ML Ops Final Project Report

Deployment of **PyTorch model** for ‘Text Grammar Categorizer’ using **TorchServe** and Amazon SageMaker

Group 18	
Mehul Gupta	**677991579**
Disha Pandey	**676191227**
Sathwik Maddi	**658204431**

Table of Contents

Aim for the Project

The project aims to deploy a Machine Learning model using TorchServe. We are trying to accomplish the integration of our project into an existing production environment and explore our understanding of the deployment process. By this, we try to enhance the PyTorch ML models' implementation, and track/run the 'Grammar Categorizer' using Machine Learning. The Grammar Categorizer identifies if the given sentences are grammatically correct or not. Here, we build the PyTorch model to get the predicted class as 1 for a correct statement and the predicted class as 0 for an incorrect statement. Finally, we try to serve the model in production using TorchServe.

Possible Solutions and Comparison

At present, we have many solutions to serve the ML models in production. ML Ops is considered a standard procedure to work with Machine Learning models during the entire lifecycle. Similarly, we are using PyTorch models to categorize our sentences if they are grammatically correct or not and eventually deploying them using TorchServe. There are various default ways to serve PyTorch models, such as;

- Kubeflow -
- MLflow
- Sagemaker
- Kserve: Supports both v1 and v2 API
- Vertex AI

What is PyTorch?

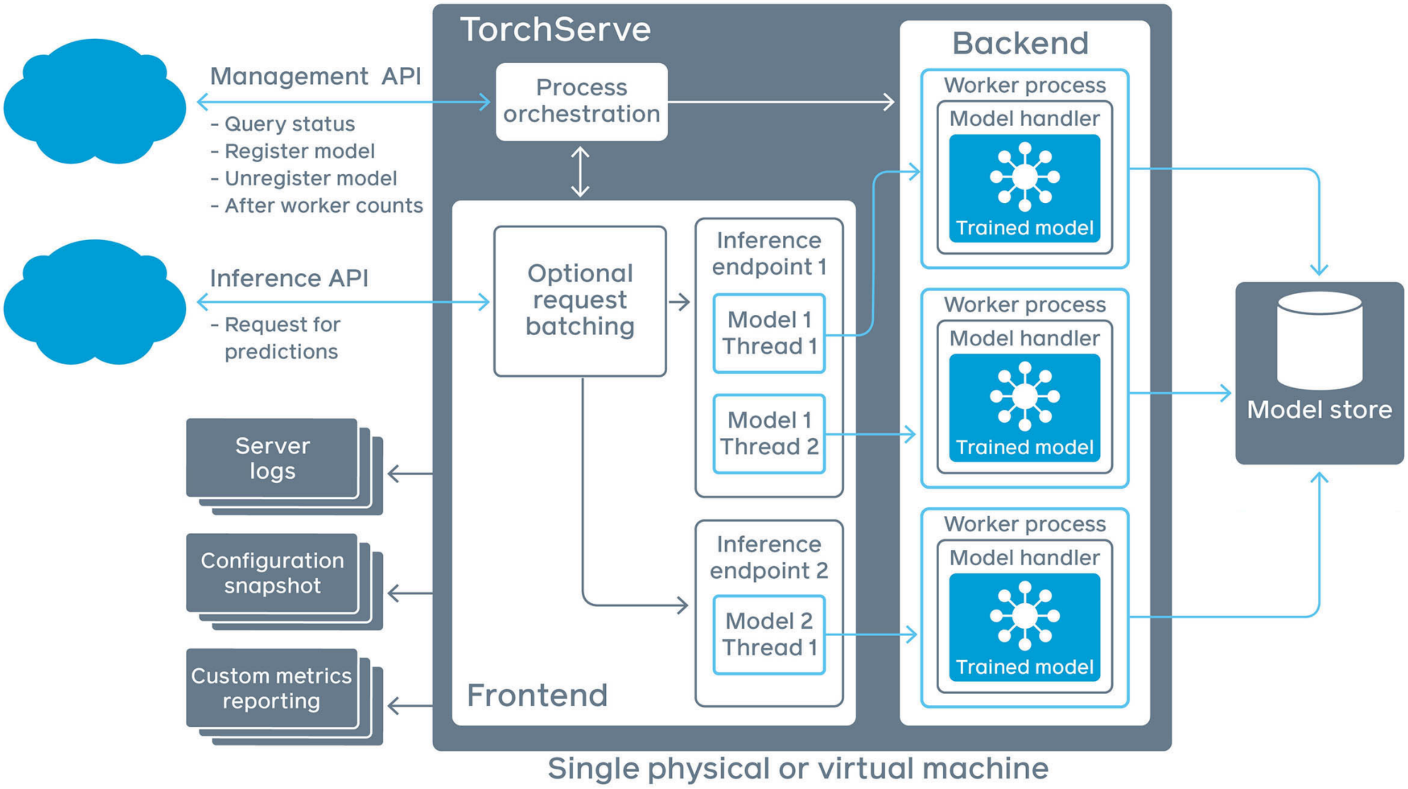
PyTorch is an open-source ML framework that helps in accelerating the path from research prototyping to its deployment in production.

TorchServe Library

TorchServe is a flexible environment and easy tool for serving and scaling the PyTorch models into a production environment. It has a Model Management API which allows multi-model management with an optimized form of worker-to-model allocation. It also has an Inference API which provides REST and gRPC support for batched inference.

The serving platform was designed as a multi-model inference framework that manages several worker processes that are extensively assigned to various models. In TorchServe this is done along with having the behavior of those workers which is determined by the handler file and also a model store where the weights are loaded.

TorchServe Architecture



Business Benefits

As TorchServe embeds default handlers for the most popular applications, like object identification and text classification, in addition to offering a low latency prediction API, PyTorch users can deploy their models faster and without the need to develop custom code. TorchServe also offers RESTful endpoints for application integration, model versioning for A/B testing, monitoring metrics, and multi-model serving. TorchServe supports any machine learning environment, including container services, Amazon SageMaker, and Amazon Elastic Compute Cloud.

