Fine-Tuning using SFT for Financial Sentiment

Introduction

In the previous lesson, we experimented with the method of fine-tuning an LLM to follow the instructions like a chatbot. Although this proves beneficial across various applications, we can similarly employ this strategy to train a model tailored for a particular domain.

In this lesson, our goal is to create a thoroughly tuned model for conducting **sentiment analysis on financial statements**. Ideally, the LLM would assess financial tweets by categorizing them as Positive, Negative, or Neutral. The dataset utilized in this lesson is the one curated in the FinGPT project.

As previously stated, the dataset remains the pivotal and influential factor. Having acknowledged that, and given that we've extensively addressed the process of Supervised Fine-Tuning (SFT) before, this lesson will predominantly touch upon the dataset we utilized and the preprocessing steps involved. Nonetheless, a comprehensive notebook script for running and experimenting is provided at the conclusion of this lesson.

The activities showcased in this tutorial involve the utilization of the 4th Generation Intel® Xeon® Scalable Processors (with 64GB RAM), with the use of Intel® Advanced Matrix Extensions (Intel® AMX). Both finetuning and inference can be accomplished by leveraging its optimization technologies. You can spin up a virtual machine using GCP Compute Engine as explained in the previous lesson.

Follow the instructions in the course introduction to spin up a VM with Compute Engine with high-end Intel CPUs.

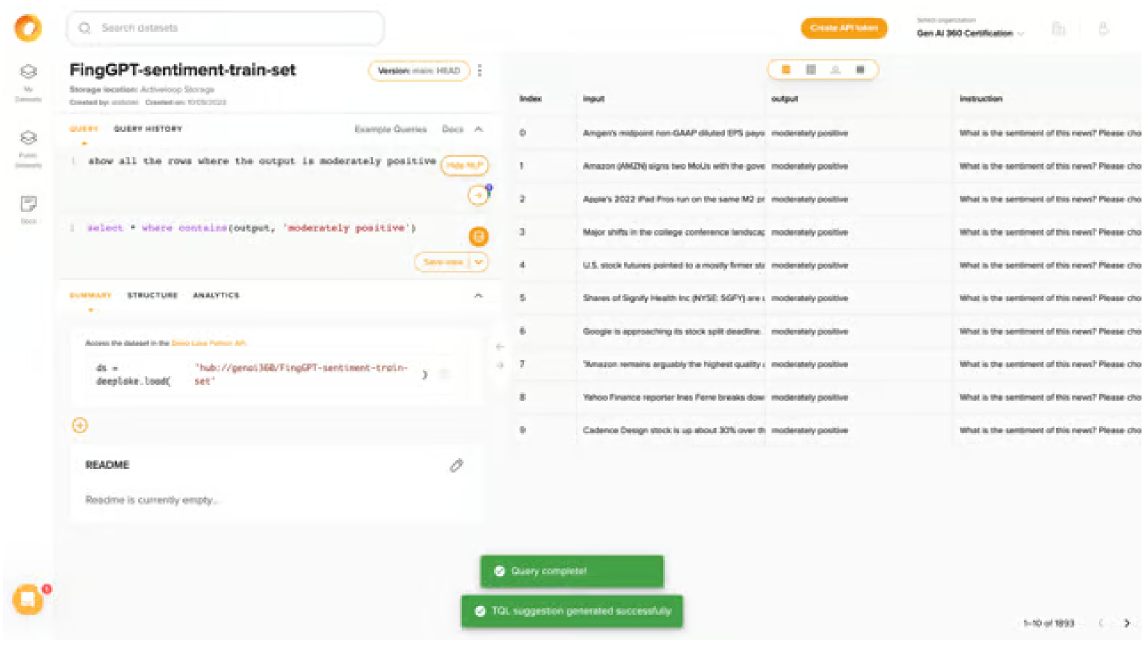
⚠️ Beware of costs when you spin up virtual machines. The total cost will depend on the machine type and the up time of the machine. Always remember to monitor your costs in the billing section of GCP and to spin off your virtual machines when you don't use them.

💡 If you just want to replicate the code in the lesson spending very few money, you can just run the training in your virtual machine and stop it after a few iterations.

Load the Dataset

We are set to utilize the FinGPT sentiment dataset, comprising a set of financial tweets along with their corresponding labels. Additionally, this dataset features an `instruction` column containing the initial task directive. Typically, this instruction prompts something akin to "What is the sentiment of the following content? Choose from Positive, Negative, or Neutral."

A smaller subset of the dataset can be accessed from the 300+ free public datasets curated by team Activeloop accessible in Deep Lake format. We've deliberately chosen a smaller subset to expedite the fine-tuning process. Specifically, the training set comprises 20,000 data points, while we employ 2,000 samples for validation purposes. The dataset can be explored and queried using the Deep Lake Web UI or filtered using the Python package. The Deep Lake visualization engine enables us to query the dataset and filter relevant rows using its query field. The NLP feature allows you to compose your query in English and receive the corresponding TQL query.



