AutoML Implementation on BERT Classifier

—

- Debdeep Ghosal

Roadmap

- Dataset Collection
- Data Preprocessing
- Initial Model Training
- Model deployment on Sagemaker Endpoint
- Set up Model inference API
- AutoML Implementation

Dataset Collection

- The Dataset has been collected from http://qwone.com/~jason/20Newsgroups/

- It contains 18846 documents of 20 Newsgroups with multiple labels (eg. baseball,comp,crypt,talk,religion etc.)

Data Preprocessing

- The dataset is loaded in the Kaggle Environment.
- The dataset which is present in a directory structure corresponding to the classes is inserted into a pandas dataframe.
- After that the dataset is randomly shuffled and then categorical encoding is applied on the column.
- After that categories with low frequency and empty rows are removed.
- Now the Dataframe is ready and is exported to be trained on.

Initial Model Training

The initial model is trained on Free Kaggle GPU. The following steps describe it :

- The required libraries and the dataset is loaded.
- The dataset is divided into train,test and validation split , then the encoder is loaded.
- A custom dataset class is created which serves the tokenized datapoints, and then the train,test and validation dataset instance is created with the new class .
- Then the train, test and validation dataset instance are used to create the respective dataloaders.

cont..

- After that the <BERTClass> is created and instantiated which loads the bert-base-uncased model.
- In the next step the loss function,optimizer and the hyperparameters are defined.
- Then a training and evaluation function is created with the given hyperparameters.
- Then the training function and evaluation function is run <epoch> times and upon the improvements of the model the the new model weights are saved .

Model Deployment on Sagemaker Endpoint

- In this step a sagemaker notebook instance is created , then the weights of the previously trained model is uploaded to s3 bucket.

- After that an inference script is created where functions are defined on how the endpoint will load the model, how the input will be processed, how the prediction will be made and how the output is sent.

- After that a <PyTorchModel> instance is created and the inference script along with the model weight(s3 URI) is passed to it.

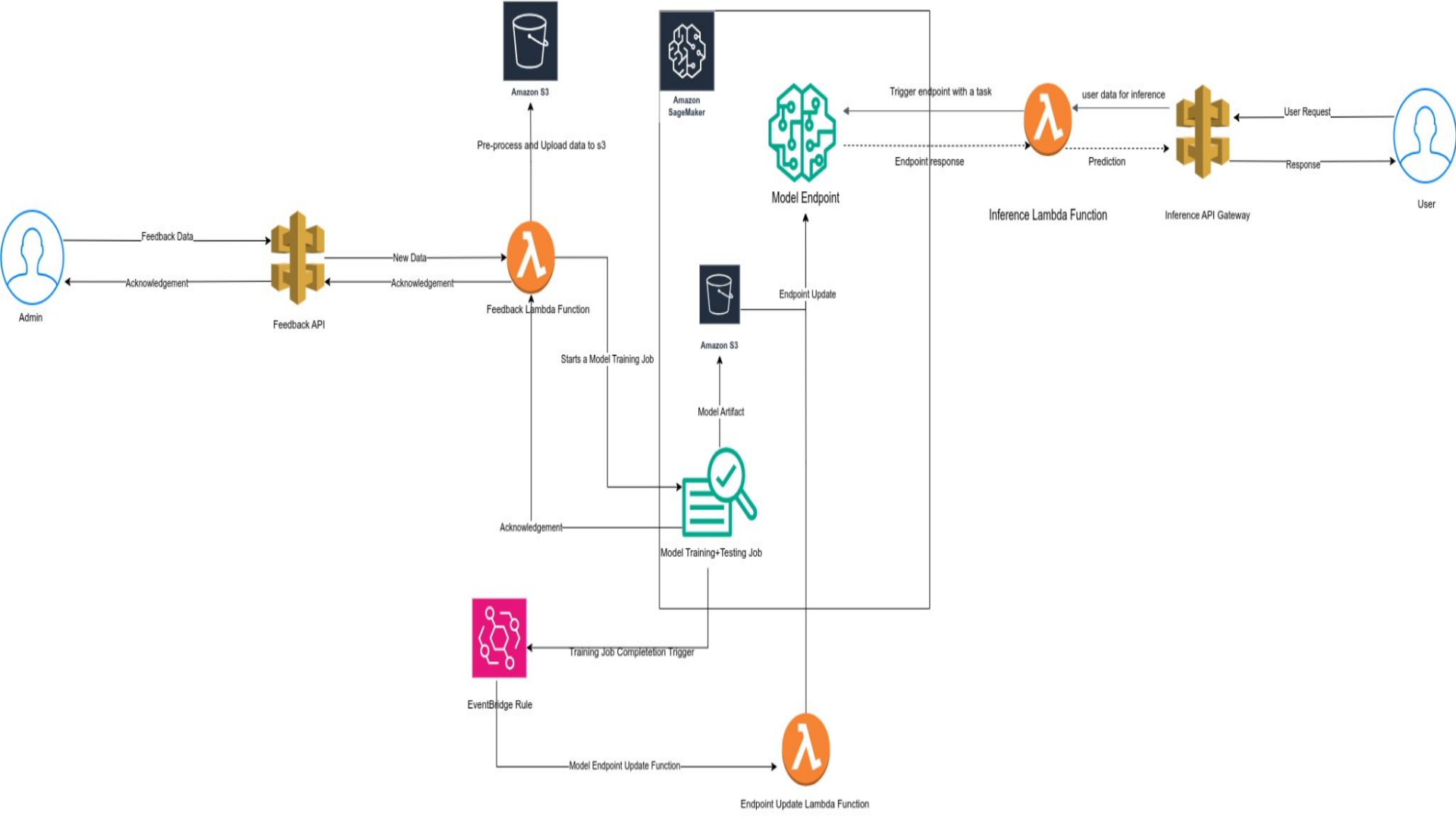
- Then this model is deployed choosing a instance type of our choice .

