REALTIME AND BATCH PREDICTION SERVICE

The first 2 pages contain the project proposal, and the following pages go into the implementation details.

Abstract: The following 2 goals were implemented.

GOAL #1: Batch prediction. An automated workflow to generate batch predictions using S3, Lambda and EC2.

GOAL #2: Realtime prediction using API gateway, Lambda and EC2.

Prediction refers to generating summarized text (using ML model) for a given corpus/body of text. For this project we have used the Amazon Food Reviews dataset (https://www.kaggle.com/snap/amazon-fine-food-reviews, and we will generating summarized reviews of this dataset. For example: If the review is 10 sentences long, the summarized review will be about 3 sentences long. The ML model will pick the key sentences. An example is shown below.

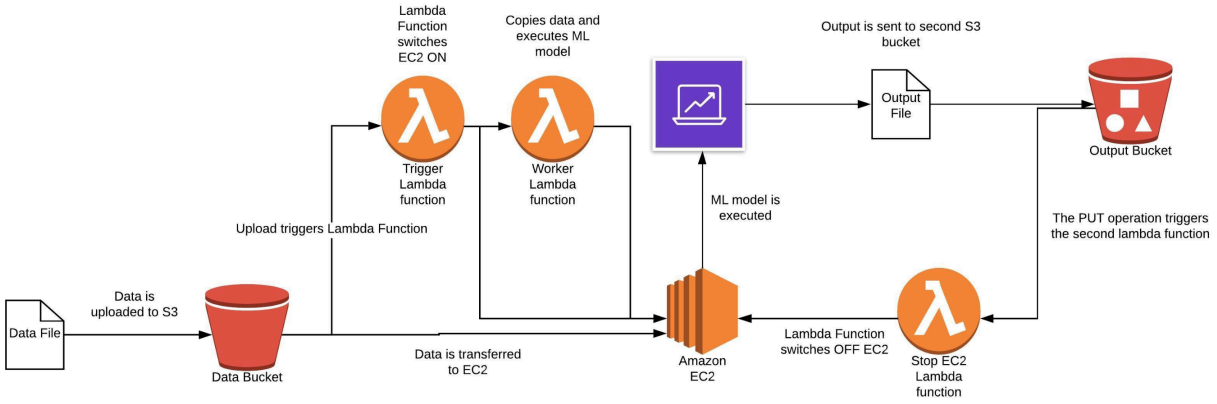
Amazon Food Review

This taffy is so good. It is very soft and chewy. The flavors are amazing. I would definitely recommend you buying it. Very satisfying!!

Summarized Review Output

"This taffy is so good. The flavors are amazing"

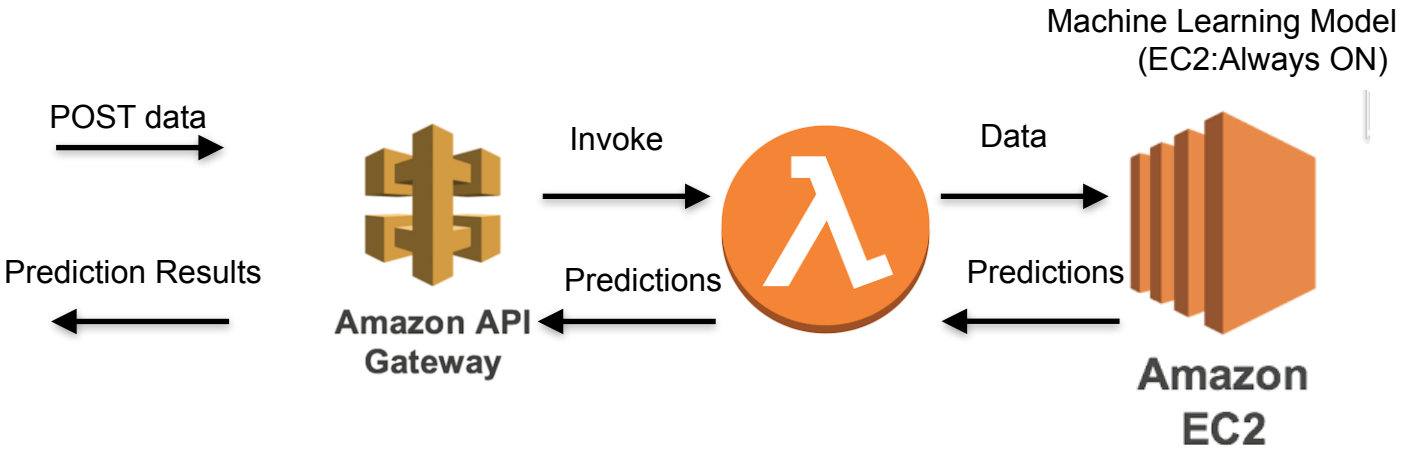
GOAL #1("Compute as Glue"): Visualization of automatic workflow. The below diagram has been borrowed from https://towardsdatascience.com/automating-machine-learning-models-on-aws-bfa183fe4065



