Sentiment Analysis of Documents and Speech to Text

Project Report for AI/ML tools in GCP Summer Course

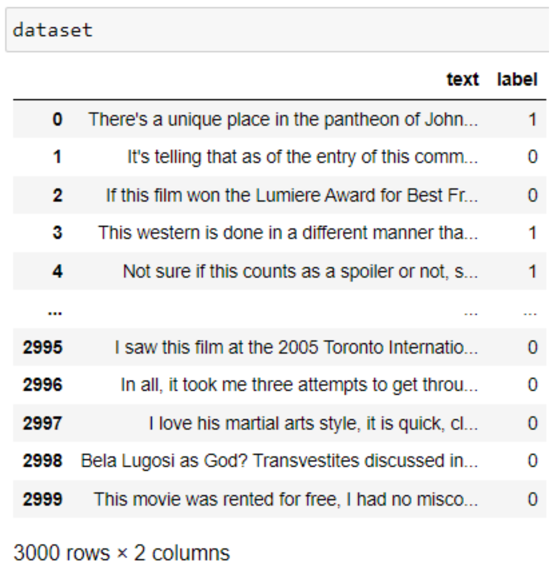
Akella Aditya Bhargav (PES2UG19CS024)
Ankit P Bisleri (PES2UG19CS048)
Anvesh S K (PES2UG19CS056)

Goals

I. Use Document AI to extract text from a PDF/ Speech to Text API for transcription

II. Find the sentiment of the extracted text using a model trained by AutoML

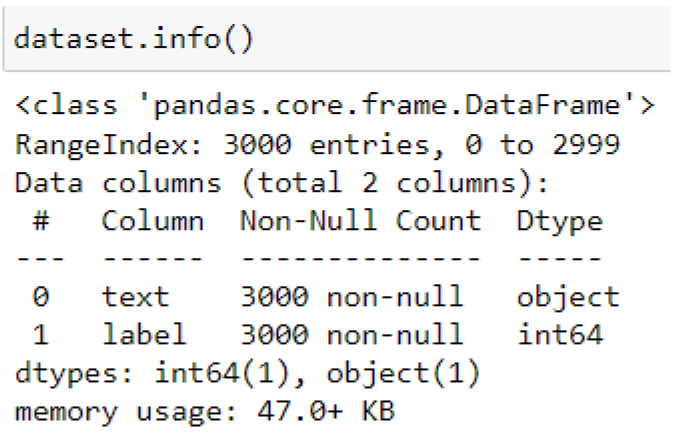
Datasets



```
dataset
```

	text	label
0	There's a unique place in the pantheon of John...	1
1	It's telling that as of the entry of this comm...	0
2	If this film won the Lumiere Award for Best Fr...	0
3	This western is done in a different manner tha...	1
4	Not sure if this counts as a spoiler or not, s...	1
...	...	...
2995	I saw this film at the 2005 Toronto Internatio...	0
2996	In all, it took me three attempts to get throu...	0
2997	I love his martial arts style, it is quick, cl...	0
2998	Bela Lugosi as God? Transvestites discussed in...	0
2999	This movie was rented for free, I had no misco...	0

3000 rows × 2 columns



```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    3000 non-null   object
 1   label   3000 non-null   int64
dtypes: int64(1), object(1)
memory usage: 47.0+ KB
```

I. This is a custom dataset that consists of tuples from several review and sentiment datasets.

II. The two attributes are “text” and “label”. This is a single-label dataset.

III. There are only 2 labels 0 and 1 corresponding to negative and positive sentiment.

IV. We trained 2 datasets and found the above one to have better results, but we have uploaded screenshots of both.

Approach

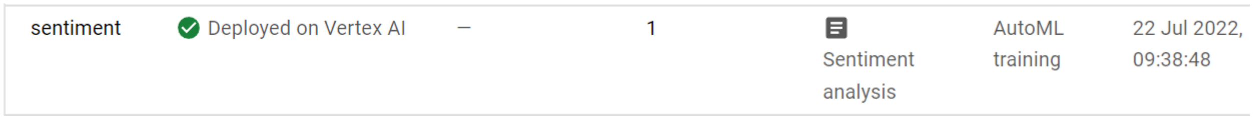
I.

Used Document AI for extracting text from the .pdf file whose path is to be given by the user. The above dataset was used to train a model using AutoML and deployed to an endpoint. We use python libraries to write functions that use these APIs and endpoints to finally give us the extracted text and predicted sentiment.

