COS-X BUSINESS SOLUTIONS

Internship Project

Video/Audio Transcription (speech to text)

Setup

This project involves transcripting audio and video data for hiring platform. Important libraries used are **google.cloud speech**, **gcloud storage** and **audioread**. This project uses **enhanced model** of google speech to text api to get the most accurate results. It has 2 jupyter notebooks working (for reference and extensions), final (for usage). So before using it one should create a project, service account and bucket and fill up the constant parameters in the final_notebook.

Description and Specifications

The data_trancriptor function takes input file name as first argument and use already defined path to directory to fetch the file. The second argument is google language code of the audio file, if not familiar, one can look here. Finally, the third argument is whether it involves multiple channels or not (different speakers). The constant parameter naming is self explanatory and is not described here.

Conditions on input file:

1. Input formats are FLAC(lossless) or wav only
2. Preferred bitrate > 16000 Hz
3. For multiple speakers each person's voice should be in different channels
4. For single speaker mono audio channel is preferred (not a compulsion)

Functioning

The code flow goes like, firstly the specs of input file is extracted i.e. its bitrate, duration, channels etc., giving in these numerics to speech to text api increases the accuracy and

makes cloud functioning faster. Based on duration, the audio is classified as lengthy (duration > 1 minute). If its lengthy it has to be uploaded to cloud for processing. The output of processing is confidence and the text. We will be having a limited storage space so file uploaded should be deleted after processing. Any output with confidence less than 70% (lot of noise and disturbance) are not returned to avoid meaningless transcripts. Any input error is referred as “Bad Input”.

Extensions

Google speech to text API support is a domain where development is still going on, so some of the features are in beta testing stage. These features are not used in final notebook to avoid mistakes in the core of the project but can be implemented once they are publicly released. Two of these features are explored and code for the same is available as add ons in working notebook. They are

1. Automatic language detection - detects the language from the given domain of a maximum of 4 languages.
2. Speaker diarization - detects speaker even in the mono audio channel based on speaker’s natural frequency and other parameters.

Arabhi Subhash

CS17B005

IIT Madras