ASSIGNMENT 2 SUBTASK 1

COP290

Jahnabi Roy

14^{th} March, 2024

1 Introduction

This assignment aims to parse text from different types of media like audio (`.wav`), video (`.mp4`) and text document (`.pdf`). Python and its libraries were used to extract the text from these multimedia files. For the assignment only free APIs and libraries were used with unlimited usage.

1.1 Parse text from Video

Implementation Details : For this part, the audio is first extracted from the video clip in `.wav` form. Library used for this purpose is `Moviepy`. Once the audio file is generated, we use the code and libraries of the second part of the subtask. This part was done smoothly by `Moviepy` library.

Issues Faced : Other libraries like `SpeechRecognition` were not as smooth as using `Moviepy`.

1.2 Parse text from Audio

Implementation Details : This part was implemented with the library SpeechRecognition using the API - Google Web Speech-to-Text Service.

Issues Faced : The main problem that came through during the implementation was that Google Web Speech-to-Text had a limit of processing : about 120 seconds (gives an error of Broken Pipe). So, any video length longer than that was not possible to be processed by the Service. The text transcription is not very accurate and Other APIs like Google Cloud Speech to Text, IBM Watson Speech to Text, Whisper_API were all paid services for unlimited use so were not opted for this subtask.
Another biggest issue is the amount of time taken for processing data. A 15 minute video could take upto 30-40 minutes for processing. Better APIs could be used for this but the ones I came across were all paid and had limited usage only. `CMU Sphinx` was an interesting one I had come across but could not implement it.

Solution Implemented : To solve this, the audio was divided into chunks of maximum interval length of 115 seconds (to minimise space used and process as much as it could take in one go). Default parameters for `pydub.silence` were used as 0.5s for `min_silence_threshold` and -16 as the `silence_threshold` below which the audio is considered to be silent. These chunks were then stored in a separated folder which was then processed in a separate iteration over the chunks and concatenated in a string. The `UnknownValueErrors` were handled were adjusting the text content to `[incoherent]`. The `RequestError` is also handled with an appropriate error message.

1.3 Parse text from PDF

Implementation Details : Here, `PyMuPDF` was used for implementation. The pros of this choice is that it does not lose data and it also removes unnecessary spacing from the extracted text thus giving an overall clean look. The cons is that it does not preserve tabular

data; but this is taken care by the fact that the structure in which the data is printed with every column in a row printed in a new line and this structure is preserved throughout the text extracted.

Issues Faced : On research, several libraries came up for this functionality like `PyPDF2`, `Tika` and `Textract` but they were not as efficient as `PyMuPDF2` was. `Textract` was significantly closer to preservation of the structure of text, but inconsistent and incoherent textual data was more frequent than necessary and thus was opted out of it. Tabulapy was also a good library but was specific to tabular data only.

2 Submission Details

The submission details include :

- `main.py`
- `extract_text_from_video.py`
- `extract_text_from_audio.py`
- `extract_text_from_PDF.py`
- `write_to_file.py`
- `mp3_to_wav.py` (unused as of now)
- `COP290 ASSIGNMENT 2 REPORT.pdf`
- Link for resources - **Resources Tested**