```
# hf_MRpkyxxxxxxEXAfnwxxgnnDB
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
# ✅ Install Required Libraries
!pip install -q bitsandbytes accelerate transformers datasets peft huggingface_hub
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.0/67.0 MB 11.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.5/491.5 kB 34.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 11.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.5/143.5 kB 14.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 125.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 95.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 58.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 13.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 75.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 17.2 MB/s eta 0:00:00
```

```
# ✅ Login to Hugging Face (to push model to Hub)
from huggingface_hub import notebook_login
notebook_login()
```

```
import pandas as pd

df = pd.read_csv("/content/drive/MyDrive/llm_codegen/CodeAlpaca-20k dataset.csv")
df.head()
```

	instruction	input	output
0	What are the distinct values from the given list?	dataList = [3, 9, 3, 5, 7, 9, 5]	The distinct values from the given list are 3,...
1	How would you order a sequence of letters alph...	A, B, C, D	The sequence of letters ordered alphabetically...
2	Write a JavaScript code to loop over all eleme...	numbersArray = [45, 6, 23, 12, 35]	for(let i = 0; i < numbersArray.length; i++) {...
3	Write a Python function to calculate the facto...	NaN	def factorial(number):\n fact = 1\n for ...
4	What would be the output of the following Java...	let area = 6 * 5;\nlet radius = area / 3.14;	The output of the JavaScript snippet is the ra...

```
from transformers import AutoModelForCausalLM, AutoTokenizer, BitsAndBytesConfig
```

```
model_id = "bigcode/starcoderbase-1b"
```

```
# 4-bit quantization config (for QLoRA)
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype="float16"
)
```

```
# Load model and tokenizer
model = AutoModelForCausalLM.from_pretrained(model_id, quantization_config=bnb_config, device_map="auto")
tokenizer = AutoTokenizer.from_pretrained(model_id, trust_remote_code=True)
```

```
config.json: 100%  1.05k/1.05k [00:00<00:00, 82.2kB/s]
model.safetensors: 100%  4.55G/4.55G [00:29<00:00, 66.6MB/s]
generation_config.json: 100%  111/111 [00:00<00:00, 11.6kB/s]
tokenizer_config.json: 100%  677/677 [00:00<00:00, 74.1kB/s]
vocab.json: 100%  777k/777k [00:00<00:00, 10.1MB/s]
merges.txt: 100%  442k/442k [00:00<00:00, 993kB/s]
tokenizer.json: 100%  2.06M/2.06M [00:00<00:00, 6.48MB/s]
special_tokens_map.json: 100%  532/532 [00:00<00:00, 59.8kB/s]
```

```
# Ensure padding is handled
tokenizer.pad_token = tokenizer.eos_token
model.config.pad_token_id = tokenizer.eos_token_id
```

```
!pip install -q peft

from peft import prepare_model_for_kbit_training, LoraConfig, get_peft_model
import torch # Import torch to print model structure

# Prepare for QLoRA
model = prepare_model_for_kbit_training(model)

# --- Add this section to inspect model modules ---
print("Model modules:")
for name, module in model.named_modules():
    print(name)
# Look for module names that correspond to 'q_proj' and 'v_proj' in other models.
# Common names might be 'c_attn' for QKV combined, or separate 'c_proj' or similar.
# Based on the starcoder architecture, it uses 'c_attn' for the attention layer
# which includes QKV, and 'c_proj' for the output projection.
# We likely want to target 'c_attn' for LoRA.
# ------------------------------------------------


# Configure LoRA
lora_config = LoraConfig(
    r=8,
    lora_alpha=32,
    # Update target_modules based on the inspection above.
    # For starcoderbase, 'c_attn' is a common target for LoRA.
    target_modules=["c_attn"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)

# Inject adapters
model = get_peft_model(model, lora_config)

# Show trainable parameters (should be small)
def print_trainable_parameters(model):
    trainable = sum(p.numel() for p in model.parameters() if p.requires_grad)
    total = sum(p.numel() for p in model.parameters())
    print(f"Trainable params: {trainable} / {total} ({100 * trainable / total:.2f}%)")

print_trainable_parameters(model)
```

```
transformer.h.21.attn.attn_dropout
transformer.h.21.attn.resid_dropout
transformer.h.21.ln_2
transformer.h.21.mlp
transformer.h.21.mlp.c_fc
transformer.h.21.mlp.c_proj
transformer.h.21.mlp.act
transformer.h.21.mlp.dropout
transformer.h.22
transformer.h.22.ln_1
transformer.h.22.attn
transformer.h.22.attn.c_attn
transformer.h.22.attn.c_proj
transformer.h.22.attn.attn_dropout
transformer.h.22.attn.resid_dropout
transformer.h.22.ln_2
transformer.h.22.mlp
transformer.h.22.mlp.c_fc
transformer.h.22.mlp.c_proj
transformer.h.22.mlp.act
transformer.h.22.mlp.dropout
transformer.h.23
transformer.h.23.ln_1
transformer.h.23.attn
transformer.h.23.attn.c_attn
transformer.h.23.attn.c_proj
transformer.h.23.attn.attn_dropout
transformer.h.23.attn.resid_dropout
transformer.h.23.ln_2
transformer.h.23.mlp
transformer.h.23.mlp.c_fc
transformer.h.23.mlp.c_proj
transformer.h.23.mlp.act
transformer.h.23.mlp.dropout
transformer.ln_f
lm_head
Trainable params: 835584 / 628434944 (0.13%)
```

```
# Show trainable parameters (should be small)
def print_trainable_parameters(model):
    trainable = sum(p.numel() for p in model.parameters() if p.requires_grad)
    total = sum(p.numel() for p in model.parameters())
    print(f"Trainable params: {trainable} / {total} ({100 * trainable / total:.2f}%)")

print_trainable_parameters(model)
```

```
Trainable params: 835584 / 628434944 (0.13%)
```