FingGPT-sentiment-train-set

```
show all the rows where the output is moderately positive
```

```
select * where contains(output, 'moderately positive')
```

Access the dataset in the Deep Lake Python API

```
ds = deeplake.load('hub://genai360/FingGPT-sentiment-train-set')
```

Index	input	output	instruction
0	Amgen's midpoint non-GAAP diluted EPS payo	moderately positive	What is the sentiment of this news? Please cho
1	Amazon (AMZN) signs two MoUs with the gove	moderately positive	What is the sentiment of this news? Please cho
2	Apple's 2022 iPad Pros run on the same M2 pr	moderately positive	What is the sentiment of this news? Please cho
3	Major shifts in the college conference landscap	moderately positive	What is the sentiment of this news? Please cho
4	U.S. stock futures pointed to a mostly firmer st	moderately positive	What is the sentiment of this news? Please cho
5	Shares of Signify Health Inc (NYSE: SGFY) are	moderately positive	What is the sentiment of this news? Please cho
6	Google is approaching its stock split deadline.	moderately positive	What is the sentiment of this news? Please cho
7	"Amazon remains arguably the highest quality	moderately positive	What is the sentiment of this news? Please cho
8	Yahoo Finance reporter Ines Ferre breaks dow	moderately positive	What is the sentiment of this news? Please cho
9	Cadence Design stock is up about 30% over th	moderately positive	What is the sentiment of this news? Please cho

The Deep Lake Visualization Engine table view with filtering.

By utilizing the `deeplake.load()` function, we can create the Dataset object and load the samples.

```python
import deeplake

# Connect to the training and testing datasets
ds = deeplake.load('hub://genai360/FingGPT-sentiment-train-set')
ds_valid = deeplake.load('hub://genai360/FingGPT-sentiment-valid-set')

print(ds)
```

The sample code.

```
Dataset(path='hub://genai360/FingGPT-sentiment-train-set', read_only=True, tensors=['input
```

The output.

At this point, we can proceed to create the function that formats a sample from the dataset into a suitable input for the model. The primary distinction from our previous approach lies in incorporating the instructions at the start of the prompt. The structure is as outlined below: `<instruction>\n\nContent: <tweet>\n\nSentiment: <sentiment>`. The placeholders enclosed in `<>` will be substituted with corresponding values from the dataset.

```python
def prepare_sample_text(example):
    """Prepare the text from a sample of the dataset."""
    text = f"{example['instruction'].text()}\n\nContent: {example['input'].text()}\n\nSent
    return text
```

The sample code.

Presented here is a formatted input derived from an entry in the dataset.

```
What is the sentiment of this news? Please choose an answer from {negative/neutral/positiv

Content: Diageo Shares Surge on Report of Possible Takeover by Lemann

Sentiment: positive
```

The output.

The subsequent steps should be recognizable from earlier lessons. We initialize the tokenizer for the OPT-1.3B large language model and use the `ConstantLengthDataset` to structure the samples. The tokenizer is then employed to convert them into token IDs. Additionally, the class packs multiple samples until the sequence length threshold is reached, thus enhancing the efficiency of the training process.

```
# Load the tokenizer
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained("facebook/opt-1.3b")

# Create the ConstantLengthDataset
from trl.trainer import ConstantLengthDataset

train_dataset = ConstantLengthDataset(
    tokenizer,
    ds,
    formatting_func=prepare_sample_text,
    infinite=True,
    seq_length=1024
)

eval_dataset = ConstantLengthDataset(
    tokenizer,
    ds_valid,
    formatting_func=prepare_sample_text,
    seq_length=1024
)

# Show one sample from train set
iterator = iter(train_dataset)
sample = next(iterator)
print(sample)
```

The sample code.

```
{'input_ids': tensor([50118, 35212,  8913,  ...,  2430,     2,     2]),'labels': tensor([5
```

The output.

💡 Remember to execute the following code to reset the iterator pointer, if the iterator is used to print a sample from the dataset,

```
train_dataset.start_iteration = 0
```

Initialize the Model and Trainer

The "Fine-Tuning using SFT" tutorial clarifies the code snippets within this subsection. For additional inquiries, kindly refer to that resource for further understanding. We will quickly walk through the code.

Please bear in mind that the fine-tuned checkpoint will be accessible in the Inference section if resources for the fine-tuning process are needed. Additionally, the specifics of the training process are recorded and can be accessed through Weights and Biases. During the training process, system activity can be tracked, including metrics such as memory usage, CPU utilization, duration, loss values, and a range of other parameters. Here's the Weights and Biases report of the finetuning of this lesson.

We start by defining the arguments necessary to configure the training process. We use the `LoraConfig` class from the PEFT library for that. Subsequently, we employ the `TrainingArguments` class from the transformers library to control the training loop.

```
# Define LoRAConfig
from peft import LoraConfig

lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM",
)

# Define TrainingArguments
from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir="./OPT-fine_tuned-FinGPT-CPU",
    dataloader_drop_last=True,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    num_train_epochs=10,
    logging_steps=5,
    per_device_train_batch_size=12,
    per_device_eval_batch_size=12,
    learning_rate=1e-4,
    lr_scheduler_type="cosine",
    warmup_steps=100,
    gradient_accumulation_steps=1,
    gradient_checkpointing=False,
    fp16=False,
    bf16=True,
    weight_decay=0.05,
    ddp_find_unused_parameters=False,
    run_name="OPT-fine_tuned-FinGPT-CPU",
    report_to="wandb",
)
```

The sample code.

