1. Introduction

Text-to-speech is an innovative technology that as seen many advances in its area and due to its many applications and heavily researched areas part such as recognition of text and recreation of voice. Sounds such as reading words due to its easy analogue to digital easy conversion have become an excellent test for new coming algorithms such as machine learning and deep learning methods to learn and try to re-create from scratch by learning with data from real voice of people.

2. Program Explanation

These new tools have become increase famous and many companies that are actively developing new AI technologies such as Amazon, Google, Microsoft, OpenAi as well as many others less know, are starting to offer these products to the public mainly by cloud services. Services that are designed to provide easy, affordable access to applications and resources, without the need for internal infrastructure or hardware.

This program was almost entirely create using google cloud service resources using python just as the interactor to the platform. In google cloud the goggle they offer over 150 products and offer a 300$ in free credits in the signup, besides this when compared google voice syntheses its quality was one of the best.

3. Methods

To start the project, you must import 6 libraries, the first three are built in python and are the os (operating system module), the re (regular expression module) and the json module. Then to use the google cloud services install the google-cloud-vision, google-cloud-texttospeech and google-cloud-storage. After these imports and signing up for the Google cloud we need to create a bucket in or personal server to import the files we will want to use for the project.

For this start by creating an environmental variable for the google application credential that can be found in the bucket list in google cloud interface site, next call the vision Api client using vison.ImageAnnotatorClient().

Next Create a function for the creation of the bucket and inside create 4 variables, the first is for the storage called storage.Client(), secound for the bucket in storage, storage.bucket(“name_for_bucket”), the third for the storage class (bucket.storage_class) and at last to create the bucket using storage_client.create_bucket() .

Next Create a function for the upload of the file for the bucket and inside create 4 variables, first for the client and then for the bucket and then the third for the “blob” that are main information inside these buckets, using bucekt.blob(“name_for_output”). Lastly upload the file using blob.upload_from_filename(“name_of_file”).

Done this confirm if the files where successfully upload in the google cloud site and navegate to inside the bucket to get its URI that starts with “gs:// “. This URI is the file location in the google cloud server.

Now to start developing the main part of the program we will get access to the google vison Api that will allow us to upload a pdf and return the text that contains in it and if we want to so select its parts. Mainly this part of the code is available at google documentation of vison API.

Start by defining the input for the neural network, we create two variable one for the batch size for 2 and another for mime type of input (pdf), then we create the vison features variable at set it to detection of text do this using the dictionary type.

Next set the gcs_source link to the vison.InputConfig as well as its mime_type and set it as a dictionary. Do a similar process of the output but this time create your gsc output name and instead of the mime_type use the batch size. Use vision.AsyncAnnotateFileRequest to input the features, input configurations and output configuration. Finally call async_batch_annote_files and set a timeout for the result as 180. Created a storage client and call acess theoutput by loop the blob locating the first item that ios the text.

For the speech synthesis start by creating the client using .TextToSpeechClient(), acess to configure the synthesis_input where you pass the text from the previous fucntion to the dictionary key named “text”.

Then select the best parameters for the voice synthesis where we use langue, gender and the neural network. Finally set the audio encoding as mp3 and create a response and use it to write the response to the output file.

Appendix:

```
import os
import re
from google.cloud import storage
from google.cloud import texttospeech
from google.cloud import vision
import json
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'service-key-gcloud.json'
client = vision.ImageAnnotatorClient()


# def create_bucket(bucket_name, storage_class='STANDARD', location='eu'):
#     """ Create a new bucket"""
#     storage_client = storage.Clien"t()
#
#     bucket = storage_client.bucket(bucket_name)  # Bucket names must start and end with a number or
letter.
#     bucket.storage_class = storage_class
#
#     bucket = storage_client.create_bucket(bucket, location="eu")
#
#     return f'Bucket {bucket.name} successfully created.'
# create_bucket("dados666")


def async_detect_document(name_pdf):
    """OCR with PDF/TIFF as source files on GCS"""
    batch_size = 2


    mime_type = 'application/pdf'
    feature = vision.Feature({"type_": vision.Feature.Type.DOCUMENT_TEXT_DETECTION})


    gcs_source_uri = f"gs://dados666/{name_pdf}"
    gcs_source = vision.GcsSource({"uri": gcs_source_uri})
    input_config = vision.InputConfig({"gcs_source": gcs_source, "mime_type": mime_type})


    gcs_destination_uri = f"gs://dados666/namefff_pdf"  # <----- Possible problems
    gcs_destination = vision.GcsDestination({"uri": gcs_destination_uri})
    output_config = vision.OutputConfig({"gcs_destination": gcs_destination, "batch_size":
batch_size})


    async_request = vision.AsyncAnnotateFileRequest(
        {"features": [feature], "input_config": input_config, "output_config": output_config})


    operation = client.async_batch_annotate_files(requests=[async_request])
    print('Waiting for the operation to finish.')
    operation.result(timeout=180)


    # Once the request has completed and the output has been
    # written to GCS, we can list all the output files.
    storage_client = storage.Client()
    match = re.match(r'gs://([^/]+)/(.+)', gcs_destination_uri)
    bucket_name = match.group(1)
    prefix = match.group(2)


    bucket = storage_client.get_bucket(bucket_name)  # ?


    # List objects with the given prefix, filtering out folders.
    blob_list = list(bucket.list_blobs(prefix=prefix))
    print('Output files:')
    for blob in blob_list:
        print(blob.name)
```

```
    # Process the first output file from GCS.
    # Since we specified batch_size=2, the first response contains
    # the first two pages of the input file.
    output = blob_list[0]

    json_string = output.download_as_string()
    response = json.loads(json_string)

    # The actual response for the first page of the input file.
    first_page_response = response["responses"][0]
    annotation = first_page_response["fullTextAnnotation"]

    print('Full text:\n')
    print(annotation["text"])

    return annotation["text"]


def speech_syntesis(pdf_text):
    # Instantiates a client
    client = texttospeech.TextToSpeechClient()

    # Set the text input to be synthesized
    synthesis_input = texttospeech.SynthesisInput({"text": pdf_text})

    # Build the voice request, select the language code ("en-US") and the ssml
    # voice gender ("neutral")
    voice = texttospeech.VoiceSelectionParams(
        {"language_code": 'en-US', "name": "en-US-Neural2-C", "ssml_gender": 'FEMALE'})
    # (       language_code="en-US", ssml_gender=texttospeech.SsmlVoiceGender.NEUTRAL    )

    # Select the type of audio file you want returned
    audio_config = texttospeech.AudioConfig({"audio_encoding": 'MP3'})

    # Perform the text-to-speech request on the text input with the selected
    # voice parameters and audio file type
    response = client.synthesize_speech(input=synthesis_input, voice=voice, audio_config=audio_config)

    # The response's audio_content is binary.
    with open("output.mp3", "wb") as out:
        # Write the response to the output file.
        out.write(response.audio_content)
        print('Audio content written to file "output.mp3"')


upload_cs_file("angels2.pdf", "angels2.pdf")
pdf_text = async_detect_document("angels2.pdf")
speech_syntesis(pdf_text)
```