Steps:

1) Data posted to input S3 will trigger a Lambda function (“Trigger Lambda”) which will turn on the EC2 instance.
2) “Trigger Lambda” will invoke “Worker Lambda” that copies the data and code from S3 to EC2 instance, and directs the EC2 instance to execute the text summarizer ML model.
3) “EC2” instance executes the ML model and writes the model to output S3.
4) The output S3 triggers a Lambda function (“Stop Ec2 Lambda”) to stop the EC2 instance.

Part 2 (“Compute as Backend”): We will use the following building blocks to achieve the 2nd goal. This use case also enables the end users easy access to text summarizer service via the API Gateway.

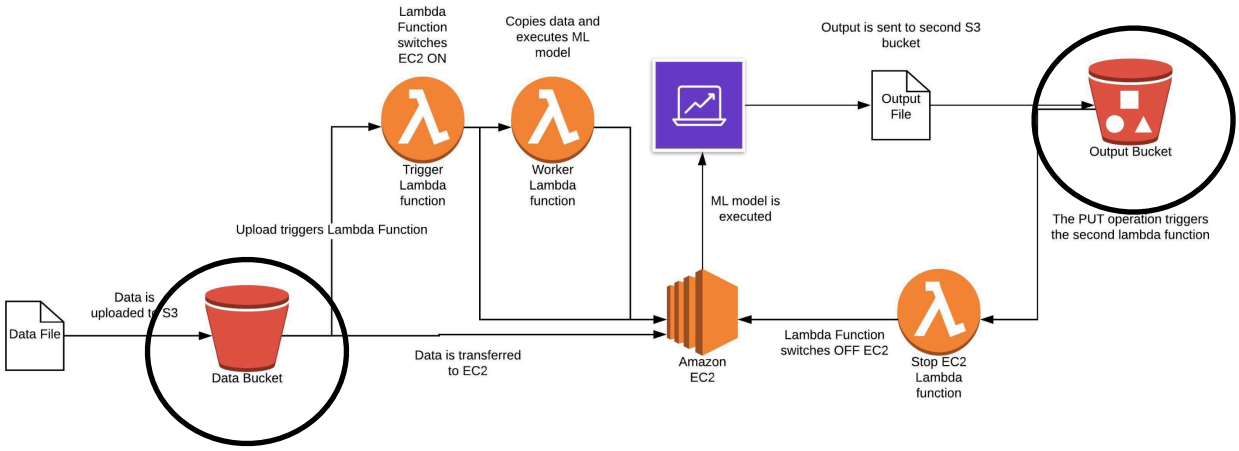


Steps:

1. The API gateway invokes a lambda function when a “POST” request is sent on the configured resource.
2. The lambda function orchestrates the downloading of resources needed for the text summarizer model, executes the model, and collects the results, and passes the summarized text back to the api gateway.

