|  |
| --- |
| GSP329 : Integrate with Machine Learning APIs: Challenge Lab :- |
|  |  |
|  | ---------------------------------------------------------------------------------------------------------------------------------------------- |
|  |  |
|  | // Run in Cloud Shell :- |
|  |  |
|  | export SANAME=challenge |
|  | gcloud iam service-accounts create $SANAME |
|  | gcloud projects add-iam-policy-binding $DEVSHELL\_PROJECT\_ID --member=serviceAccount:$SANAME@$DEVSHELL\_PROJECT\_ID.iam.gserviceaccount.com --role=roles/bigquery.admin |
|  | gcloud projects add-iam-policy-binding $DEVSHELL\_PROJECT\_ID --member=serviceAccount:$SANAME@$DEVSHELL\_PROJECT\_ID.iam.gserviceaccount.com --role=roles/storage.admin |
|  | gcloud iam service-accounts keys create sa-key.json --iam-account $SANAME@$DEVSHELL\_PROJECT\_ID.iam.gserviceaccount.com |
|  | export GOOGLE\_APPLICATION\_CREDENTIALS=${PWD}/sa-key.json |
|  | gsutil cp gs://$DEVSHELL\_PROJECT\_ID/analyze-images.py . |
|  |  |
|  |  |
|  | // Open Editor and replace the content of "analyze-images.py" file with :- |
|  |  |
|  |  |
|  | # Dataset: image\_classification\_dataset |
|  |  |
|  | # Table name: image\_text\_detail |
|  |  |
|  | import os |
|  |  |
|  | import sys |
|  |  |
|  |  |
|  |  |
|  | # Import Google Cloud Library modules |
|  |  |
|  | from google.cloud import storage, bigquery, language, vision, translate\_v2 |
|  |  |
|  |  |
|  |  |
|  | if ('GOOGLE\_APPLICATION\_CREDENTIALS' in os.environ): |
|  |  |
|  | if (not os.path.exists(os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'])): |
|  |  |
|  | print ("The GOOGLE\_APPLICATION\_CREDENTIALS file does not exist.\n") |
|  |  |
|  | exit() |
|  |  |
|  | else: |
|  |  |
|  | print ("The GOOGLE\_APPLICATION\_CREDENTIALS environment variable is not defined.\n") |
|  |  |
|  | exit() |
|  |  |
|  |  |
|  |  |
|  | if len(sys.argv)<3: |
|  |  |
|  | print('You must provide parameters for the Google Cloud project ID and Storage bucket') |
|  |  |
|  | print ('python3 '+sys.argv[0]+ '[PROJECT\_NAME] [BUCKET\_NAME]') |
|  |  |
|  | exit() |
|  |  |
|  |  |
|  |  |
|  | project\_name = sys.argv[1] |
|  |  |
|  | bucket\_name = sys.argv[2] |
|  |  |
|  |  |
|  |  |
|  | # Set up our GCS, BigQuery, and Natural Language clients |
|  |  |
|  | storage\_client = storage.Client() |
|  |  |
|  | bq\_client = bigquery.Client(project=project\_name) |
|  |  |
|  | nl\_client = language.LanguageServiceClient() |
|  |  |
|  |  |
|  |  |
|  | # Set up client objects for the vision and translate\_v2 API Libraries |
|  |  |
|  | vision\_client = vision.ImageAnnotatorClient() |
|  |  |
|  | translate\_client = translate\_v2.Client() |
|  |  |
|  |  |
|  |  |
|  | # Setup the BigQuery dataset and table objects |
|  |  |
|  | dataset\_ref = bq\_client.dataset('image\_classification\_dataset') |
|  |  |
|  | dataset = bigquery.Dataset(dataset\_ref) |
|  |  |
|  | table\_ref = dataset.table('image\_text\_detail') |
|  |  |
|  | table = bq\_client.get\_table(table\_ref) |
|  |  |
|  |  |
|  |  |
|  | # Create an array to store results data to be inserted into the BigQuery table |
|  |  |
|  | rows\_for\_bq = [] |
|  |  |
|  |  |
|  |  |
|  | # Get a list of the files in the Cloud Storage Bucket |
|  |  |
|  | files = storage\_client.bucket(bucket\_name).list\_blobs() |
|  |  |
|  | bucket = storage\_client.bucket(bucket\_name) |
|  |  |
|  |  |
|  |  |
|  | print('Processing image files from GCS. This will take a few minutes..') |
|  |  |
|  |  |
|  |  |
|  | # Process files from Cloud Storage and save the result to send to BigQuery |
|  |  |
|  | for file in files: |
|  |  |