Stage 4

```
from datasets import Dataset

# Convert pandas DataFrame to Hugging Face Dataset
hf_dataset = Dataset.from_pandas(df)
hf_dataset = hf_dataset.remove_columns(["__index_level_0__"]) if "__index_level_0__" in hf_dataset.column_names else hf_datas
```

```
def format_prompt(example):
    instruction = example["instruction"]
    input_text = example["input"]
    output = example["output"]

    # Check if input_text is None or empty before stripping
    if input_text is not None and str(input_text).strip(): # Ensure input_text is treated as a string before stripping
        prompt = f"### Instruction:\n{instruction}\n\n### Input:\n{input_text}\n\n### Response:\n"
    else:
        prompt = f"### Instruction:\n{instruction}\n\n### Response:\n"

    # Check if output is None and handle it
    if output is None:
        output_str = "" # Or some other default string like "[No Output Provided]"
    else:
        output_str = str(output) # Ensure output is treated as a string

    full_prompt = prompt + output_str # Concatenate with the handled output string
    return {"text": full_prompt}


# Format prompts
hf_dataset = hf_dataset.map(format_prompt)
```

```
Map: 100% 2017/2017 [00:00<00:00, 8217.89 examples/s]
```

```
def tokenize(example):
    return tokenizer(
        example["text"],
        truncation=True,
        padding="max_length",
        max_length=512
    )

tokenized_dataset = hf_dataset.map(tokenize, remove_columns=["instruction", "input", "output", "text"])
```

```
Map: 100% 2017/2017 [00:02<00:00, 1139.00 examples/s]
```

```
from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir="/content/drive/MyDrive/llm_codegen/output",
    per_device_train_batch_size=4,
    gradient_accumulation_steps=4,
    learning_rate=2e-4,
    num_train_epochs=1,
    fp16=True,
    logging_steps=10,
    save_strategy="epoch",
    report_to="none",
)
```

```
from transformers import Trainer, DataCollatorForLanguageModeling

data_collator = DataCollatorForLanguageModeling(tokenizer, mlm=False)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    data_collator=data_collator,
)
trainer.train()
```

```
/usr/local/lib/python3.11/dist-packages/torch/_dynamo/eval_frame.py:632: UserWarning: torch.utils.checkpoint: the use_reentrant
  return fn(*args, **kwargs)
```

[126/126 09:28, Epoch 0/1]

