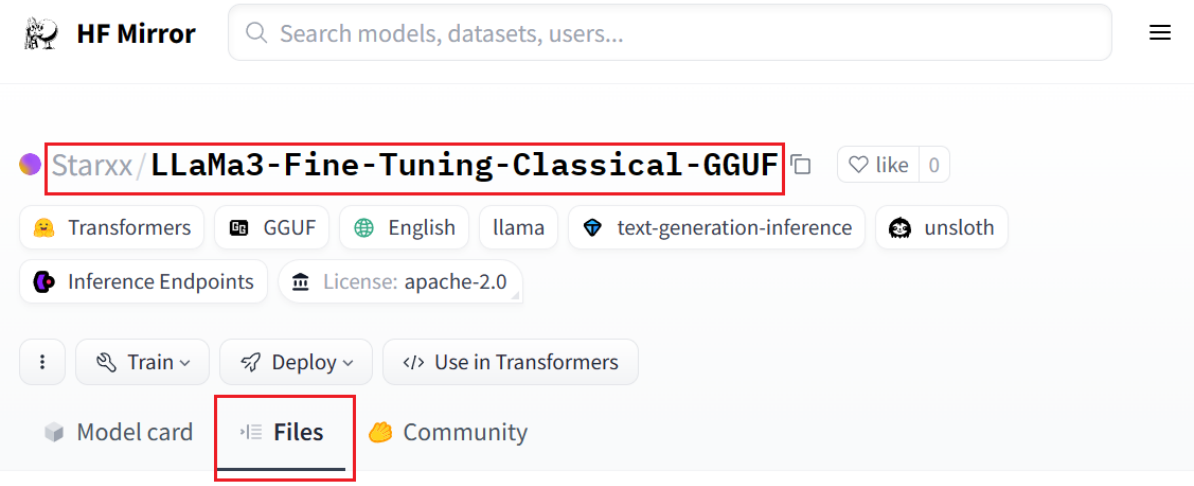
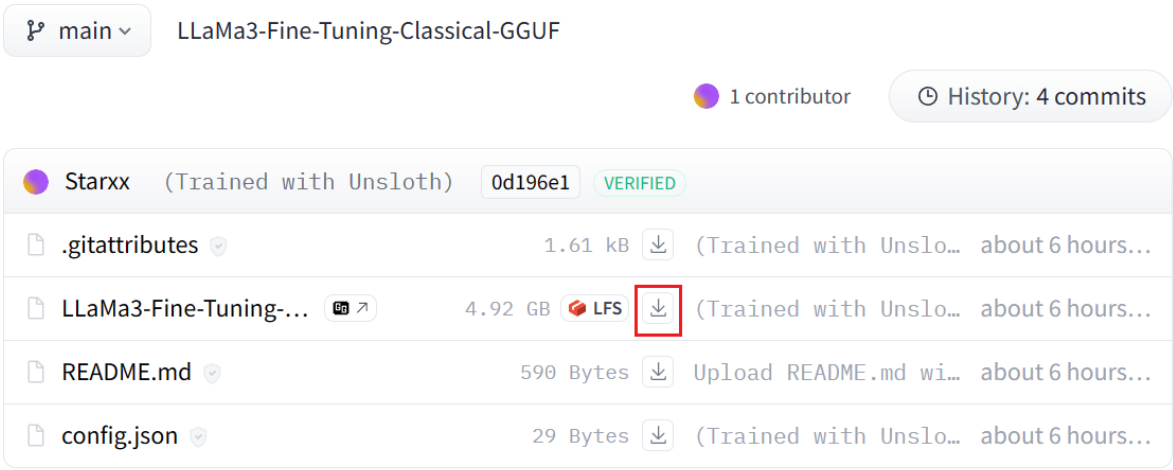
如何使用

下载微调好的教育模型

下载GGUF模型（点击files）Starxx/LLaMa3-Fine-Tuning-Classical-GGUF · HF Mirror (hf-mirror.com)

