Project 1

Welcome to **Vido!**

Vido! is an innovative video processing startup focused on extracting valuable insights from video content for clients. Using advanced natural language processing (NLP) and computer vision techniques, **Vido!** analyses sentiments and emotions in video content. This analysis is crucial for businesses seeking to understand their audience, improve customer experiences, and make data-driven decisions.

On your first day at **Vido** as a data engineer, you were assigned a critical project: determining the optimal strategy for downloading and analysing videos, including audio, translations, sentiments, emotions, etc. Videos uploaded by users on YouTube need to be analysed, and results should be extracted and shared with customers worldwide.

You have been paired to work with a client from Madrid, so your task is to provide Video translations in Spanish!

Thanks to Python, you have the opportunity to showcase your skills and automate all your tasks, while parallel processing concepts can help you speed up your analysis tasks, contributing to the success of our company!

Project:

- Objective: Develop a Python application using multiprocessing or threading concepts to download and analyse YouTube videos and compare serial versus parallel execution.
- Timeline: You have four weeks to complete all the tasks and document your solutions.

Tasks:

1. **Retrieve 10-15 random video URLs from YouTube.**
    - Save the data in a text file called `video_urls.txt`.
    - Each URL should be stored on a separate line.
2. **Reading Input URLs.**
    - Assuming you have the text file named `video_urls.txt` containing the URLs of YouTube videos.
    - Load the file in Python and extract it using your preferred data structure.

    [2.5 marks]
3. **Downloading Videos.**
    - Implement functionality to download YouTube videos.
    - Use parallel programming concepts (multiprocessing or threading) to handle downloads. Your decision will determine the best strategy.
    - For testing reasons, ensure the script can download up to `5` videos simultaneously to avoid YouTube blocks.

    [12.5 marks]
4. **Comparing Serial vs Parallel Execution.**

- Discuss the time and space complexity of your functions with explanations.
- Implement both serial and parallel downloading of videos.
- Measure and compare the physical time taken for serial vs parallel execution. Feel free to compare and contrast multiple methods until you can find the best in terms of time.

[15 marks]

5. **Logging Downloads.**
   - After downloading each video, create a logger to record which video was downloaded by which process or thread.
   - Save the log entries to the same file, e.g., `download_log.txt`.

   [15 marks]

6. **Video Analysis.**
   - You must organise tasks for each video and create a strategy for organising your scripts.
   - After a video is downloaded, perform the following tasks:
     - Extract audio from video using an `extract_audio_from_video` function.
     - Then, transcribe audio to text using a `transcribe_audio_to_text` function.
     - Perform the sentiment analysis on a video's content, extracting its `polarity` and `sensitivity`.
     - Translate the text in Spanish or any other language you prefer.
     - Extract the emotions of a text.
   - Each output task should be stored in a separate folder designated for each video. Feel free to organise your folder structure as you prefer.
   - You can use any library, including `moviepy` for loading video and `speech_recognition` or `textblob` for sentiment analysis.
   - Skeleton programs are provided below, but feel free to use your preferred libraries.

   [15 marks]

7. **Document your solutions in a technical report.**
   - **This project is open to your interpretations, and you can use any programming strategy that fits better to this project, including processes, threads, or asynchronous programming.**
   - Discuss your strategy for developing this solution.
   - A 1-2 page discussion (600-1200 words) will suffice, but you can include more if necessary.
   - There is no penalty for submitting more or less text.
   - There is no need to include references if you did not use existing resources. You don't have to reference the supporting material contained in this document. If you use external resources, e.g. code from online sources, then provide a reference in the report.
   - Finally, submit a zipped file of all your scripts.
   - You can run your scripts in any Python IDE you prefer, including Visual Studio Code, Colab, etc.

[20 marks]

8. Submit quality scripts following coding practices such as functional or object-oriented programming, separate Python files to break functionality, use of `main`, and other standard practices.
    1. Organise your scripts to be easy to follow and readable, with comments as needed.

[20 marks]

Supporting material

The following supporting scripts can help you kickstart your project. It is advised that you spend some time familiarising yourself with the libraries and scripts. However, you can use any library you prefer and not necessirlty the ones provided below.

