Step 3: Model Fine-tuning

In this notebook, you'll fine-tune the Meta Llama 2 7B large language model, deploy the fine-tuned model, and test it's text generation and domain knowledge capabilities.

Fine-tuning refers to the process of taking a pre-trained language model and retraining it for a different but related task using specific data. This approach is also known as transfer learning, which involves transferring the knowledge learned from one task to another. Large language models (LLMs) like Llama 2 7B are trained on massive amounts of unlabeled data and can be fine-tuned on domain domain datasets, making the model perform better on that specific domain.

Input: A train and an optional validation directory. Each directory contains a CSV/JSON/TXT file. For CSV/JSON files, the train or validation data is used from the column called 'text' or the first column if no column called 'text' is found. The number of files under train and validation should equal to one.

- **You'll choose your dataset below based on the domain you've chosen**

Output: A trained model that can be deployed for inference.
After you've fine-tuned the model, you'll evaluate it with the same input you used in project step 2: model evaluation.

Set up

Install and import the necessary packages. Restart the kernel after executing the cell below.

```
!pip install --upgrade sagemaker datasets
!pip install awscli --quiet
!aws configure
!aws configure get region
```

```
    Requirement already satisfied: deprecated>=1.2.6 in /usr/local/lib/python3.10/dist-packages (from opentelemetry-api<3,>=1.9.
    Requirement already satisfied: opentelemetry-semantic-conventions==0.48b0 in /usr/local/lib/python3.10/dist-packages (from o
    Requirement already satisfied: wrapt<2,>=1.10 in /usr/local/lib/python3.10/dist-packages (from deprecated>=1.2.6->openteleme
    Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from gitdb<5,>=4.0.1->gitpython<4
    Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth~=2.0->data
    Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth~=2.0->databricks-s
    Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->
    AWS Access Key ID [****************B5MN]:
    AWS Secret Access Key [****************yta9]:
    Default region name [us-east-1]:
    Default output format [None]:
    us-east-1
```

Select the model to fine-tune

```
model_id, model_version = "meta-textgeneration-llama-2-7b", "2.*"
```

In the cell below, choose the training dataset text for the domain you've chosen and update the code in the cell below:

To create a finance domain expert model:

- `"training": f"s3://genaiwithawsproject2024/training-datasets/finance"`

To create a medical domain expert model:

- `"training": f"s3://genaiwithawsproject2024/training-datasets/medical"`

To create an IT domain expert model:

- `"training": f"s3://genaiwithawsproject2024/training-datasets/it"`

```
from sagemaker.jumpstart.estimator import JumpStartEstimator
import boto3
!aws configure

role_arn = "arn:aws:iam::317957367373:role/SageMakerExecutionRole"

model_id = "meta-textgeneration-llama-2-7b"

estimator = JumpStartEstimator(
    model_id=model_id,
    environment={"accept_eula": "true"},
    instance_type="ml.g5.8xlarge",
    role=role_arn
)

estimator.set_hyperparameters(instruction_tuned="False", epoch="5")

training_data_s3_path = "s3://sagemaker-us-east-1-317957367373/financialDataset.txt"

estimator.fit({"training": training_data_s3_path})
```

```
evaluating Epoch:  67%|#033[32m██████    #033[0m| 2/3 [00:00<00:00,  3.12it/s]
evaluating Epoch: 100%|#033[32m██████████#033[0m| 3/3 [00:00<00:00,  3.12it/s]
evaluating Epoch: 100%|#033[32m██████████#033[0m| 3/3 [00:00<00:00,  3.12it/s]
eval_ppl=tensor(11.3218, device='cuda:0') eval_epoch_loss=tensor(2.4267, device='cuda:0')
we are about to save the PEFT modules
PEFT modules are saved in saved_peft_model directory
best eval loss on epoch 4 is 2.4267330169677734
Epoch 5: train_perplexity=15.9653, train_epoch_loss=2.7704, epcoh time 7.364354001000038s
INFO:root:Key: avg_train_prep, Value: 19.065526962280273
INFO:root:Key: avg_train_loss, Value: 2.940845489501953
INFO:root:Key: avg_eval_prep, Value: 12.657987594604492
INFO:root:Key: avg_eval_loss, Value: 2.5352256298065186
INFO:root:Key: avg_epoch_time, Value: 7.43241102940001
INFO:root:Key: avg_checkpoint_time, Value: 0.7406779179999716
INFO:root:Combining pre-trained base model with the PEFT adapter module.
Loading checkpoint shards:   0%|          | 0/2 [00:00<?, ?it/s]
Loading checkpoint shards:  50%|█████     | 1/2 [00:01<00:01,  1.91s/it]
Loading checkpoint shards: 100%|██████████| 2/2 [00:02<00:00,  1.21s/it]
Loading checkpoint shards: 100%|██████████| 2/2 [00:02<00:00,  1.31s/it]
INFO:root:Saving the combined model in safetensors format.
INFO:root:Saving complete.
INFO:root:Copying tokenizer to the output directory.
INFO:root:Putting inference code with the fine-tuned model directory.
2024-10-09 03:19:45,247 sagemaker-training-toolkit INFO     Waiting for the process to finish and give a return code.
2024-10-09 03:19:45,247 sagemaker-training-toolkit INFO     Done waiting for a return code. Received 0 from exiting process.
2024-10-09 03:19:45,247 sagemaker-training-toolkit INFO     Reporting training SUCCESS

2024-10-09 03:19:51 Uploading - Uploading generated training model
2024-10-09 03:20:29 Completed - Training job completed
Training seconds: 604
Billable seconds: 604
```