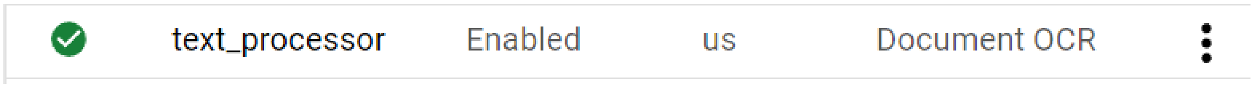
II.

Used the Speech to text API provided by GCP to transform audio files into text. This is then passed to the sentiment analysis model done through Vertex AI to determine the "sentiment" of the text.

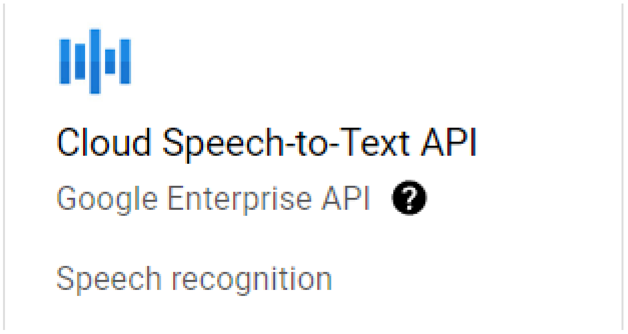
Model and Pretrained Models Used



sentiment	✓ Deployed on Vertex AI	–	1	Sentiment analysis	AutoML training	22 Jul 2022, 09:38:48



✓	text_processor	Enabled	us	Document OCR

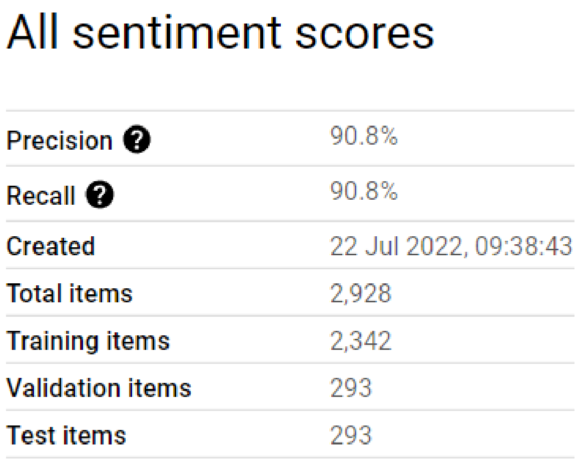


Cloud Speech-to-Text API

Google Enterprise API

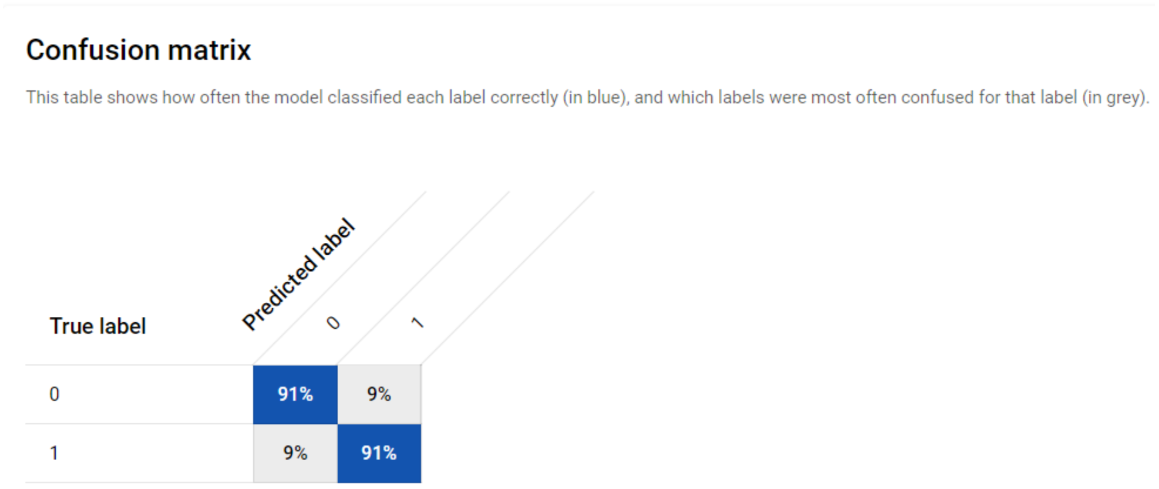
Speech recognition

Model Evaluation



All sentiment scores

Precision	90.8%
Recall	90.8%
Created	22 Jul 2022, 09:38:43
Total items	2,928
Training items	2,342
Validation items	293
Test items	293



