




 [edumunozsala](#) / [llama-2-7B-4bit-python-coder](#) Public

Fine-tune Llama-2 7B to generate Python code

 GPL-3.0 license 34 stars  10 forks  Activity Star Notifications

<> Code

 Issues Pull requests Actions Projects Security Insights main ▾

Go to file



edumunozsala ...

2 weeks ago [View code](#) README.md

Fine-tune a Llama 2 7B parameters in 4-bit to generate Python Code

LlaMa-2 7B model fine-tuned on the `python_code_instructions_18k_alpaca` Code instructions dataset by using the method QLoRA in 4-bit with [PEFT](#) and `bitsandbytes` library.

Additionally, we include a **GPTQ quantized version** of the model, **LlaMa-2 7B 4-bit GPTQ** using Auto-GPTQ integrated with Hugging Face transformers. The quantization parameters for the GPTQ algo are:

- 4-bit quantization
- Group size is 128
- Dataset C4
- Decreasing activation is False

The dataset

For our tuning process, we will take a [dataset](#) containing about 18,000 examples where the model is asked to build a Python code that solves a given task. This is an extraction of the [original dataset](#) where only the Python language examples are selected. Each row contains the description of the task to be solved, an example of data input to the task if applicable, and the generated code fragment that solves the task is provided.

Problem description

Our goal is to fine-tune the pretrained model, Llama 2 7B parameters, using 4-bit quantization to produce a Python coder. We will run the training on Google Colab using a A100 to get better performance. But you can try out to run it on a T4 adjusting some parameters to reduce memory consumption like batch size.

Once the model is fine-tuned, we apply the GPTQ quantization to get a new model with a better inference time.

Note: This is still in progress and some models may be included.

The base model

Llama-2

Meta developed and publicly released the Llama 2 family of large language models (LLMs), a collection of pretrained and fine-tuned generative text models ranging in scale from 7 billion to 70 billion parameters.

Model Architecture Llama 2 is an auto-regressive language model that uses an optimized transformer architecture. The tuned versions use supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) to align to human preferences for helpfulness and safety

Quantization

A quick definition extracted from a great article in Medium by Benjamin Marie "[GPTQ or bitsandbytes: Which Quantization Method to Use for LLMs — Examples with Llama 2](#)" (Only for Medium subscribers)

"GPTQ (Frantar et al., 2023) was first applied to models ready to deploy. In other words, once the model is fully fine-tuned, GPTQ will be applied to reduce its size. GPTQ can lower the weight precision to 4-bit or 3-bit. In practice, GPTQ is mainly used for 4-bit quantization. 3-bit has been shown very unstable (Dettmers and Zettlemoyer, 2023). It quantizes without loading the entire model into memory. Instead, GPTQ loads and quantizes the LLM module by module. Quantization also requires a small sample of data for calibration which can take more than one hour on a consumer GPU."

Content

Still In progress

- Fine-tuning notebook `Llama-2-finetune-qlora-python-coder.ipynb` : In this notebook we fine-tune the model.
- Fine-tuning script `train.py` : An script to run training process.
- Notebook to run the script `run-script-finetune-llama-2-python-coder.ipynb` : An very simple example on how to use NER to search for relevant articles.
- Quantization notebook `Quantize-Llama-2-7B-python-coder-GPTQ.ipynb` : In this notebook we quantize the model with GPTQ.

Example of usage

```
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer

model_id = "edumunozsala/llama-2-7b-int4-python-code-20k"

tokenizer = AutoTokenizer.from_pretrained(model_id)

model = AutoModelForCausalLM.from_pretrained(model_id, load_in_4bit=True, torch_device_map=device_map)

instruction="Write a Python function to display the first and last elements of
input="

prompt = f"""### Instruction:
Use the Task below and the Input given to write the Response, which is a progr

### Task:
{instruction}

### Input:
{input}

### Response:
"""
```

```
input_ids = tokenizer(prompt, return_tensors="pt", truncation=True).input_ids.  
# with torch.inference_mode():  
outputs = model.generate(input_ids=input_ids, max_new_tokens=100, do_sample=True)  
  
print(f"Prompt:\n{prompt}\n")  
print(f"Generated instruction:\n{tokenizer.batch_decode(outputs.detach().cpu().numpy().tolist())}")
```

Citation

```
@misc {edumunozsala_2023,  
  author      = { {Eduardo Muñoz} },  
  title       = { llama-2-7b-int4-python-coder },  
  year        = 2023,  
  url         = { https://huggingface.co/edumunozsala/llama-2-7b-int4-python-18k-alpaca },  
  publisher   = { Hugging Face }  
}
```



Contributing

If you find some bug or typo, please let me know or fix it and push it to be analyzed.

License

These notebooks are under a public GNU License version 3.

Releases

No releases published

Packages

No packages published

Languages

