Name: Viplavi Wade

Student Id: 13922741

Subject: Big Data Analytics

Course: MSC Advanced Computing

Date: June 24, 2024

# Big Data Analytics: CineSense Project

GitHub Repo Link: https://github.com/ViplaviWade/Cinesense-Big-Data-Project

## 1.0 Introduction

CineSense is an innovative video-processing startup that extracts valuable insights from social media video content using advanced natural language processing (NLP) and computer vision techniques. This analysis is crucial for businesses seeking to understand their audience, improve customer experiences, and make data-driven decisions.

The project's primary objective is to develop a Python application that efficiently downloads and analyzes YouTube videos using parallel processing techniques. This involves tasks such as downloading videos, extracting audio, transcribing audio to text, performing sentiment and emotion analysis, and translating the text. The project emphasizes the use of multiprocessing, threading, or asynchronous programming to optimize the workflow.

## 2.0 Tools and technologies used:

- Python Programming language (for implementation) Python3 as Python has a wide range of libraries that could be used for the implementation of the project such as using pytube, using speech recognition, spacy, textblob for sentiment analysis etc.
- Git and GitHub: Implementing version control for the project, as it provides visibility and contribution for repository hosting, and tracking the changes from each phase of the project development

## 3.0 Implementation Phases of the Project:

### 3.1 Phase-1

Tasks:

1. Manually retrieve 10-15 random video URLs from YouTube. Save the URLs in a text file called video_urls.txt , where each URL should be stored on a separate line. Consider YouTube videos that are 2-3 minutes in duration.

video_urls.txt

```
https://www.youtube.com/shorts/tRWlMXKNkG0
https://www.youtube.com/shorts/dJWFzVCGUlw
https://www.youtube.com/shorts/NsU98S_LBLM
https://www.youtube.com/shorts/-2n1UnRYgLA
https://www.youtube.com/shorts/RiOqgmmcSvc
https://www.youtube.com/shorts/G-RwDL_b-0I
https://www.youtube.com/shorts/tT1JRa28iL0
https://www.youtube.com/shorts/qXfnoJdWezo
https://www.youtube.com/shorts/WSkVdEoYTFc
https://www.youtube.com/shorts/ANEokonLIPg
```

2. Develop a Python script to read the URLs. Assuming you have the text file named video_urls.txt containing the URLs of YouTube videos, load it in Python and extract the URLs using your preferred data structure.

```python
def read_video_urls(file_path):
    urls = []
    with open(file_path, 'r') as file:
        urls = [line.strip() for line in file.readlines()]
    return urls
```

3. Develop a Python script to download the videos using their URLs. Test your solution by downloading the files serially. Use parallel programming such as 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. You are advised to use threads and semaphores to control the downloads. Compare serial and parallel executions for your video download script. Discuss the complexity of your video download scripts time and space.

   To download the videos in a serial as well as parallel fashion I have created this input choice where the users have the choice to select the downloading execution mechanism where the user can select ‘1’ for serial execution of downloading and ‘2’ for parallel execution of download.

```python
if __name__ == "__main__":
    with open("video_urls.txt", "r") as file:
        video_urls = [line.strip() for line in file.readlines()]

    print("Choose download method:")
    print("1. Serial Download")
    print("2. Parallel Download")

    choice = input("Enter 1 or 2: ")

    if choice == "1":
        download_videos_serially(video_urls)
    elif choice == "2":
        download_videos_parallely(video_urls)
    else:
        print("Invalid choice. Please enter 1 or 2.")
        sys.exit(1)
```

```
PS C:\Users\wadev\OneDrive\Desktop\viplavi\BBK Folder\Semester-3\Big Data Analytics\BDA Project\BDA Project> python main.py
Choose download method:
1. Serial Download
2. Parallel Download
Enter 1 or 2: 1
Serial downloaded: Three rules for a happy life 🍀Beautiful Quotes🍀#shorts
Serial downloaded: You absolutely have to know this expression in English! 🧠🚫
Serial downloaded: Mom Asks Son A Question About Texting #Shorts
Serial downloaded: When You Never Get Any Credit !! #Shorts
Serial downloaded: Dumb Things People Say (Part 25) #Shorts
Serial downloaded: When Your Girlfriend Does Not Finish Her FOOD #shorts
Serial downloaded: 5 Second Trick to get WAY MORE Views on YouTube Shorts
Serial downloaded: Random Fact #youtubeshorts #shorts
Serial downloaded: Naming Random NBA Players Until Someone Messes Up! 🏀 #shorts
Serial downloaded: Success is not a comfortable procedure - Steve Harvey Motivational Speech
Serial download completed in 43.01 seconds
PS C:\Users\wadev\OneDrive\Desktop\viplavi\BBK Folder\Semester-3\Big Data Analytics\BDA Project\BDA Project>
```

```
PS C:\Users\wadev\OneDrive\Desktop\viplavi\BBK Folder\Semester-3\Big Data Analytics\BDA Project\BDA Project> python main.py
Choose download method:
1. Serial Download
2. Parallel Download
Enter 1 or 2: 2
Thread-2 downloaded: You absolutely have to know this expression in English! 🧠🚫
Thread-1 downloaded: Three rules for a happy life 🍀Beautiful Quotes🍀#shorts
Thread-4 downloaded: When You Never Get Any Credit !! #Shorts
Thread-6 downloaded: When Your Girlfriend Does Not Finish Her FOOD #shorts
Thread-5 downloaded: Dumb Things People Say (Part 25) #Shorts
Thread-7 downloaded: 5 Second Trick to get WAY MORE Views on YouTube Shorts
Thread-3 downloaded: Mom Asks Son A Question About Texting #Shorts
Thread-9 downloaded: Naming Random NBA Players Until Someone Messes Up! 🏀 #shorts
Thread-8 downloaded: Random Fact #youtubeshorts #shorts
Thread-10 downloaded: Success is not a comfortable procedure - Steve Harvey Motivational Speech
Parallel download completed in 27.67 seconds
```