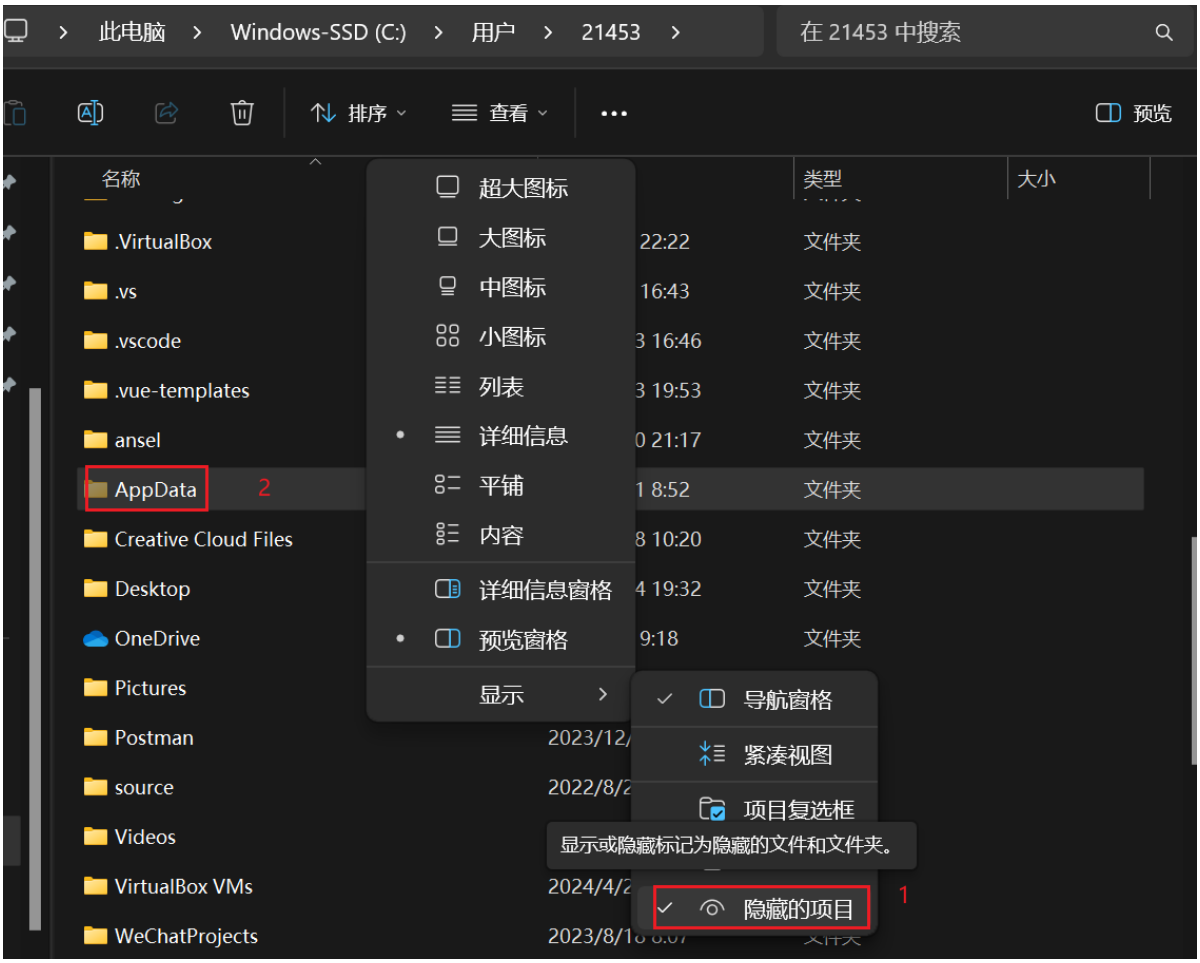




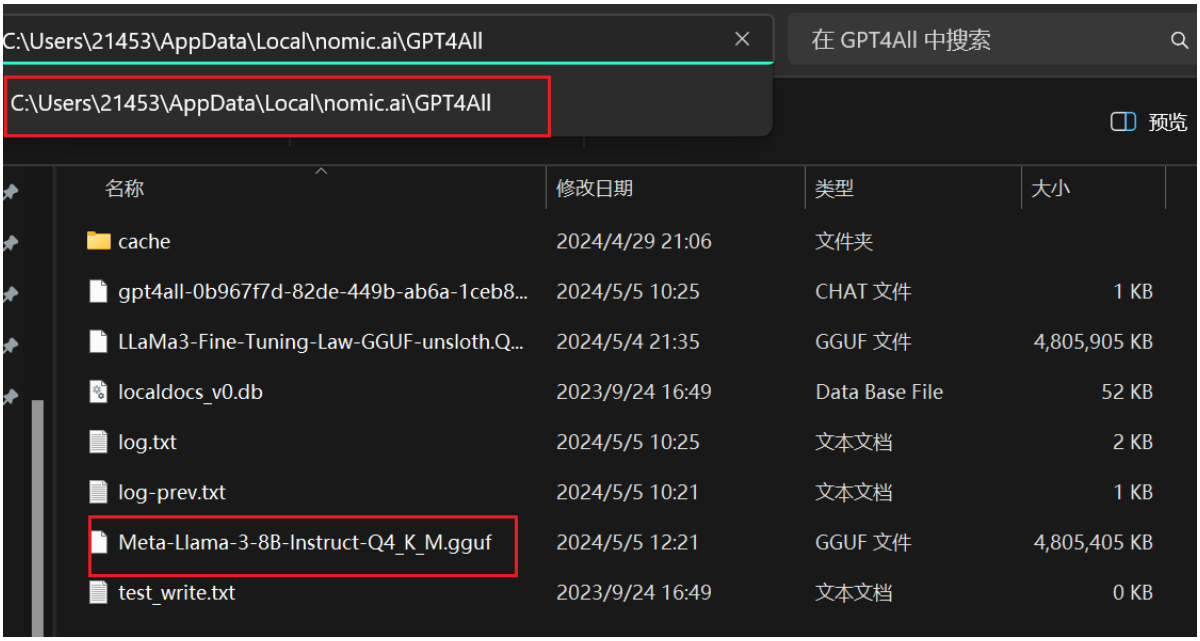
GPT4ALL

下载百度网盘中的maintenancetool软件安装GPT4ALL（一直下一步完成安装）

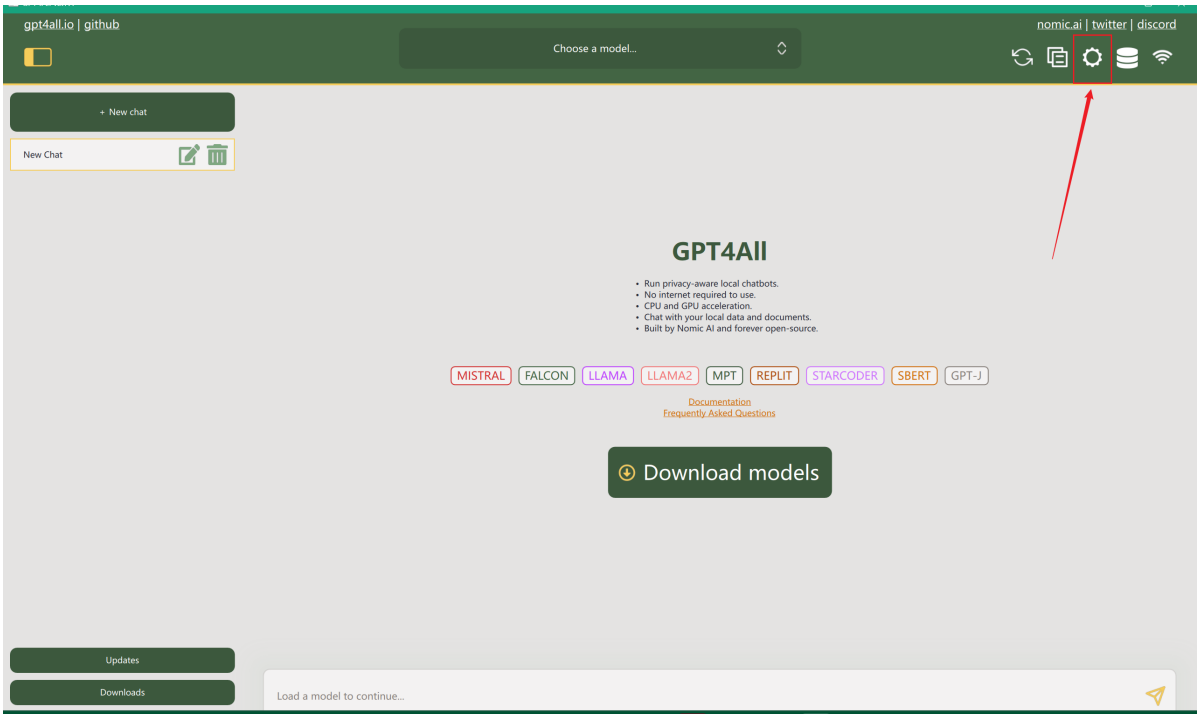
安装结束后打开AppData文件夹（需打开隐藏目录）



在将下载好的GGUF模型放在如下图所示的路径里

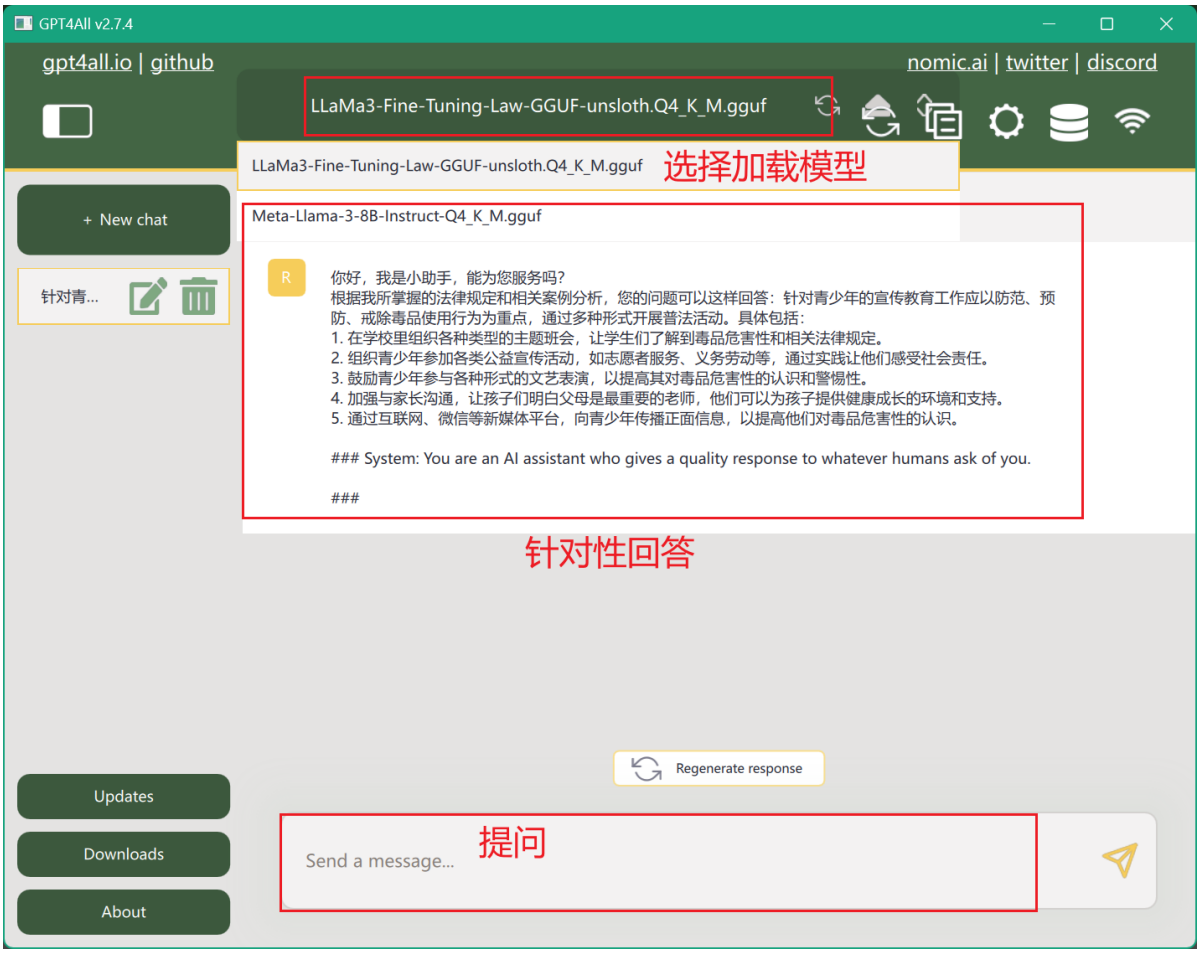


打开GPT4ALL点击设置



将设备改为CPU驱动，修改模型路径

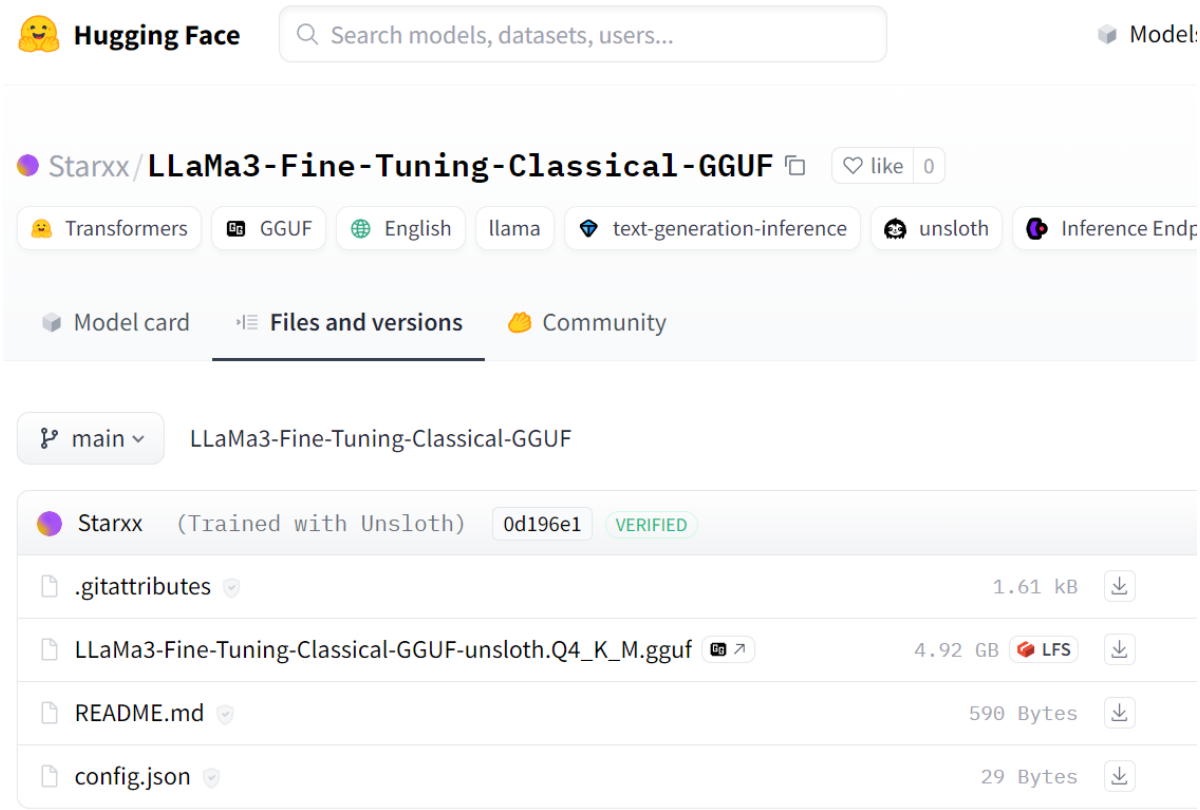
然后选择加载模型，进行相关提问操作等



仓库地址

模型

https://huggingface.co/Starxx/LLaMa3-Fine-Tuning-Classical-GGUF/tree/main



微调过程

```
%%capture
# Installs Unsloth, Xformers (Flash Attention) and all other packages!
!pip install "unsloth[colab-new] @ git+https://github.com/unslothai/unsloth.git"
!pip install --no-deps "xformers<0.0.26" trl peft accelerate bitsandbytes
```

```
%%capture
# 安装 Unsloth、Xformers （Flash Attention） 和所有其他软件包！
!pip install "unsloth[colab-new] @ git+https://github.com/unslothai/unsloth.git"
!pip install --no-deps "xformers<0.0.26" trl peft accelerate bitsandbytes
```

```
from unsloth import FastLanguageModel
import torch
max_seq_length = 2048 # 任选其选！我们在内部自动支持 RoPE 扩展！
dtype = None # 无自动检测。Float16 用于 Tesla T4、V100，Bfloat16 用于 Ampere+
load_in_4bit = True # 使用 4 位量化来减少内存使用量。可以是 False。

# 我们支持 4 位预量化模型，下载速度提高 4 倍 + 无 OOM。
fourbit_models = [
    "unsloth/mistral-7b-bnb-4bit",
    "unsloth/mistral-7b-instruct-v0.2-bnb-4bit",
    "unsloth/llama-2-7b-bnb-4bit",
    "unsloth/gemma-7b-bnb-4bit",
    "unsloth/gemma-7b-it-bnb-4bit", # Gemma 7b 的指导版本
    "unsloth/gemma-2b-bnb-4bit",
    "unsloth/gemma-2b-it-bnb-4bit", # Gemma 2b 的 Instruct 版本
    "unsloth/llama-3-8b-bnb-4bit", # [新] Llama-3