1. Install the required libraries using the following command.

```
pip3 install pytube moviepy speechrecognition spacy nltk NRCLex Path
```

> **Tip**
>
> Depending on your Python installation, you might need to use `pip3` instead of `pip`. You might need to install more libraries.

2. Explore the following script that downloads a video in your local folder.

```python
from pytube import YouTube

url = "https://www.youtube.com/watch?v=jNQXAC9IVRw"
yt = YouTube(url)
stream = yt.streams.get_highest_resolution()
print(f"Downloading video: {yt.title}")
stream.download()
print(f"Download completed: {yt.title}")
```

> The output is as follows:

```
Downloading video: Me at the zoo
Download completed: Me at the zoo
```

> **Note**
>
> The files are stored as `mp4` files; to create a folder, you can use the following script.
>
> ```python
> from pathlib import Path
> Path("output").mkdir(parents=True, exist_ok=True)
> ```

3. The following script extracts the audio file `wav` from a video clip `mp4`.

```python
import moviepy.editor as mp

video = mp.VideoFileClip("Me at the zoo.mp4")
video.audio.write_audiofile("Me at the zoo.wav")
```

> ⓘ **Note**
>
> The script extracts the audio files in `wav` format.

4. The following script extracts the text from the audio file.

```python
import speech_recognition as sr
recognizer = sr.Recognizer()
with sr.AudioFile("Me at the zoo.wav") as source:
    audio = recognizer.record(source)
text = recognizer.recognize_google(audio)
print(text)
```

> ⓘ **Note**
>
> The scripts extract the audio as a string. The following script saves the string to a text file.
>
> ```python
> file_path = 'Me at the zoo.txt'
> with open(file_path, 'w') as file:
>  file.write(text)
> print(f'Text has been written to {file_path}')
> ```

5. The following script extracts the sentiment from a text file.

```python
from textblob import TextBlob

text = "all right so here we are one of the elephants and cool thing about these guys
today is that they have really really really really really really long trunks and that's
that's cool"

blob = TextBlob(text)
print(blob.sentiment)
print(blob.sentiment.polarity)
print(blob.sentiment.subjectivity)
```

> ⓘ **Note**
>
> - Polarity measures the text's positive or negative tone. It ranges from `-1.0` (very negative) to `1.0` (very positive). A value of `0` indicates a neutral sentiment.
> - Subjectivity measures how subjective or objective the text is. It ranges from `0.0` (very objective) to `1.0` (very subjective). A subjective text contains personal opinions, emotions, or judgments, whereas an objective text is based on factual information.

7. Extract the video subtitles in Spanish - or any other language you prefer.

```python
from textblob import TextBlob

text = "all right so here we are one of the elephants and cool thing about these guys
today is that they have really really really really really really long trunks and that's
that's cool"

blob = TextBlob(text)
blob_translated = blob.translate(from_lang='en', to='es')
print(blob_translated)
```

> **Note**
>
> The output will be stored in the `blob_translated` in Spanish.

6. Extract the emotions of the text using the `spacy`, `nltk` and the `NRCLex` libraries.

```python
import spacy,nltk
from nrclex import NRCLex
nlp = spacy.load('en_core_web_sm')
nltk.download('punkt')

text = "all right so here we are one of the elephants and cool thing about these guys
today is that they have really really really really really really long trunks and that's
that's cool"

doc = nlp(text)

full_text = ' '.join([sent.text for sent in doc.sents])

emotion = NRCLex(text)

print("Detected Emotions and Frequencies:")
print(emotion.affect_frequencies)
```

> **Note**
>
> The output will be the video emotions in a dictionary, such as:
>
> ```
> Detected Emotions and Frequencies:
> {'fear': 0.0, 'anger': 0.0, 'anticip': 0.0, 'trust': 0.0, 'surprise': 0.0,
> 'positive': 0.6666666666666666, 'negative': 0.0, 'sadness': 0.0, 'disgust': 0.0,
> 'joy': 0.0, 'anticipation': 0.3333333333333333}
> ```