AWS Sagemaker

Amazon A completely managed machine learning service is SageMaker. Data scientists and developers can quickly and effortlessly develop and train machine learning models with SageMaker, then instantly deploy them into a hosted environment that is ready for production. You don't need to manage servers because it has an integrated Jupyter writing notebook instance for quick access to your data sources for exploration and analysis. Additionally, it offers popular machine learning methods that have been enhanced for distributed environments and extremely big data sets. SageMaker provides adaptable distributed training alternatives that fit your unique workflows with native support for bring-your-own-algorithms and frameworks. Launch a model with a few clicks from SageMaker Studio or the SageMaker console to deploy it into a safe and scalable environment.

Cost Benefit Analysis

Serving Platform	Tier	Cost Control	Management
TorchServe	Free Tier	Open - source framework	Facebook
AWS SageMaker	Free Tier usage per month for the first 2 months	Based on Usage	AWS

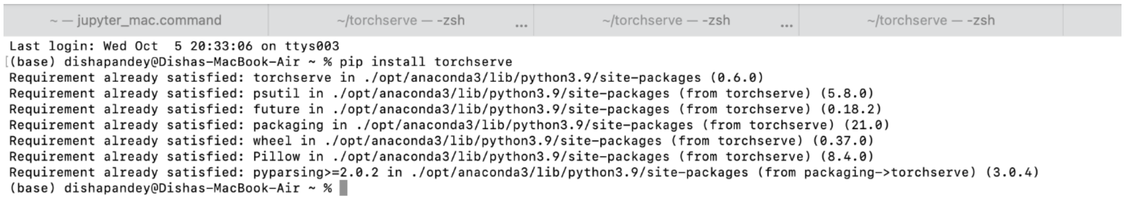
Implementation in the Environment - TorchServe

Requirements -

- JDK 11
- Torch 1.7

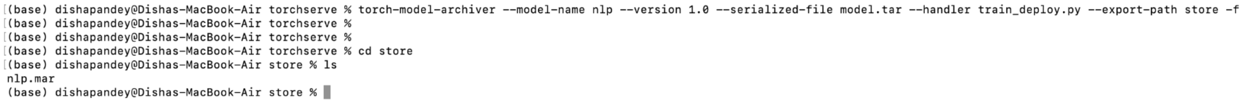
Steps -

1. We installed JDK 11 and necessary IDEs
2. Then we proceeded with the installation of necessary PyTorch Python packages required for both training and serving the model



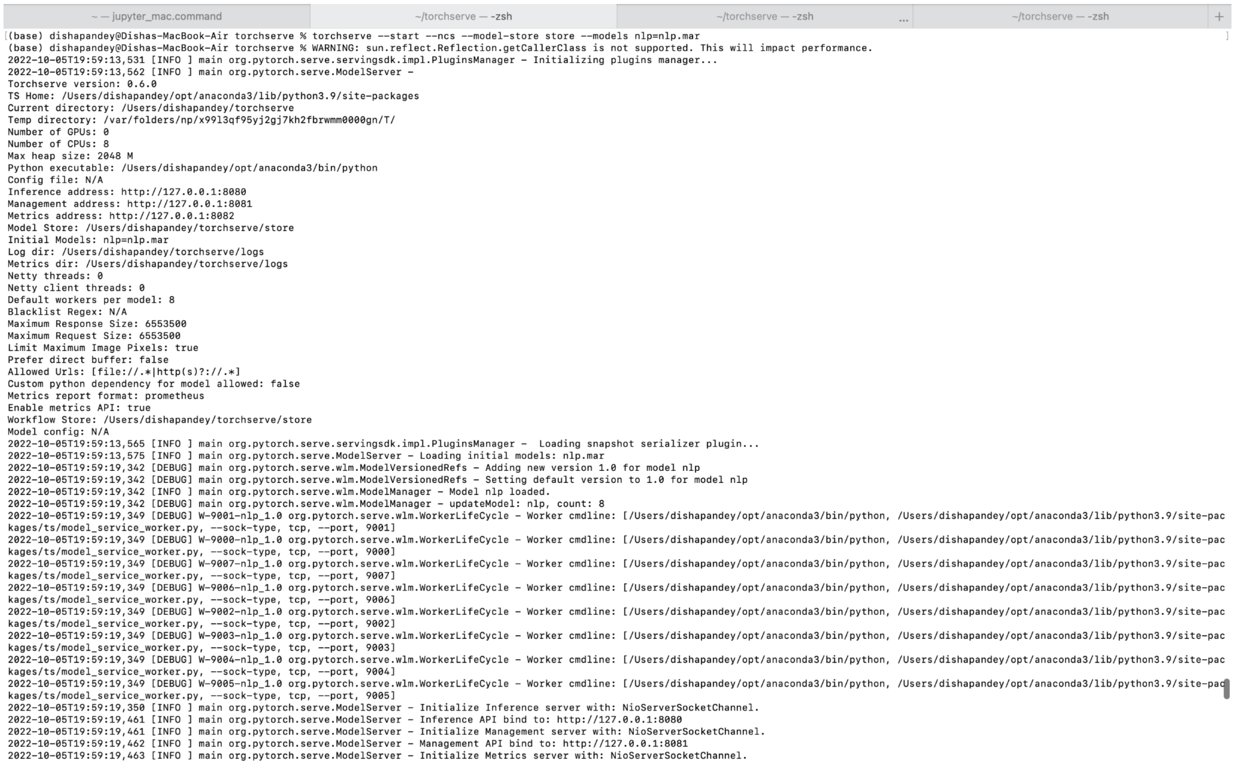
```
Last login: Wed Oct  5 20:33:06 on ttys003
(base) dishapandey@Dishas-MacBook-Air ~ % pip install torchserve
Requirement already satisfied: torchserve in ./opt/anaconda3/lib/python3.9/site-packages (0.6.0)
Requirement already satisfied: psutil in ./opt/anaconda3/lib/python3.9/site-packages (from torchserve) (5.8.0)
Requirement already satisfied: future in ./opt/anaconda3/lib/python3.9/site-packages (from torchserve) (0.18.2)
Requirement already satisfied: packaging in ./opt/anaconda3/lib/python3.9/site-packages (from torchserve) (21.0)
Requirement already satisfied: wheel in ./opt/anaconda3/lib/python3.9/site-packages (from torchserve) (0.37.0)
Requirement already satisfied: Pillow in ./opt/anaconda3/lib/python3.9/site-packages (from torchserve) (8.4.0)
Requirement already satisfied: pyparsing>=2.0.2 in ./opt/anaconda3/lib/python3.9/site-packages (from packaging->torchserve) (3.0.4)
(base) dishapandey@Dishas-MacBook-Air ~ %
```

