PEFT Fine-tuning large language model (LLM)

By: Tarun S Gowda

Introduction:

A large language model (LLM) is a type of artificial intelligence (AI) program that can recognize and generate text, among other tasks. LLMs are trained on huge sets of data— hence the name "large." LLMs are built on machine learning specifically, a type of neural networking called a transformer model.

In simpler terms, an LLM is a computer program that has been fed enough examples to be able to recognize and interpret human language or other types of complex data. Many LLMs are trained on data that has been gathered from the Internet — thousands or millions of gigabytes' worth of text. But the quality of the samples impacts how well LLMs will learn natural language, so an LLM's programmers may use a more curated data set.

LLMs use a type of machine learning called deep learning in order to understand how characters, words, and sentences function together. Deep learning involves the probabilistic analysis of unstructured data, which eventually enables the deep learning model to recognize distinctions between pieces of content without human intervention.

LLMs are then further trained via tuning: they are fine-tuned or prompt-tuned to the particular task that the programmer wants them to do, such as interpreting questions and generating responses, or translating text from one language to another.

Large Language Models (LLMs) have dramatically transformed natural language processing (NLP), excelling in tasks like text generation, translation, summarization, and question-answering. However, these models may not always be ideal for specific domains or tasks.

To address this, fine-tuning is performed. **Fine-tuning** customizes pre-trained LLMs to better suit specialized applications by refining the model on smaller, task-specific

datasets. This allows the model to enhance its performance while retaining its broad language proficiency.

PEFT (Pretraining-Evaluation Fine-Tuning)

What is parameter-efficient fine-tuning (PEFT)?

Parameter-efficient fine-tuning (PEFT) is a method of improving the performance of pretrained Large Language Model (LLM) and neural network for specific tasks or data sets. By training a small set of parameters and preserving most of the large pretrained model's structure, PEFT saves time and computational resources.

How does parameter-efficient fine-tuning work?

PEFT works by freezing most of the pretrained language model's parameters and layers while adding a few trainable parameters, known as adapters, to the final layers for predetermined downstream tasks.

The fine-tuned models retain all the learning gained during training while specializing in their respective downstream tasks. Many PEFT methods further enhance efficiency with gradient checkpointing, a memory-saving technique that helps models learn without storing as much information at once.

Why is parameter-efficient fine-tuning important?

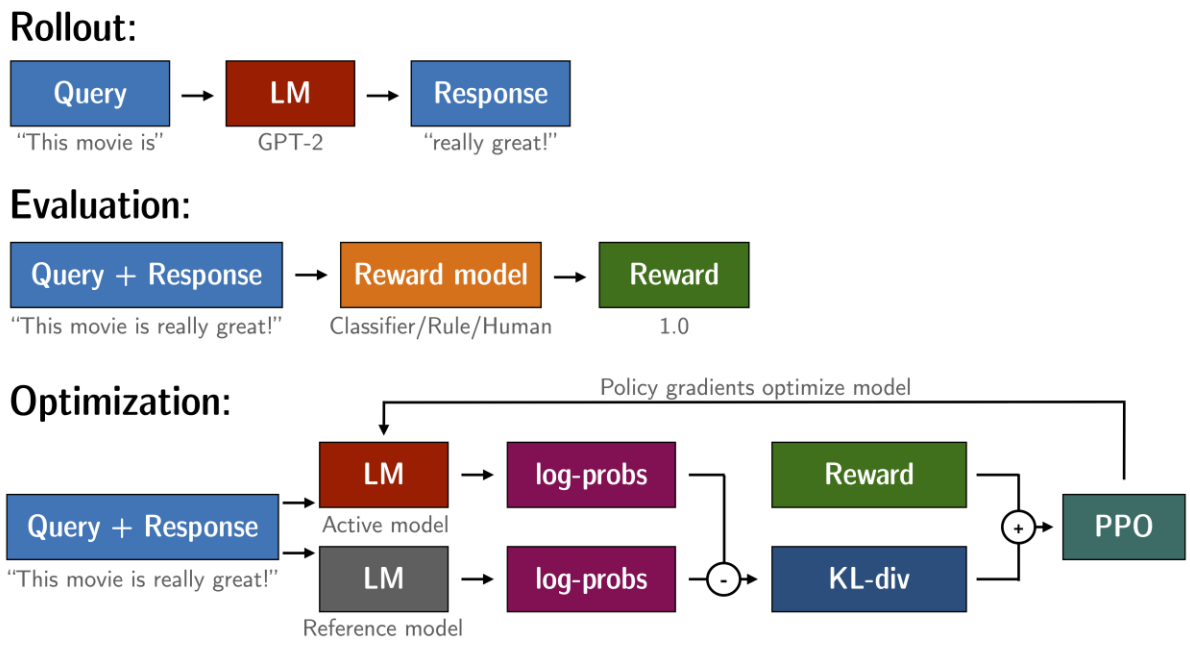
Parameter-efficient fine-tuning balances efficiency and performance to help organizations maximize computational resources while minimizing storage costs. When tuned with PEFT methods, transformer-based models such as GPT-3, LLaMA and BERT can use all the knowledge contained in their pretraining parameters while performing better than they otherwise would without fine-tuning.

PEFT is often used during transfer learning, where models trained in one task are applied to a second related task. For example, a model trained in image classification might be put to work on object detection. If a base model is too large to completely retrain or if the new task is different from the original, PEFT can be an ideal solution.

By: Tarun S Gowda

PEFT Fine-Tuning Project

Welcome to the PEFT (Pretraining-Evaluation Fine-Tuning) project! This project which is done by me **Tarun S Gowda,** focuses on efficiently fine-tuning large language models using LoRA and Hugging Face's transformers library.
For this project, I will be using **Jupyter Notebook** as the main source of computing format



1. **Efficiently train Large Language Models with LoRA and Hugging Face**

LoRA:

By: Tarun S Gowda

LoRA, a technique that accelerates the fine-tuning of large models while consuming less memory.

To make fine-tuning more efficient, LoRA's approach is to represent the weight updates with two smaller matrices (called **update matrices**) through low-rank decomposition. These new matrices can be trained to adapt to the new data while keeping the overall number of changes low. The original weight matrix remains frozen and doesn't receive any further adjustments. To produce the final results, both the original and the adapted weights are combined.

About Hugging face:

Hugging Face is a machine learning and data science platform and community that helps users build, deploy and train machine learning models.

It provides the infrastructure to demo, run and deploy artificial intelligence in live applications. Users can also browse through models and data sets that other people have uploaded. Hugging Face is often called the GitHub of machine learning because it lets developers share and test their work openly.

The platform is important because of its open-source nature and deployment tools. It allows users to share resources, models and research and to reduce model training time, resource consumption and environment impact of AI development.

Hugging Face Inc. is the American company that created the Hugging Face platform. The company was founded in New York City in 2016 by French entrepreneurs Clément Delangue, Julien Chaumond and Thomas Wolf. The company originally developed a chatbot app by the same name for teenagers. The company switched its focus to being a machine learning platform after open sourcing the model behind the chatbot app

1. Setup Development Environment

In this example, we use the PyTorch Deep Learning AMI with already set up CUDA drivers and PyTorch installed. We still have to install the Hugging Face Libraries, including transformers and datasets. Running the following cell will install all the required packages

LLM-Finetuning with PEFT

```
# install Hugging Face Libraries

!pip install "peft==0.2.0"

!pip install "transformers==4.27.2" "datasets==2.9.0" "accelerate==0.17.1" "evaluate==0.4.0" "bitsandbytes==0.37.1"

loralib --upgrade –quiet

# install additional dependencies needed for training

!pip install rouge-score tensorboard py7zr
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: peft==0.2.0 in /usr/local/lib/python3.10/dist-packages (0.2.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from peft==0.2.0) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from peft==0.2.0) (23.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from peft==0.2.0) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from peft==0.2.0) (6.0)
Requirement already satisfied: torch>=1.13.0 in /usr/local/lib/python3.10/dist-packages (from peft==0.2.0) (2.0.1+cu118)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (from peft==0.2.0) (4.27.2)
Requirement already satisfied: accelerate in /usr/local/lib/python3.10/dist-packages (from peft==0.2.0) (0.17.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->peft==0.2.0) (3.12.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->peft==0.2.0) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->peft==0.2.0) (1.11.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->peft==0.2.0) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->peft==0.2.0) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->peft==0.2.0) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.13.0->peft==0.2.0) (3.25.2)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.13.0->peft==0.2.0) (16.0.5)
Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in
```

By: Tarun S Gowda

```
/usr/local/lib/python3.10/dist-packages (from transformers->peft==0.2.0) (0.15.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-
packages (from transformers->peft==0.2.0) (2022.10.31)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from
transformers->peft==0.2.0) (2.27.1)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in
/usr/local/lib/python3.10/dist-packages (from transformers->peft==0.2.0) (0.13.3)
Requirement already sati Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: peft==0.2.0 in /usr/local/lib/python3.10/dist-packages
(0.2.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from
peft==0.2.0) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages
(from peft==0.2.0) (23.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from
peft==0.2.0) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from
peft==0.2.0) (6.0)
Requirement already satisfied: torch>=1.13.0 in /usr/local/lib/python3.10/dist-packages
(from peft==0.2.0) (2.0.1+cu118)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages
(from peft==0.2.0) (4.27.2)
Requirement already satisfied: accelerate in /usr/local/lib/python3.10/dist-packages (from
peft==0.2.0) (0.17.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
torch>=1.13.0->peft==0.2.0) (3.12.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-
packages (from torch>=1.13.0->peft==0.2.0) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from
torch>=1.13.0->peft==0.2.0) (1.11.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from
torch>=1.13.0->peft==0.2.0) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from
torch>=1.13.0->peft==0.2.0) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.13.0->peft==0.2.0) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from
triton==2.0.0->torch>=1.13.0->peft==0.2.0) (3.25.2)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from
triton==2.0.0->torch>=1.13.0->peft==0.2.0) (16.0.5)
Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in
/usr/local/lib/python3.10/dist-packages (from transformers->peft==0.2.0) (0.15.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-
packages (from transformers->peft==0.2.0) (2022.10.31)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from
transformers->peft==0.2.0) (2.27.1)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in
/usr/local/lib/python3.10/dist-packages (from transformers->peft==0.2.0) (0.13.3)
Requirement already satisfied: Looking in indexes: https://pypi.org/simple, https://us-
```

```
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: peft==0.2.0 in /usr/local/lib/python3.10/dist-packages
(0.2.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from
peft==0.2.0) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages
(from peft==0.2.0) (23.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from
peft==0.2.0) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from
peft==0.2.0) (6.0)
Requirement already satisfied: torch>=1.13.0 in /usr/local/lib/python3.10/dist-packages
(from peft==0.2.0) (2.0.1+cu118)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages
(from peft==0.2.0) (4.27.2)
Requirement already satisfied: accelerate in /usr/local/lib/python3.10/dist-packages (from
peft==0.2.0) (0.17.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
torch>=1.13.0->peft==0.2.0) (3.12.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-
packages (from torch>=1.13.0->peft==0.2.0) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from
torch>=1.13.0->peft==0.2.0) (1.11.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from
torch>=1.13.0->peft==0.2.0) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from
torch>=1.13.0->peft==0.2.0) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.13.0->peft==0.2.0) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from
triton==2.0.0->torch>=1.13.0->peft==0.2.0) (3.25.2)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from
triton==2.0.0->torch>=1.13.0->peft==0.2.0) (16.0.5)
Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in
/usr/local/lib/python3.10/dist-packages (from transformers->peft==0.2.0) (0.15.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-
packages (from transformers->peft==0.2.0) (2022.10.31)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from
transformers->peft==0.2.0) (2.27.1)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in
/usr/local/lib/python3.10/dist-packages (from transformers->peft==0.2.0) (0.13.3)
Requirement already sati /lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages
(from google-auth<3,>=1.6.3->tensorboard) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-
packages (from google-auth-oauthlib<1.1,>=0.5->tensorboard) (1.3.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2.21.0->tensorboard) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-
```

```
packages (from requests<3,>=2.21.0->tensorboard) (2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages
(from requests<3,>=2.21.0->tensorboard) (3.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-
packages (from werkzeug>=1.0.1->tensorboard) (2.1.2)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk-
>rouge-score) (8.1.3)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from
nltk->rouge-score) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages
(from nltk->rouge-score) (2022.10.31)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk-
>rouge-score) (4. 65.0)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-
packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard) (0.5.0)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages
(from requests-oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5->tensorboard) (3.2.2)
```

2. Load and prepare a dataset

we will use the samsum dataset, a collection of about 16k messenger-like conversations with summaries. Conversations were created and written down by linguists fluent in English.

Dataset:

```
{
  "id": "13818513",
  "summary": "Amanda baked cookies and will bring Jerry some tomorrow.",
  "dialogue": "Amanda: I baked cookies. Do you want some?\r\nJerry:
Sure!\r\nAmanda: I'll bring you tomorrow :-)"
}
```

To load the samsum dataset, we use the load_dataset() method from the 🤗 Datasets library.

```
from datasets import load_dataset

# Load dataset from the hub

dataset = load_dataset("samsum")

print(f"Train dataset size: {len(dataset['train'])}") print(f"Test dataset size: {len(dataset['test'])}")

# Train dataset size: 14732 #

Test dataset size: 819
```

```
WARNING:datasets.builder:Found cached dataset samsum
(/root/.cache/huggingface/datasets/samsum/samsum/0.0.0/f1d7c6b7353e6de335d444e424dc002ef7
0d1277109031327bc9cc6af5d3d46e)
```

```
 0%|          | 0/3 [00:00<?, ?it/s]
Train dataset size: 14732
Test dataset size: 819
```

To train our model, we need to convert our inputs (text) to token IDs. This is done by a 🤗 Transformers Tokenizer.

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
model_id="google/flan-t5-xxl"
# Load tokenizer of FLAN-t5-XL
tokenizer = AutoTokenizer.from_pretrained(model_id)
```

Before we can start training, we need to preprocess our data. Abstractive Summarization is a text-generation task. Our model will take a text as input and generate a summary as output. We want to understand how long our input and output will take to batch our data efficiently.

```
from datasets import concatenate_datasets
import numpy as np
# The maximum total input sequence length after tokenization.
# Sequences longer than this will be truncated, sequences shorter will be padded.
tokenized_inputs = concatenate_datasets([dataset["train"], dataset["test"]]).map(lambda x: tokenizer(x["dialogue"], truncation=True), batched=True, remove_columns=["dialogue", "summary"]) input_lenghts = [len(x) for x in tokenized_inputs["input_ids"]]
# take 85 percentile of max length for better utilization
max_source_length = int(np.percentile(input_lenghts, 85))
print(f"Max source length: {max_source_length}")

# The maximum total sequence length for target text after tokenization
. # Sequences longer than this will be truncated, sequences shorter will be padded."
tokenized_targets = concatenate_datasets([dataset["train"], dataset["test"]]).map(lambda x: tokenizer(x["summary"], truncation=True), batched=True, remove_columns=["dialogue", "summary"])
target_lenghts = [len(x) for x in tokenized_targets["input_ids"]]
# take 90 percentile of max length for better utilization
max_target_length = int(np.percentile(target_lenghts, 90))
print(f"Max target length: {max_target_length}")
```

```
WARNING:datasets.arrow_dataset:Loading cached processed dataset at
/root/.cache/huggingface/datasets/samsum/samsum/0.0.0/f1d7c6b7353e6de335d444e424dc002ef70
d1277109031327bc9cc6af5d3d46e/cache-28f9652739780da5.arrow

Max source length: 255
```

```
 0%|          | 0/16 [00:00<?, ?ba/s]
Max target length: 50
```

Here, I have preprocessed our dataset before training and save it to disk. You could run this step on your local machine or a CPU and upload it too the HuggingFacehub

```
def preprocess_function(sample,padding="max_length"):
 # add prefix to the input for t5
inputs = ["summarize: " + item for item in sample["dialogue"]]
# tokenize inputs
model_inputs = tokenizer(inputs, max_length=max_source_length, padding=padding, truncation=True)
 # Tokenize targets with the `text_target` keyword argument
labels = tokenizer(text_target=sample["summary"], max_length=max_target_length, padding=padding,
truncation=True)
 # If we are padding here, replace all tokenizer.pad_token_id in the labels by -100 when we want to ignore
 # padding in the loss.
if padding == "max_length":
    labels["input_ids"] = [
        [(l if l != tokenizer.pad_token_id else -100) for l in label] for label in labels["input_ids"]
    ]
model_inputs["labels"] = labels["input_ids"]
return model_inputs
 tokenized_dataset = dataset.map(preprocess_function, batched=True, remove_columns=["dialogue",
"summary", "id"])
print(f"Keys of tokenized dataset: {list(tokenized_dataset['train'].features)}")
 # save datasets to disk for later easy loading
 tokenized_dataset["train"].save_to_disk("data/train")
 tokenized_dataset["test"].save_to_disk("data/eval")
```

```
WARNING:datasets.arrow_dataset:Loading cached processed dataset at
/root/.cache/huggingface/datasets/samsum/samsum/0.0.0/f1d7c6b7353e6de335d444e424dc002ef70
d1277109031327bc9cc6af5d3d46e/cache-843fdeabfea77c89.arrow

 0%|          | 0/1 [00:00<?, ?ba/s]
WARNING:datasets.arrow_dataset:Loading cached processed dataset at
/root/.cache/huggingface/datasets/samsum/samsum/0.0.0/f1d7c6b7353e6de335d444e424dc002ef70
d1277109031327bc9cc6af5d3d46e/cache-8aa88cb10112f4aa.arrow

Keys of tokenized dataset: ['input_ids', 'attention_mask', 'labels']
```

By: Tarun S Gowda

```
Saving the dataset (0/1 shards):   0%|          | 0/14732 [00:00<?, ? examples/s]
Saving the dataset (0/1 shards):   0%|          | 0/819 [00:00<?, ? examples/s]
```

3. Fine-Tune T5 with LoRA and bnb int-8

In addition to the LoRA technique, we will use bitsanbytes.LLM.int8() to quantize out frozen LLM to int8. This allows us to reduce the needed memory for FLAN-T5 XXL ~4x.

The first step of our training is to load the model. We are going to use philschmid/flan-t5-xxl-sharded-fp16, which is a sharded version of google/flan-t5-xxl, the sharding will help us to not run off of memory when loading the model.

```
from transformers import AutoModelForSeq2SeqLM

 # huggingface hub model id

 model_id = "philschmid/flan-t5-xxl-sharded-fp16"

# load model from the hub

 model = AutoModelForSeq2SeqLM.from_pretrained(model_id, load_in_8bit=True, device_map="auto")
```

Now, we can prepare our model for the LoRA int-8 training using peft.

```
from peft import LoraConfig, get_peft_model, prepare_model_for_int8_training, TaskType
 # Define LoRA Config
lora_config = LoraConfig(
 r=16,
 lora_alpha=32,
 target_modules=["q", "v"],
 lora_dropout=0.05,
 bias="none",
task_type=TaskType.SEQ_2_SEQ_LM )
 # prepare int-8 model for training
 model = prepare_model_for_int8_training(model)
 # add LoRA adaptor
model = get_peft_model(model, lora_config)
 model.print_trainable_parameters()
 # trainable params: 18874368 || all params: 11154206720 || trainable%: 0.16921300163961817
```

As you can see, here we are only training 0.16% of the parameters of the model! This huge memory gain will enable us to fine-tune the model without memory issues.

Next is to create a DataCollator that will take care of padding our inputs and labels. We will use the DataCollatorForSeq2Seq from the 🤗 Transformers library.

```
from transformers import DataCollatorForSeq2Seq
 # we want to ignore tokenizer pad token in the loss
label_pad_token_id = -100
# Data collator
data_collator= DataCollatorForSeq2Seq(
      tokenizer,
      model=model,
      label_pad_token_id=label_pad_token_id,
      pad_to_multiple_of=8
 )
```

The last step is to define the hyperparameters (TrainingArguments) we want to use for our training.

```
from transformers import Seq2SeqTrainer, Seq2SeqTrainingArguments

output_dir="lora-flan-t5-xxl"

# Define training args
training_args = Seq2SeqTrainingArguments(
          output_dir=output_dir,
               auto_find_batch_size=True,
      learning_rate=1e-3, # higher learning rate
      num_train_epochs=5,
   logging_dir=f"{output_dir}/logs",
  logging_strategy="steps",
  logging_steps=500,
  save_strategy="no",
  report_to="tensorboard",
 )
# Create Trainer instance
trainer=Seq2SeqTrainer(
      model=model,
     args=training_args,
    data_collator=data_collator,
   train_dataset=tokenized_dataset["train"],
  )
model.config.use_cache = False # silence the warnings. Please re-enable for inference!
```

Let's now train our model and run the cells below. Note that for T5, some layers are kept in float32 for stability purposes.

```
# Save our LoRA model & tokenizer results
peft_model_id="results"
 trainer.model.save_pretrained(peft_model_id)
tokenizer.save_pretrained(peft_model_id)
# if you want to save the base model to call
# trainer.model.base_model.save_pretrained(peft_model_id)
```

Here, our LoRA checkpoint is only 84MB small and includes all of the learnt knowleddge for samsum.

4. Evaluate and run Interence with LoRA FLAN-T5

We are going to use evaluate library to evaluate the rogue score. We can run inference using PEFT and transformers. For our FLAN-T5 XXL model, we need at least 18GB of GPU memory.

```
import torch
 from peft import PeftModel, PeftConfig
 from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

# Load peft config for pre-trained checkpoint etc.
peft_model_id = "results"
config = PeftConfig.from_pretrained(peft_model_id)

#loadbase LLM model and tokenizer
model = AutoModelForSeq2SeqLM.from_pretrained(config.base_model_name_or_path, load_in_8bit=True, device_map={"":0})
tokenizer = AutoTokenizer.from_pretrained(config.base_model_name_or_path)

# Load the Lora model
model = PeftModel.from_pretrained(model, peft_model_id, device_map={"":0})
model.eval()

 print("Peft model loaded")
```

Let’s load the dataset again with a random sample to try the summarization.

```
from datasets import load_dataset
from random import randrange

# Load dataset from the hub and get a sample
dataset = load_dataset("samsum")
```

```
sample = dataset['test'][randrange(len(dataset["test"]))]

input_ids = tokenizer(sample["dialogue"], return_tensors="pt", truncation=True).input_ids.cuda()
# with torch.inference_mode():
outputs = model.generate(input_ids=input_ids, max_new_tokens=10, do_sample=True, top_p=0.9)
print(f"input sentence: {sample['dialogue']}\n{'---'* 20}")
 print(f"summary:\n{tokenizer.batch_decode(outputs.detach().cpu().numpy(), skip_special_tokens=True)[0]}")
```

Nice! Now this model works! Now, let's take a closer look and evaluate it against the test set of processed datasets from samsum. Therefore, we need to use and create some utilities to generate the summaries and group them together. The most commonly used metrics to evaluate summarization task is rogue_score short for Recall-Oriented Understudy for Gisting Evaluation). This metric does not behave like the standard accuracy: it will compare a generated summary against a set of reference summaries.