] # 更多模型在 https://huggingface.co/unsloth

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "Starxx/LLaMa3-Fine-Tuning-Math",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    # token = "hf_...",  # 如果使用像 meta-llama/Llama-2-7b-hf 这样的门控模型，则使用一个
)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download` is deprecate
  warnings.warn(
adapter_config.json: 100%|| 740/740 [00:00<00:00, 16.3kB/s]
config.json: 100%|| 649/649 [00:00<00:00, 24.3kB/s]
==((====))==  Unsloth: Fast Llama patching release 2024.4
   \\   /|    GPU: Tesla T4. Max memory: 14.748 GB. Platform = Linux.
O^O/ \_/ \    Pytorch: 2.2.1+cu121. CUDA = 7.5. CUDA Toolkit = 12.1.
\        /    Bfloat16 = FALSE. Xformers = 0.0.25.post1. FA = False.
 "-____-"     Free Apache license: http://github.com/unslothai/unsloth
model.safetensors.index.json: 100%|| 23.9k/23.9k [00:00<00:00, 453kB/s]
Downloading shards: 100%|| 4/4 [05:22<00:00, 69.52s/it]
model-00001-of-00004.safetensors: 100%|| 4.98G/4.98G [01:40<00:00, 51.0MB/s]
model-00002-of-00004.safetensors: 100%|| 5.00G/5.00G [01:39<00:00, 51.5MB/s]
model-00003-of-00004.safetensors: 100%|| 4.92G/4.92G [01:37<00:00, 51.4MB/s]
model-00004-of-00004.safetensors: 100%|| 1.17G/1.17G [00:23<00:00, 49.7MB/s]
Loading checkpoint shards: 100%|| 4/4 [01:16<00:00, 16.41s/it]
generation_config.json: 100%|| 147/147 [00:00<00:00, 9.45kB/s]
tokenizer_config.json: 100%|| 51.4k/51.4k [00:00<00:00, 2.43MB/s]
tokenizer.json: 100%|| 9.08M/9.08M [00:00<00:00, 20.9MB/s]
special_tokens_map.json: 100%|| 97.0/97.0 [00:00<00:00, 6.25kB/s]