Step	Training Loss
10	1.249800
20	1.087800
30	0.951100
40	0.896500
50	0.861500
60	0.891500
70	0.817700
80	0.869400
90	0.792200
100	0.835000
110	0.790200
120	0.798100

```
TrainOutput(global_step=126, training_loss=0.8975822433592782, metrics={'train_runtime': 574.9232, 'train_samples_per_second':
3.508, 'train_steps_per_second': 0.219, 'total_flos': 6320745596583936.0, 'train_loss': 0.8975822433592782, 'epoch':
0.998019801980198})
```

```
from peft import PeftModel
import os # Import the os module

output_dir = "/content/drive/MyDrive/llm_codegen/output"

# Find the latest checkpoint directory
```

```
# This assumes checkpoint directories follow the naming convention 'checkpoint-XXXX'
checkpoints = [d for d in os.listdir(output_dir) if os.path.isdir(os.path.join(output_dir, d)) and d.startswith("checkpoint-"

# Sort checkpoints by name to get the latest one
checkpoints.sort()

if not checkpoints:
    raise FileNotFoundError(f"No checkpoint directories found in {output_dir}")

latest_checkpoint_dir = os.path.join(output_dir, checkpoints[-1])

print(f"Loading adapter from: {latest_checkpoint_dir}")

# Load your fine-tuned adapter model from the specific checkpoint folder
merged_model = PeftModel.from_pretrained(model, latest_checkpoint_dir)
merged_model = merged_model.merge_and_unload()

# Save the merged model to a new directory
save_dir = "/content/drive/MyDrive/llm_codegen/merged_model"
merged_model.save_pretrained(save_dir)
tokenizer.save_pretrained(save_dir)

print(f"Merged model saved to: {save_dir}")
```

```
dapter from: /content/drive/MyDrive/llm_codegen/output/checkpoint-126
l/lib/python3.11/dist-packages/peft/peft_model.py:599: UserWarning: Found missing adapter keys while loading the checkpoint: ['
s.warn(f"Found missing adapter keys while loading the checkpoint: {missing_keys}")
l/lib/python3.11/dist-packages/peft/tuners/lora/bnb.py:355: UserWarning: Merge lora module to 4-bit linear may get different ge
s.warn(
del saved to: /content/drive/MyDrive/llm_codegen/merged_model
```

```
from huggingface_hub import notebook_login
notebook_login()
```

```
merged_model.push_to_hub("key-life/codegen-alpaca-1b")
tokenizer.push_to_hub("key-life/codegen-alpaca-1b")
```

```
README.md: 100%  24.0/24.0 [00:00<00:00, 436B/s]

adapter_model.safetensors: 100%  40.0/40.0 [00:00<00:00, 380B/s]

CommitInfo(commit_url='https://huggingface.co/key-life/codegen-alpaca-1b/commit/4dc5045a0db8537026d472908105044299098c6c',
commit_message='Upload tokenizer', commit_description='', oid='4dc5045a0db8537026d472908105044299098c6c', pr_url=None,
repo_url=RepoUrl('https://huggingface.co/key-life/codegen-alpaca-1b', endpoint='https://huggingface.co', repo_type='model',
repo_id='key-life/codegen-alpaca-1b'), pr_revision=None, pr_num=None)
```

Testing

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

model_id = "key-life/codegen-alpaca-1b"  # your HF model repo

tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(model_id, device_map="auto")

prompt = "### Instruction:\nWrite a Python function to check if a number is prime.\n\n### Response:\n"
inputs = tokenizer(prompt, return_tensors="pt").to(model.device)

outputs = model.generate(**inputs, max_new_tokens=128)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

```
tokenizer_config.json: 100%  4.14k/4.14k [00:00<00:00, 241kB/s]
vocab.json: 100%  777k/777k [00:00<00:00, 5.87MB/s]
merges.txt: 100%  442k/442k [00:00<00:00, 8.83MB/s]
tokenizer.json: 100%  3.48M/3.48M [00:00<00:00, 28.4MB/s]
special_tokens_map.json: 100%  906/906 [00:00<00:00, 45.9kB/s]
adapter_config.json: 100%  706/706 [00:00<00:00, 30.2kB/s]
adapter_model.safetensors: 100%  40.0/40.0 [00:00<00:00, 919B/s]
Loading adapter weights from key-life/codegen-alpaca-1b led to missing keys in the model: transformer.h.0.attn.c_attn.lora_A.de
Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.
### Instruction:
Write a Python function to check if a number is prime.

### Response:
```python
def is_prime(n):
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for i in range(3, int(n ** 0.5) + 1, 2):
        if n % i == 0:
            return False
    return True
```

Solution:
```python
def is_prime(n):
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for i in range(3, int(n ** 0.
```
```