Set up Model Inference API

- In this step a Lambda function is created with necessary permission that can invoke a Sagemaker Enpoint and get the prediction.

- After that this Lambda function is attached with an API Gateway which can receive data from user in form of json and respond with the prediction it got from the lambda function.

AutoML Implementation


Amazon S3
Amazon SageMaker
Trigger endpoint with a task
user data for inference
User Request
Pre-process and Upload data to s3
Endpoint response
Prediction
Response
Model Endpoint
Inference Lambda Function
Inference API Gateway
User
Feedback Data
New Data
Acknowledgement
Acknowledgement
Acknowledgement
Admin
Feedback API
Feedback Lambda Function
Endpoint Update
Amazon S3
Starts a Model Training Job
Model Artifact
Acknowledgement
Model Training+Testing Job
Training Job Completetion Trigger
EventBridge Rule
Model Endpoint Update Function
Endpoint Update Lambda Function

- For AutoML implementation we have created a lambda function and attached it to an API Gateway.

- The API receives the feedback data and the passes it to the lambda function, inside the lambda function we perform the following two operations:
  - The received feedback data is cleaned and preprocessed and then merged with the previous dataset and saved in the s3 bucket.
  - Then a new training job is started with the updated dataset, and then the lambda function closes after sending the response to the API gateway.

- We have also set up an EventBridge rule which gets triggered when the training job is finished , then it calls a lambda function which updates the current endpoint with a new endpoint with the new weights.

- We have kept the endpoint name as same to maintain consistency.

Future Scope and Improvements:

- Hyperparameter Tuning (Not implemented due to large compute need)
- Using secondary Endpoint to ensure ~100% uptime
- Setting Event to train model when the traffic is less using cloudwatch and EventBridge.

Author’s Note

- Each Lambda function needs specific policies attached to its role to perform actions including other AWS services.(e.g. IAM,Sagemaker,s3)

- The lambda function associated with training the model also needs to pass the training script. The script is in my github repo .

- The name of the scheduled training job is saved in s3 and further used by the bert-update lambda function while deploying endpoint.

Thank You!