GCP Speech-To-Text (STT) API

Speech-To-Text is a Google cloud API that can be used to transcribe audio sent to the API to receive a text transcription. This Google Cloud’s API can be used via two methods – REST API or Cloud STT Console.

# REST API

Focusing on REST API to send audio to STT, the below information depicts how to setup the environment, specify the usage of the service, and provide information on possible responses.

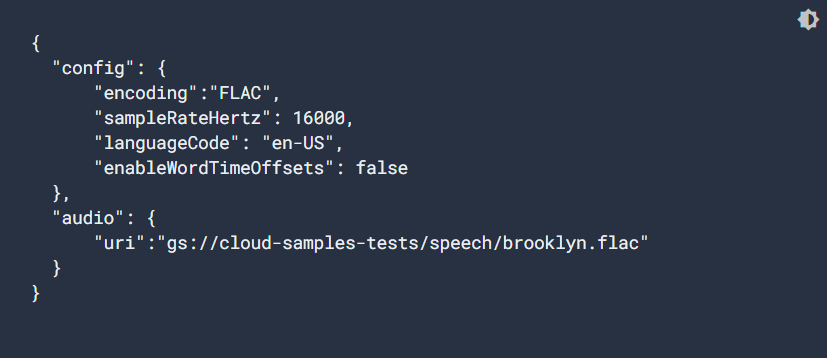
## Setup:

This part of the STT solution is used to setup the environment and API service in Google Cloud to receive requests and respond appropriately:

* Create a GCP project in your service account and enable billing for the project.
* Enable Speech-to-Text function on the project.
* Create a service account credential key and set Authentication Environment Variable.
* Create a Cloud Storage bucket to store Audio data (optional)
* Enable ‘Data Logging’ to record audio data sent to STT for improving the STT model. (optional)

## Usage:

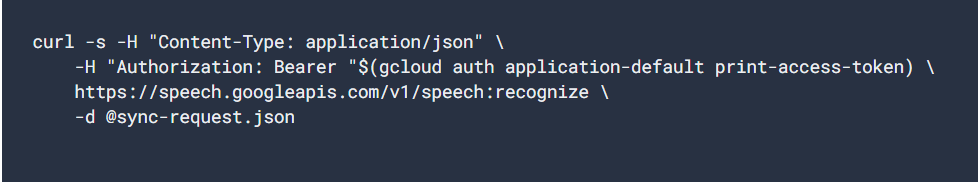
The REST request to STT is done using the REST interface and the curl command. Sending a transcription REST request to the STT API uses a JSON request file like the one below:



The JSON file above has the audio file in a public google cloud storage bucket specified as the ‘uri’ parameter.

The above specifies the details of the passed audio in the config part of the JSON file, and the audio part specifies the location of the audio itself.

Sequel to the above, a curl command is used to make a **speech:recognize** request to the STT API, passing the filename of the JSON request setup above. Given that the above is named **‘sync-request.json’**:



* The **“gcloud…token”** part of the above code is used to obtain authentication token to the cloud storage.
* The -d option is used to specify the audio data location.

Typical response to the above request will be:



This shows the transcribes text and the confidence in the response, which ranges from 0 – 1.

# Breaking Captcha Project Application

## Speech Request

There are 3 different types of speech requests that can be made to the STT API:

* Synchronous Recognition (REST and gRPC): Audio data length is limited to 1 minute per request. It performs recognition on the data and returns results after it has been processed. **Recommended for the project**
* Asynchronous Recognition (REST and gRPC): sends audio data for STT API and initiates a Long Running Operation.
* Streaming Recognition (gRPC only)

The chosen method above is **blocking**, meaning it can only process one request at a time. *PS: Poor audio quality can result in longer processing time.*

Below is an image of a Synchronous Speech Recognition Request:



The recognition request (of type **RecognitionConfig**) contains numerous sub-fields that can be used to streamline the response from the API.

## RecognitionConfig Object

### Recognition Requests Sub-Fields:

* encoding (required): this specifies the encoding scheme of the supplied audio (of type **AudioEncoding**). In the case of the breaking Captcha project, the value of this will be WAV, or based on the audio captcha dataset used.
* sampleRateHertz (required): specified the sample rate of the audio supplied. For FLAC and WAV, this can be optional if the sample rate is specified in the audio file header. Else, **a rate of 16000 is recommended** as values below this may affect the quality of the audio and the response, and value above this does not have any significant effect on the API response.
* languageCode (required): this must be a BCP-47 identifier. It contains the language + region/locale of the supplied audio.
* maxAlternatives (optional, defaults to 1): this is the number of alternatives that should be present in the API response. By default, the best alternatives will be the order of the alternative response, and an alternative will only be included if it is deemed to be of sufficient quality by the recognizer. **For this project, the default value is fine**.
* timestamps (optional): this can be used to specify the beginning and end of the recognized word in the processed audio file. Can be set to **true** or **false**. To set this in the RecognitionConfig object, set the **enableWordTimeOffsets** parameter to **true**.
* model: specifies the ML model to use

Other subfields are available, but only the above listed are suitable for this project.