|  | if file.name.endswith('jpg') or file.name.endswith('png'): |
|  |  |
|  | file\_content = file.download\_as\_string() |
|  |  |
|  |  |
|  |  |
|  | # TBD: Create a Vision API image object called image\_object |
|  |  |
|  | # Ref: https://googleapis.dev/python/vision/latest/gapic/v1/types.html#google.cloud.vision\_v1.types.Image |
|  |  |
|  | from google.cloud import vision\_v1 |
|  |  |
|  | import io |
|  |  |
|  | client = vision.ImageAnnotatorClient() |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | # TBD: Detect text in the image and save the response data into an object called response |
|  |  |
|  | # Ref: https://googleapis.dev/python/vision/latest/gapic/v1/api.html#google.cloud.vision\_v1.ImageAnnotatorClient.document\_text\_detection |
|  |  |
|  | image = vision\_v1.types.Image(content=file\_content) |
|  |  |
|  | response = client.text\_detection(image=image) |
|  |  |
|  |  |
|  |  |
|  | # Save the text content found by the vision API into a variable called text\_data |
|  |  |
|  | text\_data = response.text\_annotations[0].description |
|  |  |
|  |  |
|  |  |
|  | # Save the text detection response data in <filename>.txt to cloud storage |
|  |  |
|  | file\_name = file.name.split('.')[0] + '.txt' |
|  |  |
|  | blob = bucket.blob(file\_name) |
|  |  |
|  | # Upload the contents of the text\_data string variable to the Cloud Storage file |
|  |  |
|  | blob.upload\_from\_string(text\_data, content\_type='text/plain') |
|  |  |
|  |  |
|  |  |
|  | # Extract the description and locale data from the response file |
|  |  |
|  | # into variables called desc and locale |
|  |  |
|  | # using response object properties e.g. response.text\_annotations[0].description |
|  |  |
|  | desc = response.text\_annotations[0].description |
|  |  |
|  | locale = response.text\_annotations[0].locale |
|  |  |
|  |  |
|  |  |
|  | # if the locale is English (en) save the description as the translated\_txt |
|  |  |
|  | if locale == 'en': |
|  |  |
|  | translated\_text = desc |
|  |  |
|  | else: |
|  |  |
|  | # TBD: For non EN locales pass the description data to the translation API |
|  |  |
|  | # ref: https://googleapis.dev/python/translation/latest/client.html#google.cloud.translate\_v2.client.Client.translate |
|  |  |
|  | # Set the target\_language locale to 'en') |
|  |  |
|  | from google.cloud import translate\_v2 as translate |
|  |  |
|  |  |
|  |  |
|  | client = translate.Client() |
|  |  |
|  | translation = translate\_client.translate(text\_data, target\_language='en') |
|  |  |
|  | translated\_text = translation['translatedText'] |
|  |  |
|  | print(translated\_text) |
|  |  |
|  |  |
|  |  |
|  | # if there is response data save the original text read from the image, |
|  |  |
|  | # the locale, translated text, and filename |
|  |  |
|  | if len(response.text\_annotations) > 0: |
|  |  |
|  | rows\_for\_bq.append((desc, locale, translated\_text, file.name)) |
|  |  |
|  |  |
|  |  |
|  | print('Writing Vision API image data to BigQuery...') |
|  |  |
|  | # Write original text, locale and translated text to BQ |
|  |  |
|  | # TBD: When the script is working uncomment the next line to upload results to BigQuery |
|  |  |
|  | errors = bq\_client.insert\_rows(table, rows\_for\_bq) |
|  |  |
|  |  |
|  |  |
|  | assert errors == [] |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | // In Cloud Shell run :- |
|  |  |
|  | python3 analyze-images.py $DEVSHELL\_PROJECT\_ID $DEVSHELL\_PROJECT\_ID |
|  |  |
|  | // Navigation Menu -> BigQuery, Run :- |
|  |  |
|  | SELECT locale,COUNT(locale) as lcount FROM image\_classification\_dataset.image\_text\_detail GROUP BY locale ORDER BY lcount DESC |