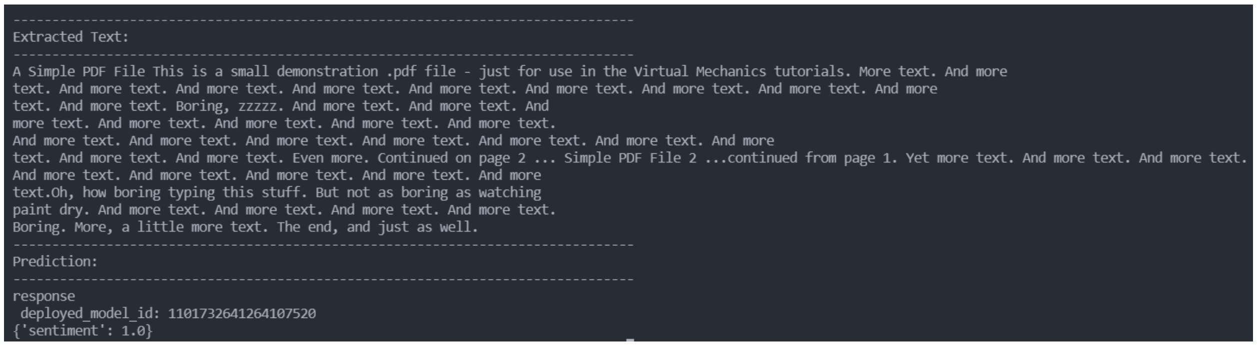
Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in grey).