GOAL #1: IMPLEMENTATION DETAILS

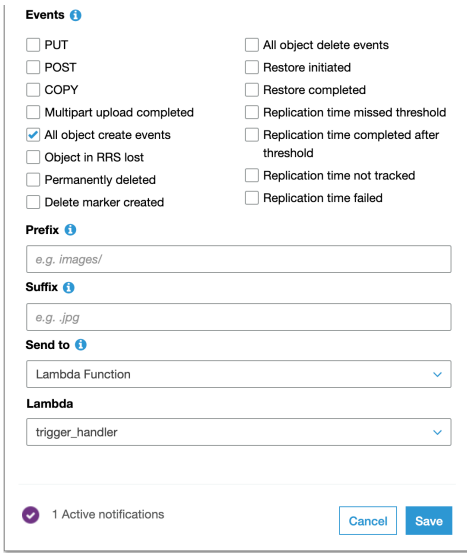
Generating batches of text summary

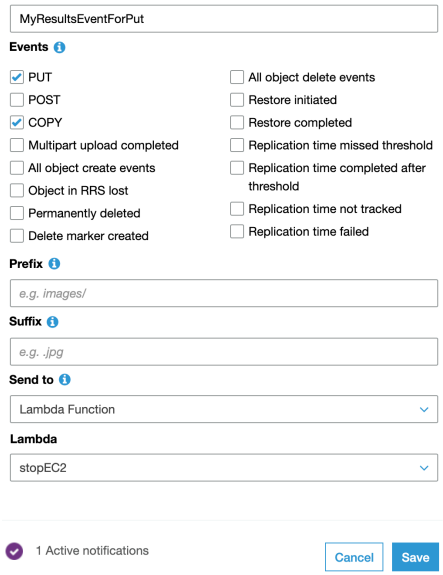


S3 bucket: Two S3 buckets were implemented namely cscie90-finalproject-inputdata and cscie90-finalproject-results. The S3 bucket cscie90-finalproject-inputdata was used to source the dataset, and the second S3 bucket cscie90-finalproject-results was used to write the summarized text result outputs.

Amazon Food Reviews file (https://www.kaggle.com/snap/amazon-fine-food-reviews) was uploaded to S3 bucket cscie90-finalproject-inputdata. This bucket was configured to trigger the lambda function on copy/ put object event.

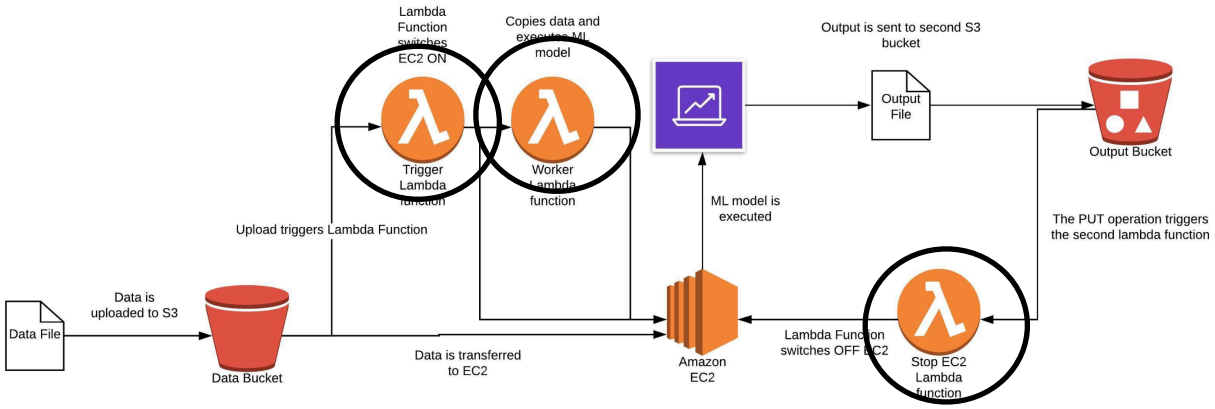
Under cscie90-finalproject-inputdata properties, events were configured as below to call **"trigger_handler"** lambda function when any object was written/uploaded to this bucket.





Similarly the bucket cscie90-finalproject-results was configured to call **“stopEC2”** lambda function once results were posted/copied to this bucket.