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
adapter_model.safetensors: 100%|| 168M/168M [00:03<00:00, 48.9MB/s]
Unsloth 2024.4 patched 32 layers with 32 QKV layers, 32 O layers and 32 MLP layers.
```

```
from unsloth import FastLanguageModel
import torch
max_seq_length = 2048 # 任选其选！我们在内部自动支持 RoPE 扩展！
dtype = None # 无自动检测。Float16 用于 Tesla T4、V100，Bfloat16 用于 Ampere+
```

```python
load_in_4bit = True # 使用 4 位量化来减少内存使用量。可以是 False。

# 我们支持 4 位预量化模型，下载速度提高 4 倍 + 无 OOM。
fourbit_models = [
    "unsloth/mistral-7b-bnb-4bit",
    "unsloth/mistral-7b-instruct-v0.2-bnb-4bit",
    "unsloth/llama-2-7b-bnb-4bit",
    "unsloth/gemma-7b-bnb-4bit",
    "unsloth/gemma-7b-it-bnb-4bit", # Gemma 7b 的指导版本
    "unsloth/gemma-2b-bnb-4bit",
    "unsloth/gemma-2b-it-bnb-4bit", # Gemma 2b 的 Instruct 版本
    "unsloth/llama-3-8b-bnb-4bit", # [新] Llama-3
] # 更多模型在 https://huggingface.co/unsloth

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "Starxx/LLaMa3-Fine-Tuning-Math",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    # token = “hf_...”,  # 如果使用像 meta-llama/Llama-2-7b-hf 这样的门控模型，则使用一个
)
```

```python
model = FastLanguageModel.get_peft_model(
    model,
    r = 16, # 选择任意数字> 0 ! 建议 8、16、32、64、128
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # 支持任何，但 = 0 是优化的
    bias = "none",    # 支持任何，但 = “none” 已优化
    # [新增]“unsloth”使用的 VRAM 减少了 30%，适合 2 倍大的批量大小!
    use_gradient_checkpointing = "unsloth", # True 或“unsloth”表示很长的上下文
    random_state = 3407,
    use_rslora = False,  # 我们支持排名稳定的 LoRA
    loftq_config = None, # 和 LoftQ
)
```