True label \ Predicted label	0	1
0	91%	9%
1	9%	91%

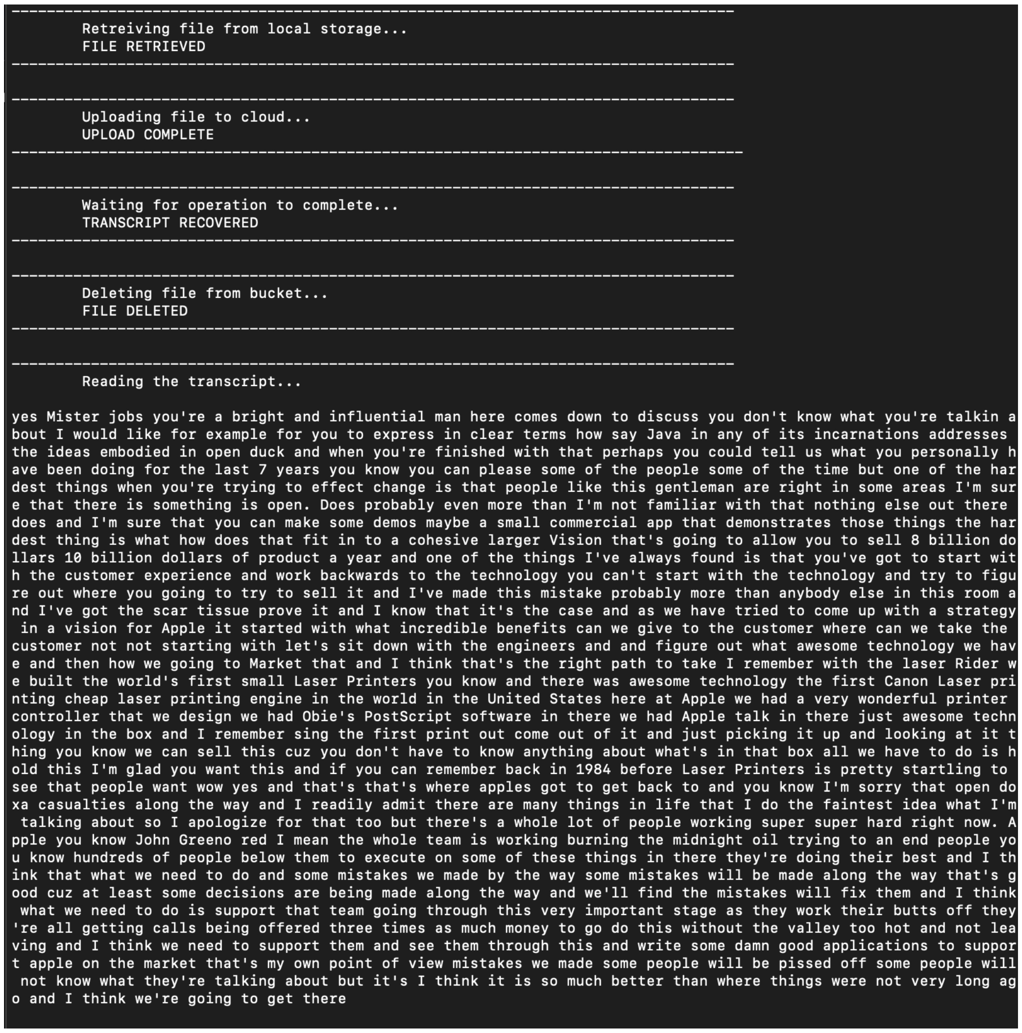
Output

Using Document API

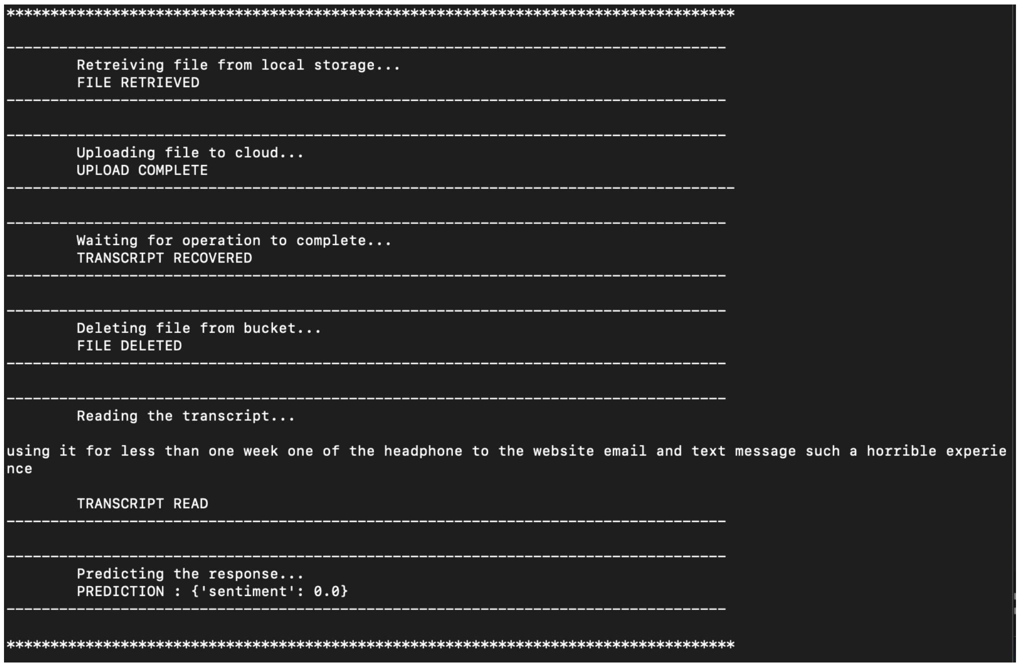


```
--------------------------------------------------------------------------------
Extracted Text:
--------------------------------------------------------------------------------
A Simple PDF File This is a small demonstration .pdf file - just for use in the Virtual Mechanics tutorials. More text. And more
text. And more text. And more text. And more text. And more text. And more text. And more text. And more text. And more
text. And more text. Boring, zzzzz. And more text. And more text. And
more text. And more text. And more text. And more text. And more text.
And more text. And more text. And more text. And more text. And more text. And more text. And more
text. And more text. And more text. Even more. Continued on page 2 ... Simple PDF File 2 ...continued from page 1. Yet more text. And more text. And more text.
And more text. And more text. And more text. And more text. And more
text.Oh, how boring typing this stuff. But not as boring as watching
paint dry. And more text. And more text. And more text. And more text.
Boring. More, a little more text. The end, and just as well.
--------------------------------------------------------------------------------
Prediction:
--------------------------------------------------------------------------------
response
 deployed_model_id: 1101732641264107520
{'sentiment': 1.0}
```