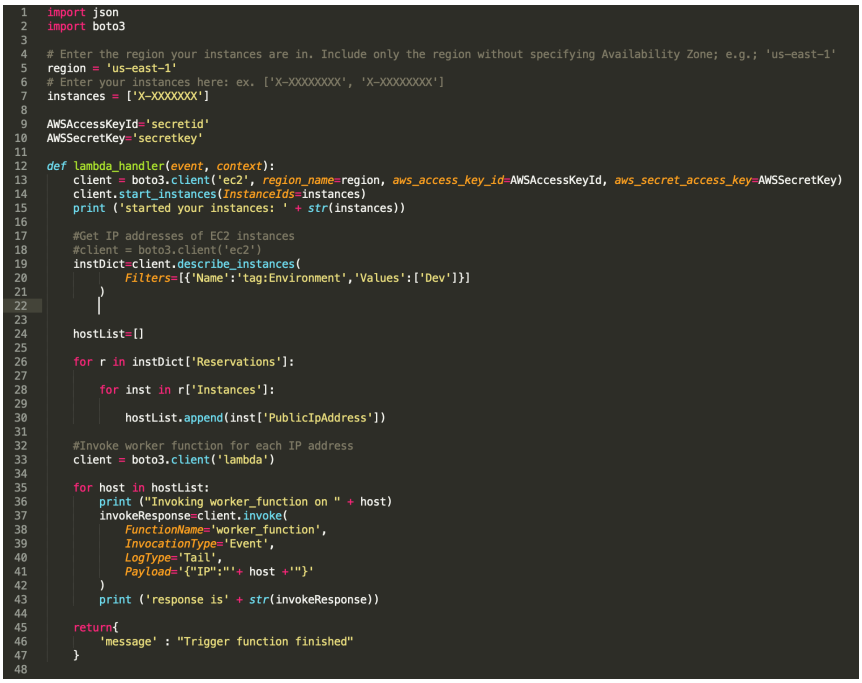
Lambda Functions



Trigger_Handler Lambda:

When dataset is uploaded to the input s3 bucket, the lambda function trigger_handler is invoked.

This handler starts an EC2 instance, and invokes the lambda function worker_function. Please note a policy was attached to this lambda function to allow access to EC2, and allow invocation of another lambda function. The source code and policy are shown in the below screenshots.



```
import json
import boto3

# Enter the region your instances are in. Include only the region without specifying Availability Zone; e.g.; 'us-east-1'
region = 'us-east-1'
# Enter your instances here: ex. ['X-XXXXXXXX', 'X-XXXXXXXX']
instances = ['X-XXXXXXX']

AWSAccessKeyId='secretid'
AWSSecretKey='secretkey'

def lambda_handler(event, context):
    client = boto3.client('ec2', region_name=region, aws_access_key_id=AWSAccessKeyId, aws_secret_access_key=AWSSecretKey)
    client.start_instances(InstanceIds=instances)
    print ('started your instances: ' + str(instances))

    #Get IP addresses of EC2 instances
    #client = boto3.client('ec2')
    instDict=client.describe_instances(
            Filters=[{'Name':'tag:Environment','Values':['Dev']}]
        )


    hostList=[]

    for r in instDict['Reservations']:

        for inst in r['Instances']:

            hostList.append(inst['PublicIpAddress'])

    #Invoke worker function for each IP address
    client = boto3.client('lambda')

    for host in hostList:
        print ("Invoking worker_function on " + host)
        invokeResponse=client.invoke(
            FunctionName='worker_function',
            InvocationType='Event',
            LogType='Tail',
            Payload='{"IP":"'+ host +'"}'
        )
        print ('response is' + str(invokeResponse))

    return{
        'message' : "Trigger function finished"
    }

```

Example Policy Attached to the Trigger Lambda function:


```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": "ec2:Describe*",
         "Resource": "*"
      },
      {
         "Sid": "VisualEditor1",
         "Effect": "Allow",
```

```
        "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "lambda:InvokeFunction"
        ],
        "Resource": [
            "arn:aws:lambda:us-east-1:926962607868:function:worker_function"
          ]
      }
   ]
}
```

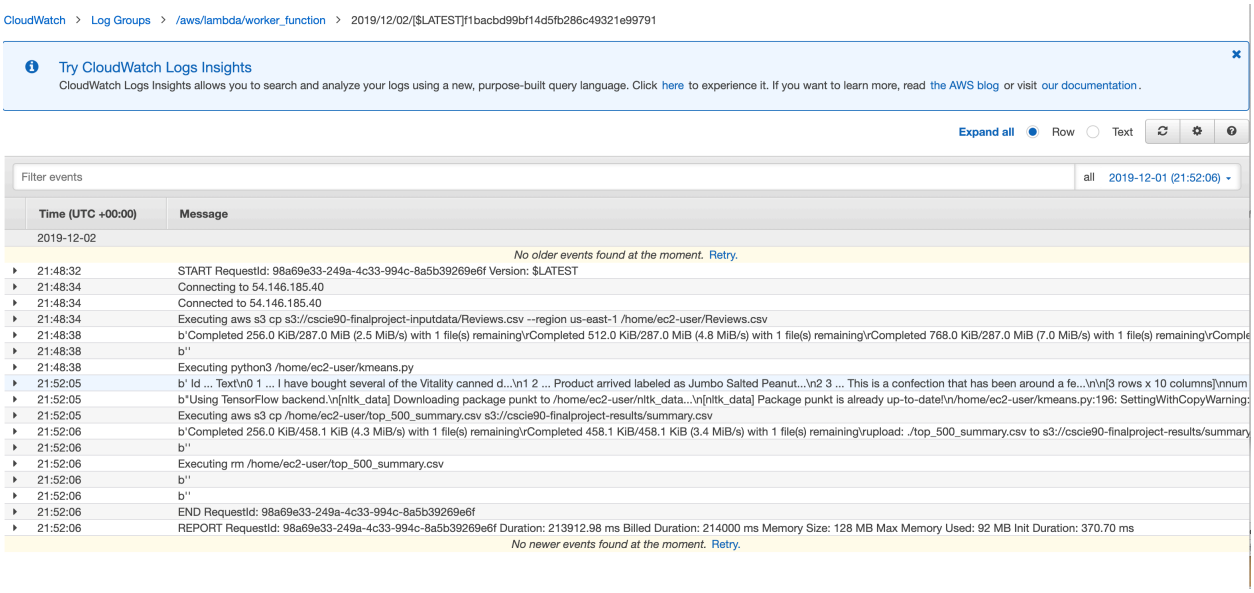
Worker Function Lambda:

This function creates a SSH tunnel into the EC2 instance, and executes various s3 commads, runs the text summarizer “kmeans.py” model, and copies the results to the output s3 results bucket (cscie90-finalproject-results). One very important aspect of this lambda function was the installation of paramiko package which provides methods to SSH into an EC2 instance. The steps outlined in this blog (https://towardsdatascience.com/hosting-your-ml-model-on-aws-lambdas-api-gateway-part-1-9052e6b63b25) were followed to install the paramiko package into the worker_function lambda layer.

Administrative EC2 access policy was attached to this lambda function to execute the s3 and python code on the EC2 instance.

```
import json
import boto3
import paramiko

def lambda_handler(event, context):

    s3_client = boto3.client('s3')
    #Download private key file from secure S3 bucket
    s3_client.download_file('cscie90-finalproject-virginia','virginia.pem', '/tmp/virginia.pem')

    k = paramiko.RSAKey.from_private_key_file("/tmp/virginia.pem")
    c = paramiko.SSHClient()
    c.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    host=event['IP']
    print ("Connecting to " + host)
    c.connect( hostname = host, username = "ec2-user", pkey = k )
    print ("Connected to " + host)

    commands = [
        "aws s3 cp s3://cscie90-finalproject-inputdata/Reviews.csv --region us-east-1 /home/ec2-user/Reviews.csv",
        "python3 /home/ec2-user/kmeans.py",
        "aws s3 cp /home/ec2-user/top_500_summary.csv s3://cscie90-finalproject-results/summary.csv",
        "rm /home/ec2-user/top_500_summary.csv"
        ]

    for command in commands:
        print ("Executing {}".format(command))
        stdin , stdout, stderr = c.exec_command(command)
        print (stdout.read())
        print (stderr.read())

    return
    {
        'message' : "Script execution completed. See Cloudwatch logs for complete output"
    }
```

A sample CloudWatch Log showing execution of worker lambda function: The logs are a useful resource to debug lambda functions.

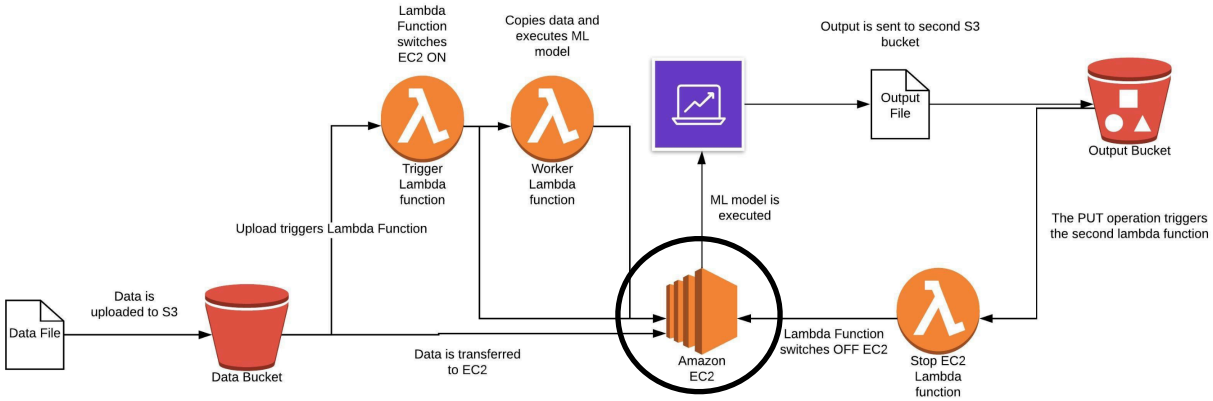


StopEC2 Lambda:

This function is invoked when results are posted to the output S3 bucket (cscie90-finalproject-results). Again, administrative EC2 access privileges were granted to this lambda function to stop the EC2 instance.

```
import json
import boto3

# Enter the region your instances are in. Include only the region without specifying Availability Zone; e.g.; 'us-east-1'
region = 'us-east-1'
# Enter your instances here: ex. ['X-XXXXXXXX', 'X-XXXXXXXX']
instances = ['X-XXXXXXXXX']

AWSAccessKeyId='secretId'
AWSSecretKey='secretKey'

def lambda_handler(event, context):

    client = boto3.client('ec2', region_name=region, aws_access_key_id=AWSAccessKeyId, aws_secret_access_key=AWSSecretKey)
    client.stop_instances(InstanceIds=instances)
    print ('stopped your instances: ' + str(instances))

    return {
        'statusCode': 200,
        'body': json.dumps('Successfully stopped EC2 from stopEC2 Lambda')
    }

```

Text Summarizer Model (kmeans.py):



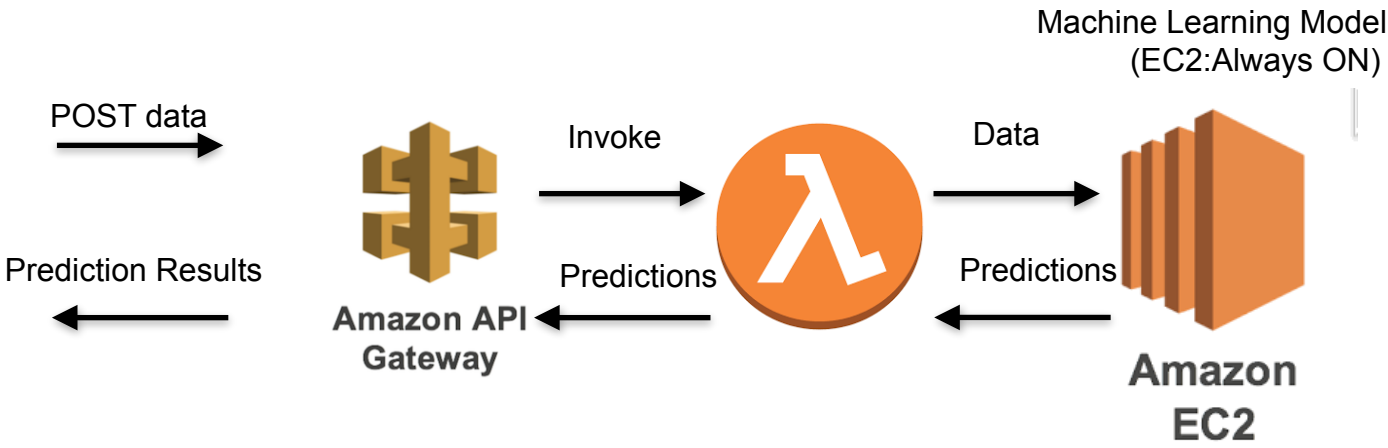
The model has been taken from the GitHub https://github.com/Preetikasri/CognitiveComputing-TextSummarizer/blob/master/Text_summarizer_unsupervised.ipynb.

The source code for this model is in “kmeans.py” and is available at the GitHub link identified in the resources section towards the end of the document.