3. TorchServe Archiver



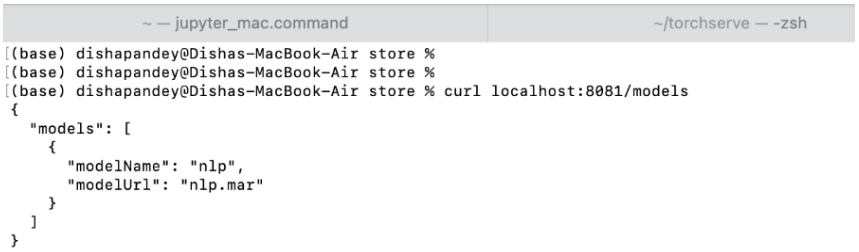
```
[(base) dishapandey@Dishas-MacBook-Air torchserve % torch-model-archiver --model-name nlp --version 1.0 --serialized-file model.tar --handler train_deploy.py --export-path store -f
[(base) dishapandey@Dishas-MacBook-Air torchserve %
[(base) dishapandey@Dishas-MacBook-Air torchserve %
[(base) dishapandey@Dishas-MacBook-Air torchserve % cd store
[(base) dishapandey@Dishas-MacBook-Air store % ls
nlp.mar
(base) dishapandey@Dishas-MacBook-Air store % 
```

4. Starting the Model



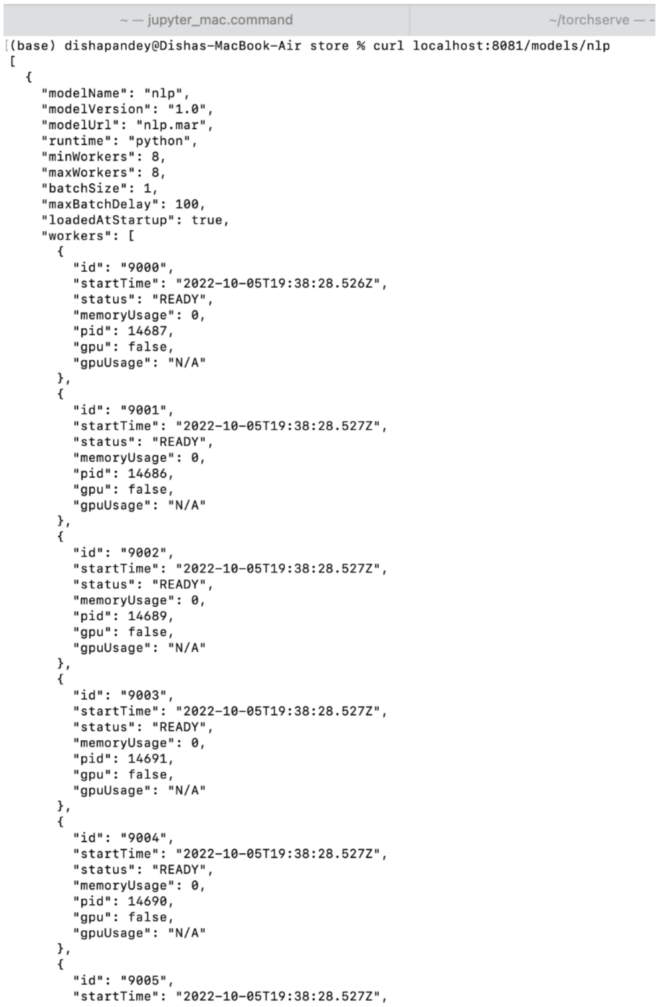
```
[(base) dishapandey@Dishas-MacBook-Air torchserve % torchserve --start --ncs --model-store store --models nlp=nlp.mar
(base) dishapandey@Dishas-MacBook-Air torchserve % WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
2022-10-05T19:59:13,531 [INFO ] main org.pytorch.serve.servingsdk.impl.PluginsManager - Initializing plugins manager...
2022-10-05T19:59:13,562 [INFO ] main org.pytorch.serve.ModelServer -
Torchserve version: 0.6.0
TS Home: /Users/dishapandey/opt/anaconda3/lib/python3.9/site-packages
Current directory: /Users/dishapandey/torchserve
Temp directory: /var/folders/np/x99l3qf95yj2gj7kh2fbrwmm0000gn/T/
Number of GPUs: 0
Number of CPUs: 8
Max heap size: 2048 M
Python executable: /Users/dishapandey/opt/anaconda3/bin/python
Config file: N/A
Inference address: http://127.0.0.1:8080
Management address: http://127.0.0.1:8081
Metrics address: http://127.0.0.1:8082
Model Store: /Users/dishapandey/torchserve/store
Initial Models: nlp=nlp.mar
Log dir: /Users/dishapandey/torchserve/logs
Metrics dir: /Users/dishapandey/torchserve/logs
Netty threads: 0
Netty client threads: 0
Default workers per model: 8
Blacklist Regex: N/A
Maximum Response Size: 6553500
Maximum Request Size: 6553500
Limit Maximum Image Pixels: true
Prefer direct buffer: false
Allowed Urls: [file://.*|http(s)?://.*]
Custom python dependency for model allowed: false
Metrics report format: prometheus
Enable metrics API: true
Workflow Store: /Users/dishapandey/torchserve/store
Model config: N/A
2022-10-05T19:59:13,565 [INFO ] main org.pytorch.serve.servingsdk.impl.PluginsManager -  Loading snapshot serializer plugin...
2022-10-05T19:59:13,575 [INFO ] main org.pytorch.serve.ModelServer - Loading initial models: nlp.mar
2022-10-05T19:59:19,342 [DEBUG] main org.pytorch.serve.wlm.ModelVersionedRefs - Adding new version 1.0 for model nlp
2022-10-05T19:59:19,342 [DEBUG] main org.pytorch.serve.wlm.ModelVersionedRefs - Setting default version to 1.0 for model nlp
2022-10-05T19:59:19,342 [INFO ] main org.pytorch.serve.wlm.ModelManager - Model nlp loaded.
2022-10-05T19:59:19,342 [DEBUG] main org.pytorch.serve.wlm.ModelManager - updateModel: nlp, count: 8
2022-10-05T19:59:19,349 [DEBUG] W-9001-nlp_1.0 org.pytorch.serve.wlm.WorkerLifeCycle - Worker cmdline: [/Users/dishapandey/opt/anaconda3/bin/python, /Users/dishapandey/opt/anaconda3/lib/python3.9/site-pac
kages/ts/model_service_worker.py, --sock-type, tcp, --port, 9001]
2022-10-05T19:59:19,349 [DEBUG] W-9000-nlp_1.0 org.pytorch.serve.wlm.WorkerLifeCycle - Worker cmdline: [/Users/dishapandey/opt/anaconda3/bin/python, /Users/dishapandey/opt/anaconda3/lib/python3.9/site-pac
kages/ts/model_service_worker.py, --sock-type, tcp, --port, 9000]
2022-10-05T19:59:19,349 [DEBUG] W-9007-nlp_1.0 org.pytorch.serve.wlm.WorkerLifeCycle - Worker cmdline: [/Users/dishapandey/opt/anaconda3/bin/python, /Users/dishapandey/opt/anaconda3/lib/python3.9/site-pac
kages/ts/model_service_worker.py, --sock-type, tcp, --port, 9007]
2022-10-05T19:59:19,349 [DEBUG] W-9006-nlp_1.0 org.pytorch.serve.wlm.WorkerLifeCycle - Worker cmdline: [/Users/dishapandey/opt/anaconda3/bin/python, /Users/dishapandey/opt/anaconda3/lib/python3.9/site-pac
kages/ts/model_service_worker.py, --sock-type, tcp, --port, 9006]
2022-10-05T19:59:19,349 [DEBUG] W-9002-nlp_1.0 org.pytorch.serve.wlm.WorkerLifeCycle - Worker cmdline: [/Users/dishapandey/opt/anaconda3/bin/python, /Users/dishapandey/opt/anaconda3/lib/python3.9/site-pac
kages/ts/model_service_worker.py, --sock-type, tcp, --port, 9002]
2022-10-05T19:59:19,349 [DEBUG] W-9003-nlp_1.0 org.pytorch.serve.wlm.WorkerLifeCycle - Worker cmdline: [/Users/dishapandey/opt/anaconda3/bin/python, /Users/dishapandey/opt/anaconda3/lib/python3.9/site-pac
kages/ts/model_service_worker.py, --sock-type, tcp, --port, 9003]
2022-10-05T19:59:19,349 [DEBUG] W-9004-nlp_1.0 org.pytorch.serve.wlm.WorkerLifeCycle - Worker cmdline: [/Users/dishapandey/opt/anaconda3/bin/python, /Users/dishapandey/opt/anaconda3/lib/python3.9/site-pac
kages/ts/model_service_worker.py, --sock-type, tcp, --port, 9004]
2022-10-05T19:59:19,349 [DEBUG] W-9005-nlp_1.0 org.pytorch.serve.wlm.WorkerLifeCycle - Worker cmdline: [/Users/dishapandey/opt/anaconda3/bin/python, /Users/dishapandey/opt/anaconda3/lib/python3.9/site-pac
kages/ts/model_service_worker.py, --sock-type, tcp, --port, 9005]
2022-10-05T19:59:19,350 [INFO ] main org.pytorch.serve.ModelServer - Initialize Inference server with: NioServerSocketChannel.
2022-10-05T19:59:19,461 [INFO ] main org.pytorch.serve.ModelServer - Inference API bind to: http://127.0.0.1:8080
2022-10-05T19:59:19,461 [INFO ] main org.pytorch.serve.ModelServer - Initialize Management server with: NioServerSocketChannel.
2022-10-05T19:59:19,462 [INFO ] main org.pytorch.serve.ModelServer - Management API bind to: http://127.0.0.1:8081
2022-10-05T19:59:19,463 [INFO ] main org.pytorch.serve.ModelServer - Initialize Metrics server with: NioServerSocketChannel.
```