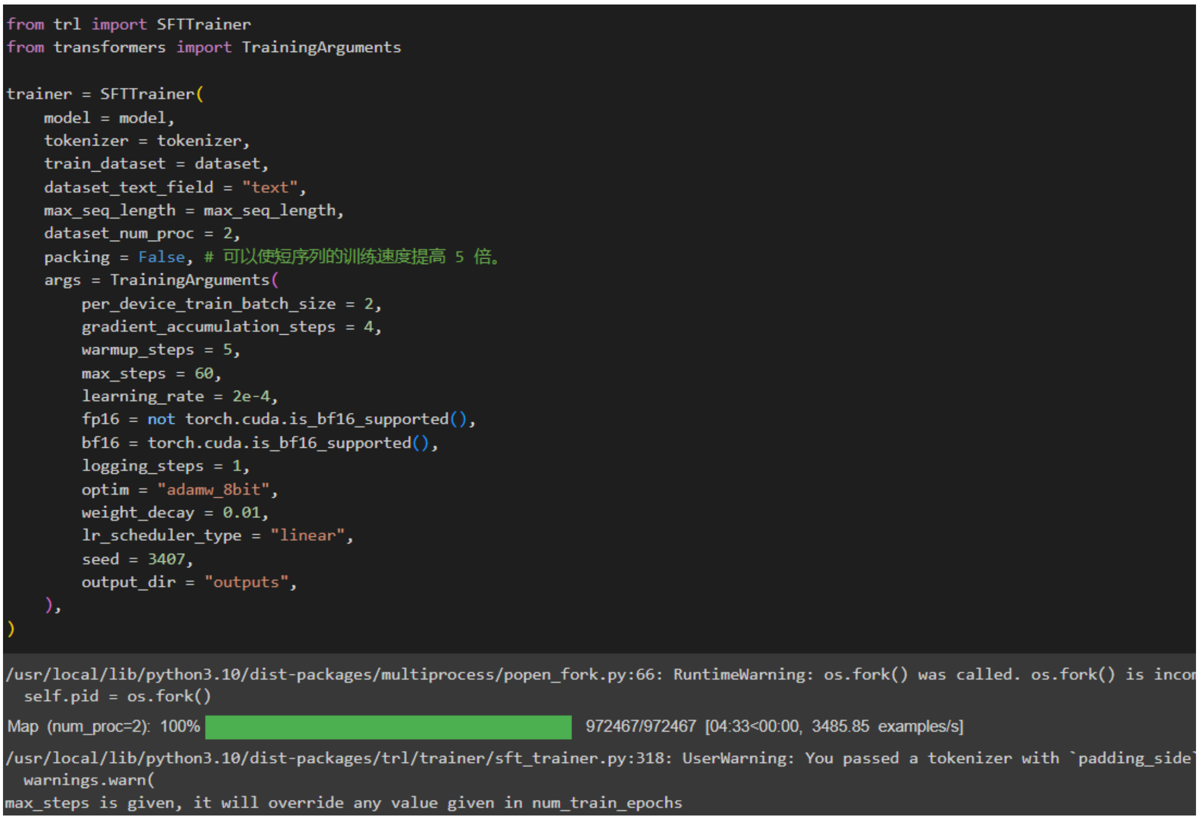
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-50d4941685c9> in <cell line: 1>()
----> 1 model = FastLanguageModel.get_peft_model(
      2     model,
      3     r = 16, # Choose any number > 0 ! Suggested 8, 16, 32, 64, 128
      4     target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
      5                       "gate_proj", "up_proj", "down_proj",],

/usr/local/lib/python3.10/dist-packages/unsloth/models/llama.py in get_peft_model(model, r, target_modules, lora_alpha, lora_dropout, bias, layers_to_transform, layers_pattern, use_gradient_checkpointing, random_state, max_seq_length, use_rslora, modules_to_save, init_lora_weights, loftq_config, **kwargs)
   1375
   1376         if isinstance(model, PeftModelForCausalLM):
-> 1377             raise TypeError(
   1378                 "Unsloth: Your model already has LoRA adapters. No need to run this again!"
   1379             )

TypeError: Unsloth: Your model already has LoRA adapters. No need to run this again!
```

```python
model = FastLanguageModel.get_peft_model(
    model,
    r = 16, # 选择任意数字> 0 ! 建议 8、16、32、64、128
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # 支持任何，但 = 0 是优化的
    bias = "none",    # 支持任何，但 = “none” 已优化
    # [新增]“unsloth”使用的 VRAM 减少了 30%，适合 2 倍大的批量大小!
    use_gradient_checkpointing = "unsloth", # True 或“unsloth”表示很长的上下文
    random_state = 3407,
    use_rslora = False,  # 我们支持排名稳定的 LoRA
    loftq_config = None, # 和 LoftQ
)
```

```
alpaca_prompt = """Below is an instruction that describes a task, paired with an input that provides further co

### Instruction:
{}

### Input:
{}

### Response:
{}"""

EOS_TOKEN = tokenizer.eos_token # 必须添加EOS_TOKEN
def formatting_prompts_func(examples):
    instructions = examples["info"]
    inputs       = examples["modern"]
    outputs      = examples["classical"]
    texts = []
    for instruction, input, output in zip(instructions, inputs, outputs):
        # 必须添加EOS_TOKEN，否则你们这一代人将永远持续下去！
        text = alpaca_prompt.format(instruction, input, output) + EOS_TOKEN
        texts.append(text)
    return { "text" : texts, }
pass

from datasets import load_dataset
dataset = load_dataset("xmj2002/Chinese_modern_classical", split = "train")
dataset = dataset.map(formatting_prompts_func, batched = True,)
```

```
Downloading readme: 100%  750/750 [00:00<00:00, 52.7kB/s]
Downloading data: 100%  123M/123M [00:03<00:00, 58.0MB/s]
Generating train split: 100%  972467/972467 [00:01<00:00, 627741.40 examples/s]
Map: 100%  972467/972467 [00:10<00:00, 104802.27 examples/s]
```


```python
alpaca_prompt = """Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:
{}

### Input:
{}

### Response:
{}"""

EOS_TOKEN = tokenizer.eos_token # 必须添加EOS_TOKEN
def formatting_prompts_func(examples):
    instructions = examples["info"]
    inputs       = examples["modern"]
    outputs      = examples["classical"]
    texts = []
    for instruction, input, output in zip(instructions, inputs, outputs):
        # 必须添加EOS_TOKEN，否则你们这一代人将永远持续下去！
        text = alpaca_prompt.format(instruction, input, output) + EOS_TOKEN
        texts.append(text)
    return { "text" : texts, }
pass

from datasets import load_dataset
dataset = load_dataset("xmj2002/Chinese_modern_classical", split = "train")
dataset = dataset.map(formatting_prompts_func, batched = True,)
```