Deploy the fine-tuned model

Next, we deploy the domain fine-tuned model. We will compare the performance of the fine-tuned and pre-trained model.

```
finetuned_predictor = estimator.deploy(
#predictor = estimator.deploy(
    initial_instance_count=1,
    instance_type="ml.g5.4xlarge",
)
print(finetuned_predictor.endpoint_name)
```

```
-----------!meta-textgeneration-llama-2-7b-2024-10-09-03-33-13-148
```

```
import boto3

client = boto3.client('sagemaker')
response = client.describe_endpoint(EndpointName=finetuned_predictor.endpoint_name)
print(response['EndpointStatus'])

logs = boto3.client('logs')
log_group = '/aws/sagemaker/Endpoints/{}'.format(finetuned_predictor.endpoint_name)
log_streams = logs.describe_log_streams(logGroupName=log_group)
for stream in log_streams['logStreams']:
    log_events = logs.get_log_events(logGroupName=log_group, logStreamName=stream['logStreamName'])
    for event in log_events['events']:
        print(event['message'])
```

```
InService
#033[2m2024-10-09T03:38:03.049482Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_launcher#033[0m#033[2m:#033[0m Args {
#033[2m2024-10-09T03:38:03.049543Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_launcher#033[0m#033[2m:#033[0m Using d
#033[2m2024-10-09T03:38:03.049603Z#033[0m #033[32m INFO#033[0m #033[1mdownload#033[0m: #033[2mtext_generation_launcher#033[0
#033[2m2024-10-09T03:38:07.799307Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_launcher#033[0m#033[2m:#033[0m Files a
#033[2m2024-10-09T03:38:08.354800Z#033[0m #033[32m INFO#033[0m #033[1mdownload#033[0m: #033[2mtext_generation_launcher#033[0
#033[2m2024-10-09T03:38:08.354965Z#033[0m #033[32m INFO#033[0m #033[1mshard-manager#033[0m: #033[2mtext_generation_launcher#
#033[2m2024-10-09T03:38:18.365267Z#033[0m #033[32m INFO#033[0m #033[1mshard-manager#033[0m: #033[2mtext_generation_launcher#
#033[2m2024-10-09T03:38:18.506723Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_launcher#033[0m#033[2m:#033[0m Server
#033[2m2024-10-09T03:38:18.565467Z#033[0m #033[32m INFO#033[0m #033[1mshard-manager#033[0m: #033[2mtext_generation_launcher#
#033[2m2024-10-09T03:38:18.664383Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_launcher#033[0m#033[2m:#033[0m Startin
#033[2m2024-10-09T03:38:18.719782Z#033[0m #033[33m WARN#033[0m #033[2mtokenizers::tokenizer::serialization#033[0m#033[2m:#03
#033[2m2024-10-09T03:38:18.720084Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_router#033[0m#033[2m:#033[0m #033[2mro
#033[2m2024-10-09T03:38:18.720099Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_router#033[0m#033[2m:#033[0m #033[2mro
#033[2m2024-10-09T03:38:18.720112Z#033[0m #033[33m WARN#033[0m #033[2mtext_generation_router#033[0m#033[2m:#033[0m #033[2mro
#033[2m2024-10-09T03:38:18.722958Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_router#033[0m#033[2m:#033[0m #033[2mro
#033[2m2024-10-09T03:38:22.368372Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_launcher#033[0m#033[2m:#033[0m Cuda Gr
#033[2m2024-10-09T03:38:23.270912Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_router#033[0m#033[2m:#033[0m #033[2mro
#033[2m2024-10-09T03:38:23.270934Z#033[0m #033[32m INFO#033[0m #033[2mtext_generation_router#033[0m#033[2m:#033[0m #033[2mro
#033[2m2024-10-09T03:38:23.270938Z#033[0m #033[33m WARN#033[0m #033[2mtext_generation_router#033[0m#033[2m:#033[0m #033[2mro
```