5. Deployment



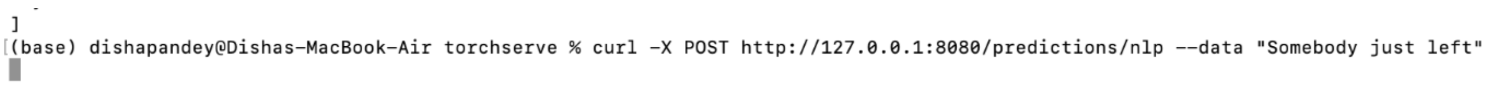
```
[(base) dishapandey@Dishas-MacBook-Air store %
[(base) dishapandey@Dishas-MacBook-Air store %
[(base) dishapandey@Dishas-MacBook-Air store % curl localhost:8081/models
{
  "models": [
    {
      "modelName": "nlp",
      "modelUrl": "nlp.mar"
    }
  ]
}
```

6. No. of Workers



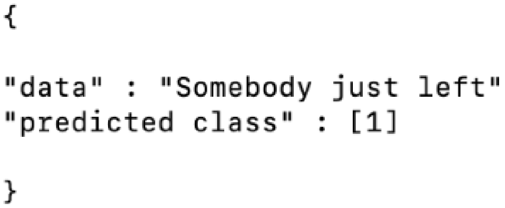
```
[(base) dishapandey@Dishas-MacBook-Air store % curl localhost:8081/models/nlp
[
  {
    "modelName": "nlp",
    "modelVersion": "1.0",
    "modelUrl": "nlp.mar",
    "runtime": "python",
    "minWorkers": 8,
    "maxWorkers": 8,
    "batchSize": 1,
    "maxBatchDelay": 100,
    "loadedAtStartup": true,
    "workers": [
      {
        "id": "9000",
        "startTime": "2022-10-05T19:38:28.526Z",
        "status": "READY",
        "memoryUsage": 0,
        "pid": 14687,
        "gpu": false,
        "gpuUsage": "N/A"
      },
      {
        "id": "9001",
        "startTime": "2022-10-05T19:38:28.527Z",
        "status": "READY",
        "memoryUsage": 0,
        "pid": 14686,
        "gpu": false,
        "gpuUsage": "N/A"
      },
      {
        "id": "9002",
        "startTime": "2022-10-05T19:38:28.527Z",
        "status": "READY",
        "memoryUsage": 0,
        "pid": 14689,
        "gpu": false,
        "gpuUsage": "N/A"
      },
      {
        "id": "9003",
        "startTime": "2022-10-05T19:38:28.527Z",
        "status": "READY",
        "memoryUsage": 0,
        "pid": 14691,
        "gpu": false,
        "gpuUsage": "N/A"
      },
      {
        "id": "9004",
        "startTime": "2022-10-05T19:38:28.527Z",
        "status": "READY",
        "memoryUsage": 0,
        "pid": 14690,
        "gpu": false,
        "gpuUsage": "N/A"
      },
      {
        "id": "9005",
        "startTime": "2022-10-05T19:38:28.527Z",
```

