec 0.437, Tokens/sec 139.477, Trained Tokens 268709, Peak mem 17.683 GB
Iter 830: Train loss 0.383, Learning Rate 1.000e-05, It/sec 0.398, Tokens/sec 136.750, Trained Tokens 272142, Peak mem 17.683 GB
Iter 840: Train loss 0.364, Learning Rate 1.000e-05, It/sec 0.439, Tokens/sec 140.850, Trained Tokens 275352, Peak mem 17.683 GB
Iter 850: Train loss 0.341, Learning Rate 1.000e-05, It/sec 0.418, Tokens/sec 135.666, Trained Tokens 278600, Peak mem 17.683 GB
Iter 860: Train loss 0.347, Learning Rate 1.000e-05, It/sec 0.453, Tokens/sec 141.799, Trained Tokens 281730, Peak mem 17.683 GB
Iter 870: Train loss 0.388, Learning Rate 1.000e-05, It/sec 0.407, Tokens/sec 140.748, Trained Tokens 285186, Peak mem 17.683 GB
Iter 880: Train loss 0.368, Learning Rate 1.000e-05, It/sec 0.352, Tokens/sec 119.112, Trained Tokens 288568, Peak mem 17.683 GB
Iter 890: Train loss 0.339, Learning Rate 1.000e-05, It/sec 0.437, Tokens/sec 138.414, Trained Tokens 291737, Peak mem 17.683 GB
Iter 900: Train loss 0.358, Learning Rate 1.000e-05, It/sec 0.407, Tokens/sec 133.491, Trained Tokens 295020, Peak mem 17.683 GB
Iter 900: Saved adapter weights to adapters/adapters.safetensors and adapters/0000900_adapters.safetensors.
Iter 910: Train loss 0.348, Learning Rate 1.000e-05, It/sec 0.399, Tokens/sec 129.861, Trained Tokens 298272, Peak mem 17.683 GB
Iter 920: Train loss 0.338, Learning Rate 1.000e-05, It/sec 0.424, Tokens/sec 138.409, Trained Tokens 301533, Peak mem 17.683 GB
Iter 930: Train loss 0.353, Learning Rate 1.000e-05, It/sec 0.290, Tokens/sec 96.350, Trained Tokens 304854, Peak mem 17.683 GB
Iter 940: Train loss 0.342, Learning Rate 1.000e-05, It/sec 0.438, Tokens/sec 139.813, Trained Tokens 308049, Peak mem 17.683 GB
Iter 950: Train loss 0.345, Learning Rate 1.000e-05, It/sec 0.417, Tokens/sec 137.112, Trained Tokens 311336, Peak mem 17.683 GB
Iter 960: Train loss 0.338, Learning Rate 1.000e-05, It/sec 0.423, Tokens/sec 141.789, Trained Tokens 314688, Peak mem 17.683 GB
Iter 970: Train loss 0.340, Learning Rate 1.000e-05, It/sec 0.438, Tokens/sec 140.452, Trained Tokens 317896, Peak mem 17.683 GB
Iter 980: Train loss 0.338, Learning Rate 1.000e-05, It/sec 0.403, Tokens/sec 138.129, Trained Tokens 321322, Peak mem 17.683 GB
Iter 990: Train loss 0.335, Learning Rate 1.000e-05, It/sec 0.438, Tokens/sec 140.262, Trained Tokens 324522, Peak mem 17.683 GB
Iter 1000: Val loss 0.897, Val took 16.256s
Iter 1000: Train loss 0.337, Learning Rate 1.000e-05, It/sec 4.644, Tokens/sec 1531.171, Trained Tokens 327819, Peak mem 17.683 GB
Iter 1000: Saved adapter weights to adapters/adapters.safetensors and adapters/0001000_adapters.safetensors.
Saved final adapter weights to adapters/adapters.safetensors.
```

Mdigital

Run with ollama

```
vi Modelfile
```

```
FROM llama3.1
ADAPTER ./adapters
~
~
:wq
```

```
ollama create commentcheck -f Modelfile
ollama run commentcheck
```

Mdigital

Thank you

robin@mdigital.co.nz

Mdigital