Assignment 2 – Subtask 1

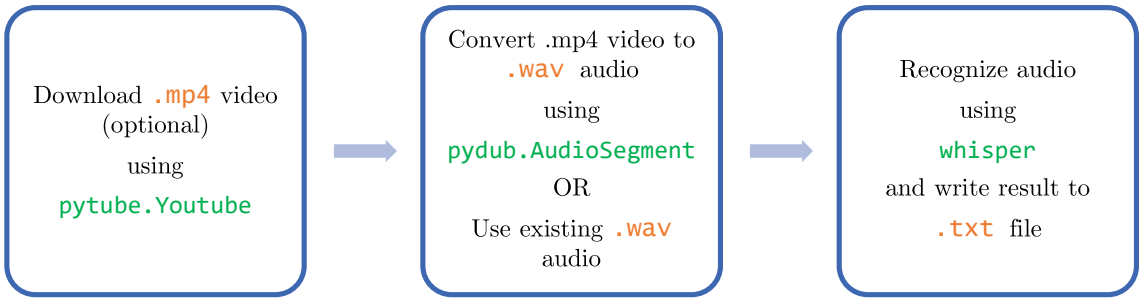
Name: AASTHA A K VERMA
Entry Number: 2022CS11607
Course: COP290

1 Video/Audio Processing

Our objectives are attained using Python because of the ease of use and availability of a variety of libraries for different functionalities.

1.1 Workflow

The following workflow is followed:



1.2 Experiments and Challenges

In the process to find a free Python library suited for speech recognition, `speechrecognition`, `speechbrain` and `whisper` libraries were tested.

- `speechrecognition` supports several APIs but most of them are paid or with limited free usage. The `recognize_google` function calls the old Google Speech API which is no longer maintained, and the result was not good. For some tests, parts of the actual speech were missing in the reported text. The API also failed with the introduction of some high-tempo music beats and reported the speech as "unintelligible", raising an `UnknownValueError`.
- `speechbrain` was difficult to implement because of the requirement for several dependencies like `symlink`, and some technicalities like administrator privileges. It can probably be explored along with the HuggingFace-hosted models for more complex applications, but it seemed to be overkill for our purposes.
- OpenAI's `whisper` is open-source and produces excellent, almost correct speech-to-text. It is also trained to recognize sounds like music and humming. It supports multiple languages, with automatic detection. It supports variations of models, "medium" has been selected here.

Size	Parameters	English-only model	Multilingual model	Required VRAM	Relative speed
tiny	39 M	`tiny.en`	`tiny`	~1 GB	~32x
base	74 M	`base.en`	`base`	~1 GB	~16x
small	244 M	`small.en`	`small`	~2 GB	~6x
medium	769 M	`medium.en`	`medium`	~5 GB	~2x
large	1550 M	N/A	`large`	~10 GB	1x

The `.en` models for English-only applications tend to perform better, especially for the `tiny.en` and `base.en` models. We observed that the difference becomes less significant for the `small.en` and `medium.en` models.

Source: https://github.com/openai/whisper/tree/main

Hence, whisper generated the best results and was finally used.

1.3 Testing

The sample videos are converted to audio to test both functionalities simultaneously. The samples are selected to test the following criteria:

- Media duration should not affect the result.
- Proper recognition for speech that is not conversational, e.g., songs with heavy instrumental influence.
- Media possibly not being in English.
- Efficient filtering of possible background noise/voices.

In this light, the selected samples consist of normal conversational-style videos, narratives, clearly-enunciated songs, some poorly-enunciated songs, some instrumentally heavy songs, and a vernacular clip in Marathi.

Marking `verbose=True` in the `transcribe()` function displays the output of the sliding window at the terminal.

```
Detecting language using up to the first 30 seconds. Use `--language` to specify the language
Detected language: English
[00:00.000 --> 00:16.000]  Is this the real life? Is this just fantasy? Caught in a landslide, no escape from re
ality.
[00:16.000 --> 00:26.000]  Open your eyes, look up to the skies and see.
[00:26.000 --> 00:32.000]  I'm just a poor boy, I need no sympathy.
[00:32.000 --> 00:40.000]  Because I'm easy come, easy go, little high, little low.
[00:40.000 --> 00:49.000]  Any way the wind blows doesn't really matter to me.
[00:49.000 --> 00:53.000]  To me.
[00:53.000 --> 01:10.000]  Mama just killed a man, put a gun against his head, pulled my trigger, now he's dead.
[01:10.000 --> 01:24.000]  Mama, life had just begun, but now I've gone and thrown it all away.
[01:24.000 --> 01:39.000]  Mama, who did mean to make you cry, if I'm not back again this time tomorrow.
[01:39.000 --> 01:56.000]  Carry on, carry on, cause if nothing really matters.
```

The Marathi video's language is correctly recognised.

```
Detecting language using up to the first 30 seconds. Use `--language` to specify the language
Detected language: Marathi
[00:00.000 --> 00:22.120]  music
[00:22.120 --> 00:30.120]  कुमल कया कि मोहे माया पुनव चापद नले
[00:30.120 --> 00:34.120]  सोपया कस जले स्पयक भोले
[00:34.120 --> 00:38.120]  रतन प्रभास नुलयले
[00:38.120 --> 00:42.120]  नटली तटली जशी मटली
[00:42.120 --> 00:45.120]  सापदणी रंग महले
[00:45.120 --> 00:49.120]  मी यकॉन भोली पअन ती जली
[00:49.120 --> 00:52.120]  इंद्र सबा भवतली
[00:52.120 --> 00:55.120]  अंसरा आली
```

2 PDF Processing

Our objectives are attained using Python because of the ease of use and availability of a variety of libraries for different functionalities.