7. Curling Prediction Query



```
]
[(base) dishapandey@Dishas-MacBook-Air torchserve % curl -X POST http://127.0.0.1:8080/predictions/nlp --data "Somebody just left"
```

8. Predicted Output



```
{

"data" : "Somebody just left"
"predicted class" : [1]

}
```

Documenting the experience

While starting our project, our initial aim was to learn the mechanism of deploying Machine Learning models to run our project efficiently in a production environment. Through this, we could comprehend the real-world implementation of how developers deploy their codes/applications or updates.

Initially, we started with an MBTI personality prediction ML model. We were able to import PyTorch libraries successfully and were able to serve them in production with the Amazon SageMaker native TorchServe integration. However, after connecting with the faculty we realized that the main purpose of our project was to deploy the PyTorch model using TorchServe. We noticed that our MBTI personality prediction model was based on Python libraries and not PyTorch. Also, we realized that PyTorch does not have support for the XGBoost model. Hence, we selected another project 'Grammar Categorizer' which supported PyTorch and carried out the same steps to deploy it in AWS SageMaker. Finally, we learned the methodologies to implement our project using TorchServe and were able to deploy it successfully. Deploying it directly on a default platform and TorchServe gave us a comparative analysis of both. The entire process was a collaborative effort of the team for understanding all the concepts we learned efficiently. of course! Google helped a lot with all the blockers we faced.

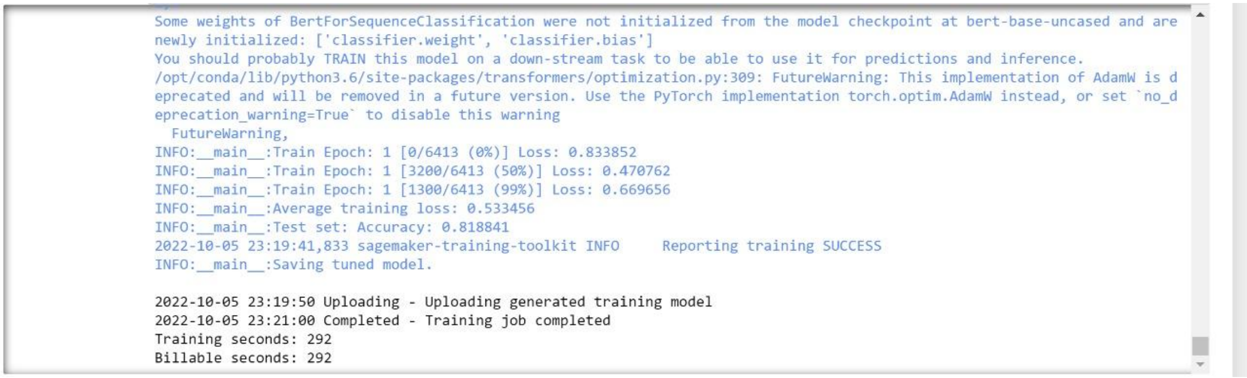
Lessons Learned/Issues Faced

1. We faced a JAVA compatibility issue while trying to deploy the model on TorchServe, finally, we had to upgrade it from JDK 8 to JDK 11
2. As there were computational limitations on our AWS free account subscription, we had to spend more time training and deploying the model on the AWS SageMaker
3. We had to upgrade our TorchServe library version from 1.3 to 1.7, as the BertForSequenceClassification classifier which had AdamW, BertForSequenceClassification, and BertTokenizer, was not compatible with TorchServe version 1.3

Code Artifacts

Below are the snippets from our Jupyter Notebook, of our 'Grammar Categorizer' using the PyTorch model and serving it in production with the Amazon SageMaker native TorchServe integration

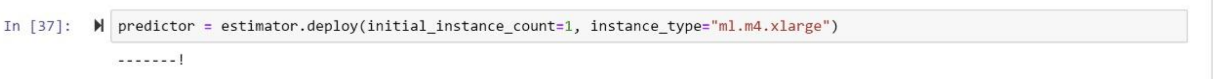
1. Pytorch model trained and saved in the AWS S3 Bucket



```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are
newly initialized: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/opt/conda/lib/python3.6/site-packages/transformers/optimization.py:309: FutureWarning: This implementation of AdamW is d
eprecated and will be removed in a future version. Use the PyTorch implementation torch.optim.AdamW instead, or set `no_d
eprecation_warning=True` to disable this warning
  FutureWarning,
INFO:__main__:Train Epoch: 1 [0/6413 (0%)] Loss: 0.833852
INFO:__main__:Train Epoch: 1 [3200/6413 (50%)] Loss: 0.470762
INFO:__main__:Train Epoch: 1 [1300/6413 (99%)] Loss: 0.669656
INFO:__main__:Average training loss: 0.533456
INFO:__main__:Test set: Accuracy: 0.818841
2022-10-05 23:19:41,833 sagemaker-training-toolkit INFO     Reporting training SUCCESS
INFO:__main__:Saving tuned model.

2022-10-05 23:19:50 Uploading - Uploading generated training model
2022-10-05 23:21:00 Completed - Training job completed
Training seconds: 292
Billable seconds: 292
```