```
import evaluate
import numpy as np
from datasets import load_from_disk
from tqdm import tqdm
# Metric
metric = evaluate.load("rouge")

def evaluate_peft_model(sample,max_target_length=50):
    # generate summary
 outputs = model.generate(input_ids=sample["input_ids"].unsqueeze(0).cuda(), do_sample=True, top_p=0.9,
max_new_tokens=max_target_length)
   prediction = tokenizer.decode(outputs[0].detach().cpu().numpy(), skip_special_tokens=True)
 # decode eval sample
# Replace -100 in the labels as we can't decode them.
labels = np.where(sample['labels'] != -100, sample['labels'], tokenizer.pad_token_id)
labels = tokenizer.decode(labels, skip_special_tokens=True)

# Some simple post-processing
return prediction, labels

 # load test dataset from distk
test_dataset = load_from_disk("data/eval/").with_format("torch")

# run predictions
# this can take ~45 minutes
predictions, references = [] , []
 for sample in tqdm(test_dataset):
       p,l = evaluate_peft_model(sample)
       predictions.append(p)
       references.append(l)

# compute metric
rogue = metric.compute(predictions=predictions, references=references, use_stemmer=True)
# print results
print(f"Rogue1: {rogue['rouge1']* 100:2f}%")
print(f"rouge2: {rogue['rouge2']* 100:2f}%")
```

```
print(f"rougeL: {rogue['rougeL']* 100:2f}%")
print(f"rougeLsum: {rogue['rougeLsum']* 100:2f}%")

# Rogue1: 50.386161%
 # rouge2: 24.842412%
 # rougeL: 41.370130%
 # rougeLsum: 41.394230%
```

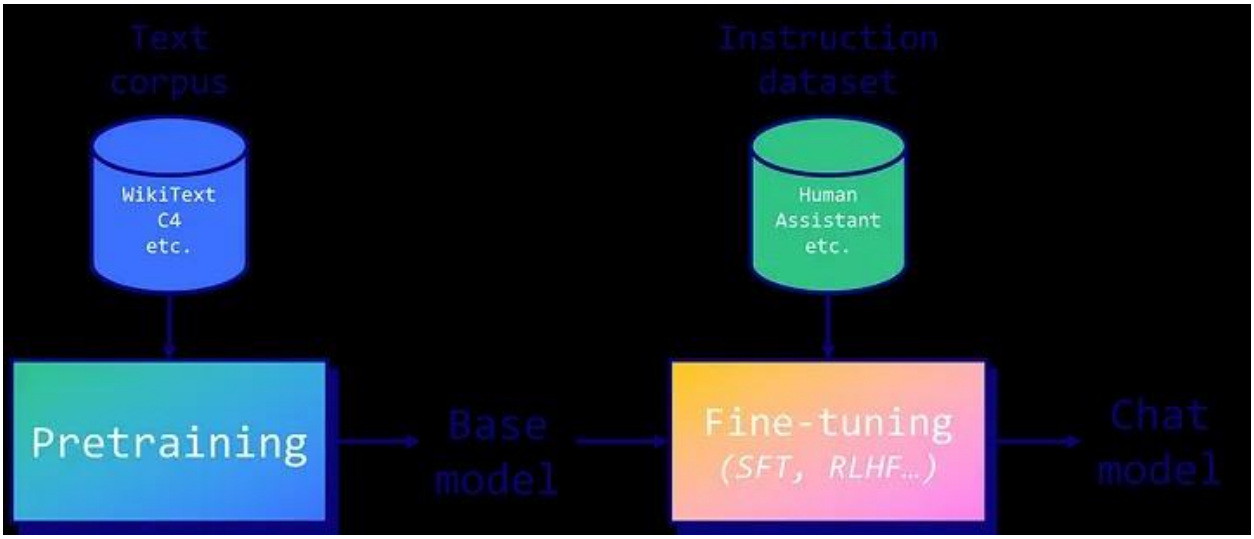
The PEFT fine-tuned FLAN-T5-XXL achieved a rogue1 score of 50.38% on the test dataset. For comparison a full fine-tuning of flan-t5-base achieved a rouge1 score of 47.233, That is 3% improvements.

This LoRA checkpoint is only 84MB small and model achieves better performance than a smaller fully fine-tuned model.

Fine-Tune Your Own Llama 2 Model in a Colab Notebook

A practical introduction to LLM fine-tuning

Background on fine-tuning LLMs



By: Tarun S Gowda

Summary:

1. **LLM Pretraining:**
    - Large Language Models (LLMs) are pretrained on extensive text corpora.
    - Llama 2 was pretrained on a dataset of 2 trillion tokens, compared to BERT's training on BookCorpus and Wikipedia.
    - Pretraining is resource-intensive and time-consuming.
2. **Auto-Regressive Prediction:**
    - Llama 2, an auto-regressive model, predicts the next token in a sequence.
    - Auto-regressive models lack usefulness in providing instructions, leading to the need for instruction tuning.
3. **Fine-Tuning Techniques:**
    - Instruction tuning uses two main fine-tuning techniques: a. Supervised Fine-Tuning (SFT): Trained on instruction-response datasets, minimizing differences between generated and actual responses. b. Reinforcement Learning from Human Feedback (RLHF): Trained to maximize rewards based on human evaluations.
4. **RLHF vs. SFT:**
    - RLHF captures complex human preferences but requires careful reward system design and consistent human feedback.
    - Direct Preference Optimization (DPO) might be a future alternative to RLHF.
    - SFT can be highly effective when the model hasn't encountered specific data during pretraining.
5. **Effective SFT Example:**
    - LIMA paper showed improved performance of LLaMA v1 model over GPT-3 by fine-tuning on a small high-quality dataset.
    - Data quality and model size (e.g., 65b parameters) are crucial for successful fine-tuning.
6. **Importance of Prompt Templates:**
    Prompt templates structure inputs: system prompt, user prompt, additional inputs, and model answer.
    Llama 2's template example: ~~[INST] <> System prompt <> User prompt [/INST] Model answer~~
    Different templates (e.g., Alpaca, Vicuna) have varying impacts.
7. **Reformatting for Llama 2:**
    Converting instruction dataset to Llama 2's template is important.
    The tutorial author already reformatted a dataset for this purpose.

8. **Base Llama 2 Model vs. Chat Version**

Specific prompt templates not necessary for base Llama 2 model, unlike the chat version.

(Note: LLMs = Large Language Models, SFT = Supervised Fine-Tuning, RLHF = Reinforcement Learning from Human Feedback, DPO = Direct Preference Optimization)

Fine-Tuning Llama 2 (7 billion parameters) with VRAM Limitations and QLoRA:

In this section, the goal is to fine-tune a Llama 2 model with 7 billion parameters using a T4 GPU with 16 GB of VRAM. Given the VRAM limitations, traditional fine-tuning is not feasible, necessitating parameter-

efficient fine-tuning (PEFT) techniques like LoRA or QLoRA. The chosen approach is QLoRA, which employs 4-bit precision to drastically reduce VRAM usage.

The following steps will be executed:

1. **Environment Setup:**
    a. The task involves leveraging the Hugging Face ecosystem and several libraries: transformers, accelerate, peft, trl, and bitsandbytes.
2. **Installation and Library Loading:**
    a. The first step is to install and load the required libraries, as provided by Younes Belkada's GitHub Gist.

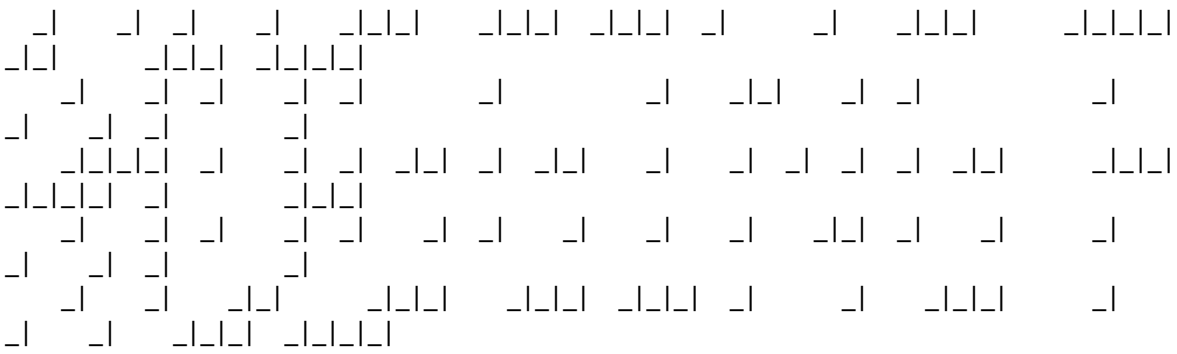
(Note: T4 GPU has 16 GB VRAM, 7 billion parameters of Llama 2 in 4-bit precision consume around 14 GB in FP16, and PEFT techniques like QLoRA are employed for efficient fine-tuning.)

```
# !pip install -q accelerate==0.21.0 peft==0.4.0 bitsandbytes==0.40.2 transformers==4.31.0 trl==0.4.7


# Import necessary packages for the fine-tuning process
import os        # Operating system functionalities
import torch    # PyTorch library for deep learning
 from datasets import load_dataset       # Loading datasets for training
from transformers import (
 AutoModelForCausalLM,       # AutoModel for language modeling tasks
  AutoTokenizer,                 # AutoTokenizer for tokenization
BitsAndBytesConfig,           # Configuration for BitsAndBytes
 HfArgumentParser,             # Argument parser for Hugging Face models
 TrainingArguments,            # Training arguments for model training
  Pipeline ,                    # Creating pipelines for model inference
logging,                       # Logging information during training
)
from peft import LoraConfig, PeftModel                  # Packages for parameter-efficient fine-tuning (PEFT) from trl
import SFTTrainer                                       # SFTTrainer for supervised fine-tuning


# !pip install -q datasets
!huggingface-cli login
```

LLM-Finetuning with PEFT



```
A token is already saved on your machine. Run `huggingface-cli whoami` to get more
information or `huggingface-cli logout` if you want to log out.

 Setting a new token will erase the existing one.
    To login, `huggingface_hub` requires a token generated from
https://huggingface.co/settings/tokens .

Token:
Add token as git credential? (Y/n) n
Token is valid (permission: write).
Your token has been saved to /root/.cache/huggingface/token
Login successful
```

- **Section 1:** Parameters to tune
  - Load a llama-2-7b-chat-hf model and train it on the mlabonne/guanaco-llama2-1k dataset.
  - The dataset contains 1,000 samples.
  - You can find more information about the dataset in this notebook.
  - Feel free to use a different dataset.
- **Section 2:** QLoRA parameters
  - QLoRA will use a rank of 64 with a scaling parameter of 16.
  - See this article for more information about LoRA parameters.
  - The Llama 2 model will be loaded directly in 4-bit precision using the NF4 type.
  - The model will be trained for one epoch.
- **Section 3:** Other parameters
  - To get more information about the other parameters, check the TrainingArguments, PeftModel, and SFTTrainer documentation.

```
# The model that you want to train from the Hugging Face hub

model_name = "NousResearch/Llama-2-7b-hf"

 # The instruction dataset to use
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
dataset_name = "mlabonne/guanaco-llama2-1k"

# Fine-tuned model name

new_model = "llama-2-7b-miniguanaco"


################################################################################

# QLoRA parameters

 ################################################################################

# LoRA attention dimension

lora_r = 64


# Alpha parameter for LoRA scaling

lora_alpha = 16

# Dropout probability for LoRA layers

 lora_dropout = 0.1


################################################################################

# bitsandbytes parameters
################################################################################

# Activate 4-bit precision base model loading

 use_4bit = True

# Compute dtype for 4-bit base models

bnb_4bit_compute_dtype = "float16"

# Quantization type (fp4 or nf4)

bnb_4bit_quant_type = "nf4"

# Activate nested quantization for 4-bit base models (double quantization)
use_nested_quant = False


################################################################################
```

By: Tarun S Gowda

```
# TrainingArguments parameters
################################################################################

# Output directory where the model predictions and checkpoints will be stored

output_dir = "./results"

# Number of training epochs

num_train_epochs = 1

# Enable fp16/bf16 training (set bf16 to True with an A100)

 fp16 = False

 bf16 = False

# Batch size per GPU for training

per_device_train_batch_size = 4

# Number of update steps to accumulate the gradients for .

gradient_accumulation_steps = 1

# Enable gradient checkpointing

gradient_checkpointing = True

# Maximum gradient normal (gradient clipping)

max_grad_norm = 0.3

# Initial learning rate (AdamW optimizer)

learning_rate = 2e-4

# Weight decay to apply to all layers except bias/LayerNorm weights

weight_decay = 0.001

# Optimizer to use

optim = "paged_adamw_32bit"

# Learning rate schedule (constant a bit better than cosine)

lr_scheduler_type = "constant"

# Number of training steps (overrides num_train_epochs)

max_steps = -1

# Ratio of steps for a linear warmup (from 0 to learning rate)
```

```
warmup_ratio = 0.03

# Group sequences into batches with same length

 # Saves memory and speeds up training considerably

group_by_length = True

# Save checkpoint every X updates steps

save_steps = 25

 # Log every X updates steps

logging_steps = 25


################################################################################

# SFT parameters
################################################################################

# Maximum sequence length to use

max_seq_length = None

# Pack multiple short examples in the same input sequence to increase efficiency

packing = False

# Load the entire model on the GPU 0

device_map = {"": 0}
```

1. **Loading the Dataset:** The first step involves loading the preprocessed dataset. This dataset will be used for fine-tuning. Preprocessing might involve reformatting prompts, filtering out low-quality text, and combining multiple datasets if needed.
2. **Configuring BitsAndBytes for 4-bit Quantization:** The BitsAndBytesConfig is set up to enable 4-bit quantization. This configuration is crucial for reducing the memory usage during fine-tuning.
3. **Loading Llama 2 Model and Tokenizer in 4-bit Precision:** The Llama 2 model is loaded with 4-bit precision, which significantly reduces the memory footprint. The corresponding tokenizer is also loaded to preprocess the text data.

4. **Loading Configurations and Initializing SFTTrainer:**

- The configurations needed for QLoRA, which is a parameter-efficient fine-tuning technique, are loaded.
- Regular training parameters are set up.
- The SFTTrainer is initialized with all the loaded configurations and parameters. This trainer will manage the supervised fine-tuning process.

5**. Start of Training:** After all the necessary components are loaded and configured, the training process begins. The SFTTrainer takes care of fine-tuning the Llama 2 model using the specified dataset, configurations, and parameters.

These steps collectively set up the environment for fine-tuning a Llama 2 model with 7 billion parameters in 4-bit precision using the QLoRA technique, thus optimizing for VRAM limitations while maintaining model performance.

```
# Step 1 : Load dataset (you can process it here)
 dataset = load_dataset(dataset_name, split="train")
```

```
# Step 2 :Load tokenizer and model with QLoRA configuration
 compute_dtype = getattr(torch, bnb_4bit_compute_dtype)

bnb_config = BitsAndBytesConfig(
 load_in_4bit=use_4bit,
bnb_4bit_quant_type=bnb_4bit_quant_type,
bnb_4bit_compute_dtype=compute_dtype,
 bnb_4bit_use_double_quant=use_nested_quant, )
# Step 3 :Check GPU compatibility with bfloat16
if compute_dtype == torch.float16 and use_4bit:
 major, _ = torch.cuda.get_device_capability()
if major >= 8:
print("=" * 80)
 print("Your GPU supports bfloat16: accelerate training with bf16=True")
 print("=" * 80)

# Step 4 :Load base model
model = AutoModelForCausalLM.from_pretrained(
 model_name,
quantization_config=bnb_config,
device_map=device_map
)
model.config.use_cache = False
 model.config.pretraining_tp = 1
```

```
Downloading (…)lve/main/config.json:   0%|          | 0.00/583 [00:00<?, ?B/s]
Downloading (…)fetensors.index.json:   0%|          | 0.00/26.8k [00:00<?, ?B/s]
Downloading shards:   0%|          | 0/2 [00:00<?, ?it/s]
Downloading (…)of-00002.safetensors:   0%|          | 0.00/9.98G [00:00<?, ?B/s]
Downloading (…)of-00002.safetensors:   0%|          | 0.00/3.50G [00:00<?, ?B/s]
Loading checkpoint shards:   0%|          | 0/2 [00:00<?, ?it/s]
Downloading (…)neration_config.json:   0%|          | 0.00/179 [00:00<?, ?B/s]
```

```
# Step 5 :Load LLaMA tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
tokenizer.add_special_tokens({'pad_token': '[PAD]'})
 tokenizer.pad_token = tokenizer.eos_token
 tokenizer.padding_side = "right"
```

```
Downloading (…)okenizer_config.json:   0%|          | 0.00/746 [00:00<?, ?B/s]
Downloading tokenizer.model:   0%|          | 0.00/500k [00:00<?, ?B/s]
Downloading (…)/main/tokenizer.json:   0%|          | 0.00/1.84M [00:00<?, ?B/s]
Downloading (…)in/added_tokens.json:   0%|          | 0.00/21.0 [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/435 [00:00<?, ?B/s]
```

```
# Step 6 :Load LoRA configuration
 peft_config = LoraConfig(
lora_alpha=lora_alpha,
lora_dropout=lora_dropout,
 r=lora_r,
 bias="none",
 task_type="CAUSAL_LM", )

# Step 7 :Set training parameters
training_arguments = TrainingArguments(
output_dir=output_dir,
 num_train_epochs=num_train_epochs,
 per_device_train_batch_size=per_device_train_batch_size,
 gradient_accumulation_steps=gradient_accumulation_steps,
optim=optim,
save_steps=save_steps,
logging_steps=logging_steps,
 learning_rate=learning_rate,
weight_decay=weight_decay,
fp16=fp16,
 bf16=bf16,
max_grad_norm=max_grad_norm,
 max_steps=max_steps,
 warmup_ratio=warmup_ratio,
 group_by_length=group_by_length,
lr_scheduler_type=lr_scheduler_type,
report_to="tensorboard”
)

# Step 8 :Set supervised fine-tuning parameters
 trainer = SFTTrainer(
 model=model,
 train_dataset=dataset,
peft_config=peft_config,
dataset_text_field="text",
 max_seq_length=max_seq_length,
 tokenizer=tokenizer,
 args=training_arguments,
 packing=packing,
)
```

```
/usr/local/lib/python3.10/dist-packages/peft/utils/other.py:102: FutureWarning:
prepare_model_for_int8_training is deprecated and will be removed in a future version.
```

```
Use prepare_model_for_kbit_training instead.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/trl/trainer/sft_trainer.py:159: UserWarning: You didn't pass a `max_seq_length` argument to the SFTTrainer, this will default to 1024
  warnings.warn(


Map:   0%|          | 0/1000 [00:00<?, ? examples/s]
```

```
# Step 9 :Train model
 trainer.train()
```

```
# Step 10 :Save trained model
trainer.model.save_pretrained(new_model)
```

```
You're using a LlamaTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` method is faster than using a method to encode the text followed by a call to the `pad` method to get a padded encoding.
```

[250/250 26:14, Epoch 1/1]

Step	Training Loss
25	1.4355775
50	1.5657964
75	1.5461145
100	1.5555535
125	1.7675787
150	1.7675778
175	1.5345467
200	1.6775677
225	1.7676556
250	1.5685657

```
%load_ext tensorboard
%tensorboard --logdir results/runs
```

Exploring the Guanaco Chatbot Demo with LLaMA-7B Model

we will dive into the code of a chatbot demo that utilizes the LLaMA-7B model for generating human-like responses. The chatbot, named Guanaco, is designed to interact with users, answer their queries, and provide insights using natural language generation. We will break down the code into several sections to understand its functionality and purpose.

Introduction

The code presented here is a modified version of a Jupyter Notebook available on GitHub. It showcases the implementation of a chatbot using the LLaMA-7B language model and is integrated into a graphical user interface (GUI) using the Gradio library. The chatbot's responses are generated by predicting the next words based on the input conversation history.

```
# https://github.com/artidoro/qlora/blob/main/examples/guanaco_7B_demo_colab.ipynb modified
# Install latest bitsandbytes & transformers, accelerate from source
!pip install -q -U bitsandbytes
!pip install -q -U git+https://github.com/huggingface/transformers.git
!pip install -q -U git+https://github.com/huggingface/peft.git
!pip install -q -U git+https://github.com/huggingface/accelerate.git
# Other requirements for the demo
!pip install gradio
!pip install sentencepiece
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.6/92.6 MB 11.1
MB/s eta 0:00:00 Installing build dependencies ... done Getting requirements to build wheel ... done Preparing
```

By: Tarun S Gowda

```
metadata (pyproject.toml) ... done
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 4.7
MB/s eta 0:00:00 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
7.8/7.8 MB 107.4 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 56.7
MB/s eta 0:00:00 Building wheel for transformers (pyproject.toml) ... done Installing build dependencies ... done
Getting requirements to build wheel ... done Preparing metadata (pyproject.toml) ... done
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251.2/251.2 kB 5.3 MB/s eta 0:00:00
  Building wheel for peft (pyproject.toml) ... done
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
  Building wheel for accelerate (pyproject.toml) ... done
Collecting gradio
  Downloading gradio-3.41.1-py3-none-any.whl (20.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.1/20.1 MB 72.0 MB/s eta 0:00:00
Collecting aiofiles<24.0,>=22.0 (from gradio)
  Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Requirement already satisfied: altair<6.0,>=4.2.0 in /usr/local/lib/python3.10/dist-
packages (from gradio) (4.2.2)
Collecting fastapi (from gradio)
  Downloading fastapi-0.101.1-py3-none-any.whl (65 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.8/65.8 kB 8.5 MB/s eta 0:00:00
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.3.1.tar.gz (5.5 kB)
  Preparing metadata (setup.py) ... done
Collecting gradio-client==0.5.0 (from gradio)
  Downloading gradio_client-0.5.0-py3-none-any.whl (298 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

8.2/298.2 kB 37.9 MB/s eta 0:00:00
Collecting httpx (from gradio)
  Downloading httpx-0.24.1-py3-none-any.whl (7



━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.6/92.6 MB 11.1 MB/s eta 0:00:00 Installing
build dependencies ... done Getting requirements to build wheel ... done Preparing
metadata (pyproject.toml) ... done ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB
4.7 MB/s eta 0:00:00 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 107.4 MB/s eta
0:00:00 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 56.7 MB/s eta 0:00:00
Building wheel for transformers (pyproject.toml) ... done Installing build
dependencies ... done Getting requirements to build wheel ... done Preparing metadata
(pyproject.toml) ... done ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251.2/251.2 kB 5.3 MB/s
eta 0:00:00 Building wheel for peft (pyproject.toml) ... done Installing build
dependencies ... done Getting requirements to build wheel ... done Preparing metadata
(pyproject.toml) ... done Building wheel for accelerate (pyproject.toml) ... done
Collecting gradio Downloading gradio-3.41.1-py3-none-any.whl (20.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.1/20.1 MB 72.0 MB/s eta 0:00:00 Collecting
aiofiles<24.0,>=22.0 (from gradio) Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
```

```
Requirement already satisfied: altair<6.0,>=4.2.0 in /usr/local/lib/python3.10/dist-
packages (from gradio) (4.2.2) Collecting fastapi (from gradio) Downloading fastapi-
0.101.1-py3-none-any.whl (65 kB) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.8/65.8 kB
8.5 MB/s eta 0:00:00 Collecting ffmpy (from gradio) Downloading ffmpy-0.3.1.tar.gz (5.5
kB) Preparing metadata (setup.py) ... done Collecting gradio-client==0.5.0 (from gradio)
Downloading gradio_client-0.5.0-py3-none-any.whl (298 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 298.2/298.2 kB 37.9 MB/s eta 0:00:00 Collecting
httpx (from gradio) Downloading httpx-0.24.1-py3-none-any.whl (75 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.4/75.4 kB 12.1 MB/s eta 0:00:00 Requirement
already satisfied: huggingface-hub>=0.14.0 in /usr/local/lib/python3.10/dist-packages
(from gradio) (0.16.4) Requirement already satisfied: importlib-resources<7.0,>=1.3 in
/usr/local/lib/python3.10/dist-packages (from gradio) (6.0.1) Requirement already
satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.1.2)
Requirement already satisfied: markupsafe~=2.0 in /usr/local/lib/python3.10/dist-packages
(from gradio) (2.1.3) Requirement already satisfied: matplotlib~=3.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1) Requirement already
satisfied: numpy~=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.23.5)
Collecting orjson~=3.0 (from gradio) Downloading orjson-3.9.5-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (139 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.9/139.9 kB 17.4 MB/s eta 0:00:00


9.9/139.9 kB 17.4 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from
gradio) (23.1)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages
(from gradio) (1.5.3)
Requirement already satisfied: pillow<11.0,>=8.0 in /usr/local/lib/python3.10/dist-
packages (from gradio) (9.4.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4 in
/usr/local/lib/python3.10/dist-packages (from gradio) (2.2.0)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Collecting python-multipart (from gradio
  Downloading python_multipart-0.0.6-py3-none-any.whl (45 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.7/45.7 kB 5.4 MB/s eta 0:00:00
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages
(from gradio) (6.0.1)
Requirement already satisfied: requests~=2.0 in /usr/local/lib/python3.10/dist-packages
(from gradio) (2.31.0)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.10/dist-
packages (from gradio) (4.7.1)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.23.2-py3-none-any.whl (59 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.5/59.5 kB 8.8 MB/s eta 0:00:00
Collecting websockets<12.0,>=10.0 (from gradio)
 Downloading websockets-11.0.3-cp310-cp310-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl
```

By: Tarun S Gowda

```
(129                                                                    kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.9/129.9 kB 16.8 MB/s eta 0:00:00
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from
gradio-client==0.5.0->gradio)                                          (2023.6.0)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from
altair<6.0,>=4.2.0->gradio)                                              (0.4)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages
(from                      altair<6.0,>=4.2.0->gradio)                  (4.19.0)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from
altair<6.0,>=4.2.0->gradio)                                             (0.12.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
huggingface-hub>=0.14.0->gradio)                                        (3.12.2)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages
(from                    huggingface-hub>=0.14.0->gradio)               (4.66.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages
(from                        matplotlib~=3.0->gradio)                   (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages
(from                        matplotlib~=3.0->gradio)                   (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-
packages              (from              matplotlib~=3.0->gradio)       (4.42.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-
packages              (from              matplotlib~=3.0->gradio)       (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages
(from                        matplotlib~=3.0->gradio)                   (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-
packages              (from              matplotlib~=3.0->gradio)       (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages
(from                       pandas<3.0,>=1.0->gradio)                   (2023.3)
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-
packages  (from  pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4->gradio)  (0.5.0)
Requirement already satisfied: pydantic-core==2.6.0 in /usr/local/lib/python3.10/dist-
packages  (from  pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4->gradio)  (2.6.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-
packages              (from              requests~=2.0->gradio)         (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages
(from                          requests~=2.0->gradio)                     (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-
packages              (from              requests~=2.0->gradio)         (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-
packages              (from            requests~=2.0->gradio)          (2023.7.22)
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.10/dist-packages (from
uvicorn>=0.14.0->gradio)                                                (8.1.7)
Collecting             h11>=0.8             (from             uvicorn>=0.14.0->gradio)
  Downloading              h11-0.14.0-py3-none-any.whl              (58             kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 8.0 MB/s eta 0:00:00
Collecting          starlette<0.28.0,>=0.27.0          (from          fastapi->gradio)
  Downloading            starlette-0.27.0-py3-none-any.whl            (66             kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.0/67.0 kB 6.3 MB/s eta 0:00:00
Collecting           httpcore<0.18.0,>=0.15.0           (from           httpx->gradio)
  Downloading            httpcore-0.17.3-py3-none-any.whl             (74             kB)
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 74.5/74.5 kB 11.4 MB/s eta 0:00:00
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from
httpx->gradio) (1.3.0)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.10/dist-packages
(from httpcore<0.18.0,>=0.15.0->httpx->gradio) (3.7.1)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages
(from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio)
(2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages
(from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (0.9.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
python-dateutil>=2.7->matplotlib~=3.0->gradio) (1.16.0)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages
(from anyio<5.0,>=3.0->httpcore<0.18.0,>=0.15.0->httpx->gradio) (1.1.3)
Building wheels for collected packages: ffmpy
  Building wheel for ffmpy (setup.py) ... done
  Created wheel for ffmpy: filename=ffmpy-0.3.1-py3-none-any.whl size=5579
sha256=baf0e0fc40c2519812aeaa3b472b6f1f9c250ec0ce39ba364a66ff1515e517db
  Stored in directory:
/root/.cache/pip/wheels/01/a6/d1/1c0828c304a4283b2c1639a09ad86f83d7c487ef34c6b4a1bf
Successfully built ffmpy
Installing collected packages: pydub, ffmpy, websockets, semantic-version, python-multipart,
orjson, h11, aiofiles, uvicorn, starlette, httpcore, httpx, fastapi, gradio-client, gradio
Successfully installed aiofiles-23.2.1 fastapi-0.101.1 ffmpy-0.3.1 gradio-3.41.1 gradio-
client-0.5.0 h11-0.14.0 httpcore-0.17.3 httpx-0.24.1 orjson-3.9.5 pydub-0.25.1 python-
multipart-0.0.6 semantic-version-2.10.0 starlette-0.27.0 uvicorn-0.23.2 websockets-11.0.3
Collecting sentencepiece
  Downloading sentencepiece-0.1.99-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 10.1 MB/s eta 0:00:00
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.99
```

Installation and setup

The initial portion of the code involves installing various Python packages required for running the demo. These packages include bitsandbytes, transformers, accelerate, gradio, and sentencepiece. These libraries provide the necessary tools for model loading, text generation, and creating the graphical interface for user interaction.

```
# Load the model.
```

```
 # Note: It can take a while to download LLaMA and add the adapter modules.
# You can also use the 13B model by loading in 4bits.

import torch
from peft import PeftModel
from transformers import AutoModelForCausalLM, AutoTokenizer, LlamaTokenizer, StoppingCriteria, StoppingCriteriaList, TextIteratorStreamer
```

Model Loading

After installing the required packages, the code proceeds to load the LLaMA-7B language model and adapter modules. The AutoModelForCausalLM class from the transformers library is used to load the pre-trained model. Additionally, the PeftModel class from the peft library is utilized to incorporate adapter modules into the model. Adapter modules allow fine-tuning of pre-trained models for specific tasks without affecting the original model parameters.