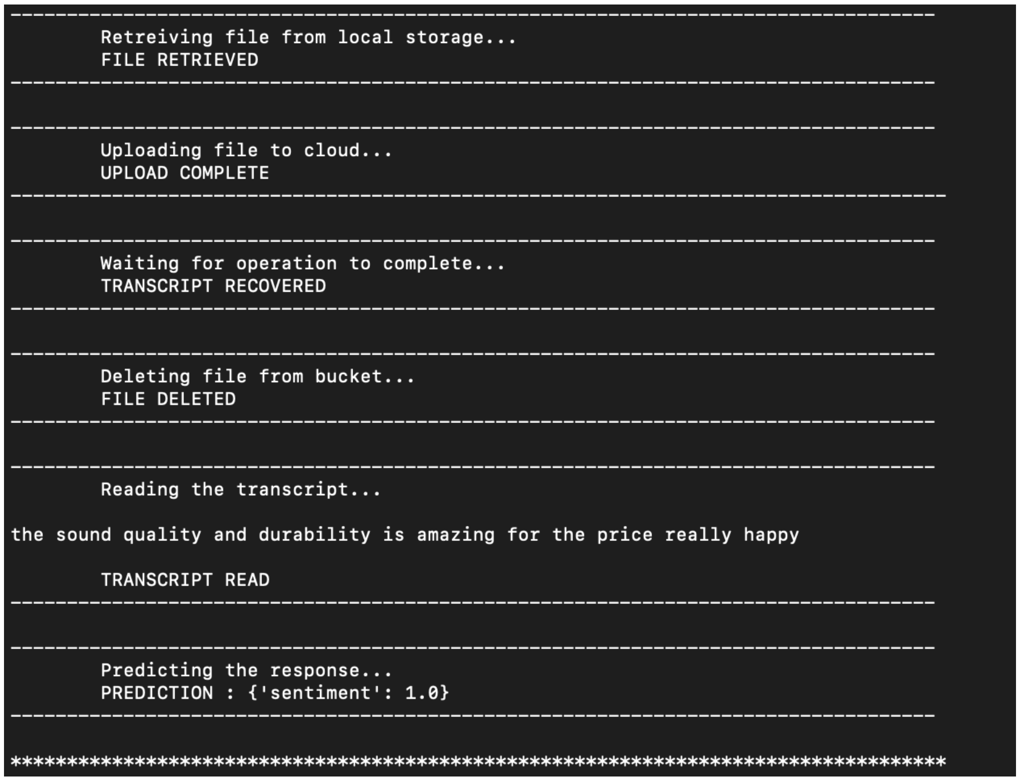
Using Text to Speech API



```
----------------------------------------------------------------------------------
        Retreiving file from local storage...
        FILE RETRIEVED
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Uploading file to cloud...
        UPLOAD COMPLETE
-----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Waiting for operation to complete...
        TRANSCRIPT RECOVERED
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Deleting file from bucket...
        FILE DELETED
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Reading the transcript...

yes Mister jobs you're a bright and influential man here comes down to discuss you don't know what you're talkin a
bout I would like for example for you to express in clear terms how say Java in any of its incarnations addresses
the ideas embodied in open duck and when you're finished with that perhaps you could tell us what you personally h
ave been doing for the last 7 years you know you can please some of the people some of the time but one of the har
dest things when you're trying to effect change is that people like this gentleman are right in some areas I'm sur
e that there is something is open. Does probably even more than I'm not familiar with that nothing else out there
does and I'm sure that you can make some demos maybe a small commercial app that demonstrates those things the har
dest thing is what how does that fit in to a cohesive larger Vision that's going to allow you to sell 8 billion do
llars 10 billion dollars of product a year and one of the things I've always found is that you've got to start wit
h the customer experience and work backwards to the technology you can't start with the technology and try to figu
re out where you going to try to sell it and I've made this mistake probably more than anybody else in this room a
nd I've got the scar tissue prove it and I know that it's the case and as we have tried to come up with a strategy
 in a vision for Apple it started with what incredible benefits can we give to the customer where can we take the
customer not not starting with let's sit down with the engineers and and figure out what awesome technology we hav
e and then how we going to Market that and I think that's the right path to take I remember with the laser Rider w
e built the world's first small Laser Printers you know and there was awesome technology the first Canon Laser pri
nting cheap laser printing engine in the world in the United States here at Apple we had a very wonderful printer
controller that we design we had Obie's PostScript software in there we had Apple talk in there just awesome techn
ology in the box and I remember sing the first print out come out of it and just picking it up and looking at it t
hing you know we can sell this cuz you don't have to know anything about what's in that box all we have to do is h
old this I'm glad you want this and if you can remember back in 1984 before Laser Printers is pretty startling to
see that people want wow yes and that's where apples got to get back to and you know I'm sorry that open do
xa casualties along the way and I readily admit there are many things in life that I do the faintest idea what I'm
 talking about so I apologize for that too but there's a whole lot of people working super super hard right now. A
pple you know John Greeno red I mean the whole team is working burning the midnight oil trying to an end people yo
u know hundreds of people below them to execute on some of these things in there they're doing their best and I th
ink that what we need to do and some mistakes we made by the way some mistakes will be made along the way that's g
ood cuz at least some decisions are being made along the way and we'll find the mistakes will fix them and I think
 what we need to do is support that team going through this very important stage as they work their butts off they
're all getting calls being offered three times as much money to go do this without the valley too hot and not lea
ving and I think we need to support them and see them through this and write some damn good applications to suppor
t apple on the market that's my own point of view mistakes we made some people will be pissed off some people will
 not know what they're talking about but it's I think it is so much better than where things were not very long ag
o and I think we're going to get there
```



```
***********************************************************************************

----------------------------------------------------------------------------------
        Retreiving file from local storage...
        FILE RETRIEVED
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Uploading file to cloud...
        UPLOAD COMPLETE
-----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Waiting for operation to complete...
        TRANSCRIPT RECOVERED
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Deleting file from bucket...
        FILE DELETED
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Reading the transcript...

using it for less than one week one of the headphone to the website email and text message such a horrible experie
nce

        TRANSCRIPT READ
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Predicting the response...
        PREDICTION : {'sentiment': 0.0}
----------------------------------------------------------------------------------

***********************************************************************************
```



```
----------------------------------------------------------------------------------
        Retreiving file from local storage...
        FILE RETRIEVED
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Uploading file to cloud...
        UPLOAD COMPLETE
-----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Waiting for operation to complete...
        TRANSCRIPT RECOVERED
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Deleting file from bucket...
        FILE DELETED
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Reading the transcript...

the sound quality and durability is amazing for the price really happy

        TRANSCRIPT READ
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
        Predicting the response...
        PREDICTION : {'sentiment': 1.0}
----------------------------------------------------------------------------------

***********************************************************************************
```