### Audio Sub-fields:

In the audio part of the config file, audio file can be submitted using two parameters:

* content: passed directly in the request. The audio must be compatible with JSON serialization and be Base64-encoded. Hence, the audio file in binary need to be converted to base64 encoding:

The following commands are used for this:

Windows:



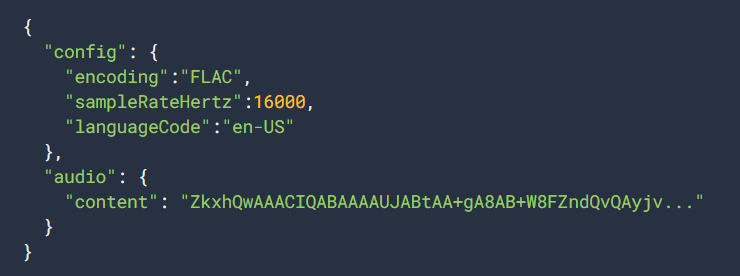
MacOS:



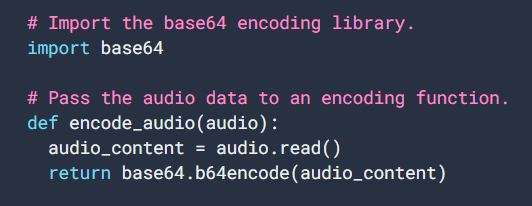
Linux:



And then passed into the **content** parameter:



A python program can also be used to encode the audio file to base64:



* uri: contain URI pointing to the audio content. Hence, passing the audio reference. This requires access to read the Google Cloud Storage files. Access permission could include:
  + publicly readable
  + readable by your service account, if using service account authorization **(this will be used for this project)**.
  + Readable by a user account, if using 3-legged oAuth for user authorization.

## Model Selection

STT has a list of pre-trained machine learning models for audio recognition of different types and sources. In the RecognitionConfig object, the source of the audio file can be specified to help STT select the right ML model to recognize the audio.

You can also specify the model to be used for the supplied audio. The **model** field can be added to the ReconitionConfig object for your request.

For the Breaking CAPTCHA project, the below models are suitable to use:

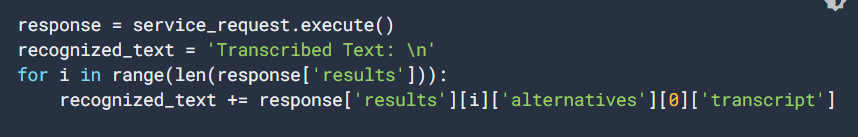
* ASR: Command and Search – best for transcribing shorter audio clips
* Latest Short – used for short utterances that are of a short length
* ASR: Default – used if you can’t categorize the audio to a type

## Speech-to-Text API responses

The response from synchronous STT API request takes time dependent on the length of the audio supplied. The response is in the JSON format and each **alternative** has the **confidence** in and **transcript** for the audio file.

## Handling Transcriptions

Below code concatenates the transcriptions of all responses together. It takes the first alternative (0th) of each response.