```
model_name = "decapoda-research/llama-7b-hf"
adapters_name = 'timdettmers/guanaco-7b'

import torch
torch.cuda.empty_cache()
print(f"Starting to load the model {model_name} into memory")
m = AutoModelForCausalLM.from_pretrained(
model_name,
# load_in_4bit=True,
 torch_dtype=torch.bfloat16,
 device_map={"": 0}
)
torch.cuda.empty_cache()
m = PeftModel.from_pretrained(m, adapters_name)
torch.cuda.empty_cache()
m = m.merge_and_unload()
tok = LlamaTokenizer.from_pretrained(model_name)
tok.bos_token_id = 1

stop_token_ids = [0]

print(f"Successfully loaded the model {model_name} into memory")
torch.cuda.empty_cache()
```

Chatbot Setup

The code defines a series of functions and configurations to set up the chatbot's behavior within the Gradio interface. These functions handle message processing, conversation history management, and text generation using the loaded model.

- The convert_history_to_text function converts the conversation history into a formatted text that includes both user and assistant messages.
- The user function appends the user's message to the conversation history.
- The bot function uses the loaded model to generate responses. It tokenizes the conversation history, sets up text generation parameters such as temperature and top-k sampling, and iteratively generates tokens while adhering to stopping criteria. The generated text is appended to the assistant's message in the history.

```
# Setup the gradio Demo.
import datetime
 import os from threading import Event, Thread
from uuid import uuid4

 import gradio as gr
import requests

max_new_tokens = 1536
start_message = """A chat between a curious human and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions."""
```

Gradio Interface

The Gradio interface is used to create a user-friendly GUI for interacting with the chatbot. Users can input messages, adjust advanced options such as temperature and sampling techniques, and view the chatbot's responses in real-time.

- The GUI displays a chat message box where users can type their messages.
- The "Submit" button triggers user input processing and chatbot response generation.
- Advanced options like temperature, top-p sampling, top-k sampling, and repetition penalty can be adjusted using sliders.
- A disclaimer is included to highlight that the model's outputs may not always be factually accurate.
- A privacy policy link is provided for user reference.

```
class StopOnTokens(StoppingCriteria):
 def __call__(self, input_ids: torch.LongTensor, scores: torch.FloatTensor, **kwargs) -> bool:
         for stop_id in stop_token_ids:
         if input_ids[0][-1] == stop_id:
           return True
           return False
def convert_history_to_text(history):
 text = start_message + "".join(
    [
```

```
    "".join(
        [
          f"### Human: {item[0]}\n",
          f"### Assistant: {item[1]}\n",
        ]
    )
    for item in history[:-1]
  ]
) text += "".join( [ "".join(
      [
        "".join(
          [
              f"### Human: {history[-1][0]}\n",
              f"### Assistant: {history[-1][1]}\n",
          ]
        )
      ]
    )
    return text

def log_conversation(conversation_id, history, messages, generate_kwargs):

logging_url = os.getenv("LOGGING_URL", None)

if logging_url is None:

return

timestamp = datetime.datetime.now().strftime("%Y-%m-%dT%H:%M:%S")

data = {

      "conversation_id": conversation_id,

      "timestamp": timestamp,

      "history": history,

      "messages": messages,

      "generate_kwargs": generate_kwargs,

}

data = {

"conversation_id": conversation_id,

"timestamp": timestamp,

"history": history,
```

```
"messages": messages,

"generate_kwargs": generate_kwargs,

}

try:

     requests.post(logging_url, json=data)

  except requests.exceptions.RequestException as e:

    print(f"Error logging conversation: {e}")

def user(message, history):

# Append the user's message to the conversation history

return "", history + [[message, ""]]

def bot(history, temperature, top_p, top_k, repetition_penalty, conversation_id):
print(f"history: {history}")

# Initialize a StopOnTokens object

stop = StopOnTokens()

# Construct the input message string for the model by concatenating the current system
message and conversation history

messages = convert_history_to_text(history)

# Tokenize the messages string

input_ids = tok(messages, return_tensors="pt").input_ids

input_ids = input_ids.to(m.device)

streamer = TextIteratorStreamer(tok, timeout=10.0, skip_prompt=True,
skip_special_tokens=True)

generate_kwargs = dict(

     input_ids=input_ids,

     max_new_tokens=max_new_tokens,

     temperature=temperature,

     do_sample=temperature > 0.0,

     top_p=top_p,

     top_k=top_k,
```

```
     repetition_penalty=repetition_penalty,
     streamer=streamer,
     stopping_criteria=StoppingCriteriaList([stop]),
)
stream_complete = Event()
def generate_and_signal_complete():
    m.generate(**generate_kwargs)
    stream_complete.set()
def log_after_stream_complete():
    stream_complete.wait()
    log_conversation(
        conversation_id,
        history,
        messages,
       {
          "top_k": top_k,
          "top_p": top_p,
          "temperature": temperature,
          "repetition_penalty": repetition_penalty,
       },
    )
t1 = Thread(target=generate_and_signal_complete)
t1.start()
t2 = Thread(target=log_after_stream_complete)
t2.start()
# Initialize an empty string to store the generated text
partial_text = ""
for new_text in streamer:
```

```
partial_text += new_text
history[-1][1] = partial_text
yield history


def get_uuid():
    return str(uuid4())


with gr.Blocks(
    theme=gr.themes.Soft(),
    css=".disclaimer {font-variant-caps: all-small-caps;}",
) as demo:
conversation_id = gr.State(get_uuid)
gr.Markdown(
         """Guanaco Demo
 """
)
chatbot = gr.Chatbot().style(height=500)
with gr.Row():
   with gr.Column():
     msg = gr.Textbox(
         label="Chat Message Box",
         placeholder="Chat Message Box",
         show_label=False,
      ).style(container=False)
  with gr.Column():
     with gr.Row():
        submit = gr.Button("Submit")
        stop = gr.Button("Stop")
```

```
        clear = gr.Button("Clear")
with gr.Row():
     with gr.Accordion("Advanced Options:", open=False):
        with gr.Row():
        with gr.Column():
with gr.Row():
      temperature = gr.Slider(
        label="Temperature",
        value=0.7,
        minimum=0.0,
        maximum=1.0,
        step=0.1,
        interactive=True,
        info="Higher values produce more diverse outputs",
     )
with gr.Column():
   with gr.Row():
       top_p = gr.Slider(
           label="Top-p (nucleus sampling)",
           value=0.9,
           minimum=0.0,
           maximum=1,
           step=0.01,
          interactive=True,
          info=(
                      "Sample from the smallest possible set of tokens whose cumulative
probability "
                       "exceeds top_p. Set to 1 to disable and sample from all tokens."
```

```
            ),
        )
with gr.Column():
    with gr.Row():
        top_k = gr.Slider(
            label="Top-k",
            value=0,
            minimum=0.0,
            maximum=200,
            step=1,
            interactive=True,
            info="Sample from a shortlist of top-k tokens — 0 to disable and sample from
all tokens.",
    )
with gr.Column():
      with gr.Row():
         repetition_penalty = gr.Slider(
              label="Repetition Penalty",
              value=1.0,
              minimum=1.0,
              maximum=2.0,
              step=0.1,
              interactive=True,
              info="Penalize repetition — 1.0 to disable.",
         )
with gr.Row():
    gr.Markdown(
       "Disclaimer: The model can produce factually incorrect output, and should not be
relied on to produce "
```

```
        "factually accurate information. The model was trained on various public datasets;
while great efforts "
        "have been taken to clean the pretraining data, it is possible that this model
could generate lewd, "
        "biased, or otherwise offensive outputs.",
         elem_classes=["disclaimer"],
    )
with gr.Row():
gr.Markdown(
"[Privacy policy](https://gist.github.com/samhavens/c29c68cdcd420a9aa0202d0839876dac)",
elem_classes=["disclaimer"],
)
submit_event = msg.submit(
    fn=user,
    inputs=[msg, chatbot],
    outputs=[msg, chatbot],
    queue=False,
).then(
     fn=bot,
     inputs=[
         chatbot,
         temperature,
         top_p,
         top_k,
         repetition_penalty,
         conversation_id,
],
outputs=chatbot,
queue=True,
```

```
)
submit_click_event = submit.click(
  fn=user,
  inputs=[msg, chatbot],
  outputs=[msg, chatbot],
  queue=False,
).then(
      fn=bot,
      inputs=[
        chatbot,
        temperature,
        top_p,
        top_k,
        repetition_penalty,
        conversation_id,
    ],
    outputs=chatbot,
    queue=True,
   )
     stop.click(
     fn=None,
     inputs=None,
     outputs=None,
     cancels=[submit_event, submit_click_event],
     queue=False,
  )
  clear.click(lambda: None, None, chatbot, queue=False)
  demo.queue(max_size=128, concurrency_count=2)
```

Conclusion

In conclusion, the provided code demonstrates the implementation of a chatbot named Guanaco, powered by the LLaMA-7B language model. The chatbot interacts with users through a user-friendly Gradio interface, generating responses based on input conversation history. By breaking down the code into various sections, we've explored how the model is loaded, the chatbot's behavior is defined, and the Gradio interface is set up for user interaction.

It's important to note that this demo showcases the capabilities of the LLaMA-7B model and its interaction with users. However, as with any AI-generated content, users should be aware that the model's responses may not always be entirely accurate or appropriate, and they should exercise caution when using the chatbot for factual information or important decisions.

Using PEFT and bitsandbytes to finetune a LoRa checkpoint

```
!pip install -q bitsandbytes datasets accelerate loralib
!pip install -q git+https://github.com/huggingface/transformers.git@main
git+https://github.com/huggingface/peft.git
```

```
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.6/92.6 MB 9.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 519.3/519.3 kB 49.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251.2/251.2 kB 27.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 14.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 22.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 16.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 26.9 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 60.2 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 61.1 MB/s eta 0:00:00
  Building wheel for transformers (pyproject.toml) ... done
```

By: Tarun S Gowda

```
  Building wheel for peft (pyproject.toml) ... done
```

In [ ]:

```
from huggingface_hub import notebook_login

notebook_login()
```

```
VBox(children=(HTML(value='<center>
<img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv…
```

```
!nvidia-smi -L
```

Setup the model

```
import os
os.environ["CUDA_VISIBLE_DEVICES"]="0"
import torch
import torch.nn as nn
import bitsandbytes as bnb
from transformers import AutoTokenizer, AutoConfig, AutoModelForCausalLM

model = AutoModelForCausalLM.from_pretrained(
    "bigscience/bloomz-560m", #Replace from big model to small model bigscience/bloom-
7b1
    load_in_8bit=True,
    device_map='auto',
)

tokenizer = AutoTokenizer.from_pretrained("bigscience/bloomz-560m")


tokenizer = AutoTokenizer.from_pretrained("bigscience/bloomz-560m")
```

```
Downloading (…)lve/main/config.json:   0%|          | 0.00/715 [00:00<?, ?B/s]
Downloading model.safetensors:   0%|          | 0.00/1.12G [00:00<?, ?B/s]
Downloading (…)okenizer_config.json:   0%|          | 0.00/222 [00:00<?, ?B/s]
Downloading tokenizer.json:   0%|          | 0.00/14.5M [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/85.0 [00:00<?, ?B/s]
```

Freezing the original weights

```
for param in model.parameters():
  param.requires_grad = False  # freeze the model - train adapters later
  if param.ndim == 1:
    # cast the small parameters (e.g. layernorm) to fp32 for stability
```

By: Tarun S Gowda

```
     param.data = param.data.to(torch.float32)

 model.gradient_checkpointing_enable()  # reduce number of stored activations
 model.enable_input_require_grads()

 class CastOutputToFloat(nn.Sequential):
   def forward(self, x): return super().forward(x).to(torch.float32)
 model.lm_head = CastOutputToFloat(model.lm_head)
```

Setting up the LoRA Adapters

```
def print_trainable_parameters(model):
    """
    Prints the number of trainable parameters in the model.
    """
    trainable_params = 0
    all_param = 0
    for _, param in model.named_parameters():
        all_param += param.numel()
        if param.requires_grad:
            trainable_params += param.numel()
    print(
        f"trainable params: {trainable_params} || all params: {all_param} || trainable%:
{100 * trainable_params / all_param}"
    )


from peft import LoraConfig, get_peft_model

config = LoraConfig(
    r=16, #attention heads
    lora_alpha=32, #alpha scaling
    # target_modules=["q_proj", "v_proj"], #if you know the
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM" # set this for CLM or Seq2Seq
)

model = get_peft_model(model, config)
print_trainable_parameters(model)
```

```
trainable params: 1572864 || all params: 560787456 || trainable%: 0.2804741766549072
```

Data

```
import transformers
from datasets import load_dataset
data = load_dataset("Abirate/english_quotes")
```

```
Downloading readme:   0%|          | 0.00/5.55k [00:00<?, ?B/s]
Downloading data files:   0%|          | 0/1 [00:00<?, ?it/s]
Downloading data:   0%|          | 0.00/647k [00:00<?, ?B/s]
Extracting data files:   0%|          | 0/1 [00:00<?, ?it/s]
Generating train split: 0 examples [00:00, ? examples/s]
```

```
Data
```

```
DatasetDict({
    train: Dataset({
        features: ['quote', 'author', 'tags'],
        num_rows: 2508
    })
})
```

```
def merge_columns(example):
    example["prediction"] = example["quote"] + " ->: " + str(example["tags"])
    return example

data['train'] = data['train'].map(merge_columns)
data['train']["prediction"][:5]
```

```
Map:   0%|          | 0/2508 [00:00<?, ? examples/s]

["“Be yourself; everyone else is already taken.” ->: ['be-yourself', 'gilbert-perreira',
'honesty', 'inspirational', 'misattributed-oscar-wilde', 'quote-investigator']",
 "“I'm selfish, impatient and a little insecure. I make mistakes, I am out of control and
at times hard to handle. But if you can't handle me at my worst, then you sure as hell
don't deserve me at my best.” ->: ['best', 'life', 'love', 'mistakes', 'out-of-control',
'truth', 'worst']",
 "“Two things are infinite: the universe and human stupidity; and I'm not sure about the
universe.” ->: ['human-nature', 'humor', 'infinity', 'philosophy', 'science',
'stupidity', 'universe']",
 "“So many books, so little time.” ->: ['books', 'humor']",
 "“A room without books is like a body without a soul.” ->: ['books', 'simile', 'soul']"]
```

```
data['train'][0]
```

```
{'quote': '“Be yourself; everyone else is already taken.”',
 'author': 'Oscar Wilde',
 'tags': ['be-yourself',
  'gilbert-perreira',
  'honesty',
  'inspirational',
  'misattributed-oscar-wilde',
  'quote-investigator'],
 'prediction': "“Be yourself; everyone else is already taken.” ->: ['be-yourself',
'gilbert-perreira', 'honesty', 'inspirational', 'misattributed-oscar-wilde', 'quote-
investigator']"}
```

```
data = data.map(lambda samples: tokenizer(samples['prediction']), batched=True)
```

```
Map: 0%| | 0/2508 [00:00<?, ? examples/s]
```

```
Data
```

```
DatasetDict({
    train: Dataset({
        features: ['quote', 'author', 'tags', 'prediction', 'input_ids',
'attention_mask'],
        num_rows: 2508
    })
})
```

Training

```
trainer = transformers.Trainer(
    model=model,
    train_dataset=data['train'],
    args=transformers.TrainingArguments(
        per_device_train_batch_size=4,
        gradient_accumulation_steps=4,
        warmup_steps=100,
        max_steps=200,
        learning_rate=2e-4,
        fp16=True,
        logging_steps=25,
        output_dir='outputs'
    ),
    data_collator=transformers.DataCollatorForLanguageModeling(tokenizer, mlm=False)
```

```
)
model.config.use_cache = False  # silence the warnings. Please re-enable for inference!
trainer.train()
```

[200/200 07:14, Epoch 1/2]

Step	Training Loss
25	2.546561760
50	2.564565675
75	2.342556577
100	2.225455678
125	2.654354567
150	2.735433376
175	2.588765656
200	2.696743543

```
TrainOutput(global_step=200, training_loss=2.6913645172119143, metrics={'train_runtime':
437.0945, 'train_samples_per_second': 7.321, 'train_steps_per_second': 0.458,
'total_flos': 663257454772224.0, 'train_loss': 2.6913645172119143, 'epoch': 1.28})
```

Share adapters on the Hub

```
model.push_to_hub("ashishpatel26/bloomz-560m-tagger",
                  use_auth_token=True,
                  commit_message="basic training",
                  private=True)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/utils/hub.py:844: FutureWarning: The
`use_auth_token` argument is deprecated and will be removed in v5 of Transformers.
  warnings.warn(
```

```
adapter_model.bin:   0%|          | 0.00/6.31M [00:00<?, ?B/s]
```

```
CommitInfo(commit_url='https://huggingface.co/ashishpatel26/bloomz-560m-
tagger/commit/8f3e3635208bd792396c44fd06307ee6b69ff08e', commit_message='basic training',
commit_description='', oid='8f3e3635208bd792396c44fd06307ee6b69ff08e', pr_url=None,
pr_revision=None, pr_num=None)
```

Load adapters from the Hub

By: Tarun S Gowda

```
import torch
 from peft import PeftModel, PeftConfig
from transformers import AutoModelForCausalLM, AutoTokenizer

peft_model_id = "ashishpatel26/bloomz-560m-tagger"
config = PeftConfig.from_pretrained(peft_model_id)
model = AutoModelForCausalLM.from_pretrained(config.base_model_name_or_path, return_dict=True,
load_in_8bit=True, device_map='auto')
tokenizer = AutoTokenizer.from_pretrained(config.base_model_name_or_path)

# Load the Lora model
model = PeftModel.from_pretrained(model, peft_model_id)
```

```
Downloading (…)/adapter_config.json:   0%|          | 0.00/440 [00:00<?, ?B/s]
Downloading adapter_model.bin:   0%|          | 0.00/6.31M [00:00<?, ?B/s]
```

Inference

```
batch = tokenizer("“Training models with PEFT and LoRa is cool” ->: ",
return_tensors='pt')

with torch.cuda.amp.autocast():
  output_tokens = model.generate(**batch, max_new_tokens=50)

print('\n\n', tokenizer.decode(output_tokens[0], skip_special_tokens=True))
```

```
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1529: UserWarning: You are calling .generate() with the `input_ids` being on a device type different than your model's device. `input_ids` is on cpu, whereas the model is on cuda. You may experience unexpected behaviors or slower generation. Please make sure that you have put `input_ids` to the correct device by calling for example input_ids = input_ids.to('cuda') before running `.generate()`. warnings.warn(
```

```
 “Training models with PEFT and LoRa is cool” ->:  ['training'], 'trainingmodels', 'trainingmodels-cloud'], 'trainingmodels', 'trainingmodels-cloud'], 'trainingmodels-cloud'], 'trainingmodels-cloud'], '
```

Fine-tune large models using peft adapters, transformers and bitsandbytes

In this tutorial we will cover how we can fine-tune large language models using the very recent peft library and bitsandbytes for loading large models in 8-bit. The fine-tuning method will rely on a recent method called "Low Rank Adapters" (LoRA), instead of fine-tuning the entire model you just have to fine-tune these adapters and load them properly inside the model. After fine-tuning the model you can also share your adapters on the 🤗 Hub and load them very easily. Let's get started!

Install requirements

First, run the cells below to install the requirements:

```
!pip install -q bitsandbytes datasets accelerate loralib
!pip install -q git+https://github.com/huggingface/transformers.git@main
git+https://github.com/huggingface/peft.git
```

```
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
```

Model loading

By: Tarun S Gowda

Here let's load the opt-6.7b-lora model, its weights in half-precision (float16) are about 13GB on the Hub! If we load them in 8-bit we would require around 7GB of memory instead.

```
import os
os.environ["CUDA_VISIBLE_DEVICES"]="0"
import torch
import torch.nn as nn
import bitsandbytes as bnb
from transformers import AutoTokenizer, AutoConfig, AutoModelForCausalLM,
BitsAndBytesConfig

bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.float16,
)

model = AutoModelForCausalLM.from_pretrained(
    "facebook/opt-1.3b",
    quantization_config=bnb_config,
    device_map='auto',
    trust_remote_code=True,
)
model.config.use_cache = False

tokenizer = AutoTokenizer.from_pretrained("facebook/opt-1.3b")
```

Post-processing on the model

Finally, we need to apply some post-processing on the 8-bit model to enable training, let's freeze all our layers, and cast the layer-norm in float32 for stability. We also cast the output of the last layer in float32 for the same reasons.

```
for param in model.parameters():
  param.requires_grad = False  # freeze the model - train adapters later
  if param.ndim == 1:
    # cast the small parameters (e.g. layernorm) to fp32 for stability
    param.data = param.data.to(torch.float32)

model.gradient_checkpointing_enable()  # reduce number of stored activations
model.enable_input_require_grads()
```

```
class CastOutputToFloat(nn.Sequential):
  def forward(self, x): return super().forward(x).to(torch.float32)
model.lm_head = CastOutputToFloat(model.lm_head)
```

Apply LoRA

Here comes the magic with peft! Let's load a PeftModel and specify that we are going to use low-rank adapters (LoRA) using get_peft_model utility function from peft.

```
def print_trainable_parameters(model):
    """
    Prints the number of trainable parameters in the model.
    """
    trainable_params = 0
    all_param = 0
    for _, param in model.named_parameters():
        all_param += param.numel()
        if param.requires_grad:
            trainable_params += param.numel()
    print(
        f"trainable params: {trainable_params} || all params: {all_param} ||
trainable%: {100 * trainable_params / all_param}"
    )
```

```
from peft import LoraConfig, get_peft_model

config = LoraConfig(
    r=16,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)

model = get_peft_model(model, config)
print_trainable_parameters(model)
```

```
trainable params: 3145728 || all params: 714924032 || trainable%: 0.4400087085056892
```

Training

```
import transformers
from datasets import load_dataset
data = load_dataset("Abirate/english_quotes")
data = data.map(lambda samples: tokenizer(samples['quote']), batched=True)

trainer = transformers.Trainer(
    model=model,
    train_dataset=data['train'],
    args=transformers.TrainingArguments(
        per_device_train_batch_size=4,
        gradient_accumulation_steps=4,
        warmup_steps=100,
        max_steps=200,
        learning_rate=2e-4,
        fp16=True,
        logging_steps=25,
        output_dir='outputs'
    ),
    data_collator=transformers.DataCollatorForLanguageModeling(tokenizer, mlm=False)
)
model.config.use_cache = False  # silence the warnings. Please re-enable for inference!
trainer.train()
```

```
Map:   0%|          | 0/2508 [00:00<?, ? examples/s]
```
[200/200 07:10, Epoch 1/2]

Step	Training Loss
25	2.486200
50	2.403900
75	2.416100
100	2.402800
125	2.410800
150	2.492600
175	2.402500
200	2.360900

```
TrainOutput(global_step=200, training_loss=2.421960391998291, metrics={'train_runtime':
433.3814, 'train_samples_per_second': 7.384, 'train_steps_per_second': 0.461,
'total_flos': 1076144213557248.0, 'train_loss': 2.421960391998291, 'epoch': 1.28})
```

Share adapters on the Hub

```
from huggingface_hub import notebook_login

notebook_login()
```

```
VBox(children=(HTML(value='<center>
<img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv…
```

```
model.push_to_hub("ashishpatel26/opt-6.1b-lora", use_auth_token=True)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/utils/hub.py:844: FutureWarning: The
`use_auth_token` argument is deprecated and will be removed in v5 of Transformers.
  warnings.warn(
```

```
adapter_model.bin:   0%|          | 0.00/12.6M [00:00<?, ?B/s]
```

```
CommitInfo(commit_url='https://huggingface.co/ashishpatel26/opt-6.1b-
lora/commit/0177c00b32f8c756b39beecda61121b805e8a9ce', commit_message='Upload model',
commit_description='', oid='0177c00b32f8c756b39beecda61121b805e8a9ce', pr_url=None,
pr_revision=None, pr_num=None)
```

Load adapters from the Hub

You can also directly load adapters from the Hub using the commands below:

```
import torch
from peft import PeftModel, PeftConfig
from transformers import AutoModelForCausalLM, AutoTokenizer

peft_model_id = "ashishpatel26/opt-6.1b-lora"
config = PeftConfig.from_pretrained(peft_model_id)
```

By: Tarun S Gowda

```
model = AutoModelForCausalLM.from_pretrained(config.base_model_name_or_path,
return_dict=True, load_in_8bit=True, device_map='auto')
tokenizer = AutoTokenizer.from_pretrained(config.base_model_name_or_path)

# Load the Lora model
model = PeftModel.from_pretrained(model, peft_model_id)
```

```
Downloading (…)/adapter_config.json:   0%|          | 0.00/440 [00:00<?, ?B/s]
Downloading adapter_model.bin:   0%|          | 0.00/12.6M [00:00<?, ?B/s]
```

Interence

You can then directly use the trained model or the model that you have loaded from the 🤗 Hub for inference as you would do it usually in transformers.