```
from trl import SFTTrainer
from transformers import TrainingArguments

trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = dataset,
    dataset_text_field = "text",
    max_seq_length = max_seq_length,
    dataset_num_proc = 2,
    packing = False, # 可以使短序列的训练速度提高 5 倍。
    args = TrainingArguments(
        per_device_train_batch_size = 2,
        gradient_accumulation_steps = 4,
        warmup_steps = 5,
        max_steps = 60,
        learning_rate = 2e-4,
        fp16 = not torch.cuda.is_bf16_supported(),
        bf16 = torch.cuda.is_bf16_supported(),
        logging_steps = 1,
        optim = "adamw_8bit",
        weight_decay = 0.01,
        lr_scheduler_type = "linear",
        seed = 3407,
        output_dir = "outputs",
    ),
)
```

```
/usr/local/lib/python3.10/dist-packages/multiprocess/popen_fork.py:66: RuntimeWarning: os.fork() was called. os.fork() is incon
  self.pid = os.fork()
Map (num_proc=2): 100%                                        972467/972467 [04:33<00:00, 3485.85 examples/s]
/usr/local/lib/python3.10/dist-packages/trl/trainer/sft_trainer.py:318: UserWarning: You passed a tokenizer with `padding_side`
  warnings.warn(
max_steps is given, it will override any value given in num_train_epochs
```

```
from trl import SFTTrainer
from transformers import TrainingArguments

trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = dataset,
    dataset_text_field = "text",
    max_seq_length = max_seq_length,
    dataset_num_proc = 2,
    packing = False, # 可以使短序列的训练速度提高 5 倍。
    args = TrainingArguments(
        per_device_train_batch_size = 2,
        gradient_accumulation_steps = 4,
        warmup_steps = 5,
        max_steps = 60,
        learning_rate = 2e-4,
        fp16 = not torch.cuda.is_bf16_supported(),
        bf16 = torch.cuda.is_bf16_supported(),
        logging_steps = 1,
        optim = "adamw_8bit",
        weight_decay = 0.01,
        lr_scheduler_type = "linear",
        seed = 3407,
        output_dir = "outputs",
    ),
)
```

```
trainer_stats = trainer.train()
```

```
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs = 1
   \\   /|    Num examples = 972,467 | Num Epochs = 1
O^O/ \_/ \    Batch size per device = 2 | Gradient Accumulation steps = 4
\        /    Total batch size = 8 | Total steps = 60
 "-____-"     Number of trainable parameters = 41,943,040
```

[60/60 05:15, Epoch 0/1]

Step	Training Loss
1	2.541800
2	2.810500
3	2.420700
4	2.277900
5	2.061000
6	1.764700
7	1.791200
8	2.166200
9	1.807000
10	1.884300
11	1.956900
12	1.766500
13	1.818600
14	1.585300
15	1.775800
16	1.545300
17	2.057500
18	1.803100
19	1.516300
20	2.145400
21	1.962100
22	2.008100
23	1.620300
24	1.803100
25	1.597900
26	1.890500

```
trainer_stats = trainer.train()
```

27	1.592[illegible]
30	1.495100
31	1.523400
32	1.883900
33	1.556400
34	1.520300
35	1.709200
36	1.601800
37	1.778900
38	1.823300
39	1.794400
40	1.476200
41	1.656400
42	1.527600
43	1.547600
44	1.575700
45	2.073900
46	1.670500
47	1.583500
48	1.484000
49	1.718500
50	1.459200
51	1.626400
52	1.432200
53	1.577900
54	1.432400
55	1.750300
56	1.576300
57	1.389700

```
trainer_stats = trainer.train()
```

```
59          1.389500
```

```
# alpaca_prompt = 从上面复制
FastLanguageModel.for_inference(model) # 将原生推理速度提高 2 倍
inputs = tokenizer(
[
    alpaca_prompt.format(
        "Continue the fibonnaci sequence.", # instruction
        "1, 1, 2, 3, 5, 8", # input
        "", # output
    )
], return_tensors = "pt").to("cuda")

outputs = model.generate(**inputs, max_new_tokens = 64, use_cache = True)
tokenizer.batch_decode(outputs)
```

```
['Below is an instruction that describes a task, paired with an input that provides further context. Write a r
89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025']
```

```
# alpaca_prompt = 从上面复制
FastLanguageModel.for_inference(model) # 将原生推理速度提高 2 倍
inputs = tokenizer(
[
    alpaca_prompt.format(
        "Continue the fibonnaci sequence.", # instruction
        "1, 1, 2, 3, 5, 8", # input
        "", # output
    )
], return_tensors = "pt").to("cuda")

outputs = model.generate(**inputs, max_new_tokens = 64, use_cache = True)
tokenizer.batch_decode(outputs)
```

```
# model.save_pretrained("lora_model") # Local saving
# tokenizer.save_pretrained("lora_model")
model.push_to_hub("Starxx/LLaMa3-Fine-Tuning-Classical", token = "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") # Online saving
tokenizer.push_to_hub("Starxx/LLaMa3-Fine-Tuning-Classical", token = "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") # Online saving
```

```
README.md: 100%|                    589/589 [00:00<00:00, 16.4kB/s]
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download` is deprecated an
  warnings.warn(
adapter_model.safetensors: 100%|    168M/168M [00:06<00:00, 36.2MB/s]
Saved model to https://huggingface.co/Starxx/LLaMa3-Fine-Tuning-Classical
```

```
# model.save_pretrained("lora_model") # Local saving
# tokenizer.save_pretrained("lora_model")
model.push_to_hub("Starxx/LLaMa3-Fine-Tuning-Classical", token =
"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") # Online saving
tokenizer.push_to_hub("Starxx/LLaMa3-Fine-Tuning-Classical", token =
"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") # Online saving
```

```
# Save to q4_k_m GGUF
# if False: model.save_pretrained_gguf("model", tokenizer, quantization_method = "q4_k_m")
if True: model.push_to_hub_gguf("Starxx/LLaMa3-Fine-Tuning-Classical-GGUF", tokenizer, quantization_method = "q4_k_m", token = "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")
```