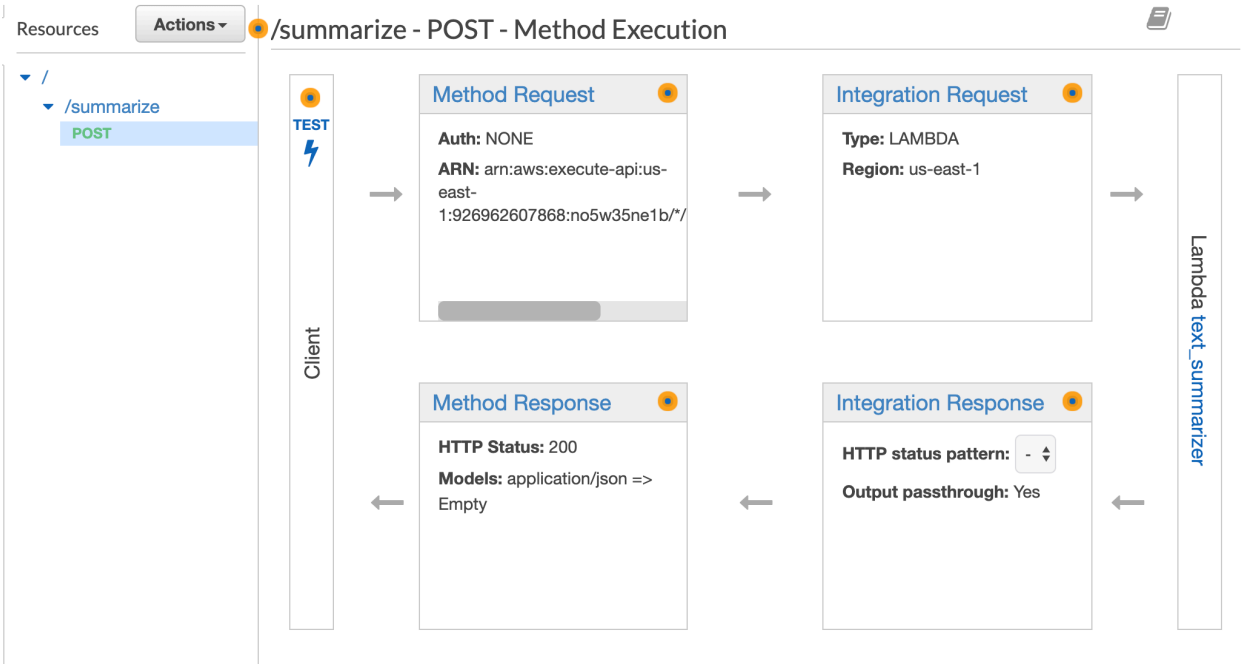
The model loads the glove embeddings for the tokens, and implements a kmeans algorithm. It essentially finds the cluster centroid and picks the sentences closest to the centroid. Essentially the centroid is the gist of the individual amazon food reviews. A sample of summarized text reviews are available in the file top_500_summary.csv on GitHub.

GOAL #2 IMPLEMENTATION DETAILS:

Real time text summarization (predictions) via REST API call.

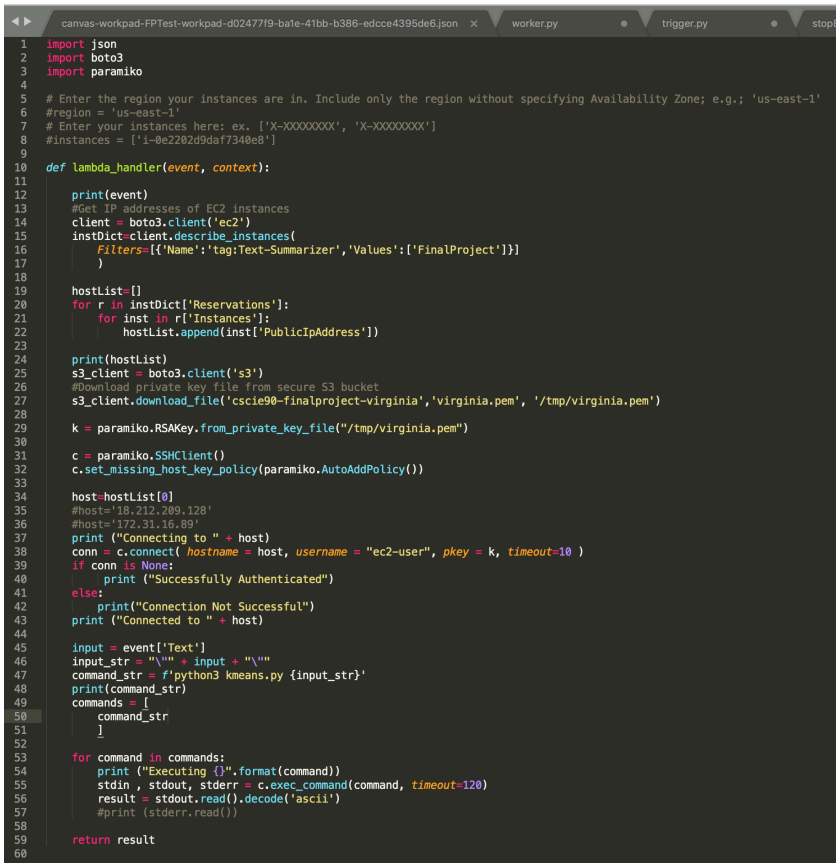


API Gateway:The API Gateway was configured on summarize resource, with action "POST". The API Gateway calls the text_summarizer lambda function when the end-user sends a POST message with the text that needs to be summarized.