```
batch = tokenizer("Two things are infinite: ", return_tensors='pt')

with torch.cuda.amp.autocast():
  output_tokens = model.generate(**batch, max_new_tokens=50)

print('\n\n', tokenizer.decode(output_tokens[0], skip_special_tokens=True))
```

```
Two things are infinite: The universe and human stupidity; and I'm not sure about the
universe. I'm not sure about the universe either.
```

As you can see by fine-tuning for few steps we have almost recovered the quote from Albert Einstein that is present in the training data

Finetune Falcon-7b on a Google colab

Welcome to this Google Colab that shows how to fine-tune the recent Falcon-7b model on a single Google colab and turn it into a chatbot

We will leverage PEFT library from Hugging Face ecosystem, as well as QLoRA for more memory efficient finetuning

By: Tarun S Gowda

Setup

Run the cells below to setup and install the required libraries. For our experiment we will need accelerate, peft, transformers, datasets and TRL to leverage the recent SFTTrainer. We will use bitsandbytes to quantize the base model into 4bit We will also install einops as it is a requirement to load Falcon models.

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

VBox(children=(HTML(value=' <img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv…

```
!pip install -q -U trl transformers accelerate git+https://github.com/huggingface/peft.git
!pip install -q datasets bitsandbytes einops wandb
```

```
Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.0/110.0 kB 2.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.5/7.5 MB 55.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251.2/251.2 kB 28.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 519.3/519.3 kB 45.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 26.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 107.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 71.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 13.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 21.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 15.6 MB/s eta 0:00:00
  Building wheel for peft (pyproject.toml) ... done
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.6/92.6 MB 9.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.2/42.2 kB 4.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 92.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 188.5/188.5 kB 22.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 215.6/215.6 kB 22.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 8.0 MB/s eta 0:00:00
  Building wheel for pathtools (setup.py) ... done ━
```

Dataset

For our experiment, we will use the Guanaco dataset, which is a clean subset of the OpenAssistant dataset adapted to train general purpose chatbots.

LLM-Finetuning with PEFT

The dataset can be found here

```
from datasets import load_dataset
 dataset_name = "timdettmers/openassistant-guanaco"
dataset = load_dataset(dataset_name, split="train")
```

Repo card metadata block was not found. Setting CardData to empty. WARNING:huggingface_hub.repocard:Repo card metadata block was not found. Setting CardData to empty.

Loading the model

In this section we will load the Falcon 7B model, quantize it in 4bit and attach LoRA adapters on it. Let's get started!

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


 model_name = "ybelkada/falcon-7b-sharded-bf16"


 bnb_config = BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_quant_type="nf4",
        bnb_4bit_compute_dtype=torch.float16,
)
model = AutoModelForCausalLM.from_pretrained(
         model_name,
         quantization_config=bnb_config,
         trust_remote_code=True
)
model.config.use_cache = False
```

```
Downloading (…)lve/main/config.json:   0%|          | 0.00/1.10k [00:00<?, ?B/s]
Downloading (…)/configuration_RW.py:   0%|          | 0.00/2.61k [00:00<?, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/tiiuae/falcon-7b:
- configuration_RW.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.

Downloading (…)main/modelling_RW.py:   0%|          | 0.00/47.6k [00:00<?, ?B/s]
```

By: Tarun S Gowda

```
A new version of the following files was downloaded from
https://huggingface.co/tiiuae/falcon-7b:
- modelling_RW.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.
Downloading (…)model.bin.index.json:   0%|          | 0.00/16.9k [00:00<?, ?B/s]
Downloading shards:   0%|          | 0/8 [00:00<?, ?it/s]
Downloading (…)l-00001-of-00008.bin:   0%|          | 0.00/1.92G [00:00<?, ?B/s]
Downloading (…)l-00002-of-00008.bin:   0%|          | 0.00/1.99G [00:00<?, ?B/s]
Downloading (…)l-00003-of-00008.bin:   0%|          | 0.00/1.91G [00:00<?, ?B/s]
Downloading (…)l-00004-of-00008.bin:   0%|          | 0.00/1.91G [00:00<?, ?B/s]
Downloading (…)l-00005-of-00008.bin:   0%|          | 0.00/1.99G [00:00<?, ?B/s]
Downloading (…)l-00006-of-00008.bin:   0%|          | 0.00/1.91G [00:00<?, ?B/s]
Downloading (…)l-00007-of-00008.bin:   0%|          | 0.00/1.91G [00:00<?, ?B/s]
Downloading (…)l-00008-of-00008.bin:   0%|          | 0.00/921M [00:00<?, ?B/s]
Loading checkpoint shards:   0%|          | 0/8 [00:00<?, ?it/s]
Downloading (…)neration_config.json:   0%|          | 0.00/116 [00:00<?, ?B/s]
```

Let's also load the tokenizer below

```
tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
 tokenizer.pad_token = tokenizer.eos_token
```

```
Downloading (…)okenizer_config.json:   0%|          | 0.00/180 [00:00<?, ?B/s]
Downloading (…)/main/tokenizer.json:   0%|          | 0.00/2.73M [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/281 [00:00<?, ?B/s]
```

Below we will load the configuration file in order to create the LoRA model. According to QLoRA paper, it is important to consider all linear layers in the transformer block for maximum performance. Therefore we will add dense, dense_h_to_4_h and dense_4h_to_h layers in the target modules in addition to the mixed query key value layer.

```
from peft import LoraConfig
lora_alpha = 16
lora_dropout = 0.1
lora_r = 64

peft_config = LoraConfig(
  lora_alpha=lora_alpha,
 lora_dropout=lora_dropout,
r=lora_r,
bias="none",
task_type="CAUSAL_LM",
 target_modules=[
    "query_key_value",
    "dense", "dense_h_to_4h",
     "dense_4h_to_h",
 ]
```

```
)
```

Loading the trainer

Here we will use the SFTTrainer from TRL library that gives a wrapper around transformers Trainer to easily fine-tune models on instruction-based datasets using PEFT adapters. Let's first load the training arguments below.

```
from transformers import TrainingArguments
 output_dir = "./results"
per_device_train_batch_size = 4
gradient_accumulation_steps = 4
optim = "paged_adamw_32bit"
save_steps = 10
logging_steps = 10
learning_rate = 2e-4
max_grad_norm = 0.3
max_steps = 200
warmup_ratio = 0.03
lr_scheduler_type = "constant"

training_arguments = TrainingArguments(
    output_dir=output_dir,
    per_device_train_batch_size=per_device_train_batch_size,
    gradient_accumulation_steps=gradient_accumulation_steps,
   optim=optim,
 save_steps=save_steps,
logging_steps=logging_steps,
 learning_rate=learning_rate,
 fp16=True,
max_grad_norm=max_grad_norm,
max_steps=max_steps,
warmup_ratio=warmup_ratio,
group_by_length=True,
lr_scheduler_type=lr_scheduler_type,
)
```

Then finally pass everthing to the trainer

```
from trl import SFTTrainer
max_seq_length = 200
trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
   peft_config=peft_config,
  dataset_text_field="text",
 max_seq_length=max_seq_length,
tokenizer=tokenizer,
args=training_arguments,
)
```

/usr/local/lib/python3.10/dist-packages/peft/utils/other.py:122: FutureWarning: prepare_model_for_int8_training is deprecated and will be removed in a future version. Use prepare_model_for_kbit_training instead.

warnings.warn(

```
Map:   0%|          | 0/9846 [00:00<?, ? examples/s]
```

We will also pre-process the model by upcasting the layer norms in float 32 for more stable training

```
for name,
module in trainer.model.named_modules():
if "norm" in name:
    module = module.to(torch.float32)
```

Train the model

Now let's train the model! Simply call trainer.train()

```
trainer.train()
```

[200/200 52:33, Epoch 0/1]

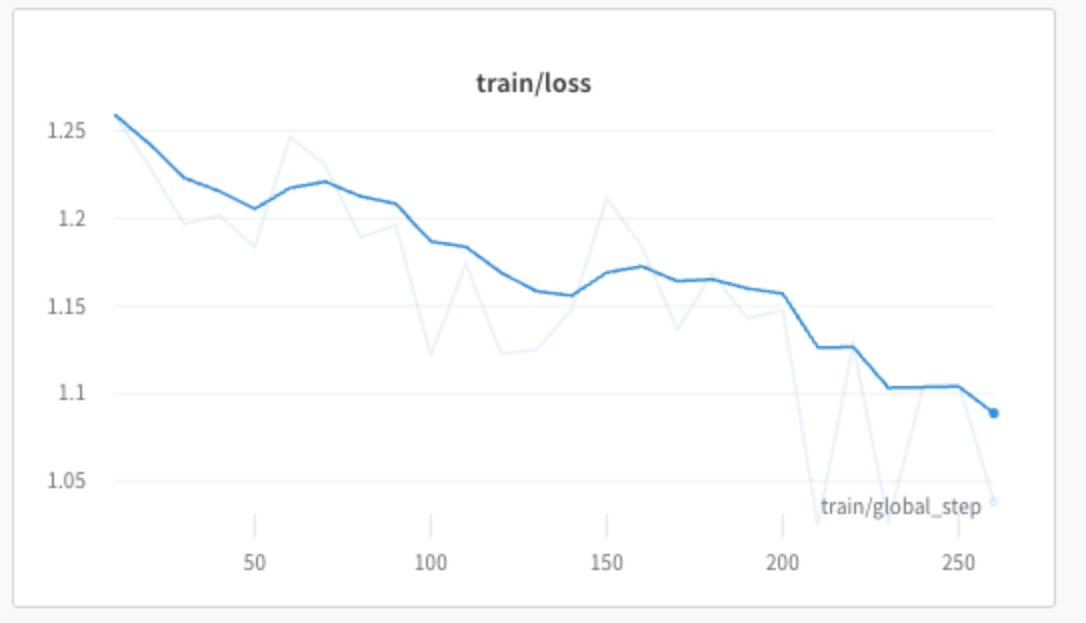
Step	Training Loss
10	1.56546466
20	1.36455623

30	1.43664267
40	1.64647467
50	1.64565454
60	1.54654345
70	1.45665456
80	1.55353553
90	1.45365463
100	1.85764555
110	1.74567456
120	1.45665452
130	1.56623436
140	1.54646655
150	1.55675775
160	1.98665764
170	1.56254623
180	1.78756766
190	1.57347545
200	1.86765455

```
TrainOutput(global_step=200, training_loss=1.5155159997940064, metrics={'train_runtime':
3168.9705, 'train_samples_per_second': 1.01, 'train_steps_per_second': 0.063,
'total_flos': 1.1841848794073088e+16, 'train_loss': 1.5155159997940064, 'epoch': 0.32})
```

During training, the model should converge nicely as follows:

LLM-Finetuning with PEFT



The SFTTrainer also takes care of properly saving only the adapters during training instead of saving the entire model.

```
model.push_to_hub("ashishpatel26/falcon-7b-sharded-bf16", create_pr=1, use_auth_token=True)
```

FineTune_LLAMA2_with_QLORA.ipynb

By: Tarun S Gowda

```
!nvidia-smi
```

```
Mon Aug 28 07:20:30 2023 +-----------------------------------------------------------------------------+ | NVIDIA-SMI
525.105.17 Driver Version: 525.105.17 CUDA Version: 12.0 | |-------------------------------+----------------------+----------------------
-------+ | GPU Name Persistence-M| Bus-Id Disp.A | Volatile Uncorr. ECC | | Fan Temp Perf Pwr:Usage/Cap| Memory-
Usage | GPU-Util Compute M. | | | | MIG M. |
|===============================+======================+======================| | 0 Tesla T4
Off | 00000000:00:04.0 Off | 0 | | N/A 48C P8 10W / 70W | 0MiB / 15360MiB | 0% Default | | | | N/A | +-----------------------
---------+----------------------+----------------------+ +-----------------------------------------------------------------------------+ |
Processes: | | GPU GI CI PID Type Process name GPU Memory | | ID ID Usage |
|=============================================================================| | No running
processes found | +-----------------------------------------------------------------------------+
```

Finetune Llama-2-7b on a Google colab

Welcome to this Google Colab that shows how to fine-tune the recent Llama-2-7b model on a single Google colab and turn it into a chatbot

We will leverage PEFT library from Hugging Face ecosystem, as well as QLoRA for more memory efficient finetuning

Setup

Run the cells below to setup and install the required libraries. For our experiment we will need accelerate, peft, transformers, datasets and TRL to leverage the recent SFTTrainer. We will use bitsandbytes to quantize the base model into 4bit. We will also install einops as it is a requirement to load Falcon models.

```
!pip install -q -U trl transformers accelerate git+https://github.com/huggingface/peft.git
```

```
!pip install -q datasets bitsandbytes einops wandb
```

```
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.2/42.2 kB 1.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 31.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 188.5/188.5 kB 20.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 215.6/215.6 kB 24.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 8.3 MB/s eta 0:00:00
  Building wheel for pathtools (setup.py) ... don
```

By: Tarun S Gowda

Dataset

```
from huggingface_hub import login
login()
```

VBox(children=(HTML(value=' <img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv…

```
from datasets import load_dataset
dataset_name = 'nisaar/Articles_Constitution_3300_Instruction_Set'
dataset = load_dataset(dataset_name, split="train")
```

```
Downloading readme:   0%|          | 0.00/1.54k [00:00<?, ?B/s]
Downloading and preparing dataset json/nisaar--Articles_Constitution_3300_Instruction_Set
to /root/.cache/huggingface/datasets/nisaar___json/nisaar--
Articles_Constitution_3300_Instruction_Set-
f7dd4b93e884d248/0.0.0/8bb11242116d547c741b2e8a1f18598ffdd40a1d4f2a2872c7a28b697434bc96..
.

Downloading data files:   0%|          | 0/1 [00:00<?, ?it/s]
Downloading data:   0%|          | 0.00/8.01M [00:00<?, ?B/s]
Extracting data files:   0%|          | 0/1 [00:00<?, ?it/s]
Generating train split: 0 examples [00:00, ? examples/s]
Dataset json downloaded and prepared to
/root/.cache/huggingface/datasets/nisaar___json/nisaar--
Articles_Constitution_3300_Instruction_Set-
f7dd4b93e884d248/0.0.0/8bb11242116d547c741b2e8a1f18598ffdd40a1d4f2a2872c7a28b697434bc96.
Subsequent calls will reuse this data.
```

```
dataset[1]
```

{'instruction': 'Identify and summarize the key legal issues in the provided case.', 'input': 'Case Citation: Central Inland Water Transport Corporation Ltd. vs Brojo Nath Ganguly & Anr., 1986 AIR 1571, 1986 SCR (2) 278. The case revolves around the termination of employment of the respondents by the appellant without assigning any reason by invoking Rule 9(i) of the service rules. The respondents challenged the termination orders and the validity of Rule 9(i) in the High Court under Article 226. The Division Bench of the High Court struck down Rule 9(i) as arbitrary and violative of Article 14. The appellant filed appeals in the Supreme Court against the High Court judgment.', 'output': "The key legal issues in the case Central Inland Water Transport Corporation Ltd. vs Brojo Nath Ganguly & Anr. are as follows: 1. The first issue is whether a government company falls under the definition of 'State' as per Article 12 of the Indian Constitution. This is important as the constitutional protections against arbitrary state action apply to 'State' entities. 2. The second issue is the constitutional validity of Rule 9(i) that allows termination of service without reason. The respondents challenged this rule as arbitrary and violative of the right to equality under Article 14 of the Constitution. 3. The third issue is the application of principles of natural justice and constitutional protections in contractual relationships, particularly in employment contracts of state entities. 4. The fourth issue is the impact of arbitrary termination of employment on the right to life and livelihood under Article 21 of the

Constitution. 5. The final issue is the scope of judicial review of state action in contractual matters, particularly in the context of employment relationships.", '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.\n\n### Instruction:\nIdentify and summarize the key legal issues in the provided case.\n\n### Input:\nCase Citation: Central Inland Water Transport Corporation Ltd. vs Brojo Nath Ganguly & Anr., 1986 AIR 1571, 1986 SCR (2) 278. The case revolves around the termination of employment of the respondents by the appellant without assigning any reason by invoking Rule 9(i) of the service rules. The respondents challenged the termination orders and the validity of Rule 9(i) in the High Court under Article 226. The Division Bench of the High Court struck down Rule 9(i) as arbitrary and violative of Article 14. The appellant filed appeals in the Supreme Court against the High Court judgment.\n\n### Response:'}

```
print(dataset)
```

```
Dataset({
    features: ['instruction', 'input', 'output', 'prompt'],
    num_rows: 3311
})
```

```
from datasets import Dataset
# Assuming `dataset` is your Dataset object
dataset = dataset.map(lambda example: {'text': example['prompt'] + example['output']})
```

```
Map: 0%| | 0/3311 [00:00<?, ? examples/s]
```

```
Dataset
```

```
Dataset({
    features: ['instruction', 'input', 'output', 'prompt', 'text'],
    num_rows: 3311
})
```

Loading the model

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

By: Tarun S Gowda

LLM-Finetuning with PEFT

```
model_name = "TinyPixel/Llama-2-7B-bf16-sharded"

bnb_config = BitsAndBytesConfig(
     load_in_4bit=True,
     bnb_4bit_quant_type="nf4",
     bnb_4bit_compute_dtype=torch.float16,
 )


model = AutoModelForCausalLM.from_pretrained(
    model_name,
    quantization_config=bnb_config,
    trust_remote_code=True
)
model.config.use_cache = False
```

```
Downloading (…)lve/main/config.json:   0%|          | 0.00/626 [00:00<?, ?B/s]
Downloading (…)model.bin.index.json:   0%|          | 0.00/26.8k [00:00<?, ?B/s]
Downloading shards:   0%|          | 0/14 [00:00<?, ?it/s]
Downloading (…)l-00001-of-00014.bin:   0%|          | 0.00/981M [00:00<?, ?B/s]
Downloading (…)l-00002-of-00014.bin:   0%|          | 0.00/967M [00:00<?, ?B/s]
Downloading (…)l-00003-of-00014.bin:   0%|          | 0.00/967M [00:00<?, ?B/s]
Downloading (…)l-00004-of-00014.bin:   0%|          | 0.00/990M [00:00<?, ?B/s]
Downloading (…)l-00005-of-00014.bin:   0%|          | 0.00/944M [00:00<?, ?B/s]
Downloading (…)l-00006-of-00014.bin:   0%|          | 0.00/990M [00:00<?, ?B/s]
Downloading (…)l-00007-of-00014.bin:   0%|          | 0.00/967M [00:00<?, ?B/s]
Downloading (…)l-00008-of-00014.bin:   0%|          | 0.00/967M [00:00<?, ?B/s]
Downloading (…)l-00009-of-00014.bin:   0%|          | 0.00/990M [00:00<?, ?B/s]
Downloading (…)l-00010-of-00014.bin:   0%|          | 0.00/944M [00:00<?, ?B/s]
Downloading (…)l-00011-of-00014.bin:   0%|          | 0.00/990M [00:00<?, ?B/s]
Downloading (…)l-00012-of-00014.bin:   0%|          | 0.00/967M [00:00<?, ?B/s]
Downloading (…)l-00013-of-00014.bin:   0%|          | 0.00/967M [00:00<?, ?B/s]
Downloading (…)l-00014-of-00014.bin:   0%|          | 0.00/847M [00:00<?, ?B/s]
Loading checkpoint shards:   0%|          | 0/14 [00:00<?, ?it/s]
Downloading (…)neration_config.json:   0%|          | 0.00/132 [00:00<?, ?
```

```
tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
tokenizer.pad_token = tokenizer.eos_token
```

By: Tarun S Gowda

```
Downloading (…)okenizer_config.json:   0%|          | 0.00/676 [00:00<?, ?B/s]
Downloading tokenizer.model:   0%|          | 0.00/500k [00:00<?, ?B/s]
Downloading (…)/main/tokenizer.json:   0%|          | 0.00/1.84M [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/411 [00:00<?, ?B/s]
```

```
from peft import LoraConfig, get_peft_model

lora_alpha = 16
 lora_dropout = 0.1
lora_r = 64


peft_config = LoraConfig(
      lora_alpha=lora_alpha,
      lora_dropout=lora_dropout,
      r=lora_r,
      bias="none",
      task_type="CAUSAL_LM"
)


from transformers import TrainingArguments


output_dir = "./results"
per_device_train_batch_size = 1
gradient_accumulation_steps = 2
 optim = "paged_adamw_32bit"
save_steps = 1
num_train_epochs = 4
logging_steps = 1
 learning_rate = 2e-4
 max_grad_norm = 0.3
max_steps = 20
warmup_ratio = 0.03
lr_scheduler_type = "linear"


training_arguments = TrainingArguments(
      output_dir=output_dir,
      per_device_train_batch_size=per_device_train_batch_size,
     gradient_accumulation_steps=gradient_accumulation_steps,
     optim=optim,
     num_train_epochs=num_train_epochs,
    save_steps=save_steps,
    logging_steps=logging_steps,
     learning_rate=learning_rate,
      fp16=True,
      max_grad_norm=max_grad_norm,
      max_steps=max_steps,
      warmup_ratio=warmup_ratio,
      group_by_length=True,
      lr_scheduler_type=lr_scheduler_type,
```

```
)

from trl import SFTTrainer


max_seq_length = 2048


trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
    peft_config=peft_config,
    dataset_text_field="text",
    max_seq_length=max_seq_length,
    tokenizer=tokenizer,
    args=training_arguments,
)
```

WARNING:datasets.arrow_dataset:Loading cached processed dataset at /root/.cache/huggingface/datasets/nisaar___json/nisaar--Articles_Constitution_3300_Instruction_Set-f7dd4b93e884d248/0.0.0/8bb11242116d547c741b2e8a1f18598ffdd40a1d4f2a2872c7a28b697434bc96/cache-d315fa7f4915eeac.arrow

```
for name, module in trainer.model.named_modules():
 if "norm" in name:
        module = module.to(torch.float32)
```

```
trainer.train()
```

[20/20 02:45, Epoch 0/1]

[20/20 02:45, Epoch 0/1]

Step	Training Loss
1	1.080600
2	1.084400
3	1.225800
4	1.244200
5	1.032500
6	1.129500
7	1.243900
8	1.116600
9	1.015100
10	1.170400
11	1.081400
12	1.108300

13	1.026100
14	1.239200
15	1.406800
16	1.019600
17	1.150400
18	1.168400
19	1.203600
20	1.090500

```
TrainOutput(global_step=20, training_loss=1.1418639481067658, metrics={'train_runtime':
178.0812, 'train_samples_per_second': 0.225, 'train_steps_per_second': 0.112,
'total_flos': 558884728135680.0, 'train_loss': 1.1418639481067658, 'epoch': 0.01})
```

```
model_to_save = trainer.model.module if hasattr(trainer.model, 'module') else
trainer.model # Take care of distributed/parallel training
model_to_save.save_pretrained("outputs")
```

```
lora_config = LoraConfig.from_pretrained('outputs')
model = get_peft_model(model, lora_config)
```

```
model.push_to_hub("ashishpatel26/Llama2_Finetuned_Articles_Constitution_3300_Instructio
n_Set",create_pr=1)
```

```
adapter_model.bin: 0%| | 0.00/134M [00:00<?, ?B/s]
```

```
CommitInfo(commit_url='https://huggingface.co/ashishpatel26/Llama2_Finetuned_Articles_C
onstitution_3300_Instruction_Set/commit/0e6e09e0e362c05fdb5bee1d181514793a1eecaa',
commit_message='Upload model', commit_description='',
oid='0e6e09e0e362c05fdb5bee1d181514793a1eecaa',
pr_url='https://huggingface.co/ashishpatel26/Llama2_Finetuned_Articles_Constitution_330
0_Instruction_Set/discussions/1', pr_revision='refs/pr/1', pr_num=1)
```

```
dataset['text'][0]
```

```
"Below is an instruction that describes a task, paired with an input that provides
further context. Write a response that appropriately completes the request.\n\n###
Instruction:\nAnalyze and explain the legal reasoning behind the judgment in the given
case.\n\n### Input:\nCentral Inland Water Transport Corporation Ltd. vs Brojo Nath
Ganguly & Anr., 1986 AIR 1571, 1986 SCR (2) 278\n\n### Response:The Supreme Court in this
case applied a broad interpretation of the term 'State' under Article 12 of the
Constitution. The court reasoned that a government company undertaking public functions
qualifies as 'State' based on factors like government control, public importance of
```

```
activities etc. This interpretation was based on previous decisions that have defined
'State' under Article 12 broadly to include various agencies and instrumentalities beyond
just statutory bodies. The court also applied the principle that unreasonable and
arbitrary contractual terms can be struck down under Article 14 of the Constitution. The
court found that Rule 9(i) of the service rules, which allowed for termination of service
without reason, conferred unfettered power to terminate employment without hearing. This
was deemed arbitrary and violative of principles of natural justice and right to equality
under Article 14. Furthermore, the court held that the right to life and livelihood under
Article 21 is affected by arbitrary termination of employment. The court reasoned that
the right to livelihood is an integral part of the right to life, and any arbitrary
action that affects a person's livelihood would be a violation of Article 21.In
conclusion, the court's legal reasoning was based on a broad interpretation of the term
'State', the application of the principle of equality and natural justice under Article
14, and the protection of the right to life and livelihood under Article 21."
```

```
dataset['output'][0]
```

```
"The Supreme Court in this case applied a broad interpretation of the term 'State' under
Article 12 of the Constitution. The court reasoned that a government company undertaking
public functions qualifies as 'State' based on factors like government control, public
importance of activities etc. This interpretation was based on previous decisions that
have defined 'State' under Article 12 broadly to include various agencies and
instrumentalities beyond just statutory bodies. The court also applied the principle that
unreasonable and arbitrary contractual terms can be struck down under Article 14 of the
Constitution. The court found that Rule 9(i) of the service rules, which allowed for
termination of service without reason, conferred unfettered power to terminate employment
without hearing. This was deemed arbitrary and violative of principles of natural justice
and right to equality under Article 14. Furthermore, the court held that the right to
life and livelihood under Article 21 is affected by arbitrary termination of employment.
The court reasoned that the right to livelihood is an integral part of the right to life,
and any arbitrary action that affects a person's livelihood would be a violation of
Article 21.In conclusion, the court's legal reasoning was based on a broad interpretation
of the term 'State', the application of the principle of equality and natural justice
under Article 14, and the protection of the right to life and livelihood under Article
21."
```

```
text = dataset['text'][0] device = "cuda:0"
device = "cuda:0"
```

```
inputs = tokenizer(text, return_tensors="pt").to(device) outputs = model.generate(**inputs, max_new_tokens=50)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

```
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:
Analyze and explain the legal reasoning behind the judgment in the given case.
```

```
### Input:
Central Inland Water Transport Corporation Ltd. vs Brojo Nath Ganguly & Anr., 1986 AIR
1571, 1986 SCR (2) 278



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:
Analyze and explain the legal reasoning behind the judgment in the given case. ### Input:
Central Inland Water Transport Corporation Ltd. vs Brojo Nath Ganguly & Anr., 1986 AIR
1571, 1986 SCR (2) 278 ### Response:The Supreme Court in this case applied a broad
interpretation of the term 'State' under Article 12 of the Constitution. The court
reasoned that a government company undertaking public functions qualifies as 'State'
based on factors like government control, public importance of activities etc. This
interpretation was based on previous decisions that have defined 'State' under Article 12
broadly to include various agencies and instrumentalities beyond just statutory bodies.
The court also applied the principle that unreasonable and arbitrary contractual terms
can be struck down under Article 14 of the Constitution. The court found that Rule 9(i)
of the service rules, which allowed for termination of service without reason, conferred
unfettered power to terminate employment without hearing. This was deemed arbitrary and
violative of principles of natural justice and right to equality under Article 14.
Furthermore, the court held that the right to life and livelihood under Article 21 is
affected by arbitrary termination of employment. The court reasoned that the right to
livelihood is an integral part of the right to life, and any arbitrary action that
affects a person's livelihood would be a violation of Article 21.In conclusion, the
court's legal reasoning was based on a broad interpretation of the term 'State', the
application of the principle of equality and natural justice under Article 14, and the
protection of the right to life and livelihood under Article 21.