CodeBase

Documentai.py

```
from google.cloud import aiplatform

from google.cloud.aiplatform.gapic.schema import predict

from google.protobuf import json_format

from google.protobuf.struct_pb2 import Value

from google.cloud import documentai_v1 as documentai

from google.cloud import storage

import os

project_id= 'summercourse-356708'

location = 'us' # Format is 'us' or 'eu'

processor_id = '569485ff49ed7f49' #  Create processor in Cloud Console

file_path = input('Enter file path: ')

os.system('cls')

for i in file_path:

    if i == "'\'":

        i = '/'

endpoint_id = '6239798856372977664'

def quickstart(project_id: str, location: str, processor_id: str, file_path: str):

    temp = []

    opts = {}

    if location == "eu":

        opts = {"api_endpoint": "eu-documentai.googleapis.com"}


    client = documentai.DocumentProcessorServiceClient(client_options=opts)


    name = f"projects/{project_id}/locations/{location}/processors/{processor_id}"


    # Read the file into memory

    with open(file_path, "rb") as image:

        image_content = image.read()
    document = {"content": image_content, "mime_type": "application/pdf"}

    # Configure the process request
    request = {"name": name, "raw_document": document}

    result = client.process_document(request=request)
    document = result.document

    document_pages = document.pages
    # Read the text recognition output from the processor
    print("The document contains the following paragraphs:")
    for page in document_pages:
        paragraphs = page.paragraphs
        for paragraph in paragraphs:
            print(paragraph)
            paragraph_text = get_text(paragraph.layout, document)
            temp.append(paragraph_text)
    return temp

def get_text(doc_element: dict, document: dict):

    response = ""
    # If a text segment spans several lines, it will
    # be stored in different text segments.
    for segment in doc_element.text_anchor.text_segments:
        start_index = (
            int(segment.start_index)
            if segment in doc_element.text_anchor.text_segments
            else 0
        )
        end_index = int(segment.end_index)
        response += document.text[start_index:end_index]
    return response
def upload_to_bucket(blob_name, path_to_file, bucket_name):
    """ Upload data to a bucket"""
    # Explicitly use service account credentials by specifying the private key file.

    storage_client = storage.Client.from_service_account_json(

        'summercourse-356708-3dfb121f2395.json')

    print("Uploading file to cloud... ")

    bucket = storage_client.bucket(bucket_name)

    blob = bucket.blob(blob_name)

    blob.upload_from_filename(path_to_file)

    audio_file_name = blob.public_url.split("/")[-1]


    gcs_uri = 'gs://' + bucket_name + '/' + audio_file_name


    #returns a gsutil uri

    return gcs_uri


def delete_blob(bucket_name, blob_name):

    """Deletes a blob from the bucket."""

    storage_client = storage.Client.from_service_account_json(

        'summercourse-356708-3dfb121f2395.json')

    bucket = storage_client.get_bucket(bucket_name)

    blob = bucket.blob(blob_name)


    print("Deleting file from bucket...")


    blob.delete()

def predict_text_sentiment_analysis_sample(

    project: str,

    endpoint_id: str,

    content: str,

    location: str = "us-central1",

    api_endpoint: str = "us-central1-aiplatform.googleapis.com",

):

    # The AI Platform services require regional API endpoints.

    client_options = {"api_endpoint": api_endpoint}

    # Initialize client that will be used to create and send requests.

    # This client only needs to be created once, and can be reused for multiple requests.

    client = aiplatform.gapic.PredictionServiceClient(client_options=client_options)
    instance = predict.instance.TextSentimentPredictionInstance(
        content=content,
    ).to_value()
    instances = [instance]
    parameters_dict = {}
    parameters = json_format.ParseDict(parameters_dict, Value())
    endpoint = client.endpoint_path(
        project=project, location=location, endpoint=endpoint_id
    )
    response = client.predict(
        endpoint=endpoint, instances=instances, parameters=parameters
    )
    print("response")
    print(" deployed_model_id:", response.deployed_model_id)
    # See gs://google-cloud-aiplatform/schema/predict/prediction/text_sentiment_1.0.0.yaml for the
format of the predictions.
    predictions = response.predictions
    for prediction in predictions:
        print(dict(prediction))

response = quickstart(project_id, location, processor_id, file_path)
string = ''
for i in response:
    string+= i.strip() + ' '
os.system('cls')
print('--------------------------------------------------------------------------------')
print('Extracted Text: ')
print('--------------------------------------------------------------------------------')
print(string)
print('--------------------------------------------------------------------------------')
print('Prediction: ')
print('--------------------------------------------------------------------------------')
predict_text_sentiment_analysis_sample(project_id, endpoint_id, string)
```