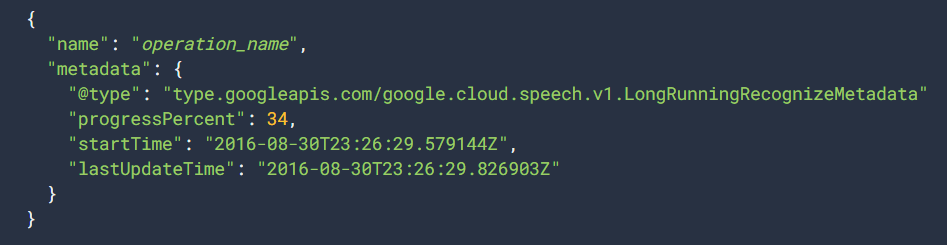


## Asynchronous Requests and Responses

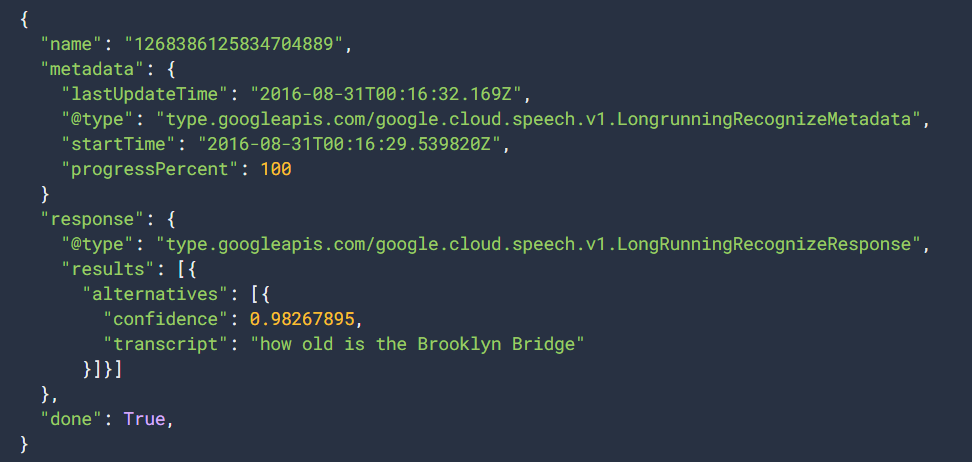
Similar to the synchronous speech request. This speech request method will start a **LongRunningRecognize** method of type **operation**, return the operation to the callee, and then the complete response is sent after the complete processing of the request.

*PS: it can take audio data duration of upto 480 minutes.*

Below is a typical operation response with progress percent of 34:



Below is the full response with progress percent of 100:



# Conclusion

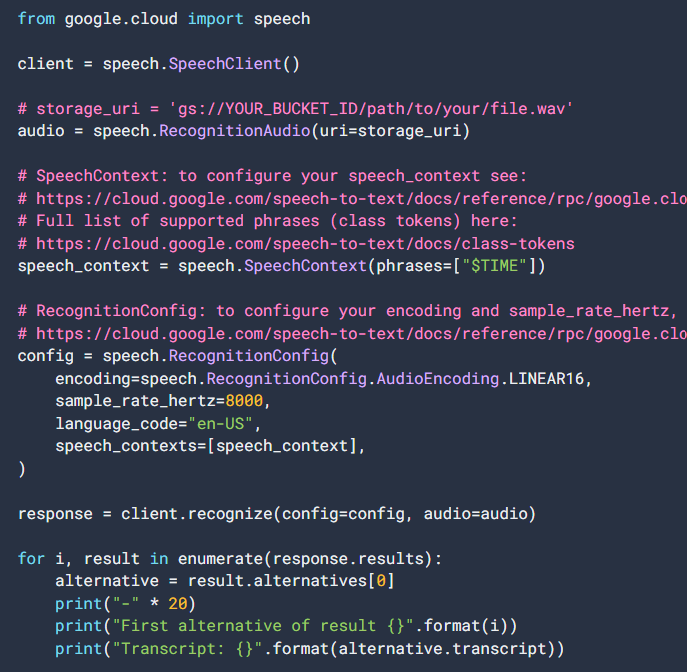
Using the GCP Speech-To-Text API provides a lot of flexibility and options on the range of audio file that can be transcribed.

This is only the first step of the using GCP for audio classification. The below list is an updated proposed architecture from Google Cloud to setup an audio categorization solution is as stated below:

* + Upload Audio File
  + Trigger Cloud Function (covered in this report)
  + Call the Speech-To-Text API (covered in this report)
  + Get Speech-to-Text API results

The above list has been modified to fit the Breaking CAPTCHA project to obtain only the transcribed audio result.

Lastly, below is a python code that brings it all together, using GCPs Speech Adaptation function to improve the accuracy of the result.



# Reference

* Google Cloud (2022-08-31), [Speech-to-text basics](https://cloud.google.com/speech-to-text/docs/basics), accessed on 2nd September 2022.
* Google Cloud (2022-08-31), [Transcribe speech to text by using the API](https://cloud.google.com/speech-to-text/docs/transcribe-api?hl=en_US), accessed on 2nd September 2022.
* Google Cloud (2022-08-31), [Send a recognition request with speech adaptation](https://cloud.google.com/speech-to-text/docs/context-strength#speech-context-strength-python), accessed on 2nd September 2022.
* Cloud Architecture Center (2019-11-08), [Categorizing audio content using machine learning](https://cloud.google.com/architecture/categorizing-audio-files-using-ml), accessed on 2nd September 2022