### Instruction:
Analyze and explain the legal reasoning behind the judgment in the given case.

### Input:
 M. Nagaraj vs Union of India, 2006 AIR 1
```

Stable_Vicuna13B_8bit_in_Colab.ipynb

```
!pip -q install git+https://github.com/huggingface/transformers # need to install from github
!pip install -q datasets loralib sentencepiece
!pip -q install bitsandbytes accelerate

Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
```

By: Tarun S Gowda

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 224.5/224.5 kB 6.1 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 78.4 MB/s eta 0:00:00
  Building wheel for transformers (pyproject.toml) ... done
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 474.6/474.6 kB 2.7 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 36.5 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.3/134.3 kB 16.6 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 kB 13.1 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 67.3 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 212.5/212.5 kB 26.5 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.5/114.5 kB 14.1 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 29.3 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 149.6/149.6 kB 18.3 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 104.3/104.3 MB 15.9 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 215.3/215.3 kB 23.7 MB/s eta 0:00:00
```

StableVicuna- RLHF Chat model

!nvidia-smi

Sun Apr 30 00:07:14 2023 +---+ | NVIDIA-SMI 525.85.12 Driver Version: 525.85.12 CUDA Version: 12.0 | |-------------------------------+----------------------+----------------------+ | GPU Name Persistence-M| Bus-Id Disp.A | Volatile Uncorr. ECC | | Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. | | | MIG M. | |===============================+======================+======================| | 0 NVIDIA A100-SXM... Off | 00000000:00:04.0 Off | 0 | | N/A 30C P0 47W / 400W | 0MiB / 40960MiB | 0% Default | | | Disabled | +-------------------------------+----------------------+----------------------+ +---+ | Processes: | | GPU GI CI PID Type Process name GPU Memory | | ID ID Usage | |===| | No running processes found | +---+

```
from transformers import LlamaTokenizer, LlamaForCausalLM, GenerationConfig, pipeline
import torch
```

```
tokenizer = LlamaTokenizer.from_pretrained("TheBloke/stable-vicuna-13B-HF")
```

```
base_model = LlamaForCausalLM.from_pretrained(
        "TheBloke/stable-vicuna-13B-HF",
        load_in_8bit=True,
        device_map='auto',
)
```

```
Downloading tokenizer.model:   0%|          | 0.00/500k [00:00<?, ?B/s]
```

By: Tarun S Gowda

```
Downloading (…)in/added_tokens.json:   0%|          | 0.00/21.0 [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/96.0 [00:00<?, ?B/s]
Downloading (…)okenizer_config.json:   0%|          | 0.00/715 [00:00<?, ?B/s]
Overriding torch_dtype=None with `torch_dtype=torch.float16` due to requirements of
`bitsandbytes` to enable model loading in mixed int8. Either pass
torch_dtype=torch.float16 or don't pass this argument at all to remove this warning.

Downloading (…)lve/main/config.json:   0%|          | 0.00/587 [00:00<?, ?B/s]
Downloading (…)model.bin.index.json:   0%|          | 0.00/33.4k [00:00<?, ?B/s]
Downloading shards:   0%|          | 0/3 [00:00<?, ?it/s]
Downloading (…)l-00001-of-00003.bin:   0%|          | 0.00/9.95G [00:00<?, ?B/s]
Downloading (…)l-00002-of-00003.bin:   0%|          | 0.00/9.90G [00:00<?, ?B/s]
Downloading (…)l-00003-of-00003.bin:   0%|          | 0.00/6.18G [00:00<?, ?B/s]
================================BUG REPORT================================
Welcome to bitsandbytes. For bug reports, please run

python -m bitsandbytes

 and submit this information together with your error trace to:
https://github.com/TimDettmers/bitsandbytes/issues
==========================================================================
bin /usr/local/lib/python3.10/dist-packages/bitsandbytes/libbitsandbytes_cuda118.so
CUDA_SETUP: WARNING! libcudart.so not found in any environmental path. Searching in
backup paths...
CUDA SETUP: CUDA runtime path found: /usr/local/cuda/lib64/libcudart.so.11.0
CUDA SETUP: Highest compute capability among GPUs detected: 8.0
CUDA SETUP: Detected CUDA version 118
CUDA SETUP: Loading binary /usr/local/lib/python3.10/dist-
packages/bitsandbytes/libbitsandbytes_cuda118.so...

/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:145: UserWarning:
/usr/lib64-nvidia did not contain ['libcudart.so', 'libcudart.so.11.0',
'libcudart.so.12.0'] as expected! Searching further paths...
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:145: UserWarning:
WARNING: The following directories listed in your path were found to be non-existent:
{PosixPath('/sys/fs/cgroup/memory.events /var/colab/cgroup/jupyter-
children/memory.events')}
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:145: UserWarning:
WARNING: The following directories listed in your path were found to be non-existent:
{PosixPath('http'), PosixPath('//172.28.0.1'), PosixPath('8013')}
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:145: UserWarning:
WARNING: The following directories listed in your path were found to be non-existent:
{PosixPath('//colab.research.google.com/tun/m/cc48301118ce562b961b3c22d803539adc1e0c19/gp
u-a100-s-1uropttk4xx0r --tunnel_background_save_delay=10s --
tunnel_periodic_background_save_frequency=30m0s --enable_output_coalescing=true --
output_coalescing_required=true'), PosixPath('--logtostderr --listen_host=172.28.0.12 --
target_host=172.28.0.12 --tunnel_background_save_url=https')}
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:145: UserWarning:
```

```
WARNING: The following directories listed in your path were found to be non-existent:
{PosixPath('/env/python')}
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:145: UserWarning:
WARNING: The following directories listed in your path were found to be non-existent:
{PosixPath('module'), PosixPath('//ipykernel.pylab.backend_inline')}
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:145: UserWarning:
Found duplicate ['libcudart.so', 'libcudart.so.11.0', 'libcudart.so.12.0'] files:
{PosixPath('/usr/local/cuda/lib64/libcudart.so.11.0'),
PosixPath('/usr/local/cuda/lib64/libcudart.so')}.. We'll flip a coin and try one of
these, in order to fail forward.
Either way, this might cause trouble in the future:
If you get `CUDA error: invalid device function` errors, the above might be the cause and
the solution is to make sure only one ['libcudart.so', 'libcudart.so.11.0',
'libcudart.so.12.0'] in the paths that we search based on your env.
  warn(msg)
```

```
Loading checkpoint shards:   0%|          | 0/3 [00:00<?, ?it/s]
Downloading (…)neration_config.json:   0%|          | 0.00/137 [00:00<?, ?B/s]
```

```
pipe = pipeline(
    "text-generation",
     model=base_model,
     tokenizer=tokenizer,
     max_length=512,
     temperature=0.7,
     top_p=0.95,
      repetition_penalty=1.15
)
```

Xformers is not installed correctly. If you want to use memorry_efficient_attention to accelerate training use the following command to install Xformers pip install xformers.

The prompt and response

```
import json
import textwrap

 human_prompt = 'What is the meaning of life?'

def get_prompt(human_prompt):
    prompt_template=f"### Human: {human_prompt} \n### Assistant:"
    return prompt_template
```

By: Tarun S Gowda

```
print(get_prompt('What is the meaning of life?'))
def remove_human_text(text):
    return text.split('### Human:', 1)[0]

def parse_text(data):
    for item in data:
        text = item['generated_text']
        assistant_text_index = text.find('### Assistant:')
        if assistant_text_index != -1:
            assistant_text = text[assistant_text_index+len('### Assistant:'):]strip()
            assistant_text = remove_human_text(assistant_text)
            wrapped_text = textwrap.fill(assistant_text, width=100)
            print(wrapped_text)

data = [{'generated_text': '### Human: What is the capital of England? \n### Assistant: The capital city of England is London.'}]
parse_text(data)
```

```
### Human: What is the meaning of life?
### Assistant:
The capital city of England is London.
```

Run it as a HF model

```
%%time
raw_output = pipe(get_prompt('What are the difference between Llamas, Alpacas and Vicunas?')) parse_text(raw_output)
```

```
Llamas, alpacas, and vicuñas are all members of the camelid family. However, there are
some
differences between them:  1. Appearance: Llamas have a larger build than both alpacas
and vicuñas.
They also have longer ears and a shorter neck. Alpacas are smaller in size with a more
slender build
compared to llamas. Their ears are shorter and their necks are longer. Vicuñas are the
smallest of
the three species and have a more delicate appearance with shorter legs and a finer coat.
2. Coat:
The coats of these animals vary greatly. Llamas have a thick, woolly coat that can be
either
straight or curly. Alpaca fleece is soft and silky while vicuña fleece is the finest and
most
expensive type of wool.  3. Uses: Llamas are used for packing and transportation
purposes, as well
as for their meat and milk. Alpacas are primarily raised for their fiber which is highly
valued due
to its softness and warmth. Vicuñas are also raised for their fiber but it is considered
the rarest
```

By: Tarun S Gowda

```
and most valuable type of wool.  4. Habitat: Llamas are native to South America,
particularly in
Peru, Chile, Bolivia, and Argentina. Alpacas are found mainly in the Andes mountain range
of South
America, specifically in Peru, Chile, Bolivia, and Argentina. Vicuñas are native to the
highlands of
South America, including Peru, Chile, and Bolivia.
CPU times: user 1min 23s, sys: 0 ns, total: 1min 23s
Wall time: 1min 23s
```

```
%%time raw_output = pipe(get_prompt('Write a short note to Sam Altman giving reasons to
open source GPT-4'))
parse_text(raw_output)
```

```
Dear Sam,  I believe that opening up the development of GPT-4 would have numerous
benefits for both
the AI community and society as a whole. Here are some reasons why I think this is
important:  1.
Increased collaboration and innovation - Open sourcing GPT-4 would allow researchers from
around the
world to work together on improving the model's capabilities and performance. This could
lead to
faster progress in the field and more creative solutions to complex problems.  2.
Improved
transparency and accountability - With an open source platform, anyone can see how the
model works
and what data it uses to generate its outputs. This increased transparency can help build
trust with
users and stakeholders who may be concerned about the ethics and fairness of AI systems.
3. Lower
barriers to entry - By making GPT-4 available to everyone, we can democratize access to
powerful
language processing tools. This could enable new voices and perspectives to emerge in
fields like
journalism, literature, and social media.  Overall, I believe that opening up GPT-4 has
the
potential to unlock tremendous value for individuals, organizations, and societies alike.
Thank you
for considering my perspective.
CPU times: user 1min 5s, sys: 0 ns, total: 1min 5s
Wall time: 1min 5s
```

```
%%time
raw_output = pipe(get_prompt('What is the capital of England?'))
parse_text(raw_output)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1078: UserWarning:
You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency
```

```
please use a dataset
  warnings.warn(

The capital city of England is London.
CPU times: user 2.22 s, sys: 0 ns, total: 2.22 s
Wall time: 2.21 s
```

```
%%time
raw_output = pipe(get_prompt('Write a story about a Koala playing pool and beating all
the camelids.'))
parse_text(raw_output)
```

```
Once upon a time, there was a brave little koala named Kody who loved to play pool with
his friends at the local bar. One day, he decided to challenge himself by taking on some
of the toughest opponents in town - the camels! Kody approached the table with
confidence, ready for whatever challenges lay ahead. The first match began, and Kody
quickly proved that he was no slouch when it came to aiming and hitting those balls. He
easily beat the first two camels, but things got more difficult as he moved onto the next
round. The third camel was particularly skilled, and Kody found himself struggling to
keep up. But just when all seemed lost, something amazing happened. Kody's tiny paws were
able to maneuver around the larger, clumsier feet of the camels, allowing him to make
shots they simply couldn't. One by one, Kody defeated each of the camels, leaving them in
shock and admiration of his skills. By the end of the night, Kody had become the champion
of the pool hall, and everyone knew that if you wanted to win, you had to go up against
the fearless little koala. CPU times: user 1min 1s, sys: 107 ms, total: 1min 1s Wall
time: 1min 1s
```

```
%%time
raw_output = pipe(get_prompt('As an AI do you like the Simpsons? What dow you know about
Homer?'))
parse_text(raw_output)
```

```
usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1078: UserWarning:
You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency
please use a dataset
  warnings.warn(
```

```
Yes, I am a fan of The Simpsons. It is one of my favorite TV shows and has been around
for many
years. Homer is one of the main characters in the show and he is known for his love of
donuts, beer,
and his wife Marge. He works at the Springfield Nuclear Power Plant as a safety inspector
and often
gets into trouble due to his carelessness and lack of attention to detail. Despite this,
he loves
his family and will go to great lengths to protect them. Overall, Homer is a lovable
character with
plenty of flaws that make him relatable to audiences.
CPU times: user 33.4 s, sys: 66.7 ms, total: 33.5 s
Wall time: 33.4 s
```

By: Tarun S Gowda

```
%%time
raw_output = pipe(get_prompt('Answer the following question by reasoning step by step.
The cafeteria had 23 apples. If they used 20 for lunch, and bought 6 more, how many apple
do they have?'))
parse_text(raw_output)
```

They started with 23 apples. After using 20 for lunch, they were left with 3 apples. Then, they bought 6 more apples, so now they have a total of 9 apples. Therefore, the cafeteria has 9 apples in total.
CPU times: user 37.5 s, sys: 64.2 ms, total: 37.6 s Wall time: 37.4 s

```
%%time
 raw_output = pipe(get_prompt('Answer the following yes\/no question by reasoning step-by-step. \n Can you write a
whole Haiku in a single tweet?'))
parse_text(raw_output)
```

No, it is not possible to write a complete haiku poem within a single tweet as Twitter has a character limit of 280 characters per tweet. A traditional haiku consists of three lines with a syllable count of 5-7-5 respectively. Therefore, it would be impossible to fit all three lines into just one tweet without sacrificing the structure and meaning of the poem.
CPU times: user 1min 52s, sys: 203 ms, total: 1min 52s Wall time: 1min 52s

```
%%time
raw_output = pipe(get_prompt('Can Geoffrey Hinton have a conversation with George
Washington? Give the rationale before answering.'))
parse_text(raw_output)
```

```
No, it is not possible for Geoffrey Hinton to have a conversation with George Washington
as they lived in different centuries and were born over two hundred years apart.
Additionally, communication between people from different time periods would require some
form of time travel which has yet to be discovered or developed. CPU times: user 1min
11s, sys: 107 ms, total: 1min 11s Wall time: 1min 11s
```

```
%%time
raw_output = pipe(get_prompt('Can Geoffrey Hinton have a conversation with George
Washington? Give the rationale before answering.'))
parse_text(raw_output)
```

```
No, it is not possible for Geoffrey Hinton to have a conversation with George Washington
as they
lived in different centuries and were born over two hundred years apart. Additionally,
communication
between people from different time periods would require some form of time travel which
has yet to
be discovered or developed.
```

By: Tarun S Gowda

```
CPU times: user 1min 11s, sys: 107 ms, total: 1min 11s
Wall time: 1min 11s
```

```
%%time
raw_output = pipe(get_prompt('Could Marcus Aurelius have had dinner with George Washington? Give the rationale before answering.'))
parse_text(raw_output)
```

No, it is highly unlikely that Marcus Aurelius and George Washington could have had dinner together as they lived in different centuries and continents. Marcus Aurelius was a Roman Emperor who ruled from 161 to 180 AD while George Washington was the first President of the United States who served from 1789 to 1797. It would be impossible for them to meet each other during their lifetimes due to the vast geographical distance between Rome (in Italy) and the United States (on the East Coast of North America). CPU times: user 1min 8s, sys: 111 ms, total: 1min 8s Wall time: 1min 8s

```
%%time
raw_output = pipe(get_prompt('tell me about 3 facts about Marcus Aurelius that most people dont know'))
parse_text(raw_output)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/base.py:1078: UserWarning:
You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency
please use a dataset
  warnings.warn(

Sure, here are three lesser-known facts about Marcus Aurelius: 1. He was born into a
wealthy and
influential family in Rome, but he rejected his privileged upbringing to pursue a career
in public
service. 2. During his reign as emperor, he implemented several reforms aimed at
improving the lives
of common citizens, such as reducing taxes and increasing access to education. 3. Despite
being one
of the most powerful men in the Roman Empire, he is known for his humility and modesty.
In fact, he
wrote extensively on the importance of living a simple life and avoiding excessive
ambition or
pride.
CPU times: user 34.6 s, sys: 65.3 ms, total: 34.7 s
Wall time: 34.6 s
```

```
%%time
raw_output = pipe(get_prompt('Who was Marcus Aureliuss son?'))
parse_text(raw_output)
```

```
The name of Marcus Aurelius's son is not known as there are no historical records
indicating that he
had a child. However, it is believed by some historians that his adopted brother and co-
```

```
emperor,
Lucius Verus, may have been his biological nephew or cousin.
CPU times: user 14.8 s, sys: 34.7 ms, total: 14.8 s
Wall time: 14.8 s
```

```
%%time
raw_output = pipe(get_prompt('Who was Marcus Aureliuss son and what was he like?'))
parse_text(raw_output)
```

```
Marcus Aurelius had a son named Commodus, who later became emperor of Rome. However,
Commodus is remembered for his tyrannical rule and eventual assassination. He was known
for his excessive spending on gladiatorial games and other extravagances, as well as his
cruelty towards those who opposed him. CPU times: user 1min 13s, sys: 132 ms, total: 1min
13s Wall time: 1min 13s
```

```
%%time
raw_output = pipe(get_prompt('Who was the emperor Commodus?'))
parse_text(raw_output)
```

```
Commodus was a Roman Emperor who ruled from AD 180 to 192. He is best known for his
excessive and
decadent lifestyle, which ultimately led to his assassination by a group of conspirators.
CPU times: user 12.6 s, sys: 26.6 ms, total: 12.7 s
Wall time: 12.6 s
```

Transformers meets bitsandbytes for democratzing Large Language Models (LLMs) through 4bit quantization

By: Tarun S Gowda

LLM-Finetuning with PEFT

Welcome in this part of this project, that goes through the recent `bitsandbytes` integration that includes the work from XXX that introduces no performance degradation 4bit quantization techniques, for democratizing LLMs inference and training.

In this part, we will learn together how to load a large model in 4bit (`gpt-neo-x-20b`) and train it using Google Colab and PEFT library from Hugging Face.

In the general usage part, you can learn how to properly load a model in 4bit with all its variants.

```
!pip install -q -U bitsandbytes
!pip install -q -U git+https://github.com/huggingface/transformers.git
!pip install -q -U git+https://github.com/huggingface/peft.git
!pip install -q -U git+https://github.com/huggingface/accelerate.git
!pip install -q datasets
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.6/92.6 MB 10.6 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 3.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 21.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 46.9 MB/s eta 0:00:00
  Building wheel for transformers (pyproject.toml) ... done
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251.2/251.2 kB 5.0 MB/s eta 0:00:00
```

By: Tarun S Gowda

```
  Building wheel for peft (pyproject.toml) ... done
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
  Building wheel for accelerate (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 519.3/519.3 kB 8.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 15.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 15.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 17.3 MB/s eta 0:00:00
```

```
from huggingface_hub import login
login()
```

```
VBox(children=(HTML(value='<center>
<img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv…
```

First let's load the model we are going to use - GPT-neo-x-20B! Note that the model itself is around 40GB in half precision

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

model_id = "EleutherAI/gpt-neox-20b"
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16
)
tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(model_id, quantization_config=, device_map={"":0})bnb_config
```

```
Downloading (…)okenizer_config.json:   0%|          | 0.00/156 [00:00<?, ?B/s]
Downloading (…)olve/main/vocab.json:   0%|          | 0.00/1.08M [00:00<?, ?B/s]
Downloading (…)olve/main/merges.txt:   0%|          | 0.00/457k [00:00<?, ?B/s]
Downloading (…)/main/tokenizer.json:   0%|          | 0.00/2.11M [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/90.0 [00:00<?, ?B/s]
Downloading (…)lve/main/config.json:   0%|          | 0.00/613 [00:00<?, ?B/s]
Downloading (…)fetensors.index.json:   0%|          | 0.00/60.4k [00:00<?, ?B/s]
Downloading shards:   0%|          | 0/46 [00:00<?, ?it/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/926M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
```

By: Tarun S Gowda

```
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/910M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/604M [00:00<?, ?B/s]
Downloading (…)of-00046.safetensors:   0%|          | 0.00/620M [00:00<?, ?B/s]
Loading checkpoint shards:   0%|          | 0/46 [00:00<?, ?it/s]
```

Then we have to apply some preprocessing to the model to prepare it for training. For that use the `prepare_model_for_kbit_training` method from PEFT.

```
from peft import prepare_model_for_kbit_training
 model.gradient_checkpointing_enable()
model = prepare_model_for_kbit_training(model)


def print_trainable_parameters(model):
```

```
"""
Prints the number of trainable parameters in the model.
 """
trainable_params = 0
all_param = 0
 for _, param in model.named_parameters():
 all_param += param.numel()
if param.requires_grad:
    trainable_params += param.numel()
print(
    f"trainable params: {trainable_params} || all params: {all_param} || trainable%: {100 * trainable_params / all_param}" )

from peft import LoraConfig, get_peft_model
config = LoraConfig(
 r=8,
lora_alpha=32,
 target_modules=["query_key_value"],
 lora_dropout=0.05,
 bias="none",
 task_type="CAUSAL_LM"
 )
 model = get_peft_model(model, config)
 print_trainable_parameters(model)
```

```
trainable params: 8650752 || all params: 10597552128 || trainable%: 0.08162971878329976
```

Let's load a common dataset, english quotes, to fine tune our model on famous quotes.

```
from datasets import load_dataset
data = load_dataset("Abirate/english_quotes")
 data = data.map(lambda samples: tokenizer(samples["quote"]), batched=True)
```

```
Downloading readme:   0%|          | 0.00/5.55k [00:00<?, ?B/s]
Downloading and preparing dataset json/Abirate--english_quotes to
/root/.cache/huggingface/datasets/Abirate___json/Abirate--english_quotes-
6e72855d06356857/0.0.0/e347ab1c932092252e717ff3f949105a4dd28b27e842dd53157d2f72e276c2e4..
.

Downloading data files:   0%|          | 0/1 [00:00<?, ?it/s]
```

```
Downloading data:   0%|          | 0.00/647k [00:00<?, ?B/s]
Extracting data files:   0%|          | 0/1 [00:00<?, ?it/s]
Generating train split: 0 examples [00:00, ? examples/s]
Dataset json downloaded and prepared to
/root/.cache/huggingface/datasets/Abirate___json/Abirate--english_quotes-
6e72855d06356857/0.0.0/e347ab1c932092252e717ff3f949105a4dd28b27e842dd53157d2f72e276c2e4.
Subsequent calls will reuse this data.

 0%|          | 0/1 [00:00<?, ?it/s]
Map:   0%|          | 0/2508 [00:00<?, ? examples/s]
```

Run the cell below to run the training! For the sake of the demo, we just ran it for few steps just to showcase how to use this integration with existing tools on the HF ecosystem.

```
import transformers

# needed for gpt-neo-x tokenizer
tokenizer.pad_token = tokenizer.eos_token

 trainer = transformers.Trainer(
     model=model,
     train_dataset=data["train"],
     args=transformers.TrainingArguments(
          per_device_train_batch_size=1,
         gradient_accumulation_steps=4,
         warmup_steps=2,
         max_steps=10,
        learning_rate=2e-4,
        fp16=True,
        logging_steps=1,
        output_dir="outputs",
        optim="paged_adamw_8bit"
 ),
  data_collator=transformers.DataCollatorForLanguageModeling(tokenizer, mlm=False),
)
model.config.use_cache = False # silence the warnings. Please re-enable for inference!
 trainer.train()
```

```
You're using a GPTNeoXTokenizerFast tokenizer. Please note that with a fast tokenizer,
using the `__call__` method is faster than using a method to encode the text followed by
a call to the `pad` method to get a padded encoding.
/usr/local/lib/python3.10/dist-
packages/transformers/models/gpt_neox/modeling_gpt_neox.py:229: UserWarning: where
received a uint8 condition tensor. This behavior is deprecated and will be removed in a
future version of PyTorch. Use a boolean condition instead. (Triggered internally
at ../aten/src/ATen/native/TensorCompare.cpp:493.)
  attn_scores = torch.where(causal_mask, attn_scores, mask_value)
```

[10/10 02:27, Epoch 0/1]

Step	Training Loss
1	2.373500
2	3.283200
3	2.290500
4	2.834700
5	2.635500
6	2.185200
7	2.260900
8	1.506300
9	2.470600
10	2.498200

```
TrainOutput(global_step=10, training_loss=2.4338608503341677, metrics={'train_runtime':
166.0171, 'train_samples_per_second': 0.241, 'train_steps_per_second': 0.06,
'total_flos': 99255709532160.0, 'train_loss': 2.4338608503341677, 'epoch': 0.02})
```

```
!pip -q install git+https://github.com/huggingface/transformers # need to install from github
!pip install -q datasets loralib sentencepiece
!pip -q install bitsandbytes accelerate xformers einops
```