2.1 Experiments and Challenges

This part was relatively less challenging because of the availability of a variety of libraries. So multiple libraries were implemented with the differences in their result mostly being the decision to put in newlines. However, benchmarks with large datasets present the following data:

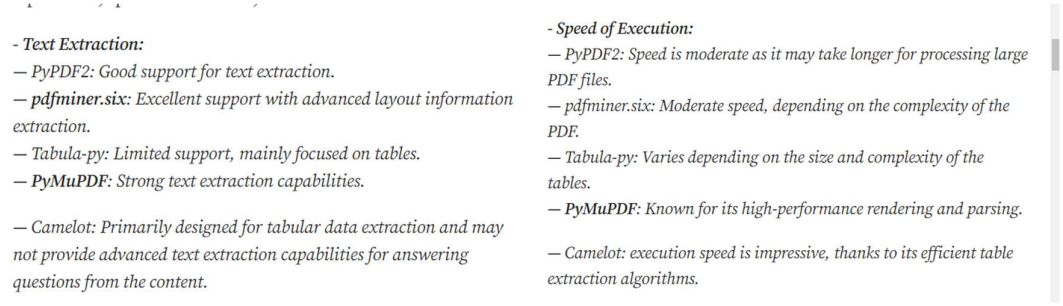
Text Extraction Speed

#	Library	Average	1	2	3	4	5	6	7	8	9	10	11	12
1	PyMuPDF	0.11s	0.46s	0.26s	0.16s	0.18s	0.04s	0.09s	0.04s	0.05s	0.03s	0.05s	0.03s	0.04s
2	Tika	0.22s	1.10s	0.43s	0.38s	0.31s	0.08s	0.15s	0.11s	0.09s	0.07s	0.10s	0.07s	0.08s
3	pdftotext	0.26s	0.76s	0.92s	0.26s	0.76s	0.07s	0.21s	0.16s	0.13s	0.06s	0.11s	0.06s	0.08s
4	pypdf	3.03s	25.86s	4.35s	6.94s	1.95s	0.25s	0.80s	0.32s	0.43s	0.31s	0.54s	0.18s	0.27s
5	pdfminer.six	7.47s	50.25s	19.13s	11.63s	7.21s	1.71s	3.26s	1.46s	1.82s	1.19s	2.25s	1.60s	1.32s
6	pdfplumber	8.46s	62.21s	14.49s	14.80s	7.79s	2.11s	3.83s	1.78s	1.87s	1.49s	2.43s	2.02s	1.47s

Text Extraction Quality

#	Library	Average	1	2	3	4	5	6	7	8	9	10	11	12
1	Tika	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
2	PyMuPDF	96%	97%	96%	97%	95%	98%	97%	93%	95%	99%	92%	98%	94%
3	pdftotext	91%	95%	93%	92%	92%	92%	96%	90%	93%	97%	78%	94%	92%
4	pdfminer.six	89%	95%	78%	89%	90%	86%	94%	90%	90%	92%	87%	94%	83%
5	pypdf	84%	89%	79%	89%	91%	95%	92%	90%	89%	97%	90%	97%	91%
6	pdfplumber	73%	93%	83%	61%	94%	61%	94%	58%	86%	57%	55%	67%	57%

Source: https://github.com/py-pdf/benchmarks/tree/main/read



- Text Extraction:
— PyPDF2: Good support for text extraction.
*— **pdfminer.six**: Excellent support with advanced layout information extraction.*
— Tabula-py: Limited support, mainly focused on tables.
*— **PyMuPDF**: Strong text extraction capabilities.*

— Camelot: Primarily designed for tabular data extraction and may not provide advanced text extraction capabilities for answering questions from the content.

- Speed of Execution:
— PyPDF2: Speed is moderate as it may take longer for processing large PDF files.
— pdfminer.six: Moderate speed, depending on the complexity of the PDF.
— Tabula-py: Varies depending on the size and complexity of the tables.
*— **PyMuPDF**: Known for its high-performance rendering and parsing.*

— Camelot: execution speed is impressive, thanks to its efficient table extraction algorithms.

> Both `pdfminer.six` and `PyMuPDF` are reliable options for preserving the original formatting during PDF text extraction. The choice between the two will depend on your specific requirements, preferences, and the overall features and functionalities provided by each library.

Source: https://pradeepundefned.medium.com/a-comparison-of-python-libraries-for-pdf-data-extraction-for-text-images-and-tables-c75e5dbcfef8

Finally, the following libraries were implemented:

- `pymupdf`
- `pypdfium`
- `pypdf2`
- `pdfminer.six`
- `tika` (Note that Tika is written in Java so we need a Java runtime installed.)
- `tabula` (for an exclusively table-containing pdf)

2.2 Testing

The pdfs were selected to test the following parameters:

- Text extraction quality
- Format preservation
- Text extraction from images in pdf
- Treatment of tables

The sample pdf set consisted of a normal text pdf, a pdf with only images, a pdf with blank underscored lines, pdfs with fancy formatting of text, and a pdf with a table. Text extraction is mostly similar for most libraries, with poor recognition of text from images. `tabula` gives excellent table comprehension with data being possibly manipulated using Pandas DataFrames and being written to CSV format.

3 Results

https://drive.google.com/drive/folders/1YK2OjW59dYu2HSRBS0gaOGVsOYJWRrU0?usp=sharing

Follow the above link to view the results of the text-extraction processes. Note that the audio samples are audio extractions of the videos themselves, found in the `gen_audios` folder, other samples are not taken.