Deployment Document: Continuous Training and Deployment Pipeline for Transformer-based Text Classification Model

This guide provides step-by-step instructions for deploying the continuous training and deployment pipeline for the transformer-based text classification model both locally and on AWS.

---

1. Local Deployment Steps

A. Prerequisites

- **Python Environment**: Ensure Python 3.7+ is installed with `pip`.

Libraries: Install required libraries with:

```
pip install transformers pandas scikit-learn flask-ngrok mlflow torch
```

MLflow Setup: Use a local directory for MLflow tracking by setting:

```
mlflow.set_tracking_uri("file:///path/to/mlruns")
```

B. Model Training and Retraining

1. **Run Initial Training**:
    - Load your dataset and initialize the tokenizer.
    - Call `train_model_with_mlflow()` to train the model. Metrics and model artifacts will be logged in MLflow.
2. **Monitoring and Retraining**:
    - The `monitor_and_retrain_with_mlflow()` function will compare current accuracy against the threshold.
    - If accuracy drops, the function retrains the model and sends an email alert using a configured Gmail account.
3. **Inference Setup**:
    - **Live Inference**: Start the Flask API server to handle real-time predictions.
    - **Batch Inference**: Use a batch inference script that loads data, processes it, and outputs predictions.

C. Running the Flask API for Live Inference

1. **In your terminal, start the Flask server:**

   ```
   python live_inference.py
   ```
2. Test the live endpoint by sending requests to the provided URL.

D. Email Alerts

1. Configure Gmail SMTP settings in the `send_alert_email()` function for alert notifications on retraining.

---

2. AWS Deployment Steps

A. Prerequisites

1. **AWS CLI**: Install and configure with your AWS credentials.

   ```
   aws configure
   ```
2. **IAM Roles**: Ensure you have the appropriate roles for SageMaker and Lambda.

B. Model Training in SageMaker

1. **Data Upload to S3**:

Upload your dataset to S3:

```
s3.upload_file("your-dataset.csv", "your-bucket-name",
"dataset-folder/your-dataset.csv")
```

2. **Run Training Job on SageMaker**:
   - Use the `train.py` script to start the training job.
   - Configure `PyTorch` estimator in SageMaker to specify training parameters, including the dataset location and model output path.

C. Deploying for Inference

1. **Live Inference**:

Deploy a SageMaker endpoint for live predictions with:

```
predictor = estimator.deploy(initial_instance_count=1,
instance_type='ml.m5.large')
```

-
2. **Batch Inference**:

Use SageMaker’s batch transform function to run batch jobs:

```
transformer = estimator.transformer(instance_count=1,
instance_type="ml.m5.large",
output_path="s3://your-bucket-name/batch-output")
transformer.transform(data="s3://your-bucket-name/dataset-folder/your-
dataset.csv", content_type='text/csv', split_type='Line')
```

-

D. Setting Up Monitoring and Retraining

1. **CloudWatch Alarms**:
   - Set up CloudWatch to monitor the model accuracy logged to CloudWatch Metrics.
   - Define a threshold for accuracy to trigger an alert if the model underperforms.
2. **Lambda and SNS for Retraining**:
   - Configure an AWS Lambda function that checks model accuracy and initiates a retraining job if the threshold is not met.
   - Use AWS SNS to send notifications when retraining is triggered.

E. Clean-Up and Cost Management

1. **Data Lifecycle Policies**: Configure S3 to clean up old model versions and dataset backups.
2. **Instance Management**: Use auto-scaling or job-specific instances to control costs.

This deployment setup enables a scalable, efficient, and cost-effective pipeline for continuous training and deployment, ensuring the model remains accurate and responsive to new data patterns.