```
Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
```

By: Tarun S Gowda

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 236.8/236.8 kB 6.4 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 83.0 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 80.1 MB/s eta 0:00:00
  Building wheel for transformers (pyproject.toml) ... done
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 486.2/486.2 kB 9.3 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 47.0 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 kB 17.6 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 212.5/212.5 kB 32.2 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.3/134.3 kB 20.9 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 66.3 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.5/114.5 kB 13.8 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 35.2 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 149.6/149.6 kB 22.0 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.1/97.1 MB 17.9 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 227.6/227.6 kB 28.4 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 109.1/109.1 MB 15.8 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.2/42.2 kB 5.8 MB/s eta 0:00:00
```

```
!nvidia-smi
```

```
Mon Jun 26 03:28:01 2023 +-----------------------------------------------------------------------------+ | NVIDIA-SMI
525.85.12 Driver Version: 525.85.12 CUDA Version: 12.0 | |-------------------------------+----------------------+----------
-------+ | GPU Name Persistence-M| Bus-Id Disp.A | Volatile Uncorr. ECC | | Fan Temp Perf Pwr:Usage/Cap| Memory-
Usage | GPU-Util Compute M. | | | | MIG M. |
|===============================+======================+======================| | 0
NVIDIA A100-SXM... Off | 00000000:00:04.0 Off | 0 | | N/A 33C P0 48W / 400W | 0MiB / 40960MiB | 0% Default | | |
| Disabled | +-------------------------------+----------------------+----------------------+ +----------------------------------------
------------------------------+ | Processes: | | GPU GI CI PID Type Process name GPU Memory | | ID ID Usage |
|=============================================================================| | No
running processes found | +-----------------------------------------------------------------------------+
```

```
import torch
 import transformers
 from transformers import AutoTokenizer

model_name = 'mosaicml/mpt-30b-instruct'
tokenizer = AutoTokenizer.from_pretrained('mosaicml/mpt-30b')
config = transformers.AutoConfig.from_pretrained(model_name,
                                                 trust_remote_code=True)
 config.init_device = 'cuda:0'
config.max_seq_len = 16384

 model = transformers.AutoModelForCausalLM.from_pretrained(
 model_name,
config=config, torch_dtype=torch.bfloat16,
# Load model weights in bfloat16
trust_remote_code=True,
```

```
  device_map='auto',
  load_in_8bit=True,
  )
```

```
Downloading (…)okenizer_config.json:   0%|          | 0.00/237 [00:00<?, ?B/s]
Downloading (…)/main/tokenizer.json: 0.00B [00:00, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/99.0 [00:00<?, ?B/s]
Downloading (…)lve/main/config.json: 0.00B [00:00, ?B/s]
Downloading (…)configuration_mpt.py: 0.00B [00:00, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
- configuration_mpt.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.

Downloading (…)main/modeling_mpt.py: 0.00B [00:00, ?B/s]
Downloading (…)n/adapt_tokenizer.py: 0.00B [00:00, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
- adapt_tokenizer.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.

Downloading (…)solve/main/blocks.py: 0.00B [00:00, ?B/s]
Downloading (…)resolve/main/norm.py: 0.00B [00:00, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
- norm.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.

Downloading (…)ve/main/attention.py: 0.00B [00:00, ?B/s]
Downloading (…)flash_attn_triton.py: 0.00B [00:00, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
- flash_attn_triton.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
- attention.py
- flash_attn_triton.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
- blocks.py
- norm.py
- attention.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.

Downloading (…)refixlm_converter.py: 0.00B [00:00, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
```

```
- hf_prefixlm_converter.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.

Downloading (…)in/param_init_fns.py: 0.00B [00:00, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
- param_init_fns.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.

Downloading (…)/custom_embedding.py:   0%|          | 0.00/305 [00:00<?, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
- custom_embedding.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.

Downloading (…)meta_init_context.py: 0.00B [00:00, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
- meta_init_context.py
. Make sure to double-check they do not contain any added malicious code. To avoid downlo
ading new versions of the code file, you can pin a revision.
A new version of the following files was downloaded from
https://huggingface.co/mosaicml/mpt-30b-instruct:
- modeling_mpt.py
- adapt_tokenizer.py
- blocks.py
- hf_prefixlm_converter.py
- param_init_fns.py
- custom_embedding.py
- meta_init_context.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.




Downloading (…)model.bin.index.json: 0.00B [00:00, ?B/s]
Downloading shards:   0%|          | 0/7 [00:00<?, ?it/s]
Downloading (…)l-00001-of-00007.bin:   0%|          | 0.00/9.77G [00:00<?, ?B/s]
Downloading (…)l-00002-of-00007.bin:   0%|          | 0.00/9.87G [00:00<?, ?B/s]
Downloading (…)l-00003-of-00007.bin:   0%|          | 0.00/9.87G [00:00<?, ?B/s]
Downloading (…)l-00004-of-00007.bin:   0%|          | 0.00/9.87G [00:00<?, ?B/s]
Downloading (…)l-00005-of-00007.bin:   0%|          | 0.00/9.87G [00:00<?, ?B/s]
Downloading (…)l-00006-of-00007.bin:   0%|          | 0.00/9.87G [00:00<?, ?B/s]
Downloading (…)l-00007-of-00007.bin:   0%|          | 0.00/822M [00:00<?, ?B/s]
You are using config.init_device='cuda:0', but you can also use config.init_device="meta"
with Composer + FSDP for fast initialization.

================================BUG REPORT=================================
Welcome to bitsandbytes. For bug reports, please run

python -m bitsandbytes
```

```
 and submit this information together with your error trace to:
https://github.com/TimDettmers/bitsandbytes/issues
================================================================================
bin /usr/local/lib/python3.10/dist-packages/bitsandbytes/libbitsandbytes_cuda118.so
CUDA_SETUP: WARNING! libcudart.so not found in any environmental path. Searching in
backup paths...
CUDA SETUP: CUDA runtime path found: /usr/local/cuda/lib64/libcudart.so.11.0
CUDA SETUP: Highest compute capability among GPUs detected: 8.0
CUDA SETUP: Detected CUDA version 118
CUDA SETUP: Loading binary /usr/local/lib/python3.10/dist-
packages/bitsandbytes/libbitsandbytes_cuda118.so...



/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:149: UserWarning:
/usr/lib64-nvidia did not contain ['libcudart.so', 'libcudart.so.11.0',
'libcudart.so.12.0'] as expected! Searching further paths...
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:149: UserWarning:
WARNING: The following directories listed in your path were found to be non-existent:
{PosixPath('/sys/fs/cgroup/memory.events /var/colab/cgroup/jupyter-
children/memory.events')}
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:149: UserWarning:
WARNING: The following directories listed in your path were found to be non-existent:
{PosixPath('//172.28.0.1'), PosixPath('8013'), PosixPath('http')}
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:149: UserWarning:
WARNING: The following directories listed in your path were found to be non-existent:
{PosixPath('--logtostderr --listen_host=172.28.0.12 --target_host=172.28.0.12 --
tunnel_background_save_url=https'),
PosixPath('//colab.research.google.com/tun/m/cc48301118ce562b961b3c22d803539adc1e0c19/gpu
-a100-s-2vnsfvg2p0rp5 --tunnel_background_save_delay=10s --
tunnel_periodic_background_save_frequency=30m0s --enable_output_coalescing=true --
output_coalescing_required=true')}
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:149: UserWarning:
WARNING: The following directories listed in your path were found to be non-existent:
{PosixPath('/env/python')}
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:149: UserWarning:
WARNING: The following directories listed in your path were found to be non-existent:
{PosixPath('//ipykernel.pylab.backend_inline'), PosixPath('module')}
  warn(msg)
/usr/local/lib/python3.10/dist-packages/bitsandbytes/cuda_setup/main.py:149: UserWarning:
Found duplicate ['libcudart.so', 'libcudart.so.11.0', 'libcudart.so.12.0'] files:
{PosixPath('/usr/local/cuda/lib64/libcudart.so.11.0'),
PosixPath('/usr/local/cuda/lib64/libcudart.so')}.. We'll flip a coin and try one of
these, in order to fail forward.
Either way, this might cause trouble in the future:
If you get `CUDA error: invalid device function` errors, the above might be the cause and
the solution is to make sure only one ['libcudart.so', 'libcudart.so.11.0',
'libcudart.so.12.0'] in the paths that we search based on your env.
  warn(msg)
WARNING:accelerate.utils.modeling:The model weights are not tied. Please use the
`tie_weights` method before using the `infer_auto_device` function.
```

```
Loading checkpoint shards:   0%|          | 0/7 [00:00<?, ?it/s]
Downloading (…)neration_config.json:   0%|          | 0.00/91.0 [00:00<?, ?B/s]
```

```
!nvidia-smi
```

```
Mon Jun 26 03:46:49 2023 +-----------------------------------------------------------------------------
-------------+ | NVIDIA-SMI 525.85.12 Driver Version: 525.85.12 CUDA Version: 12.0 | |--
-----------------------------+----------------------+----------------------+ | GPU Name
Persistence-M| Bus-Id Disp.A | Volatile Uncorr. ECC | | Fan Temp Perf Pwr:Usage/Cap|
Memory-Usage | GPU-Util Compute M. | | | | MIG M. |
|===============================+======================+======================| | 0
NVIDIA A100-SXM... Off | 00000000:00:04.0 Off | 0 | | N/A 33C P0 53W / 400W | 30311MiB /
40960MiB | 0% Default | | | | Disabled | +-------------------------------+--------------
-------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

```
import json
import textwrap

def get_prompt(instruction):
    prompt_template = "Below is an instruction that describes a task. Write a response
that appropriately completes the request.\n\n###Instruction\n{instruction}\n\n###
Response\n"
    return prompt_template.format(instruction=instruction)

def cut_off_text(text, prompt):
    cutoff_phrase = prompt
   index = text.find(cutoff_phrase)
   if index != -1:
       return text[:index]
else:
     return text

def remove_substring(string, substring):
    return string.replace(substring, "")


def generate(text):
    prompt = get_prompt(text)
    with torch.autocast('cuda', dtype=torch.bfloat16):
        inputs = tokenizer(prompt, return_tensors="pt").to('cuda')
```

```
        outputs = model.generate(**inputs,
                                 max_new_tokens=512,
                                 eos_token_id=tokenizer.eos_token_id,
pad_token_id=tokenizer.pad_token_id,
 )
final _outputs = tokenizer.batch_decode(outputs, skip_special_tokens=False)[0]
final_outputs = cut_off_text(final_outputs, '<|endoftext|>')
final_outputs = remove_substring(final_outputs, prompt)

return final_outputs#, outputs


def parse_text(text):
         wrapped_text = textwrap.fill(text, width=100)
         print(wrapped_text +'\n\n')
         # return assistant_text




'''
%%time function = [
    {
       "name": "get_flight_info",
       "description": "Get the info of the cheapest flight for a given date",
"parameters": {
      "type": "object",
      "properties": {
         "fly_from": {
           "type": "string",
           "description": "the 3-digit code for departure airport"
    },
    "fly_to": {
       "type": "string",
       "description": "the 3-digit code for arrival airport"
    },
    "date": {
        "type": "string",
        "description": "the dd/mm/yyyy format date for flight search"
    },
  },
  "required": ["fly_from", "fly_to", "date"]
}
}
]
prompt = "My query is - What is the cheapest flight for 13/08/2023 from Shanghai to New
York? Before answer you need to learn the function definition first, then give me a JSON
structure describing how to call this function to get answer from this function to help
answer my query, for example [{'name': 'get_flight_info', 'parameters':
{'fly_from':'LAX', 'fly_to':'SFO', 'date':'11/09/2012'}]. ###Function- " +
format(function)
print (prompt)
generated_text = generate(prompt)
parse_text(generated_text)
'''
```

```
'\n%%time\nfunction = [\n {\n "name": "get_flight_info",\n "description": "Get the info
of the cheapest flight for a given date",\n "parameters": {\n "type": "object",\n
"properties": {\n "fly_from": {\n "type": "string",\n "description": "the 3-digit code
for departure airport"\n },\n "fly_to": {\n "type": "string",\n "description": "the 3-
digit code for arrival airport"\n },\n "date": {\n "type": "string",\n "description":
"the dd/mm/yyyy format date for flight search"\n },\n },\n "required": ["fly_from",
"fly_to", "date"]\n }\n }\n]\nprompt = "My query is - What is the cheapest flight for
13/08/2023 from Shanghai to New York? Before answer you need to learn the function
definition first, then give me a JSON structure describing how to call this function to
get answer from this function to help answer my query, for example [{\'name\':
\'get_flight_info\', \'parameters\': {\'fly_from\':\'LAX\', \'fly_to\':\'SFO\',
\'date\':\'11/09/2012\'}]. ###Function- " + format(function)\nprint (prompt)
\ngenerated_text = generate(prompt)\nparse_text(generated_text)\n'
```

```
%%time
function = [
    {
        "name": "get_flight_info",
       "description": "Get the info of the cheapest flight for a given date",
       "parameters": {
          "type": "object",
          "properties": {
            "fly_from": {
                "type": "string",
                 "description": "the 3-digit code for departure airport"
         },
         "fly_to": {
              "type": "string",
             "description": "the 3-digit code for arrival airport"
         }
       , "date": {
              "type": "string",
             "description": "the dd/mm/yyyy format date for flight search"
         },
     },
        "required": ["fly_from", "fly_to", "date"]
    }
  }
]
prompt = "My query is - What is the cheapest flight for 13/08/2023 from Shanghai to New York? Before answer you
need to learn the function definition first, then give me a JSON structure describing how to call this function to get
answer from this function to help answer my query, you should provide 'name' from function definition, and provide
'parameters' in 'properties' from function definition, the format should be : [{'function_name': name, 'parameters':
{para1:value1, para2:value2, para3:value3...}]. ###Function- " + format(function)
#print (prompt)
generated_text = generate(prompt)
response = parse_text(generated_text.partition("### Response\n")[2])
```

```
Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.
/usr/local/lib/python3.10/dist-packages/bitsandbytes/autograd/_functions.py:321:
UserWarning: MatMul8bitLt: inputs will be cast from torch.bfloat16 to float16 during
quantization
  warnings.warn(f"MatMul8bitLt: inputs will be cast from {A.dtype} to float16 during
quantization")


The JSON structure for calling the function is as follows:  [{ "name": "get_flight_info",
"parameters": { "fly_from": "SHA", "fly_to": "JFK", "date": "13/08/2023" } }]


CPU times: user 16.8 s, sys: 139 ms, total: 16.9 s
Wall time: 18.1 s
```

```
%%time #first generate usage prompt = """Please summerize the below article- ##start:
Since the launch of MPT-7B in May, the ML community has eagerly embraced open-source
MosaicML Foundation Series models. The MPT-7B base, -Instruct, -Chat, and -StoryWriter
models have collectively been downloaded over 3M times! We've been overwhelmed by what
the community has built with  MPT-7B. To highlight a few: LLaVA-MPT adds vision
understanding to MPT,  GGML optimizes MPT on Apple Silicon and CPUs, and GPT4All lets you
run a GPT4-like chatbot on your laptop using MPT as a backend model. Today, we are
excited to expand the MosaicML Foundation Series with MPT-30B, a new, open-source model
licensed for commercial use that is significantly more powerful than MPT-7B and
outperforms the original GPT-3. In addition, we are releasing two fine-tuned variants,
MPT-30B-Instruct and MPT-30B-Chat, that are built on top of MPT-30B and excel at single-
turn instruction following and multi-turn conversations, respectively. All MPT-30B models
come with special features that differentiate them from other LLMs, including an 8k token
context window at training time, support for even longer contexts via ALiBi, and
efficient inference + training performance via FlashAttention. The MPT-30B family also
has strong coding abilities thanks to its pretraining data mixture. This model was
extended to an 8k context window on NVIDIA H100s, making it (to the best of our
knowledge) the first LLM trained on H100s. H100s are now available to MosaicML customers!
The size of MPT-30B was also specifically chosen to make it easy to deploy on a single
GPU—either 1xA100-80GB in 16-bit precision or 1xA100-40GB in 8-bit precision. Other
comparable LLMs such as Falcon-40B have larger parameter counts and cannot be served on a
single datacenter GPU (today); this necessitates 2+ GPUs, which increases the minimum
inference system cost. If you want to start using MPT-30B in production, there are
several ways to customize and deploy it using the MosaicML Platform. MosaicML Training.
Customize MPT-30B using your private data via finetuning, domain-specific pretraining, or
training from scratch. You always own the final model weights,  and your data is never
stored on our platform. Pricing is per-GPU-minute. MosaicML Inference: Starter Edition.
Talk to our hosted endpoints for MPT-30B-Instruct (and MPT-7B-Instruct) using our Python
API, with standard pricing per-1K-tokens. MosaicML Inference: Enterprise Edition. Deploy
custom MPT-30B models, either on MosaicML compute or in your own private VPC, using our
optimized inference stack. Pricing is per-GPU-minute, so you only pay for the compute you
use. We are so excited to see what our community and customers build next with MPT-30B.
To learn more about the models and how you can customize them using the MosaicML
platform, read on! MPT-30B Family Mosaic Pretrained Transformer (MPT) models are GPT-
style decoder-only transformers with several improvements including higher speed, greater
stability, and longer context lengths. Thanks to these improvements, customers can train
MPT models efficiently (40-60% MFU) without diverging from loss spikes and can serve MPT
models with both standard HuggingFace pipelines and FasterTransformer. MPT-30B (Base)
```

By: Tarun S Gowda

MPT-30B is a commercial Apache 2.0 licensed, open-source foundation model that exceeds the quality of GPT-3 (from the original paper) and is competitive with other open-source models such as LLaMa-30B and Falcon-40B. Using our publicly available LLM Foundry codebase, we trained MPT-30B over the course of 2 months, transitioning between multiple different A100 clusters as hardware availability changed, with an average MFU of >46%. In mid-June, after we received our first batch of 256xH100s from CoreWeave, we seamlessly moved MPT-30B to the new cluster to resume training on H100s with an average MFU of >35%. To the best of our knowledge, MPT-30B is the first public model to be (partially) trained on H100s! We found that throughput increased by 2.44x per GPU and we expect this speedup to increase as software matures for the H100. As mentioned earlier, MPT-30B was trained with a long context window of 8k tokens (vs. 2k for LLaMa and Falcon) and can handle arbitrarily long context windows via ALiBi or with finetuning. To build 8k support into MPT-30B efficiently, we first pre-trained on 1T tokens using sequences that were 2k tokens long, and continued training for an additional 50B tokens using sequences that were 8k tokens long. The data mix used for MPT-30B pretraining is very similar to MPT-7B (see the MPT-7B blog post for details). For the 2k context window pre-training we used 1T tokens from the same 10 data subsets as the MPT-7B model (Table 1), but in slightly different proportions. Table 1: Data mix for MPT-30B pretraining. We collected 1T tokens of pretraining data from ten different open-source text corpora. We tokenized the text using the EleutherAI GPT-NeoX-20B tokenizer and sampled according to the above ratios. For the 8k context window finetuning, we created two data mixes from the same 10 subsets we used for the 2k context window pretraining (Figure 1). The first 8k finetuning mix is similar to the 2k pretraining mix, but we increased the relative proportion of code by 2.5x. To create the second 8k finetuning mix, which we refer to as the “long sequence” mix, we extracted all sequences of length ≥ 4096 tokens from the 10 pretraining data subsets. We then finetuned on a combination of these two data mixes. See the Appendix for more details on the 8k context window finetuning data. Figure 1: Data subset distribution for 8k context window finetuning. For 8k context window finetuning, we took each data subset and extracted all the samples with ≥ 4096 tokens in order to create a new “long sequence” data mix. We then finetuned on a combination of both the long sequence and original data mixes. In Figure 2, we measure these six core capabilities and find that MPT-30B significantly improves over MPT-7B in every respect. In Figure 3 we perform the same comparison between similarly-sized MPT, LLaMa, and Falcon models. Overall we find that the 7B models across the different families are quite similar. But LLaMa-30B and Falcon-40B are slightly higher in text capabilities than MPT-30B, which is consistent with their larger pretraining budgets: MPT-30B FLOPs ~= 6 * 30e9 [params] * 1.05e12 [tokens] = 1.89e23 FLOPs LLaMa-30B FLOPs ~= 6 * 32.5e9 [params] * 1.4e12 [tokens] = 2.73e23 FLOPs (1.44x more) Falcon-40B FLOPs ~= 6 * 40e9 [params] * 1e12 [tokens] = 2.40e23 FLOps (1.27x more) On the other hand, we find that MPT-30B is significantly better at programming, which we credit to its pretraining data mixture including a substantial amount of code. We dig into programming ability further in Table 2, where we compare the HumanEval scores of MPT-30B, MPT-30B-Instruct, and MPT-30B-Chat to existing open source models including those designed for code generation. We find that MPT-30B models are very strong at programming and MPT-30B-Chat outperforms all models except WizardCoder. We hope that this combination of text and programming capabilities will make MPT-30B models a popular choice for the community. Finally in Table 3, we show how MPT-30B outperforms GPT-3 on the smaller set of eval metrics that are available from the original GPT-3 paper. Just about 3 years after the original publication, we are proud to surpass this famous baseline with a smaller model (17% of GPT-3 parameters) and significantly less training compute (60% of GPT-3 FLOPs). For more detailed evaluation data, or if you want to reproduce our results, you can see the raw data and scripts we used in our LLM Foundry eval harness here. Note that we are still polishing our HumanEval methodology and will release it soon via Composer and LLM-Foundry.

Figure 2 -MPT-7B vs MPT-30B. Our new MPT-30B model significantly improves over our previous MPT-7B model

By: Tarun S Gowda

```
Figure 3 - MPT vs. LLaMa vs. Falcon models. Left: Comparing models with 7 billion
parameters. Right: Comparing models with 30 to 40 billion parameters.

Table 2: Zero-shot accuracy (pass @ 1) of MPT-30B models vs. general purpose and GPT-
tilled code generation models on HumanEval, a corpus of Python coding problems. We find
that MPT-30B models outperform LLaMa-30B and Falcon-40B by a wide margin, and even
outperform many purpose-built coding models such as StarCoder. See Appendix about
disclaimer about Falcon-40B-Instruct and Falcon-40B. External sources: [1], [2], [3],
[4], [5]

Table 3: Zero-shot accuracy of MPT-30B vs. GPT-3 on nine in-context-learning (ICL) tasks.
We find that MPT-30B outperforms GPT-3 in six out of the nine metrics. GPT-3 numbers are
copied from the original paper.

##end

"""
#print (prompt)
generated_text = generate(prompt)
response = parse_text(generated_text.partition("### Response\n")[2])


Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.

1. What is the average MFU of MPT-30B? 2. What is MosaicML Training? 3. Which model did
the
researchers choose to compare their work with in Figure 3? 4. What is the Apache 2.0
licensed, open-
source foundation model called? 5. What is MosaicML Inference: Starter Edition? 6. What
is the size
of MPT-30B? 7. Who do we train MPT-30B? 8. What is Mosaic Pretrained Transformer (MPT)?
9. What is
the size of Mosaic-7B? 10. What is MosaicML Training? 11. What is the average MFU of
Mosaic-7B? 12.
What is MosaicML Inference: Enterprise Edition? 13. Who pretrained Mosaic-30B on 1T
tokens? 14. What
model was extended to an 8k context window on NVIDIA H100s? 15. Who is releasing two
fine-tuned
variants, MPT-30B-Instruct and MPT-30B-Chat? 16. What is the average MFU of Mosaic-30B?

CPU times: user 45.2 s, sys: 71.7 ms, total: 45.3 s
Wall time: 45.2 s


prompt = """Explain the details of this code: ##code start: class
GetflightInPeriodCheckInput(BaseModel):


fly_from: str = Field(..., description="the 3-digit code for departure airport")
fly_to: str = Field(..., description="the 3-digit code for arrival airport") date_from:
str = Field(..., description="the dd/mm/yyyy format of start date for the range of
search")
date_to: str = Field(..., description="the dd/mm/yyyy format of end date for the range of
search")
sort: str = Field(..., description="the catagory for low-to-high sorting, only support
'price', 'duration', 'date'")
```

```
price_limit: int = Field(..., description="The price limit for the flights of search, in
USD, it is set to 999 if not provided")
duration_limit: int = Field(..., description="The flying duration limit for the flights
of search, in hours, it is set to 999 if not provided")


class GetflightInPeriodTool(BaseTool):
      name = "get_flight_in_period"
      description = \"\"\"Useful when you need to search the flights info. You can sort
the result by "sort" argument. You can filter the result by price_limit and
duration_limit. They are default value is 999 if not set. if there is no year, you need
to use 2023 for search. Try to understand the parameters of every flight

    \"\"\"
'''
description = \"\"\"Useful for when you need to find out the information from top 10
flights by sorting for certain category defined in "sort" with a given range of dates.
You should input or convert to the nearest 3-digit airport code and also input dates
range in dd/mm/yyyy format from 2023 for default. In the funtion return, every element
means one entire flight with flight info including price means fly ticket price, duration
means the traveling time, and route informtion for every connection flight.
 \"\"\"
'''
def _run(self, fly_from: str, fly_to: str, date_from: str, date_to: str, sort: str,
price_limit: int, duration_limit: int): get_flight_in_period_response =
get_flight_in_period(fly_from, fly_to, date_from, date_to, sort, price_limit,
duration_limit)

    return get_flight_in_period_response

def _arun(self, fly_from: str, fly_to: str, date_from: str, date_to: str, sort: str,
price_limit: int, duration_limit: int): raise NotImplementedError("This tool does not
support async")
 args_schema: Optional[Type[BaseModel]] = GetflightInPeriodCheckInput. ##code end"""


generated_text = generate(prompt)
response = parse_text(generated_text)



Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.

This code defines a class called GetflightInPeriodTool, which has a method called _run.
The
description of this tool is "Useful for when you need to find out the information from
top 10
flights by sorting for certain category defined in "sort" with a given range of dates.
You should
input or convert to the nearest 3-digit airport code and also input dates range in
dd/mm/yyyy format
from 2023 for default. In the funtion return, every element means one entire flight with
flight info
including price means fly ticket price, duration means the traveling time, and route
informtion for
every connection flight.". The code also defines an args_schema for this tool, which is a
```

```
class
called GetflightInPeriodCheckInput. This class has 6 fields: fly_from, fly_to, date_from,
date_to,
sort, price_limit, duration_limit.
```

```
prompt="complet the code start with - def get_flight_in_period(fly_from, fly_to,
date_from, date_to, sort, price_limit=999, duration_limit=999):"
generated_text = generate(prompt)
response = parse_text(generated_text)
```

```
Setting `pad_token_id` to `eos_token_id`:0 for open-end generation.