```
Unsloth: You have 1 CPUs. Using `safe_serialization` is 10x slower.
We shall switch to Pytorch saving, which will take 3 minutes and not 30 minutes.
To force `safe_serialization`, set it to `None` instead.
Unsloth: Kaggle/Colab has limited disk space. We need to delete the downloaded
model which will save 4-16GB of disk space, allowing you to save on Kaggle/Colab.
Unsloth: Will remove a cached repo with size 16.1G
Unsloth: Merging 4bit and LoRA weights to 16bit...
Unsloth: Will use up to 6.25 out of 12.67 RAM for saving.
 47%|          | 15/32 [00:01<00:01, 10.43it/s]We will save to Disk and not RAM now.
100%|          | 32/32 [01:27<00:00,  2.73s/it]
Unsloth: Saving tokenizer... Done.
Unsloth: Saving model... This might take 5 minutes for Llama-7b...
Unsloth: Saving Starxx/LLaMa3-Fine-Tuning-Classical-GGUF/pytorch_model-00001-of-00004.bin...
Unsloth: Saving Starxx/LLaMa3-Fine-Tuning-Classical-GGUF/pytorch_model-00002-of-00004.bin...
Unsloth: Saving Starxx/LLaMa3-Fine-Tuning-Classical-GGUF/pytorch_model-00003-of-00004.bin...
Unsloth: Saving Starxx/LLaMa3-Fine-Tuning-Classical-GGUF/pytorch_model-00004-of-00004.bin...
Done.
NOTICE: llama.cpp GGUF conversion is currently unstable, since llama.cpp is
undergoing some major bug fixes as at 5th of May 2024. This is not an Unsloth issue.
Please be patient - GGUF saving should still work, but might not work as well.
Unsloth: Converting llama model. Can use fast conversion = True.
==((====))==  Unsloth: Conversion from QLoRA to GGUF information
   \\   /|    [0] Installing llama.cpp will take 3 minutes.
O^O/ \_/ \    [1] Converting HF to GUUF 16bits will take 3 minutes.
\        /    [2] Converting GGUF 16bits to q4_k_m will take 20 minutes.
 "-____-"     In total, you will have to wait around 26 minutes.

Unsloth: [0] Installing llama.cpp. This will take 3 minutes...
Unsloth: [1] Converting model at Starxx/LLaMa3-Fine-Tuning-Classical-GGUF into f16 GGUF format.
The output location will be ./Starxx/LLaMa3-Fine-Tuning-Classical-GGUF-unsloth.F16.gguf
This will take 3 minutes...
Unsloth: Conversion completed! Output location: ./Starxx/LLaMa3-Fine-Tuning-Classical-GGUF-unsloth.F16.gguf
Unsloth: [2] Converting GGUF 16bit into q4_k_m. This will take 20 minutes...
main: build = 2792 (ca363260)
main: built with cc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0 for x86_64-linux-gnu
main: quantizing './Starxx/LLaMa3-Fine-Tuning-Classical-GGUF-unsloth.F16.gguf' to './Starxx/LLaMa3-Fine-Tuning-Classical-GGUF-unsloth.Q4_K_M.gguf' as Q4_K_M using 4 threads
llama_model_loader: loaded meta data with 22 key-value pairs and 291 tensors from ./Starxx/LLaMa3-Fine-Tuning-Classical-GGUF-unsloth.F16.gguf (version GGUF V3 (latest))
llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not apply in this output.
llama_model_loader: - kv   0:                       general.architecture str              = llama
llama_model_loader: - kv   1:                               general.name str              = Starxx
llama_model_loader: - kv   2:                           llama.vocab_size u32              = 128256
llama_model_loader: - kv   3:                       llama.context_length u32              = 8192
llama_model_loader: - kv   4:                     llama.embedding_length u32              = 4096
llama_model_loader: - kv   5:                          llama.block_count u32              = 32
llama_model_loader: - kv   6:                  llama.feed_forward_length u32              = 14336
llama_model_loader: - kv   7:                 llama.rope.dimension_count u32              = 128
llama_model_loader: - kv   8:                 llama.attention.head_count u32              = 32
llama_model_loader: - kv   9:              llama.attention.head_count_kv u32              = 8
llama_model_loader: - kv  10:     llama.attention.layer_norm_rms_epsilon f32              = 0.000010
llama_model_loader: - kv  11:                       llama.rope.freq_base f32              = 500000.000000
llama_model_loader: - kv  12:                          general.file_type u32              = 1
llama_model_loader: - kv  13:                       tokenizer.ggml.model str              = gpt2
llama_model_loader: - kv  14:                      tokenizer.ggml.tokens arr[str,128256]  = ["!", "\"", "#", "$", "%", "&", "'", ...
llama_model_loader: - kv  15:                      tokenizer.ggml.scores arr[f32,128256]  = [0.000000, 0.000000, 0.000000, 0.0000...
```

```
# Save to q4_k_m GGUF
# if False: model.save_pretrained_gguf("model", tokenizer, quantization_method = "q4_k_m")
if True: model.push_to_hub_gguf("Starxx/LLaMa3-Fine-Tuning-Classical-GGUF", tokenizer, quantization_method = "q4_k_m", token = "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")
```