The subsequent task involves loading the OPT-1.3B model in the `bfloat16` format, which is easily managed by Intel® CPUs and saves memory during fine-tuning.

```
from transformers import AutoModelForCausalLM
import torch

model = AutoModelForCausalLM.from_pretrained(
    "facebook/opt-1.3b", torch_dtype=torch.bfloat16
)
```

The sample code.

The subsequent stage entails casting specific layers within the network to complete 32-bit precision, enhancing the model's stability throughout training.

```
from torch import nn

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)
```

The sample code.

Now, connect the model, dataset, training arguments, and Lora config together using the `SFTTrainer` class to start the training process by invoking the `.train()` method.

```
from trl import SFTTrainer
```

```
trainer = SFTTrainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    peft_config=lora_config,
    packing=True,
)

print("Training...")
trainer.train()
```

The sample code.

💡 Access the best checkpoint that we trained by using the following URL.
Additionally, find more information about the fine-tuning process on the Weights and Biases project page.

📎 OPT-fine_tuned-FinGPT-CPU.zip 35038.5KB

Merging LoRA and OPT

Before conducting inference and observing the results, the final task is to load the LoRA adaptors from the preceding stage and merge them with the base model.

```
# Load the base model (OPT-1.3B)
from transformers import AutoModelForCausalLM
import torch

model = AutoModelForCausalLM.from_pretrained(
  "facebook/opt-1.3b", return_dict=True, torch_dtype=torch.bfloat16
)

# Load the LoRA adaptors
from peft import PeftModel

# Load the Lora model
model = PeftModel.from_pretrained(model, "./OPT-fine_tuned-FinGPT-CPU/<desired_checkpoint>
model.eval()
model = model.merge_and_unload()

# Save for future use
model.save_pretrained("./OPT-fine_tuned-FinGPT-CPU/merged")
```

The sample code.

Inference

We randomly selected four previously unseen examples from the dataset and provided them as input to both the vanilla base model (OPT-1.3B) and the fine-tuned model in order to contrast their respective responses. The code is relatively straightforward when utilizing the `.generate()` method from the Transformers library.

```
inputs = tokenizer("""What is the sentiment of this news? Please choose an answer from {st
Content: UPDATE 1-AstraZeneca sells rare cancer drug to Sanofi for up to S300 mln.\n\nSent

generation_output = model.generate(**inputs,
                                    return_dict_in_generate=True,
                                    output_scores=True,
                                    max_length=256,
                                    num_beams=1,
                                    do_sample=True,
                                    repetition_penalty=1.5,
                                    length_penalty=2.)

print( tokenizer.decode(generation_output['sequences'][0]) )
```

The sample code.

```
What is the sentiment of this news? Please choose an answer from {strong negative/moderate
```

The output.

Observing the samples, we see that the model fine-tuned on financial tweets for the specific domain exhibits good performance in terms of adhering to instructions and comprehending the task at hand. Below, find a list of prompts to toggle the outputs by clicking on the right arrow icon.

▸ 1. UPDATE 1-AstraZeneca sells rare cancer drug to Sanofi for up to S300 mln.

▸ 2. SABMiller revenue hit by weaker EM currencies

▸ 3. Buffett's Company Reports 37 Percent Drop in 2Q Earnings

▸ 4. For a few hours this week, the FT gained access to Poly, the university where students in Hong Kong have been trapped…

These instances demonstrate that the vanilla model primarily focuses on the default language modeling task, which involves predicting the next word based on the input. In contrast, the fine-tuned model comprehends the instruction and generates the requested content.

Conclusion

This tutorial illustrated the procedure of leveraging publicly accessible datasets or curated data from your organization to develop a personalized model that caters to personalized requirements. More powerful base models, such as LLaMA2 can be employed, by modifying the model id to load both the model and its tokenizer. However, it needs more resources to initiate the fine-tuning process.

We also showcased the feasibility of conducting the fine-tuning process using 4th Generation Intel® Xeon® Scalable Processors for both the fine-tuning and inference stages. The Intel® oneAPI Math Kernel Library (Intel® MKL) toolkits offer a suite of utilities aimed at boosting the efficiency of various applications, including those in the field of Artificial Intelligence. It's intriguing to anticipate how the latest CPU architectures like Emerald Rapids, Sierra Forest, and Granite Rapids will shape and potentially transform the deep learning landscape.

In the upcoming lessons, we will integrate GPUs to fine-tune a model using the RLHF (Reinforcement Learning from Human Feedback) approach.

>> notebook.

>> Weights and Biases report.

For more information on Intel® Accelerator Engines, visit this resource page. Learn more about Intel® Extension for Transformers, an Innovative Transformer-based Toolkit to Accelerate GenAI/LLM Everywhere here.

Intel, the Intel logo, and Xeon are trademarks of Intel Corporation or its subsidiaries.