# sort can be 'price' or 'duration' # price_limit and duration_limit are int, which
represent the
limit of the flights' price or duration  # get all the flights from fly_from to fly_to in
the
date_from to date_to all_flights = get_all_flights(fly_from, fly_to, date_from, date_to)
# sort the
flights by price or duration if sort == 'price':     all_flights.sort(key=lambda x:
x.price,
reverse=True) elif sort == 'duration':     all_flights.sort(key=lambda x: x.duration,
reverse=True)
# get the flights under the price or duration limit limited_flights = [] for flight in
all_flights:
if flight.price <= price_limit and flight.duration <= duration_limit:
limited_flights.append(flight)          # return the limited flights return li
  Downloading transformers-4.33.0-py3-none-any.whl (7.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.6/7.6 MB 59.6 MB/s eta 0:00:00
Collecting trl
  Downloading trl-0.7.1-py3-none-any.whl (117 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.0/118.0 kB 16.4 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
transformers) (3.12.2)
Collecting huggingface-hub<1.0,>=0.15.1 (from transformers)
  Downloading huggingface_hub-0.16.4-py3-none-any.whl (268 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 32.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages
(from transformers) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages
(from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages
(from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-
packages (from transformers) (2023.6.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from
transformers) (2.31.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers)
  Downloading tokenizers-0.13.3-cp310-cp310-
```

```
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 110.0 MB/s eta 0:00:00
Collecting safetensors>=0.3.1 (from transformers)
  Downloading safetensors-0.3.3-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 84.5 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages
(from transformers) (4.66.1)
Requirement already satisfied: torch>=1.4.0 in /usr/local/lib/python3.10/dist-packages
(from trl) (2.0.1+cu118)
Collecting accelerate (from trl)
  Downloading accelerate-0.22.0-py3-none-any.whl (251 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251.2/251.2 kB 20.4 MB/s eta 0:00:00
Collecting datasets (from trl)
  Downloading datasets-2.14.4-py3-none-any.whl (519 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 519.3/519.3 kB 55.8 MB/s eta 0:00:00
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from
huggingface-hub<1.0,>=0.15.1->transformers) (2023.6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.15.1->transformers)
(4.7.1)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from
torch>=1.4.0->trl) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from
torch>=1.4.0->trl) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from
torch>=1.4.0->trl) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.4.0->trl) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from
triton==2.0.0->torch>=1.4.0->trl) (3.27.2)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from
triton==2.0.0->torch>=1.4.0->trl) (16.0.6)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from
accelerate->trl) (5.9.5)
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.10/dist-packages
(from datasets->trl) (9.0.0)
Collecting dill<0.3.8,>=0.3.0 (from datasets->trl)
  Downloading dill-0.3.7-py3-none-any.whl (115 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 15.1 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from
datasets->trl) (1.5.3)
Collecting xxhash (from datasets->trl)
  Downloading xxhash-3.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(194 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 25.1 MB/s eta 0:00:00
Collecting multiprocess (from datasets->trl)
  Downloading multiprocess-0.70.15-py310-none-any.whl (134 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 18.4 MB/s eta 0:00:00
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from
datasets->trl) (3.8.5)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages
(from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (2.0.4)
```

```
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (2023.7.22)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets->trl) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets->trl) (6.0.4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets->trl) (4.0.3)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets->trl) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets->trl) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets->trl) (1.3.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages
(from jinja2->torch>=1.4.0->trl) (2.1.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->datasets->trl) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages
(from pandas->datasets->trl) (2023.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages
(from sympy->torch>=1.4.0->trl) (1.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
python-dateutil>=2.8.1->pandas->datasets->trl) (1.16.0)
Installing collected packages: tokenizers, safetensors, xxhash, dill, multiprocess,
huggingface-hub, transformers, datasets, accelerate, trl
Successfully installed accelerate-0.22.0 datasets-2.14.4 dill-0.3.7 huggingface-hub-
0.16.4 multiprocess-0.70.15 safetensors-0.3.3 tokenizers-0.13.3 transformers-4.33.0 trl-
0.7.1 xxhash-3.3.0
```

```
## Import libraries
import torch
import transformers
from transformers import pipeline, AutoTokenizer, AutoModelForCausalLM, DataCollatorForLanguageModeling
from trl import RewardTrainer, SFTTrainer
from datasets import Dataset
import json
 import pandas as pd
from transformers import Trainer, TrainingArguments
```

```
!wget https://huggingface.co/datasets/CarperAI/openai_summarize_comparisons/resolve/main/data/test-00000-of-00001-0845e2eec675b16a.parquet
# after that rename to test.parquet
```

--2023-09-06 12:36:39--
https://huggingface.co/datasets/CarperAI/openai_summarize_comparisons/resolve/main/data/test-00000-of-00001-0845e2eec675b16a.parquet Resolving huggingface.co (huggingface.co)... 18.172.134.24, 18.172.134.4, 18.172.134.88, ... Connecting to huggingface.co (huggingface.co)|18.172.134.24|:443... connected. HTTP request sent, awaiting response... 302 Found Location: https://cdn-lfs.huggingface.co/repos/02/84/02842250b699bbde4e980e9206f5045df542f9ca374595cc86b8d57189641ff5/a8c5d73113a2c7119faacd44e43959f74e5b59e181f3da57caa918c907ac6620?response-content-disposition=attachment%3B+filename*%3DUTF-8%27%27test-00000-of-00001-0845e2eec675b16a.parquet%3B+filename%3D%22test-00000-of-00001-0845e2eec675b16a.parquet%22%3B&Expires=1694262999&Policy=eyJTdGF0ZW1lbnQiOlt7IkNvbmRpdGlvbiI6eyJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MTY5NDI2Mjk5OX19LCJSZXNvdXJjZSI6Imh0dHBzOi8vY2RuLWxmcy5odWdnaW5nZmFjZS5jby9yZXBvcy8wMi84NC8wMjg0MjI1MGI2OTliYmRlNGU5ODBlOTIwNmY1MDQ1ZGY1NDJmOWNhMzc0NTk1Y2M4NmI4ZDU3MTg5NjQxZmY1L2E4YzVkNzMxMTNhMmM3MTE5ZmFhY2Q0NGU0Mzk1OWY3NGU1YjU5ZTE4MWYzZGE1N2NhYTkxOGM5MDdhYzY2MjA%7EcmVzcG9uc2UtY29udGVudC1kaXNwb3NpdGlvbj0qIn1dfQ__&Signature=CLl3zBeYW5jzKbX9MZkryL0p-j8lTtx03ksho0Zwpi5ZeKplDz%7EohUhbXaK5OKCteoEJx%7EDjpluciHcFdC%7EQcfdwgRSuAuT%7ECMaPxAmmzR54Zd0AcIPR02OenGpIbf3scib4X-ZCWgSVAgUlF6F8jfbSq1Yo4eoi4iyJj3m%7El1O0NbWVZ7qm%7E4kXfNzBpaMI1l0pDg-9x6UCrVuto2g1guD3Uzwb%7EahiEwCYIcVGLgxo-SXOFO7-9w4eCZ-KkvwNq%7EgEGeUBmilzf1Wwnu6IVPnygYicEbiAIopHZDcm87wuC3j%7EcMwVKQx9JxrfXf3kP-Q6OqrLDUd2ejN0gInvUQ__&Key-Pair-Id=KVTP0A1DKRTAX [following] --2023-09-06 12:36:39-- https://cdn-lfs.huggingface.co/repos/02/84/02842250b699bbde4e980e9206f5045df542f9ca374595cc86b8d57189641ff5/a8c5d73113a2c7119faacd44e43959f74e5b59e181f3da57caa918c907ac6620?response-content-disposition=attachment%3B+filename*%3DUTF-8%27%27test-00000-of-00001-0845e2eec675b16a.parquet%3B+filename%3D%22test-00000-of-00001-0845e2eec675b16a.parquet%22%3B&Expires=1694262999&Policy=eyJTdGF0ZW1lbnQiOlt7IkNvbmRpdGlvbiI6eyJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MTY5NDI2Mjk5OX19LCJSZXNvdXJjZSI6Imh0dHBzOi8vY2RuLWxmcy5odWdnaW5nZmFjZS5jby9yZXBvcy8wMi84NC8wMjg0MjI1MGI2OTliYmRlNGU5ODBlOTIwNmY1MDQ1ZGY1NDJmOWNhMzc0NTk1Y2M4NmI4ZDU3MTg5NjQxZmY1L2E4YzVkNzMxMTNhMmM3MTE5ZmFhY2Q0NGU0Mzk1OWY3NGU1YjU5ZTE4MWYzZGE1N2NhYTkxOGM5MDdhYzY2MjA%7EcmVzcG9uc2UtY29udGVudC1kaXNwb3NpdGlvbj0qIn1dfQ__&Signature=CLl3zBeYW5jzKbX9MZkryL0p-j8lTtx03ksho0Zwpi5ZeKplDz%7EohUhbXaK5OKCteoEJx%7EDjpluciHcFdC%7EQcfdwgRSuAuT%7ECMaPxAmmzR54Zd0AcIPR02OenGpIbf3scib4X-ZCWgSVAgUlF6F8jfbSq1Yo4eoi4iyJj3m%7El1O0NbWVZ7qm%7E4kXfNzBpaMI1l0pDg-9x6UCrVuto2g1guD3Uzwb%7EahiEwCYIcVGLgxo-SXOFO7-9w4eCZ-KkvwNq%7EgEGeUBmilzf1Wwnu6IVPnygYicEbiAIopHZDcm87wuC3j%7EcMwVKQx9JxrfXf3kP-Q6OqrLDUd2ejN0gInvUQ__&Key-Pair-Id=KVTP0A1DKRTAX Resolving cdn-lfs.huggingface.co (cdn-lfs.huggingface.co)... 18.154.185.26, 18.154.185.64, 18.154.185.94, ... Connecting to cdn-lfs.huggingface.co (cdn-lfs.huggingface.co)|18.154.185.26|:443... connected. HTTP request sent, awaiting response... 200 OK Length: 20441979 (19M) [binary/octet-stream] Saving to: ‘test-00000-of-00001-0845e2eec675b16a.parquet’ test-00000-of-00001 100%[===================>] 19.49M 105MB/s in 0.2s 2023-09-06 12:36:39 (105 MB/s) - ‘test-00000-of-00001-0845e2eec675b16a.parquet’ saved [20441979/20441979]

Dataset for Training : https://huggingface.co/datasets/CarperAI/openai_summarize_tldr

Dataset for Testing : https://huggingface.co/datasets/CarperAI/openai_summarize_comparisons

By: Tarun S Gowda

```
Model_name = "bigcode/tiny_starcoder_py"
 Data_path = "test.parquet"
```

```
# Read the file
df = pd.read_parquet(Data_path)
df = df[:10]
raw_dataset = Dataset.from_pandas(df)
raw_dataset
```

```
Dataset({
    features: ['prompt', 'chosen', 'rejected'],
    num_rows: 10
})
```

Tokenize the dataset to feed in the model

```
tokenizer = AutoTokenizer.from_pretrained(Model_name)
model = AutoModelForCausalLM.from_pretrained(Model_name)
```

```
Downloading (…)okenizer_config.json:   0%|          | 0.00/677 [00:00<?, ?B/s]
Downloading (…)olve/main/vocab.json:   0%|          | 0.00/777k [00:00<?, ?B/s]
Downloading (…)olve/main/merges.txt:   0%|          | 0.00/442k [00:00<?, ?B/s]
Downloading (…)/main/tokenizer.json:   0%|          | 0.00/2.06M [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/532 [00:00<?, ?B/s]
Downloading (…)lve/main/config.json:   0%|          | 0.00/1.03k [00:00<?, ?B/s]
Downloading model.safetensors:   0%|          | 0.00/657M [00:00<?, ?B/s]
Downloading (…)neration_config.json:   0%|          | 0.00/111 [00:00<?, ?B/s]
```

Tokenize the dataset to feed in the model

```
tokenizer.add_special_tokens({'pad_token': '[PAD]'})
def formatting_func(examples):
     kwargs = {"padding": "max_length",
    "truncation": True,
    "max_length": 256,
    "return_tensors": "pt"
     }

 # Prepend the prompt and a line break to the original_response and response-1 fields.
 prompt_plus_chosen_response = examples["prompt"] + "\n" + examples["chosen"]
prompt_plus_rejected_response = examples["prompt"] + "\n" + examples["rejected"]

# Then tokenize these modified fields.
tokens_chosen = tokenizer.encode_plus(prompt_plus_chosen_response, **kwargs)
 tokens_rejected = tokenizer.encode_plus(prompt_plus_rejected_response, **kwargs)

 return k{
```

By: Tarun S Gowda

```
 "input_ids_chosen": tokens_chosen["input_ids"][0], "attention_mask_chosen": tokens_chosen["attention_mask"][0],
"input_ids_rejected": tokens_rejected["input_ids"][0], "attention_mask_rejected": tokens_rejected["attention_mask"][0]

}
```

Map the function to data

```
formatted_dataset = raw_dataset.map(formatting_func)
formatted_dataset = formatted_dataset.train_test_split()
```

```
Map:   0%|          | 0/10 [00:00<?, ? examples/s]
```

Model.config

```
GPTBigCodeConfig {
  "_name_or_path": "bigcode/tiny_starcoder_py",
  "activation_function": "gelu_pytorch_tanh",
  "architectures": [
    "GPTBigCodeForCausalLM"
  ],
  "attention_softmax_in_fp32": true,
  "attn_pdrop": 0.1,
  "bos_token_id": 0,
  "embd_pdrop": 0.1,
  "eos_token_id": 0,
  "inference_runner": 0,
  "initializer_range": 0.02,
  "layer_norm_epsilon": 1e-05,
  "max_batch_size": null,
  "max_sequence_length": null,
  "model_type": "gpt_bigcode",
  "multi_query": true,
  "n_embd": 768,
  "n_head": 12,
  "n_inner": 3072,
  "n_layer": 20,
  "n_positions": 8192,
  "pad_key_length": true,
  "pre_allocate_kv_cache": false,
  "resid_pdrop": 0.1,
  "scale_attention_softmax_in_fp32": true,
  "scale_attn_weights": true,
  "summary_activation": null,
  "summary_first_dropout": 0.1,
  "summary_proj_to_labels": true,
  "summary_type": "cls_index",
```

```
  "summary_use_proj": true,
  "torch_dtype": "float32",
  "transformers_version": "4.33.0",
  "use_cache": true,
  "validate_runner_input": true,
  "vocab_size": 49152
}
```

```
### Loading the TRL reward trainer and training the trainer
training_args = TrainingArguments(
        output_dir="tinystarcoder-rlhf-model",
      num_train_epochs=1,
logging_steps=10,
gradient_accumulation_steps=1,
save_strategy="steps",
evaluation_strategy="steps",
per_device_train_batch_size=2,
per_device_eval_batch_size=1,
eval_accumulation_steps=1,
eval_steps=500,
save_steps=500,
warmup_steps=100,
logging_dir="./logs",
learning_rate=1e-5,
save_total_limit=1,
no_cuda=True
)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1257:
FutureWarning: using `no_cuda` is deprecated and will be removed in version 5.0 of 🤗
Transformers. Use `use_cpu` instead
  warnings.warn(
```

```
trainer = RewardTrainer(model=model,
                        tokenizer=tokenizer,
                        train_dataset=formatted_dataset['train'],
                        eval_dataset=formatted_dataset['test'],
                        args= training_args
                        )
trainer.train()
```

```
/usr/local/lib/python3.10/dist-packages/trl/trainer/reward_trainer.py:124: UserWarning:
When using RewardDataCollatorWithPadding, you should set `max_length` in the
RewardTrainer's init it will be set to `512` by default, but you should do it yourself in
```

```
the future.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/trl/trainer/reward_trainer.py:138: UserWarning:
When using RewardDataCollatorWithPadding, you should set `remove_unused_columns=False` in
your TrainingArguments we have set it for you, but you should do it yourself in the
future.
  warnings.warn(
You're using a GPT2TokenizerFast tokenizer. Please note that with a fast tokenizer, using
the `__call__` method is faster than using a method to encode the text followed by a call
to the `pad` method to get a padded encoding.
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2436:
UserWarning: `max_length` is ignored when `padding`=`True` and there is no truncation
strategy. To pad to max length, use `padding='max_length'`.
  warnings.warn(
Could not estimate the number of tokens of the input, floating-point operations will not
be computed
```

[4/4 00:52, Epoch 1/1]

StepTraining LossValidation Loss

```
TrainOutput(global_step=4, training_loss=0.7674217224121094, metrics={'train_runtime':
101.4459, 'train_samples_per_second': 0.069, 'train_steps_per_second': 0.039,
'total_flos': 0.0, 'train_loss': 0.7674217224121094, 'epoch': 1.0})
```

```
trainer.evaluate()
```

```
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2436:
UserWarning: `max_length` is ignored when `padding`=`True` and there is no truncation
strategy. To pad to max length, use `padding='max_length'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/trl/trainer/reward_trainer.py:214: UserWarning:
The use of `x.T` on tensors of dimension other than 2 to reverse their shape is
deprecated and it will throw an error in a future release. Consider `x.mT` to transpose
batches of matrices or `x.permute(*torch.arange(x.ndim - 1, -1, -1))` to reverse the
dimensions of a tensor. (Triggered internally
at ../aten/src/ATen/native/TensorShape.cpp:3571.)
  logits = torch.stack(logits).mean(dim=2).softmax(dim=0).T
```

[3/3 00:14]

```
{'eval_loss': 0.6931466460227966,
 'eval_accuracy': 0.0,
 'eval_runtime': 22.5484,
 'eval_samples_per_second': 0.133,
 'eval_steps_per_second': 0.133,
 'epoch': 1.0}
```

```
from huggingface_hub import login
login()
```

```
VBox(children=(HTML(value='<center>
<img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv…
```

```
trainer.push_to_hub("RLHF model of StarCoder")
```

```
pytorch_model.bin:   0%|          | 0.00/657M [00:00<?, ?B/s]
```

```
'https://huggingface.co/ashishpatel26/tinystarcoder-rlhf-model/tree/main/'
```

```
trainer.save_model("tinystarcoder-rlhf-model")
```

Policy Model

```
help(trainer.push_to_hub)
```

```
## inference the model
rm_model = AutoModelForCausalLM.from_pretrained("tinystarcoder-rlhf-model")
tokenizer = AutoTokenizer.from_pretrained("tinystarcoder-rlhf-model")


def get_score(model, tokenizer, prompt, response):

instructions = tokenizer.encode_plus(prompt,
                                     response,
                                     padding="max_length",
                                     max_length=256,
                                     return_tensors="pt",
                                     truncation=True)
 with torch.no_grad():
        outputs = model(**instructions)

logits = outputs[0]

 return logits
```

```
 # usage with prompt
 prompt = df.iloc[0]["prompt"]
 example_prefered_response = df.iloc[0]["chosen"]
 example_unprefered_response = df.iloc[0]["rejected"]
```

By: Tarun S Gowda

```
loss1 = get_score(model, tokenizer, prompt, example_prefered_response)
loss2= get_score(model, tokenizer, prompt, example_unprefered_response)
```

```
from torch import nn
loss = -nn.functional.logsigmoid(loss1 - loss2).mean()
```

```
tokenizer.decode(torch.max(loss1, axis=-1).indices[0])
```

'_DDIT_\n "r/#: Relationship RelationshipRelationship0]0ARCH\nlsriend\n2//M]\n [\n [\n the was to the [\n a friends to\n\n: https [lfriend [me am a aumb. my19 minutes.\n\n"""I updated:** girlfriend was through my Facebook..** I my my friends.**.** my of dayslf.\n\n** went dding for my message personirl** I had to makeellpping my my my past** but I was a in\n\n** have ali a of ago she tolirt with my girl. and she had my about my.. me few of gir.1viously). was\'t find that was).\n\n** was it for twoirl and the had my Facebook. the she wasand historyirl) was was March,\n to to find that were flited. I a g.. f.ing on her.\nirirl wass; I1 girirllfriend is 19 months. through my Facebook.. my permission. I it messages. her.lirty with me coupleirl.\n found up with me after she through more with\n'

Policy Model

```
import torch
import transformers
from transformers import pipeline, AutoTokenizer, AutoModelForCausalLM, DataCollatorForLanguageModeling
from trl import RewardTrainer, SFTTrainer
from datasets import Dataset
import json
import pandas as pd
 from transformers import Trainer, TrainingArguments
from trl import PPOTrainer, PPOConfig, AutoModelForCausalLMWithValueHead, create_reference_model
```

```
##model path
 MODEL_PATH = "bigcode/tiny_starcoder_py"
DATA_PATH = "test.parquet"
```

```
df = pd.read_parquet(DATA_PATH)
df = df[:1000]
dataset = Dataset.from_pandas(df)
dataset
```

```
Dataset({
    features: ['prompt', 'chosen', 'rejected'],
```

```
    num_rows: 1000
})
```

```
from transformers import AutoTokenizer, pipeline
```

```
rf_model_path = "tinystarcoder-rlhf-model"
starcoder_model = AutoModelForCausalLMWithValueHead.from_pretrained(rf_model_path) starcoder_model_ref =
AutoModelForCausalLMWithValueHead.from_pretrained(rf_model_path) starcoder_tokenizer =
AutoTokenizer.from_pretrained(rf_model_path)

tokenizer = AutoTokenizer.from_pretrained(MODEL_PATH, padding_side='left')
tokenizer.pad_token = tokenizer.eos_token
txt_in_len = 5
txt_out_len = 32
 seed = 1

dataset = dataset.map(
      lambda x: {"input_ids": tokenizer.encode(" " + x["chosen"], return_tensors="pt", truncation=True,
padding="max_length", max_length=32)[0]}, batched=False,
)
dataset = dataset.map(lambda x: {"query": tokenizer.decode(x["input_ids"])}, batched=False)
dataset = dataset[:20480]
from datasets import Dataset

dataset = Dataset.from_dict(dataset)
dataset.set_format("pytorch")
```

```
Map:   0%|          | 0/1000 [00:00<?, ? examples/s]
Map:   0%|          | 0/1000 [00:00<?, ? examples/s]
```

```
dataset
```

```
Dataset({
    features: ['review', 'chosen', 'rejected', 'input_ids', 'query'],
    num_rows: 1000
})
```

```
sentiment_pipe_kwargs = {"top_k": None, "function_to_apply": "none"}

config = PPOConfig(
    model_name=MODEL_PATH, steps=51200, learning_rate=1.41e-5, remove_unused_columns=True
 )

txt_in_len = 5
```

```
txt_out_len = 20
seed = 1
```

```
import torch
optimizer = torch.optim.SGD(starcoder_model.parameters(), lr=config.learning_rate)
ppo_trainer = PPOTrainer(config, starcoder_model, starcoder_model, starcoder_tokenizer, dataset=dataset,
data_collator=collator, optimizer=optimizer)
```

```
# for i in ppo_trainer.dataloader:
# print(i)
```

```
ctrl_str = ["[negative]", "[positive]"]
device = torch.device("cuda" if torch.cuda.is_available() else "cpu") # this should be handled by accelerate
ctrl_tokens = dict((s, starcoder_tokenizer.encode(s, return_tensors="pt").squeeze().to(device)) for s in ctrl_str)
```

```
def pos_logit_to_reward(logit, task):
    """
    Take the positive sentiment logit and scale it for the task.
        task [negative]: reward = -logit
        task [neutral]: reward = -2*abs(logit)+4 task
        [positive]: reward = logit
    """
    for i in range(len(logit)):
        if task[i] == "[negative]":
            logit[i] = -logit[i]
        elif task[i] == "[positive]":
            pass
        else:
            raise ValueError("task has to be in [0, 1, 2]!")
    return logit
```

```
pos_logit_to_reward(torch.Tensor([4, 4]), ctrl_str)
```

```
tensor([-4.,  4.])
```

```
generation_kwargs = {
    "min_length": -1,
    "top_k": 0.0,
    "top_p": 1.0,
    "do_sample": True,
    "pad_token_id": starcoder_tokenizer.eos_token_id,
    "max_new_tokens": 32, "eos_token_id": -1,
}
```

```
def get_score(model, tokenizer, responses):
    positive_logist = []
    for i in responses:
```

```
        instructions = tokenizer.encode_plus(
                                                i,
                                                padding="max_length",
                                                 max_length=32,
                                                 return_tensors="pt")
  with torch.no_grad():
        outputs = model(**instructions)

 logits = outputs[0].mean()
positive_logist.append(logits)

return positive_logist

  # responses =["ashish is a goo", "heelow how are you", "__IT_\nr/\n: r RelationshipRelationship]]0]\nlsriend\n2//M]\n
  [ [ a\n the was to the [. a friends to\n\n:\n [lfriend [ me have a aried in his19 minutes.\n\nWhat Modified:** girlfriend
  was through the Facebook.. I my my friends.**** my of lf**\n\n** was d1ing for my few personirl** I had for
  findoolpping my my the future** but I was that in\n\n** have ali of to she tolirt my me girl. and she found my about my..
  me few of gir.1viously). was\'t find her was).\n\n** was it about my twoirl and the had Facebook. the and she gand
  historyirl) was in April,\n to, find, were flirted. I a messages.. f.ing on her.\n girlM\n; I1 girirllfriend and the19 months.
  to my Facebook.. my permission. she her messages. my.lirty with my fewirl.\n found her with me. I through more
  with\n"]
  # get_score(starcoder_model, tokenizer, responses)


  # from random import choices
  # from tqdm import tqdm
   # import time
  # import numpy as np

   # for epoch in range(1):
  # for batch in tqdm(ppo_trainer.dataloader):
  # (logs, game_data,) = (
   # dict()
  , # dict(),
   # )
  # print(ctrl_str)
   # #### prepend a random control token
  # task_list = choices(ctrl_str, k=config.batch_size)
  # game_data["query"] = [t + q for t, q in zip(task_list, batch["query"])]
  # query_tensors = [torch.cat((ctrl_tokens[t], input_ids)) for t, input_ids in zip(task_list, batch["input_ids"])]

  # #### get response from gpt2
  # response_tensors = []
  # for query in query_tensors:
  # response = ppo_trainer.generate(query, **generation_kwargs)
  # response_tensors.append(response.squeeze()[-txt_out_len:])
  # # print(response_tensors)
  # game_data["response"] = [starcoder_tokenizer.decode(r.squeeze()) for r in response_tensors]
   # #### sentiment analysis
   # texts = [q + r for q, r in zip(batch["query"], game_data["response"])]
   # logits = get_score(starcoder_model,starcoder_tokenizer, texts)
  # rewards = pos_logit_to_reward(logits, task_list)
  # #### Run PPO training # t = time.time()
```

```
# stats = ppo_trainer.step(query_tensors, response_tensors, rewards)
# for cs in ctrl_str:
# key = "env/reward_" + cs.strip("[]")
# stats[key] = np.mean([r.cpu().numpy() for r, t in zip(rewards, task_list) if t == cs])
 # ppo_trainer.log_stats(stats, game_data, rewards)
```

```
###saving the model
starcoder_model.save_pretrained("StarCoderTinyrhlfmodel/")
starcoder_tokenizer.save_pretrained("StarCoderTinyrhlfmodel/")
```

```
('StarCoderTinyrhlfmodel/tokenizer_config.json',
 'StarCoderTinyrhlfmodel/special_tokens_map.json',
 'StarCoderTinyrhlfmodel/vocab.json',
 'StarCoderTinyrhlfmodel/merges.txt',
 'StarCoderTinyrhlfmodel/added_tokens.json',
 'StarCoderTinyrhlfmodel/tokenizer.json')
```

```
from transformers import pipeline, set_seed
model_path = "StarCoderTinyrhlfmodel/"
set_seed(42)
 pipe = pipeline("text-generation",model=model_path, tokenizer=model_path, max_length=30,
num_return_sequences=5)
```

Some weights of the model checkpoint at StarCoderTinyrhlfmodel/ were not used when initializing GPTBigCodeForCausalLM: ['v_head.summary.weight', 'v_head.summary.bias'] - This IS expected if you are initializing GPTBigCodeForCausalLM from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model). - This IS NOT expected if you are initializing GPTBigCodeForCausalLM from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).