Lambda Function:

The API Gateway invokes the text_summarizer lambda function. This function is very similar in functionality to the worker lambda function described earlier. It creates a SSH tunnel with the always ON EC2 instance, and takes the “text” message from API Gateway and runs the text summarizer model (“kmeans.py”) on the text, and returns the summarized text back to the API Gateway. The screenshot for the text summarizer lambda code is below. It has similar administrative privileges policy assigned to it to execute code on the EC2 instance.

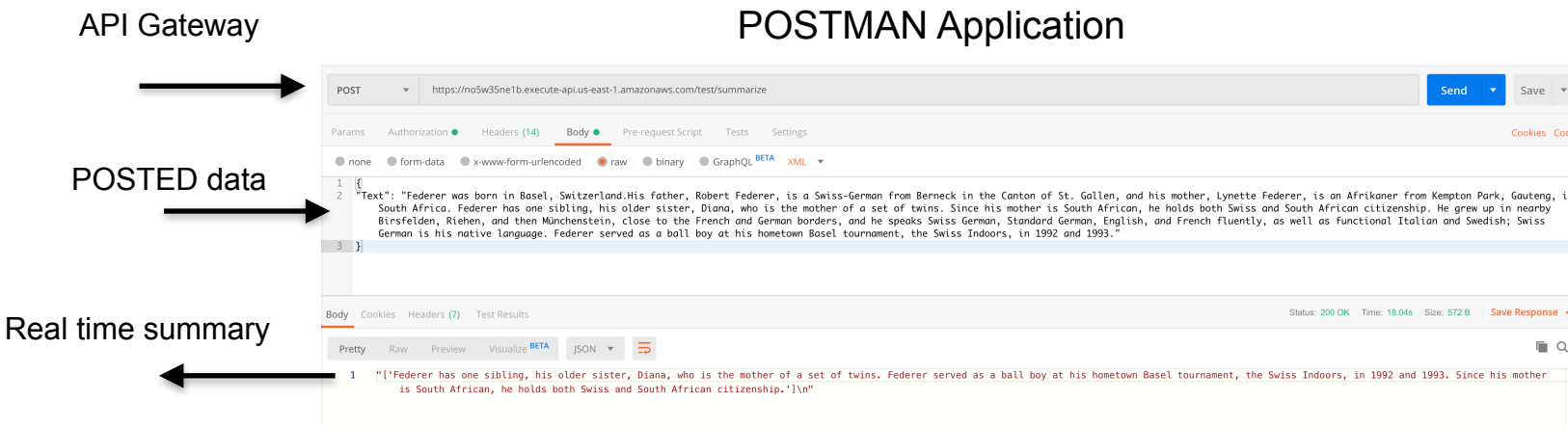


Text Summarizer Model:

The “kmeans.py” model is run on the EC2 instance. It takes the “text” argument passed to it from the text_summarizer lambda function, and prints the summarized text to its console. The lambda function grabs the console output and passes it back to the API Gateway.

Results:

Here is a screenshot of text summarizer model using POSTMAN. The sample text was taken from Roger Federer's wikipedia and sent via API Gateway POST on the summarize resource. The response in red is the summarized text from the model. The round trip time is approximately 30seconds.



Resources:

- Project Report (PDF document):https://github.com/amitrgupta27/cscie90-finalproject
- Code:https://github.com/amitrgupta27/cscie90-finalproject
- Youtube URL (Project Overview):https://youtu.be/aQPjzQAn_Ys
- Youtube URL (Deep Dive): https://www.youtube.com/watch?v=s8wfpDl0Okk
- Contact:amitrgupta27@gmail.com
- References:
  - https://towardsdatascience.com/hosting-your-ml-model-on-aws-lambdas-api-gateway-part-1-9052e6b63b25
  - https://towardsdatascience.com/hosting-your-ml-model-on-aws-lambdas-api-gateway-part-2-23517609522b
  - https://towardsdatascience.com/automating-machine-learning-models-on-aws-bfa183fe4065
  - https://towardsdatascience.com/data-scientists-guide-to-summarization-fc0db952e363