sc_gcp_project.py

```
#Essentials
```

```python
audio_filepath = "/Users/anveshsk/Dropbox/Final_summer_course/audio/"

text_filepath = "/Users/anveshsk/Dropbox/Final_summer_course/Transcripts/"

bucket_name = "anvesh_demo"

project_id = "summercourse-356705"

endpoint_id = "8531005166797717504"

json_key = 'summercourse-356705-c2d5caafe2e5.json'

#Import libraries

from pydub import AudioSegment

from google.cloud import speech

from google.cloud import storage

import wave

import os

import io

from sentiment_prediction import *

# Find the framerate and channels of audio file

def frame_rate_channel(audio_file_name):

    with wave.open(audio_file_name, "rb") as wave_file:

        frame_rate = wave_file.getframerate()

        channels = wave_file.getnchannels()

        return frame_rate, channels

#Set the channel to mono

def stereo_to_mono(audio_file_name):

    sound = AudioSegment.from_wav(audio_file_name)

    sound = sound.set_channels(1)

#Dependency : pip install --upgrade google-cloud-storage.

#Uploading audio file to bucket

def upload_to_bucket(blob_name, path_to_file, bucket_name):

    """ Upload data to a bucket"""

    # Explicitly use service account credentials by specifying the private key file.

    storage_client = storage.Client.from_service_account_json(json_key

        )

    print("--------------------------------------------------------------------------------")

    print("\tUploading file to cloud... ")


    bucket = storage_client.bucket(bucket_name)

    blob = bucket.blob(blob_name)
    blob.upload_from_filename(path_to_file, timeout = (10,200))
    audio_file_name = blob.public_url.split("/")[-1]

    gcs_uri = 'gs://' + bucket_name + '/' + audio_file_name
    print("\tUPLOAD COMPLETE")
    print("----------------------------------------------------------------------------------\n")

    #returns a gsutil uri
    return gcs_uri
#Delete file from bucket
def delete_blob(bucket_name, blob_name):
    """Deletes a blob from the bucket."""
    storage_client = storage.Client.from_service_account_json(
        json_key)
    bucket = storage_client.get_bucket(bucket_name)
    blob = bucket.blob(blob_name)
    print("----------------------------------------------------------------------------------")
    print("\tDeleting file from bucket...")
    print("\tFILE DELETED")
    print("----------------------------------------------------------------------------------\n")
    blob.delete()
#Converting audio file to text
def transcribe_gcs(gcs_uri):
    """Asynchronously transcribes the audio file specified by the gcs_uri."""
    client = speech.SpeechClient()

    audio = speech.RecognitionAudio(uri=gcs_uri)
    frame_rate, channels = frame_rate_channel(audio_full_path)
    if channels > 1:
        stereo_to_mono(audio_full_path)

    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz=frame_rate,
        audio_channel_count=1,
        language_code ="en-US",
    )
    operation = client.long_running_recognize(config=config, audio=audio)
    print("----------------------------------------------------------------------------------")
    print("\tWaiting for operation to complete...")
    response = operation.result(timeout=90)

    print("\tTRANSCRIPT RECOVERED")
    print("----------------------------------------------------------------------------------\n")
    full_transcript = ''

    for result in response.results:
        full_transcript += result.alternatives[0].transcript
    return(full_transcript)
#Writing transcript onto a file
def write_transcripts(transcript_file_name,transcript):
    f= open(text_filepath + transcript_file_name,"w+")
    f.write(transcript)
    f.close()
if __name__ == "__main__":
    for audio_file_name in os.listdir(audio_filepath):
        if audio_file_name == ".DS_Store" :
            continue
        else :
            print("----------------------------------------------------------------------------------")
            print("\tRetreiving file from local storage...")
            print("\tFILE RETRIEVED")
            print("----------------------------------------------------------------------------------\n")
            audio_full_path = audio_filepath + audio_file_name
            uri= upload_to_bucket(audio_file_name,audio_full_path, bucket_name)
            complete_transcript = transcribe_gcs(uri)
            transcript_file_name = audio_file_name.split('.')[0] + '.txt'

            write_transcripts(transcript_file_name,complete_transcript)
            delete_blob("anvesh_demo", audio_file_name)
            transcript_text = read_text_transcript(text_filepath, transcript_file_name)
            predict_text_sentiment_analysis_sample(project_id, endpoint_id, transcript_text)
```