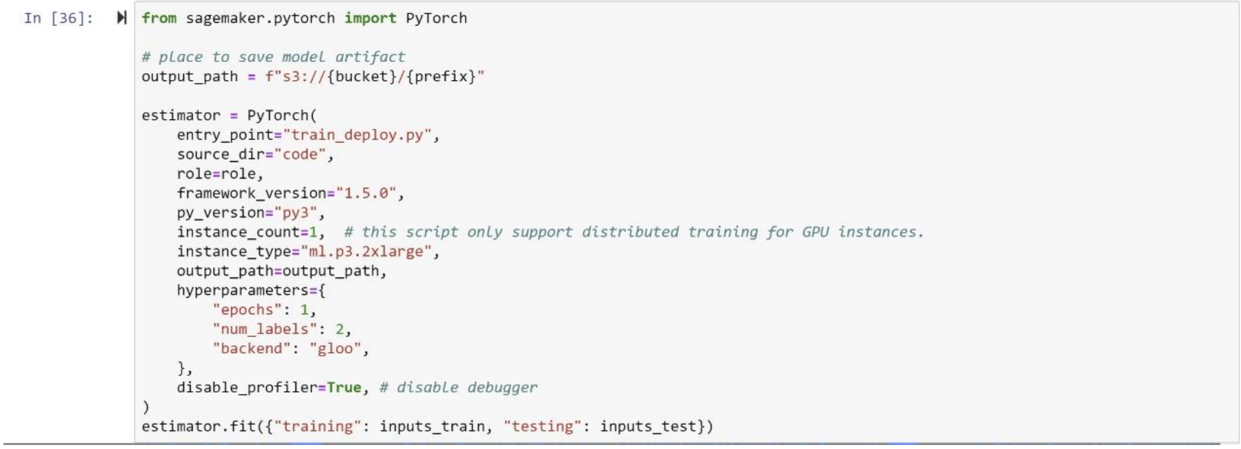
2. The PyTorch model successfully deployed



```
In [37]:  predictor = estimator.deploy(initial_instance_count=1, instance_type="ml.m4.xlarge")

          -------!
```

3. Utilizing the deployed model

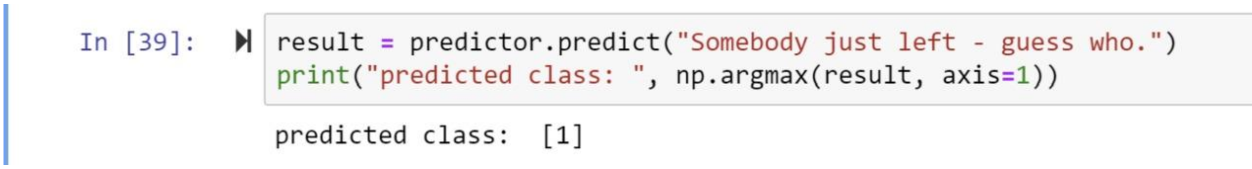


```
In [36]:  from sagemaker.pytorch import PyTorch

          # place to save model artifact
          output_path = f"s3://{bucket}/{prefix}"

          estimator = PyTorch(
              entry_point="train_deploy.py",
              source_dir="code",
              role=role,
              framework_version="1.5.0",
              py_version="py3",
              instance_count=1,  # this script only support distributed training for GPU instances.
              instance_type="ml.p3.2xlarge",
              output_path=output_path,
              hyperparameters={
                  "epochs": 1,
                  "num_labels": 2,
                  "backend": "gloo",
              },
              disable_profiler=True, # disable debugger
          )
          estimator.fit({"training": inputs_train, "testing": inputs_test})
```

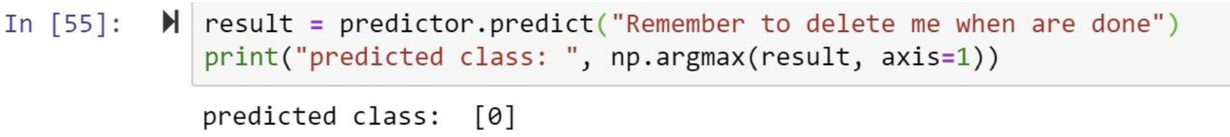
4. Testing our PyTorch model with sample output for a positive outcome (we have given our model a grammatically correct statement, hence it has passed the predicted class as 1)



```
In [39]:  result = predictor.predict("Somebody just left - guess who.")
          print("predicted class: ", np.argmax(result, axis=1))

          predicted class:  [1]
```

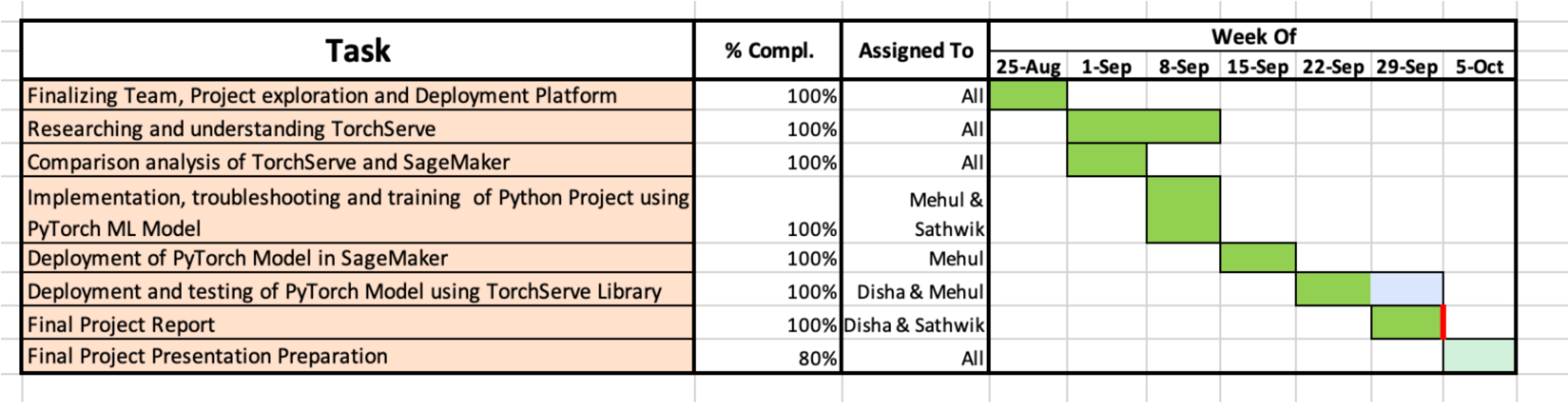
5. Testing our PyTorch model with sample output for a negative outcome (we have given our model a grammatically incorrect statement, hence it has passed the predicted class as 0)



```
In [55]:  result = predictor.predict("Remember to delete me when are done")
          print("predicted class: ", np.argmax(result, axis=1))

          predicted class:  [0]
```