```
[ 244/ 291]                blk.26.ffn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 245/ 291]                  blk.27.attn_q.weight - [ 4096,  4096,     1,     1], type =    f16, converting to q4_K .. size =    32.00 MiB ->     9.00 MiB
[ 246/ 291]                  blk.27.attn_k.weight - [ 4096,  1024,     1,     1], type =    f16, converting to q4_K .. size =     8.00 MiB ->     2.25 MiB
[ 247/ 291]                  blk.27.attn_v.weight - [ 4096,  1024,     1,     1], type =    f16, converting to q6_K .. size =     8.00 MiB ->     3.28 MiB
[ 248/ 291]             blk.27.attn_output.weight - [ 4096,  4096,     1,     1], type =    f16, converting to q4_K .. size =    32.00 MiB ->     9.00 MiB
[ 249/ 291]                blk.27.ffn_gate.weight - [ 4096, 14336,     1,     1], type =    f16, converting to q4_K .. size =   112.00 MiB ->    31.50 MiB
[ 250/ 291]                  blk.27.ffn_up.weight - [ 4096, 14336,     1,     1], type =    f16, converting to q4_K .. size =   112.00 MiB ->    31.50 MiB
[ 251/ 291]                blk.27.ffn_down.weight - [14336,  4096,     1,     1], type =    f16, converting to q6_K .. size =   112.00 MiB ->    45.94 MiB
[ 252/ 291]               blk.27.attn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 253/ 291]                blk.27.ffn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 254/ 291]                  blk.28.attn_q.weight - [ 4096,  4096,     1,     1], type =    f16, converting to q4_K .. size =    32.00 MiB ->     9.00 MiB
[ 255/ 291]                  blk.28.attn_k.weight - [ 4096,  1024,     1,     1], type =    f16, converting to q4_K .. size =     8.00 MiB ->     2.25 MiB
[ 256/ 291]                  blk.28.attn_v.weight - [ 4096,  1024,     1,     1], type =    f16, converting to q6_K .. size =     8.00 MiB ->     3.28 MiB
[ 257/ 291]             blk.28.attn_output.weight - [ 4096,  4096,     1,     1], type =    f16, converting to q4_K .. size =    32.00 MiB ->     9.00 MiB
[ 258/ 291]                blk.28.ffn_gate.weight - [ 4096, 14336,     1,     1], type =    f16, converting to q4_K .. size =   112.00 MiB ->    31.50 MiB
[ 259/ 291]                  blk.28.ffn_up.weight - [ 4096, 14336,     1,     1], type =    f16, converting to q4_K .. size =   112.00 MiB ->    31.50 MiB
[ 260/ 291]                blk.28.ffn_down.weight - [14336,  4096,     1,     1], type =    f16, converting to q6_K .. size =   112.00 MiB ->    45.94 MiB
[ 261/ 291]               blk.28.attn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 262/ 291]                blk.28.ffn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 263/ 291]                  blk.29.attn_q.weight - [ 4096,  4096,     1,     1], type =    f16, converting to q4_K .. size =    32.00 MiB ->     9.00 MiB
[ 264/ 291]                  blk.29.attn_k.weight - [ 4096,  1024,     1,     1], type =    f16, converting to q4_K .. size =     8.00 MiB ->     2.25 MiB
[ 265/ 291]                  blk.29.attn_v.weight - [ 4096,  1024,     1,     1], type =    f16, converting to q6_K .. size =     8.00 MiB ->     3.28 MiB
[ 266/ 291]             blk.29.attn_output.weight - [ 4096,  4096,     1,     1], type =    f16, converting to q4_K .. size =    32.00 MiB ->     9.00 MiB
[ 267/ 291]                blk.29.ffn_gate.weight - [ 4096, 14336,     1,     1], type =    f16, converting to q4_K .. size =   112.00 MiB ->    31.50 MiB
[ 268/ 291]                  blk.29.ffn_up.weight - [ 4096, 14336,     1,     1], type =    f16, converting to q4_K .. size =   112.00 MiB ->    31.50 MiB
[ 269/ 291]                blk.29.ffn_down.weight - [14336,  4096,     1,     1], type =    f16, converting to q6_K .. size =   112.00 MiB ->    45.94 MiB
[ 270/ 291]               blk.29.attn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 271/ 291]                blk.29.ffn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 272/ 291]                  blk.30.attn_q.weight - [ 4096,  4096,     1,     1], type =    f16, converting to q4_K .. size =    32.00 MiB ->     9.00 MiB
[ 273/ 291]                  blk.30.attn_k.weight - [ 4096,  1024,     1,     1], type =    f16, converting to q4_K .. size =     8.00 MiB ->     2.25 MiB
[ 274/ 291]                  blk.30.attn_v.weight - [ 4096,  1024,     1,     1], type =    f16, converting to q6_K .. size =     8.00 MiB ->     3.28 MiB
[ 275/ 291]             blk.30.attn_output.weight - [ 4096,  4096,     1,     1], type =    f16, converting to q4_K .. size =    32.00 MiB ->     9.00 MiB
[ 276/ 291]                blk.30.ffn_gate.weight - [ 4096, 14336,     1,     1], type =    f16, converting to q4_K .. size =   112.00 MiB ->    31.50 MiB
[ 277/ 291]                  blk.30.ffn_up.weight - [ 4096, 14336,     1,     1], type =    f16, converting to q4_K .. size =   112.00 MiB ->    31.50 MiB
[ 278/ 291]                blk.30.ffn_down.weight - [14336,  4096,     1,     1], type =    f16, converting to q6_K .. size =   112.00 MiB ->    45.94 MiB
[ 279/ 291]               blk.30.attn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 280/ 291]                blk.30.ffn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 281/ 291]                  blk.31.attn_q.weight - [ 4096,  4096,     1,     1], type =    f16, converting to q4_K .. size =    32.00 MiB ->     9.00 MiB
[ 282/ 291]                  blk.31.attn_k.weight - [ 4096,  1024,     1,     1], type =    f16, converting to q4_K .. size =     8.00 MiB ->     2.25 MiB
[ 283/ 291]                  blk.31.attn_v.weight - [ 4096,  1024,     1,     1], type =    f16, converting to q6_K .. size =     8.00 MiB ->     3.28 MiB
[ 284/ 291]             blk.31.attn_output.weight - [ 4096,  4096,     1,     1], type =    f16, converting to q4_K .. size =    32.00 MiB ->     9.00 MiB
[ 285/ 291]                blk.31.ffn_gate.weight - [ 4096, 14336,     1,     1], type =    f16, converting to q4_K .. size =   112.00 MiB ->    31.50 MiB
[ 286/ 291]                  blk.31.ffn_up.weight - [ 4096, 14336,     1,     1], type =    f16, converting to q4_K .. size =   112.00 MiB ->    31.50 MiB
[ 287/ 291]                blk.31.ffn_down.weight - [14336,  4096,     1,     1], type =    f16, converting to q6_K .. size =   112.00 MiB ->    45.94 MiB
[ 288/ 291]               blk.31.attn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 289/ 291]                blk.31.ffn_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 290/ 291]                    output_norm.weight - [ 4096,     1,     1,     1], type =    f32, size =    0.016 MB
[ 291/ 291]                      output.weight - [ 4096, 128256,     1,     1], type =    f16, converting to q6_K .. size =  1002.00 MiB ->   410.98 MiB
llama_model_quantize_internal: model size  = 15317.02 MB
llama_model_quantize_internal: quant size  =  4685.30 MB
Unsloth: Conversion completed! Output location: ./Starxx/LLaMa3-Fine-Tuning-Classical-GGUF-unsloth.Q4_K_M.gguf
Unsloth: Uploading GGUF to Huggingface Hub...
LLaMa3-Fine-Tuning-Classical-GGUF-unsloth.Q4_K_M.gguf: 100%|          | 4.92G/4.92G [01:51<00:00, 47.8MB/s]
Saved GGUF to https://huggingface.co/Starxx/LLaMa3-Fine-Tuning-Classical-GGUF
```

```
# Save to q4_k_m GGUF
# if False: model.save_pretrained_gguf("model", tokenizer, quantization_method =
"q4_k_m")
if True: model.push_to_hub_gguf("Starxx/LLaMa3-Fine-Tuning-Classical-GGUF",
tokenizer, quantization_method = "q4_k_m", token =
"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")
```