Evaluate the pre-trained and fine-tuned model

Next, we use the same input from the model evaluation step to evaluate the performance of the fine-tuned model and compare it with the base pre-trained model.

Create a function to print the response from the model

```
def print_response(payload, response):
    print(payload["inputs"])
    print(f"> {response}")
    print("\n==================================\n")
```

Now we can run the same prompts on the fine-tuned model to evaluate it's domain knowledge.

Replace "inputs" in the next cell with the input to send the model based on the domain you've chosen.

For financial domain:

"inputs": "Replace with sentence below from text"

- "The investment tests performed indicate"
- "the relative volume for the long out of the money options, indicates"
- "The results for the short in the money options"
- "The results are encouraging for aggressive investors"

For medical domain:

"inputs": "Replace with sentence below from text"

- "Myeloid neoplasms and acute leukemias derive from"
- "Genomic characterization is essential for"
- "Certain germline disorders may be associated with"
- "In contrast to targeted approaches, genome-wide sequencing"

For IT domain:

"inputs": "Replace with sentence below from text"

- "Traditional approaches to data management such as"
- "A second important aspect of ubiquitous computing environments is"
- "because ubiquitous computing is intended to"
- "outline the key aspects of ubiquitous computing from a data management perspective."

```
payload = {
    "inputs": "the relative volume for the long out of the money options, indicates",
    "parameters": {
        "max_new_tokens": 64,
        "top_p": 0.9,
        "temperature": 0.6,
        "return_full_text": False,
    },
}
try:
    response = finetuned_predictor.predict(payload, custom_attributes="accept_eula=true")
    print_response(payload, response)
except Exception as e:
    print(e)
```

```
the relative volume for the long out of the money options, indicates
> [{'generated_text': ' a high probability of the stock moving in the same direction as the underlying stock.\nThe relative

==================================
```

```
payload = {
    "inputs": "The results are encouraging for aggressive investors",
    "parameters": {
        "max_new_tokens": 64,
        "top_p": 0.9,
        "temperature": 0.6,
        "return_full_text": False,
    },
}
try:
    response = finetuned_predictor.predict(payload, custom_attributes="accept_eula=true")
    print_response(payload, response)
except Exception as e:
    print(e)
```

```
The results are encouraging for aggressive investors
> [{'generated_text': '. The S&P 500 Index has risen 14% since its low point on March 23.\nThe Dow Jones Industrial Average

==================================
```

```
payload = {
    "inputs": "The results for the short in the money options",
    "parameters": {
        "max_new_tokens": 64,
        "top_p": 0.9,
        "temperature": 0.6,
        "return_full_text": False,
    },
}
try:
    response = finetuned_predictor.predict(payload, custom_attributes="accept_eula=true")
    print_response(payload, response)
except Exception as e:
    print(e)
```

```
The results for the short in the money options
> [{'generated_text': ' are shown below.\nThe results for the short out of the money options are shown below.\nThe results f

==================================
```

```
payload = {
    "inputs": "The investment tests performed indicate",
    "parameters": {
        "max_new_tokens": 64,
        "top_p": 0.9,
        "temperature": 0.6,
        "return_full_text": False,
    },
}
try:
    response = finetuned_predictor.predict(payload, custom_attributes="accept_eula=true")
    print_response(payload, response)
except Exception as e:
    print(e)
```

```
The investment tests performed indicate
> [{'generated_text': ' that the approach described in this paper is able to capture the main features of the real financial

==================================
```

Do the outputs from the fine-tuned model provide domain-specific insightful and relevant content? You can continue experimenting with the inputs of the model to test it's domain knowledge.

Use the output from this notebook to fill out the "model fine-tuning" section of the project documentation report

After you've filled out the report, run the cells below to delete the model deployment

`IF YOU FAIL TO RUN THE CELLS BELOW YOU WILL RUN OUT OF BUDGET TO COMPLETE THE PROJECT`

```
finetuned_predictor.delete_model()
finetuned_predictor.delete_endpoint()
```