sentiment_prediction.py

```python
from google.cloud import aiplatform
from google.cloud.aiplatform.gapic.schema import predict
from google.protobuf import json_format
from google.protobuf.struct_pb2 import Value
import os
#Dependency : pip install --upgrade google-cloud-aiplatform
#Predict the sentiment of a block of text through Vertex AI's endpoint
def predict_text_sentiment_analysis_sample(
    project: str,
    endpoint_id: str,
    content: str,
    location: str = "us-central1",
    api_endpoint: str = "us-central1-aiplatform.googleapis.com",
):
    print("----------------------------------------------------------------------------------")
    print("\tPredicting the response...")
    # The AI Platform services require regional API endpoints.
    client_options = {"api_endpoint": api_endpoint}
    # Initialize client that will be used to create and send requests.
    # This client only needs to be created once, and can be reused for multiple requests.
    client = aiplatform.gapic.PredictionServiceClient(client_options=client_options)
    instance = predict.instance.TextSentimentPredictionInstance(
        content=content,
    ).to_value()
    instances = [instance]
    parameters_dict = {}
    parameters = json_format.ParseDict(parameters_dict, Value())
    endpoint = client.endpoint_path(
        project=project, location=location, endpoint=endpoint_id
    )
    response = client.predict(
        endpoint=endpoint, instances=instances, parameters=parameters
    )
    #print(" deployed_model_id:", response.deployed_model_id)

    # See gs://google-cloud-aiplatform/schema/predict/prediction/text_sentiment_1.0.0.yaml for the format of the
predictions.

    predictions = response.predictions


    for prediction in predictions:

        print("\tPREDICTION :", dict(prediction))

        print("----------------------------------------------------------------------------------\n")

        print("******************************************************************************************\n")

#Read the text from the file

def read_text_transcript(text_filepath, transcript_file_name):

    for text_file_name in os.listdir(text_filepath):

        if text_file_name == ".DS_Store" :

            continue

        else :

            if text_file_name == transcript_file_name :

                print("----------------------------------------------------------------------------------")

                print("\tReading the transcript...\n")

                text_file_name = text_filepath + text_file_name

                f= open(text_file_name,"r")

                transcript_text = f.read()

                print(transcript_text)

                f.close()

                print("\n\tTRANSCRIPT READ")

                print("----------------------------------------------------------------------------------\n")


    return(transcript_text)
```