COP 290:DESIGN PRACTICES

ASSIGNMENT-2

SUBTASK-1: EXTRACT TEXT FROM VIDEO,AUDIO,PDF

SUBMITTED BY

NAME: DEVARAPALLI AKHIL

ENTRY NUMBER:2022CS51655

What we do in Subtask 1:

This subtask aims to create a program that extracts text from different multimedia sources efficiently. It involves the following:

1. Parsing Text from Video: Developing a method to extract transcripts from video files.
2. Parsing Text from Audio: Transcribing text from audio files.
3. Parsing Text from PDF: Extracting text content from PDF documents, including plain text or structured elements like paragraphs, headings, or lists.

IMPLEMENTATION AND OPTIMIZATIONS:

For this assignment, I utilized Python as the programming language due to its rich ecosystem of libraries and tools suitable for multimedia processing tasks. The following libraries were used:

1. **OpenCV:** Utilized for handling video files, extracting frames, and retrieving audio streams.
2. **MoviePy:** Employed to work with video files, extracting audio from them efficiently.
3. **Whisper:** Utilized for transcribing audio to text. This library offers pre-trained models for speech recognition tasks.
4. **PyPDF2:** Used for parsing PDF documents and extracting text content.
5. **PyDub:** Utilized for audio manipulation, particularly for converting audio files to WAV format.

To enhance performance, I implemented the following techniques:

- ***Temporary File Handling:*** Created temporary files for intermediate audio and video conversions to ensure efficient processing and resource management.
- ***Conditional Operations:*** Incorporated conditional checks to handle different input file formats seamlessly, allowing the program to support various video and audio file types without modification.

ISSUES FACED:

During the implementation process, several challenges were encountered:

- **Format Compatibility**: Ensuring compatibility with various video, audio, and PDF file formats posed a challenge. I addressed this by implementing conditional checks and conversion functions to handle different formats.
- **Temporary File Management**: Managing temporary files and ensuring their proper deletion after use required careful handling to prevent clutter and potential issues with file access permissions.
- **Transcription Accuracy**: Achieving accurate transcription from audio files posed a challenge, particularly with noisy or low-quality recordings. I tried Google speech recognition and CMU Sphinx, which did not give satisfactory results. Most of the good speech recognition libraries have limits or pay. While Whisper provided satisfactory results in most cases, some audio files required additional preprocessing or manual correction to improve transcription accuracy.

HOW TO USE IT:

We need to have the following things in your system to run the program.
It requires the command-line tool ffmpeg to be installed on your system, which is available from most package managers:

```
# on Ubuntu or Debian
sudo apt update && sudo apt install ffmpeg

# on Arch Linux
sudo pacman -S ffmpeg

# on MacOS using Homebrew (https://brew.sh/)
brew install ffmpeg

# on Windows using Chocolatey (https://chocolatey.org/)
choco install ffmpeg

# on Windows using Scoop (https://scoop.sh/)
scoop install ffmpeg
```

After installing ffmpeg you can now simply run the make command in the current directory of the terminal and follow according to your requirements .

You will be able to generate a text file containing the extracted text.

TEST EXAMPLES:

I conducted testing using various examples, including those provided on Piazza and additional test cases. The program successfully extracted text from a diverse range of multimedia sources, including videos, audios, and PDF documents.

The test examples, along with the output text files, are provided in the following Google Drive folder: Test Examples .