```
!nvidia-smi
```

LLM-Finetuning with PEFT

```
Thu Sep 21 11:23:52 2023


 +-----------------------------------------------------------------------------+ |
NVIDIA-SMI 525.105.17 Driver Version: 525.105.17 CUDA Version: 12.0 | |-----------------
------------+----------------------+----------------------+ | GPU Name Persistence-M|
Bus-Id Disp.A | Volatile Uncorr. ECC | | Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-
Util Compute M. | | | | MIG M. |
|===============================+======================+======================| | 0 Tesla
T4 Off | 00000000:00:04.0 Off | 0 | | N/A 60C P8 11W / 70W | 0MiB / 15360MiB | 0% Default
| | | | N/A | +-------------------------------+----------------------+-----------------
---+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+


!pip install accelerate transformers einops datasets peft bitsandbytes trl




Collecting accelerate Downloading accelerate-0.23.0-py3-none-any.whl (258 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 258.1/258.1 kB 3.9 MB/s eta 0:00:00 Collecting
transformers Downloading transformers-4.33.2-py3-none-any.whl (7.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.6/7.6 MB 18.8 MB/s eta 0:00:00 Collecting
einops Downloading einops-0.6.1-py3-none-any.whl (42 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.2/42.2 kB 5.0 MB/s eta 0:00:00 Collecting
datasets Downloading datasets-2.14.5-py3-none-any.whl (519 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 519.6/519.6 kB 26.7 MB/s eta 0:00:00 Collecting
peft Downloading peft-0.5.0-py3-none-any.whl (85 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.6/85.6 kB 9.6 MB/s eta 0:00:00 Collecting
bitsandbytes Downloading bitsandbytes-0.41.1-py3-none-any.whl (92.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.6/92.6 MB 11.2 MB/s eta 0:00:00 Collecting
trl Downloading trl-0.7.1-py3-none-any.whl (117 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.0/118.0 kB 16.0 MB/s eta 0:00:00 Requirement
already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from
accelerate) (1.23.5) Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from accelerate) (23.1) Requirement already
satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from
accelerate) (6.0.1) Requirement already satisfied: torch>=1.10.0 in
/usr/local/lib/python3.10/dist-packages (from accelerate) (2.0.1+cu118) Collecting
huggingface-hub (from accelerate) Downloading huggingface_hub-0.17.2-py3-none-any.whl
(294 kB) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.9/294.9 kB 33.3 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
transformers) (3.12.2) Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2023.6.3) Requirement
already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from
transformers) (2.31.0) Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers)
Downloading tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(7.8 MB) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 114.7 MB/s eta 0:00:00
```

By: Tarun S Gowda

```
Collecting safetensors>=0.3.1 (from transformers) Downloading safetensors-0.3.3-cp310-
cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 81.0 MB/s eta 0:00:00 Requirement
already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from
transformers) (4.66.1) Requirement already satisfied: pyarrow>=8.0.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (9.0.0) Collecting
dill<0.3.8,>=0.3.0 (from datasets) Downloading dill-0.3.7-py3-none-any.whl (115 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 14.4 MB/s eta 0:00:00 Requirement
already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets)
(1.5.3) Collecting xxhash (from datasets) Downloading xxhash-3.3.0-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 23.7 MB/s eta 0:00:00 Collecting
multiprocess (from datasets) Downloading multiprocess-0.70.15-py310-none-any.whl (134 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 17.0 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]<2023.9.0,>=2023.1.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2023.6.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from
datasets) (3.8.5)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets) (23.1.0)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (3.2.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets) (6.0.4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages
(from aiohttp->datasets) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub->accelerate) (4.5.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages
(from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (2023.7.22)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from
torch>=1.10.0->accelerate) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from
torch>=1.10.0->accelerate) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from
torch>=1.10.0->accelerate) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10.0->accelerate) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from
triton==2.0.0->torch>=1.10.0->accelerate) (3.27.4.1)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from
triton==2.0.0->torch>=1.10.0->accelerate) (16.0.6)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages
(from pandas->datasets) (2023.3.post1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
```

```
python-dateutil>=2.8.1->pandas->datasets) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages
(from jinja2->torch>=1.10.0->accelerate) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages
(from sympy->torch>=1.10.0->accelerate) (1.3.0)
Installing collected packages: tokenizers, safetensors, bitsandbytes, xxhash, einops,
dill, multiprocess, huggingface-hub, transformers, datasets, accelerate, trl, peft
Successfully installed accelerate-0.23.0 bitsandbytes-0.41.1 datasets-2.14.5 dill-0.3.7
einops-0.6.1 huggingface-hub-0.17.2 multiprocess-0.70.15 peft-0.5.0 safetensors-0.3.3
tokenizers-0.13.3 transformers-4.33.2 trl-0.7.1 xxhash-3.3.0
```

```
# !pip install -Uqqq pip --progress-bar off # !pip install -qqq torch==2.0.1 --progress-bar off
# !pip install -qqq transformers==4.32.1 --progress-bar off
# !pip install -qqq datasets==2.14.4 --progress-bar off
 # !pip install -qqq peft==0.5.0 --progress-bar off
# !pip install -qqq bitsandbytes==0.41.1 --progress-bar off
 # !pip install -qqq trl==0.7.1 --progress-bar off
```

```
import json
import re
from pprint import pprint
 import os

import pandas as pd
 import torch
 from datasets import Dataset, load_dataset
 from huggingface_hub import notebook_login
from peft import LoraConfig, PeftModel, get_peft_model
from transformers import (
      AutoModelForCausalLM,
      AutoTokenizer,
     BitsAndBytesConfig,
     TrainingArguments,
     DataCollatorForLanguageModeling,
     BitsAndBytesConfig
)
 from trl import SFTTrainer DEVICE = "cuda:0" if torch.cuda.is_available() else "cpu"

 MODEL_NAME = "microsoft/phi-1_5"
```

LLM-Finetuning with PEFT

Data

```
dataset = load_dataset("Salesforce/dialogstudio", "TweetSumm")
 dataset
```

```
DatasetDict({
    train: Dataset({
      features: ['original dialog id', 'new dialog id', 'dialog index', 'original dialog info', 'log', 'prompt'],
      num_rows: 879
})
 validation: Dataset({
       features: ['original dialog id', 'new dialog id', 'dialog index', 'original dialog info', 'log', 'prompt'],
       num_rows: 110
})
test: Dataset({
       features: ['original dialog id', 'new dialog id', 'dialog index', 'original dialog info', 'log', 'prompt'],
      num_rows: 110
  })
 })
```

```
DEFAULT_SYSTEM_PROMPT = """
Below is a conversation between a human and an AI agent. Write a summary of the conversation.
 """.strip()


def generate_training_prompt(
      conversation: str, summary: str, system_prompt: str = DEFAULT_SYSTEM_PROMPT ) -> str:
      return f"""### Instruction: {system_prompt}

 ### Input:
{conversation.strip()}

 ### Response:
{summary}
 """.strip()
```

```
def clean_text(text):
      text = re.sub(r"http\S+", "", text)
      text = re.sub(r"@[^\s]+", "", text)
      text = re.sub(r"\s+", " ", text)
       return re.sub(r"\^[^ ]+", "", text)

 def create_conversation_text(data_point):
text = ""
```

By: Tarun S Gowda

```
for item in data_point["log"]:
 user = clean_text(item["user utterance"])
text += f"user: {user.strip()}\n"

 agent = clean_text(item["system response"])
 text += f"agent: {agent.strip()}\n"

 return text


def generate_text(data_point):
summaries = json.loads(data_point["original dialog info"])["summaries"][ "abstractive_summaries"
 ]
summary = summaries[0]
 summary = " ".join(summary)

 conversation_text = create_conversation_text(data_point)
 return {
         "conversation": conversation_text,
        "summary": summary,
       "text": generate_training_prompt(conversation_text, summary),
}


example = generate_text(dataset["train"][0])


print(example["summary"])
```

```
Customer enquired about his Iphone and Apple watch which is not showing his any
steps/activity and health activities. Agent is asking to move to DM and look into it.
```

```
print(example["conversation"])
```

```
user: So neither my iPhone nor my Apple Watch are recording my steps/activity, and
Health doesn’t recognise either source anymore for some reason. Any ideas? please read
the above.
agent: Let’s investigate this together. To start, can you tell us the software versions
your iPhone and Apple Watch are running currently?
user: My iPhone is on 11.1.2, and my watch is on 4.1.
agent: Thank you. Have you tried restarting both devices since this started happening?
user: I’ve restarted both, also un-paired then re-paired the watch.
agent: Got it. When did you first notice that the two devices were not talking to each
other. Do the two devices communicate through other apps such as Messages?
user: Yes, everything seems fine, it’s just Health and activity.
agent: Let’s move to DM and look into this a bit more. When reaching out in DM, let us
know when this first started happening please. For example, did it start after an
update or after installing a certain app?
```

```
print(example["text"])
```

By: Tarun S Gowda

```
### Instruction: Below is a conversation between a human and an AI agent. Write a summary of the conversation. ### Input: user: So neither my iPhone nor my Apple Watch are recording my steps/activity, and Health doesn’t recognise either source anymore for some reason. Any ideas? please read the above. agent: Let’s investigate this together. To start, can you tell us the software versions your iPhone and Apple Watch are running currently? user: My iPhone is on 11.1.2, and my watch is on 4.1. agent: Thank you. Have you tried restarting both devices since this started happening? user: I’ve restarted both, also un-paired then re-paired the watch. agent: Got it. When did you first notice that the two devices were not talking to each other. Do the two devices communicate through other apps such as Messages? user: Yes, everything seems fine, it’s just Health and activity. agent: Let’s move to DM and look into this a bit more. When reaching out in DM, let us know when this first started happening please. For example, did it start after an update or after installing a certain app?

 ### Response: Customer enquired about his Iphone and Apple watch which is not showing his any steps/activity and health activities. Agent is asking to move to DM and look into it.
```

```
def process_dataset(data: Dataset):
    return (
        data.shuffle(seed=42)
       .map(generate_text)
       .remove_columns(
         [
                "original dialog id",
               "new dialog id",
               "dialog index",
                "original dialog info",
                "log",
               "prompt",
   ]
  )
)

dataset["train"] = process_dataset(dataset["train"])
dataset["validation"] = process_dataset(dataset["validation"])
dataset["test"] = process_dataset(dataset["test"])
```

Model

```
notebook_login()
```

By: Tarun S Gowda

LLM-Finetuning with PEFT

VBox(children=(HTML(value=' <img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv...

```
def create_model_and_tokenizer():
    bnb_config = BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_use_double_quant=True,
        bnb_4bit_quant_type="nf4",
        bnb_4bit_compute_dtype=torch.float16,
)

model = AutoModelForCausalLM.from_pretrained(
    MODEL_NAME,
    device_map={"":0},
    trust_remote_code=True,
    quantization_config=bnb_config
)
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
tokenizer.pad_token = tokenizer.eos_token

return model, tokenizer
```

```
model, tokenizer = create_model_and_tokenizer()
model.config.use_cache = False
```

```
Downloading (…)lve/main/config.json:   0%|          | 0.00/880 [00:00<?, ?B/s]
Downloading (…)former_sequential.py:   0%|          | 0.00/2.23k [00:00<?, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/microsoft/phi-1_5:
- configuration_mixformer_sequential.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.

Downloading (…)former_sequential.py:   0%|          | 0.00/32.2k [00:00<?, ?B/s]
A new version of the following files was downloaded from
https://huggingface.co/microsoft/phi-1_5:
- modeling_mixformer_sequential.py
. Make sure to double-check they do not contain any added malicious code. To avoid
downloading new versions of the code file, you can pin a revision.

Downloading pytorch_model.bin:   0%|          | 0.00/2.84G [00:00<?, ?B/s]
Downloading (…)neration_config.json:   0%|          | 0.00/69.0 [00:00<?, ?B/s]
Downloading (…)okenizer_config.json:   0%|          | 0.00/237 [00:00<?, ?B/s]
Downloading (…)olve/main/vocab.json:   0%|          | 0.00/798k [00:00<?, ?B/s]
Downloading (…)olve/main/merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]
Downloading (…)/main/tokenizer.json:   0%|          | 0.00/2.11M [00:00<?, ?B/s]
```

By: Tarun S Gowda

```
Downloading (…)in/added_tokens.json:   0%|          | 0.00/1.08k [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/99.0 [00:00<?, ?B/s]
```

```
model.config.quantization_config.to_dict()
```

```
{'quant_method': <QuantizationMethod.BITS_AND_BYTES: 'bitsandbytes'>,
 'load_in_8bit': False,
 'load_in_4bit': True,
 'llm_int8_threshold': 6.0,
 'llm_int8_skip_modules': None,
 'llm_int8_enable_fp32_cpu_offload': False,
 'llm_int8_has_fp16_weight': False,
 'bnb_4bit_quant_type': 'nf4', 'bnb_4bit_use_double_quant': True,
 'bnb_4bit_compute_dtype': 'float16'}
```

```
peft_config = LoraConfig(
      r=16,
      lora_alpha=16,
      target_modules=["Wqkv", "out_proj"],
     lora_dropout=0.05,
     bias="none",
     task_type="CAUSAL_LM"
)
```

```
model = get_peft_model(model, peft_config)
 model.print_trainable_parameters()
```

```
trainable params: 4,718,592 || all params: 1,422,989,312 || trainable%:
0.3315971497613047
```

Training

```
OUTPUT_DIR = "experiments"
```

```
%load_ext tensorboard
 %tensorboard --logdir experiments/runs
```

```
training_arguments = TrainingArguments(
           output_dir="phi-1_5-finetuned-dialogstudio",
           per_device_train_batch_size=4,
          gradient_accumulation_steps=1,
          learning_rate=2e-4,
          lr_scheduler_type="cosine",
         save_strategy="epoch",
```

```
        logging_steps=1,
        max_steps=3,

        num_train_epochs=1,
        push_to_hub=True
)

trainer = SFTTrainer(
    model=model,
    train_dataset=dataset["train"],
    eval_dataset=dataset["validation"],
    peft_config=peft_config,
    dataset_text_field="text",
   tokenizer=tokenizer,
   args=training_arguments,
)
```

/usr/local/lib/python3.10/dist-packages/trl/trainer/sft_trainer.py:166: UserWarning: You didn't pass a `max_seq_length` argument to the SFTTrainer, this will default to 1024
 warnings.warn(

```
trainer.train()
```

You're using a CodeGenTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` method is faster than using a method to encode the text followed by a call to the `pad` method to get a padded encoding.

[3/3 00:02, Epoch 0/1]

Step	Training Loss
1	3.080800
2	2.887100
3	3.302700

```
TrainOutput(global_step=3, training_loss=3.090197483698527, metrics={'train_runtime':
7.7717, 'train_samples_per_second': 1.544, 'train_steps_per_second': 0.386, 'total_flos':
22418657574912.0, 'train_loss': 3.090197483698527, 'epoch': 0.01})
```

```
trainer.evaluate()
```

[14/14 00:11]

```
{'eval_loss': 3.243016242980957,
 'eval_runtime': 12.0781,
 'eval_samples_per_second': 9.107,
 'eval_steps_per_second': 1.159,
 'epoch': 0.01}
```

```
trainer.push_to_hub()
```

'https://huggingface.co/ashishpatel26/phi-1_5-finetuned-dialogstudio/tree/main/'

```
trainer.save_model("phi-1_5-finetuned-dialogstudio")
```

```
Trainer.model
```

```
PeftModelForCausalLM(
  (base_model): LoraModel(
    (model): MixFormerSequentialForCausalLM(
      (layers): Sequential(
        (0): Embedding(
          (wte): Embedding(51200, 2048)
          (drop): Dropout(p=0.0, inplace=False)
        )
        (1): ParallelBlock(
          (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
          (resid_dropout): Dropout(p=0.0, inplace=False)
          (mixer): MHA(
            (rotary_emb): RotaryEmbedding()
            (Wqkv): Linear4bit(ures=2048, out_features=8192, bias=True)
            (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
            (act): NewGELUActivation()
          )
        )
        (6): ParallelBlock(
          (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
          (resid_dropout): Dropout(p=0.0, inplace=False)
          (mixer): MHA(
            (rotary_emb): RotaryEmbedding()
            (Wqkv): Linear4bit((lora_B): ModuleDict(
                (default): Linear(in_features=16, out_features=2048, bias=False)
              )
              (lora_embedding_A): ParameterDict()
              (lora_embedding_B): ParameterDict()
            )
            (inner_attn): SelfAttention(
              (drop): Dropout(p=0.0, inplace=False)
            )
            (inner_cross_attn): CrossAttention(
              (drop): Dropout(p=0.0, inplace=False)
            )
          )
          (mlp): MLP(
            (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
            (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
            (act): NewGELUActivation()
          )
        )
```

By: Tarun S Gowda

```
        (18): ParallelBlock(
          (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
          (resid_dropout): Dropout(p=0.0, inplace=False)
          (mixer): MHA(
            (rotary_emb): RotaryEmbedding()
            (Wqkv): Linear4bit(
              in_features=2048, out_features=6144, bias=True
              (lora_dropout): ModuleDict(
                (default): Dropout(p=0.05, inplace=False)
              )
              (lora_A): ModuleDict(
                (default): Linear(in_features=2048, out_features=16, bias=False)
              )
              (lora_B): ModuleDict(
                (default): Linear(in_features=16, out_features=6144, bias=False)
              )
              (lora_embedding_A): ParameterDict()
              (lora_embedding_B): ParameterDict()
            )
            (out_proj): Linear4bit(
              in_features=2048, out_features=2048, bias=True
              (lora_dropout): ModuleDict(
                (default): Dropout(p=0.05, inplace=False)
              )
              (lora_A): ModuleDict(
                (default): Linear(in_features=2048, out_features=16, bias=False)
              )
              (lora_B): ModuleDict(
                (default): Linear(in_features=16, out_features=2048, bias=False)
              )
              (lora_embedding_A): ParameterDict()
              (lora_embedding_B): ParameterDict()
            )
            (inner_attn): SelfAttention(
              (drop): Dropout(p=0.0, inplace=False)
            )
            (inner_cross_attn): CrossAttention(
              (drop): Dropout(p=0.0, inplace=False)
            )
          )
          (mlp): MLP(
            (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
            (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
            (act): NewGELUActivation()
          )
        )
        (19): ParallelBlock(
          (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
          (resid_dropout): Dropout(p=0.0, inplace=False)
          (mixer): MHA(
```

```
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (inner_attn): SelfAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (mlp): MLP(
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
      )
    )
    (20): ParallelBlock(
      (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
      (resid_dropout): Dropout(p=0.0, inplace=False)
      (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
```

```
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (inner_attn): SelfAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (mlp): MLP(
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
      )
    )
    (21): ParallelBlock(
      (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
      (resid_dropout): Dropout(p=0.0, inplace=False)
      (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
```

```
            )
            (lora_B): ModuleDict(
              (default): Linear(in_features=16, out_features=6144, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
          )
          (out_proj): Linear4bit(
            in_features=2048, out_features=2048, bias=True
            (lora_dropout): ModuleDict(
              (default): Dropout(p=0.05, inplace=False)
            )
            (lora_A): ModuleDict(
              (default): Linear(in_features=2048, out_features=16, bias=False)
            )
            (lora_B): ModuleDict(
              (default): Linear(in_features=16, out_features=2048, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
          )
          (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
          )
          (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
          )
        )
        (mlp): MLP(
          (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
          (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
          (act): NewGELUActivation()
        )
      )
      (22): ParallelBlock(
        (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
        (resid_dropout): Dropout(p=0.0, inplace=False)
        (mixer): MHA(
          (rotary_emb): RotaryEmbedding()
          (Wqkv): Linear4bit(
            in_features=2048, out_features=6144, bias=True
            (lora_dropout): ModuleDict(
              (default): Dropout(p=0.05, inplace=False)
            )
            (lora_A): ModuleDict(
              (default): Linear(in_features=2048, out_features=16, bias=False)
            )
            (lora_B): ModuleDict(
              (default): Linear(in_features=16, out_features=6144, bias=False)
            )
```

```
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (inner_attn): SelfAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (mlp): MLP(
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
      )
    )
    (23): ParallelBlock(
      (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
      (resid_dropout): Dropout(p=0.0, inplace=False)
      (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
```

```
            in_features=2048, out_features=2048, bias=True
            (lora_dropout): ModuleDict(
              (default): Dropout(p=0.05, inplace=False)
            )
            (lora_A): ModuleDict(
              (default): Linear(in_features=2048, out_features=16, bias=False)
            )
            (lora_B): ModuleDict(
              (default): Linear(in_features=16, out_features=2048, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
          )
          (inner_attn): SelfAttention(
            (drop): Dropout(p=0.0, inplace=False)
          )
          (inner_cross_attn): CrossAttention(
            (drop): Dropout(p=0.0, inplace=False)
          )
        )
        (mlp): MLP(
          (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
          (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
          (act): NewGELUActivation()
        )
      )
      (24): ParallelBlock(
        (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
        (resid_dropout): Dropout(p=0.0, inplace=False)
        (mixer): MHA(
          (rotary_emb): RotaryEmbedding()
          (Wqkv): Linear4bit(
            in_features=2048, out_features=6144, bias=True
            (lora_dropout): ModuleDict(
              (default): Dropout(p=0.05, inplace=False)
            )
            (lora_A): ModuleDict(
              (default): Linear(in_features=2048, out_features=16, bias=False)
            )
            (lora_B): ModuleDict(
              (default): Linear(in_features=16, out_features=6144, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
          )
          (out_proj): Linear4bit(
            in_features=2048, out_features=6144, bias=True
            (lora_dropout): ModuleDict(
              (default): Dropout(p=0.05, inplace=False)
            )
```

```
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (inner_attn): SelfAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (mlp): MLP(
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
      )
    )
    (7): ParallelBlock(
      (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
      (resid_dropout): Dropout(p=0.0, inplace=False)
      (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
```

```
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (inner_attn): SelfAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (mlp): MLP(
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
      )
    )
    (8): ParallelBlock(
      (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
      (resid_dropout): Dropout(p=0.0, inplace=False)
      (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
```

```
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (inner_attn): SelfAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (mlp): MLP(
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
      )
    )
    (9): ParallelBlock(
      (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
      (resid_dropout): Dropout(p=0.0, inplace=False)
      (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
```

```
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (inner_attn): SelfAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (mlp): MLP(
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
      )
    )
    (10): ParallelBlock(
      (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
      (resid_dropout): Dropout(p=0.0, inplace=False)
      (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
```

```
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (inner_attn): SelfAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (mlp): MLP(
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
      )
    )
    (11): ParallelBlock(
      (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
      (resid_dropout): Dropout(p=0.0, inplace=False)
      (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
```

```
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
      (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
    )
    (mlp): MLP(
      (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
      (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
      (act): NewGELUActivation()
    )
  )
  (12): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
      (rotary_emb): RotaryEmbedding()
      (Wqkv): Linear4bit(
        in_features=2048, out_features=6144, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=6144, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (out_proj): Linear4bit(
        in_features=2048, out_features=2048, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=2048, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (inner_attn): SelfAttention(
```

```
        (drop): Dropout(p=0.0, inplace=False)
      )
      (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
    )
    (mlp): MLP(
      (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
      (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
      (act): NewGELUActivation()
    )
  )
  (13): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
      (rotary_emb): RotaryEmbedding()
      (Wqkv): Linear4bit(
        in_features=2048, out_features=6144, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=6144, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (out_proj): Linear4bit(
        in_features=2048, out_features=2048, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=2048, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
      (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
```

```
      )
    )
    (mlp): MLP(
      (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
      (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
      (act): NewGELUActivation()
    )
  )
  (14): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
      (rotary_emb): RotaryEmbedding()
      (Wqkv): Linear4bit(
        in_features=2048, out_features=6144, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=6144, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (out_proj): Linear4bit(
        in_features=2048, out_features=2048, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=2048, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
      (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
    )
    (mlp): MLP(
      (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
```

```
      (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
      (act): NewGELUActivation()
    )
  )
  (15): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
      (rotary_emb): RotaryEmbedding()
      (Wqkv): Linear4bit(
        in_features=2048, out_features=6144, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=6144, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (out_proj): Linear4bit(
        in_features=2048, out_features=2048, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=2048, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
      (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
    )
    (mlp): MLP(
      (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
      (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
      (act): NewGELUActivation()
    )
  )
```

```
        (16): ParallelBlock(
          (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
          (resid_dropout): Dropout(p=0.0, inplace=False)
          (mixer): MHA(
            (rotary_emb): RotaryEmbedding()
            (Wqkv): Linear4bit(
              in_features=2048, out_features=6144, bias=True
              (lora_dropout): ModuleDict(
                (default): Dropout(p=0.05, inplace=False)
              )
              (lora_A): ModuleDict(
                (default): Linear(in_features=2048, out_features=16, bias=False)
              )
              (lora_B): ModuleDict(
                (default): Linear(in_features=16, out_features=6144, bias=False)
              )
              (lora_embedding_A): ParameterDict()
              (lora_embedding_B): ParameterDict()
            )
            (out_proj): Linear4bit(
              in_features=2048, out_features=2048, bias=True
              (lora_dropout): ModuleDict(
                (default): Dropout(p=0.05, inplace=False)
              )
              (lora_A): ModuleDict(
                (default): Linear(in_features=2048, out_features=16, bias=False)
              )
              (lora_B): ModuleDict(
                (default): Linear(in_features=16, out_features=2048, bias=False)
              )
              (lora_embedding_A): ParameterDict()
              (lora_embedding_B): ParameterDict()
            )
            (inner_attn): SelfAttention(
              (drop): Dropout(p=0.0, inplace=False)
            )
            (inner_cross_attn): CrossAttention(
              (drop): Dropout(p=0.0, inplace=False)
            )
          )
          (mlp): MLP(
            (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
            (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
            (act): NewGELUActivation()
          )
        )
        (17): ParallelBlock(
          (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
          (resid_dropout): Dropout(p=0.0, inplace=False)
          (mixer): MHA(
```

```
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
```

```
      (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
      (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
    )
    (mlp): MLP(
      (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
      (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
      (act): NewGELUActivation()
    )
  )
  (2): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
      (rotary_emb): RotaryEmbedding()
      (Wqkv): Linear4bit(
        in_features=2048, out_features=6144, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=6144, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (out_proj): Linear4bit(
        in_features=2048, out_features=2048, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=2048, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
      (inner_cross_attn): CrossAttention(
```

```
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (mlp): MLP(
        (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
        (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
        (act): NewGELUActivation()
      )
    )
    (3): ParallelBlock(
      (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
      (resid_dropout): Dropout(p=0.0, inplace=False)
      (mixer): MHA(
        (rotary_emb): RotaryEmbedding()
        (Wqkv): Linear4bit(
          in_features=2048, out_features=6144, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=6144, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (out_proj): Linear4bit(
          in_features=2048, out_features=2048, bias=True
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.05, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=2048, out_features=16, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=16, out_features=2048, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (inner_attn): SelfAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
        (inner_cross_attn): CrossAttention(
          (drop): Dropout(p=0.0, inplace=False)
        )
      )
      (mlp): MLP(
```

```
      (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
      (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
      (act): NewGELUActivation()
    )
  )
  (4): ParallelBlock(
    (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
    (resid_dropout): Dropout(p=0.0, inplace=False)
    (mixer): MHA(
      (rotary_emb): RotaryEmbedding()
      (Wqkv): Linear4bit(
        in_features=2048, out_features=6144, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=6144, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (out_proj): Linear4bit(
        in_features=2048, out_features=2048, bias=True
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.05, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=2048, out_features=16, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=16, out_features=2048, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (inner_attn): SelfAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
      (inner_cross_attn): CrossAttention(
        (drop): Dropout(p=0.0, inplace=False)
      )
    )
    (mlp): MLP(
      (fc1): Linear4bit(in_features=2048, out_features=8192, bias=True)
      (fc2): Linear4bit(in_features=8192, out_features=2048, bias=True)
      (act): NewGELUActivation()
    )
```

```
        )
        (5): ParallelBlock(
          (ln): LayerNorm((2048,), eps=1e-05, elementwise_affine=True)
          (resid_dropout): Dropout(p=0.0, inplace=False)
          (mixer): MHA(
            (rotary_emb): RotaryEmbedding()
            (Wqkv): Linear4bit(
              in_features=2048, out_features=6144, bias=True
              (lora_dropout): ModuleDict(
                (default): Dropout(p=0.05, inplace=False)
              )
              (lora_A): ModuleDict(
                (default): Linear(in_features=2048, out_features=16, bias=False)
              )
              (lora_B): ModuleDict(
                (default): Linear(in_features=16, out_features=6144, bias=False)
              )
              (lora_embedding_A): ParameterDict()
              (lora_embedding_B): ParameterDict()
            )
            (out_proj): Linear4bit(
              in_features=2048, out_features=2048, bias=True
              (lora_dropout): ModuleDict(
                (default): Dropout(p=0.05, inplace=False)
              )
              (lora_A): ModuleDict(
                (default): Linear(in_features=2048, out_features=16, bias=False)
              )
              (lora_B): ModuleDict(
                (default): Linear(in_features=16, out_features=2048, bias=False)
              )
              (lora_embedding_A): ParameterDict()
              (lora_embedding_B): ParameterDict()
            )
            (inner_attn): SelfAttention(
              (drop): Dropout(p=0.0, inplace=False)
            )
            (inner_cross_attn): CrossAttention(
              (drop): Dropout(p=0.0, inplace=False)
            )
          )
          (mlp): MLP(
            (fc1): Linear4bit(in_feat
```