Project Plan



Task	% Compl.	Assigned To	Week Of						
			25-Aug	1-Sep	8-Sep	15-Sep	22-Sep	29-Sep	5-Oct
Finalizing Team, Project exploration and Deployment Platform	100%	All	🟩						
Researching and understanding TorchServe	100%	All		🟩	🟩				
Comparison analysis of TorchServe and SageMaker	100%	All		🟩					
Implementation, troubleshooting and training of Python Project using PyTorch ML Model	100%	Mehul & Sathwik			🟩				
Deployment of PyTorch Model in SageMaker	100%	Mehul				🟩			
Deployment and testing of PyTorch Model using TorchServe Library	100%	Disha & Mehul					🟩	🟦	
Final Project Report	100%	Disha & Sathwik						🟩|	
Final Project Presentation Preparation	80%	All							▨

🟩 Completed

▨ In Progress

🟦 Buffer Period

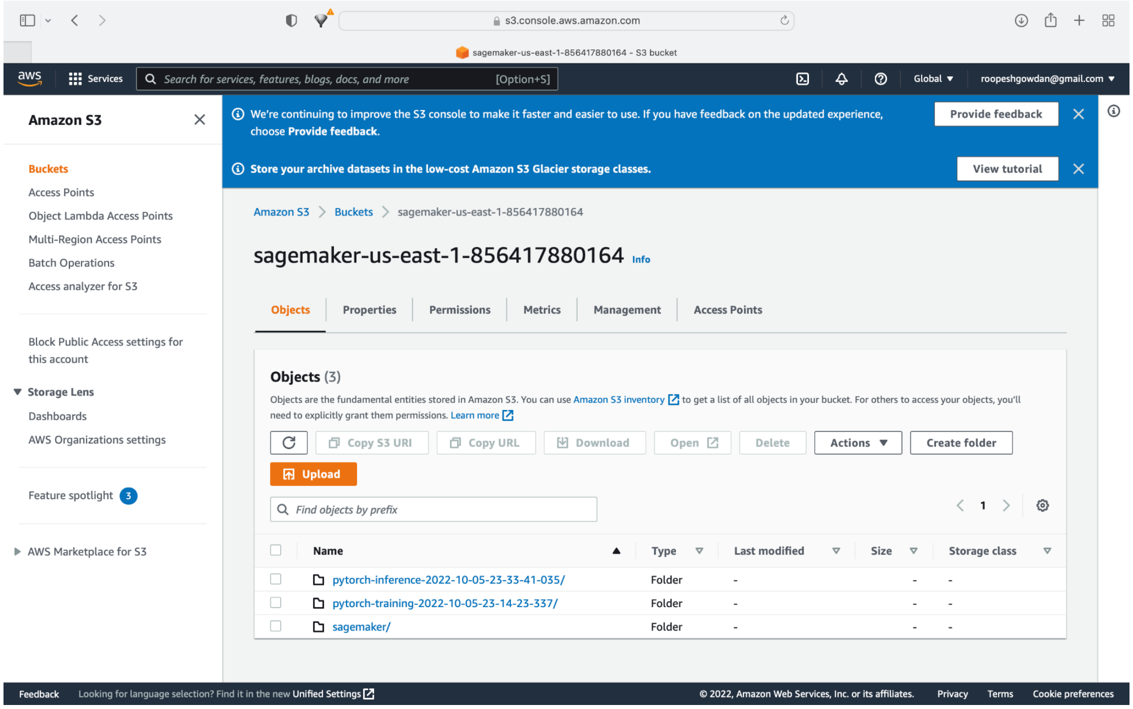
| Final Submission Milestone

References

1) https://aws.amazon.com/sagemaker/resources/?ar-cards-sagemaker.sort-by=item.additionalFields.datePublished&ar-cards-sagemaker.sort-order=desc
2) https://github.com/aws-samples/amazon-sagemaker-bert-pytorch/blob/master/bert-sm-python-SDK.ipynb
3) https://aws.amazon.com/blogs/machine-learning/serving-pytorch-models-in-production-with-the-amazon-sagemaker-native-torchserve-integration/
4) https://aws.amazon.com/blogs/aws/announcing-torchserve-an-open-source-model-server-for-pytorch/
5) https://pytorch.org/serve/
6) https://towardsdatascience.com/serving-pytorch-models-with-torchserve-6b8e8cbdb632
7) https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/

Appendix

AWS SageMaker and AWS S3 buckets implementation snippets



We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose **Provide feedback**.

Store your archive datasets in the low-cost Amazon S3 Glacier storage classes.

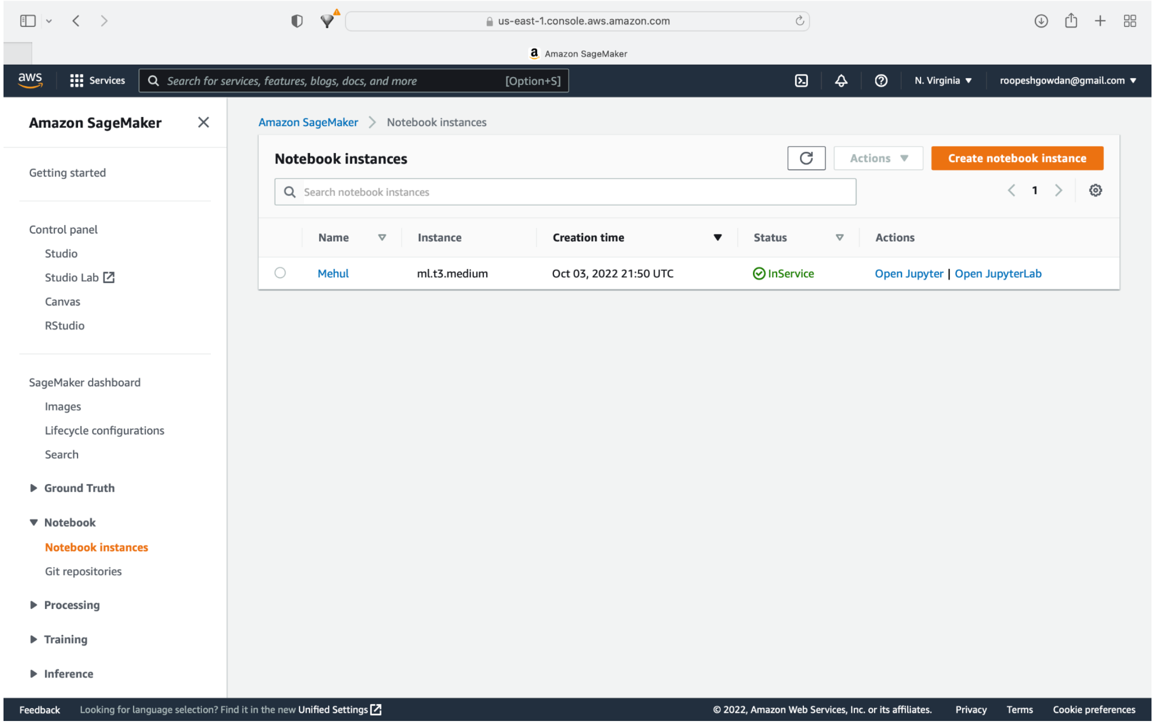
Amazon S3 > Buckets > sagemaker-us-east-1-856417880164

sagemaker-us-east-1-856417880164 Info

Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

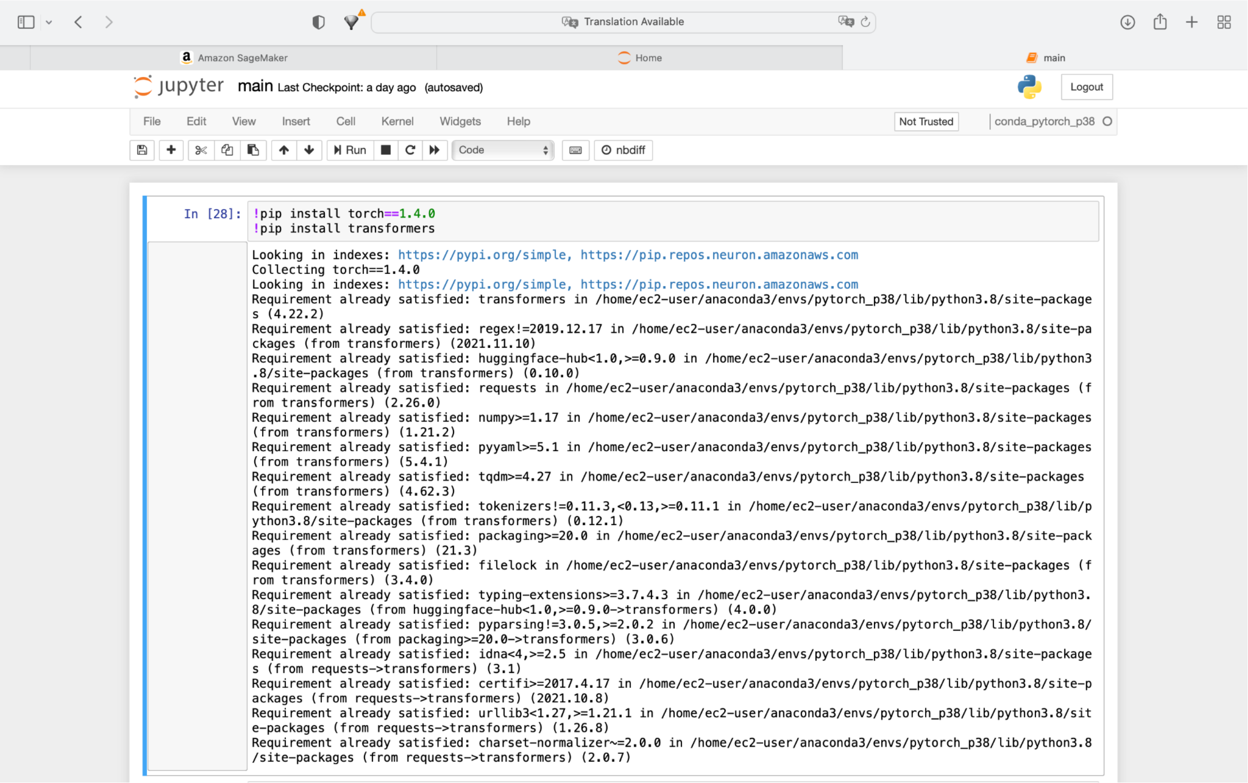
	Name	Type	Last modified	Size	Storage class
□	pytorch-inference-2022-10-05-23-33-41-035/	Folder	-	-	-
□	pytorch-training-2022-10-05-23-14-23-337/	Folder	-	-	-
□	sagemaker/	Folder	-	-	-



Amazon SageMaker > Notebook instances

Notebook instances

	Name	Instance	Creation time	Status	Actions
○	Mehul	ml.t3.medium	Oct 03, 2022 21:50 UTC	✓ InService	Open Jupyter | Open JupyterLab



```
!pip install torch==1.4.0
!pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Collecting torch==1.4.0
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Requirement already satisfied: transformers in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-package
s (4.22.2)
Requirement already satisfied: regex!=2019.12.17 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-pa
ckages (from transformers) (2021.11.10)
Requirement already satisfied: huggingface-hub<1.0,>=0.9.0 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3
.8/site-packages (from transformers) (0.10.0)
Requirement already satisfied: requests in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (f
rom transformers) (2.26.0)
Requirement already satisfied: numpy>=1.17 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages
(from transformers) (1.21.2)
Requirement already satisfied: pyyaml>=5.1 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages
(from transformers) (5.4.1)
Requirement already satisfied: tqdm>=4.27 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages
(from transformers) (4.62.3)
Requirement already satisfied: tokenizers!=0.11.3,<0.13,>=0.11.1 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/p
ython3.8/site-packages (from transformers) (0.12.1)
Requirement already satisfied: packaging>=20.0 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-pack
ages (from transformers) (21.3)
Requirement already satisfied: filelock in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-packages (f
rom transformers) (3.4.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.
8/site-packages (from huggingface-hub<1.0,>=0.9.0->transformers) (4.0.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/
site-packages (from packaging>=20.0->transformers) (3.0.6)
Requirement already satisfied: idna<4,>=2.5 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-package
s (from requests->transformers) (3.1)
Requirement already satisfied: certifi>=2017.4.17 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/site-p
ackages (from requests->transformers) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8/sit
e-packages (from requests->transformers) (1.26.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in /home/ec2-user/anaconda3/envs/pytorch_p38/lib/python3.8
/site-packages (from requests->transformers) (2.0.7)
```