Parallel programming is implemented using the multi-threading concept

```python
def download_videos_parallely(video_urls):
    threads = []
    start_time = time.perf_counter()

    for i, url in enumerate(video_urls):
        thread_name = f"Thread-{i+1}"
        thread = threading.Thread(target=download_video, args=(url, thread_name))
        threads.append(thread)
        thread.start()

    for thread in threads:
        thread.join()

    end_time = time.perf_counter()
    parallel_time = end_time - start_time
    print(f"Parallel download completed in {parallel_time:.2f} seconds")
    return parallel_time
```

Ensuring the script can download up to 5 videos simultaneously to avoid YouTube blocks.

```python
# Define a semaphore to limit concurrent downloads to 5
semaphore = threading.Semaphore(5)
```

Time and space complexity of the serial and parallel downloads.

Time Complexity:

- Serial Execution: The time complexity of serial execution is O(n). Each video download operation is independent and sequential, leading to a linear relationship between the number of videos (n) and the total download time. Thus, the total time taken increases linearly with the number of videos.
- Parallel Execution: The time complexity of parallel execution is O(n/k). Assuming the system allows downloading (k) videos concurrently without significant overhead, the time complexity can be approximated to O(n/k), where (k) is the number of concurrent threads. However, the actual speedup will depend on the network bandwidth, system resources, and how well the parallelism can be achieved.

Space Complexity:

- Serial Execution: The time complexity of serial execution is O(1). The space complexity is constant because at any given time, only one video is being processed and downloaded, regardless of the total number of videos.
- Parallel Execution: The time complexity of parallel execution is O(k). The space complexity is proportional to the number of concurrent threads (k). Each thread

consumes memory for its stack and local variables. Additionally, each video being downloaded will consume space simultaneously, but this is generally negligible compared to the overall download directory size.

Factors that might influence the actual working mechanism of the serial and parallel execution

- Network Bandwidth: The actual performance gain from parallel downloads will highly depend on the available network bandwidth. If the bandwidth is a bottleneck, adding more threads might not lead to a proportional decrease in download time.
- Thread Management: Proper management of threads using semaphores and mutexes is crucial to avoid issues like race conditions and excessive resource usage. As in my code, a semaphore is used to limit the number of concurrent downloads to 5, ensuring the system is not overwhelmed.
- Error Handling: Robust error handling is essential in both serial and parallel execution to manage issues like unavailable videos or network failures gracefully.

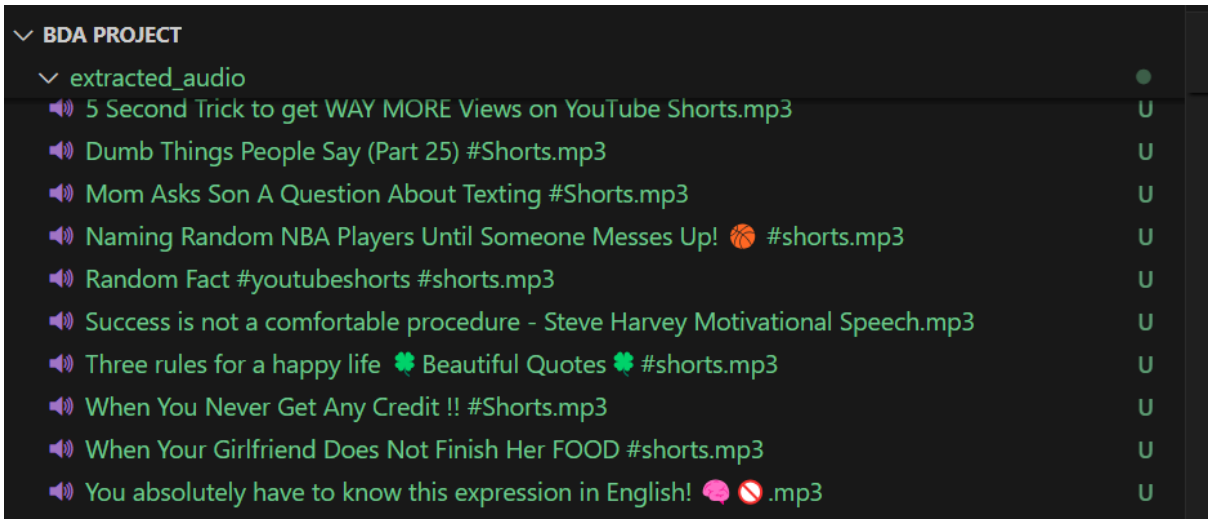
Parallel execution can significantly reduce the total download time compared to serial execution, as demonstrated by the results of my executed code the serial execution took 43.01 seconds, whereas the parallel execution is demonstrated in 27.67 seconds. However, this comes with increased complexity in managing concurrent threads and potential challenges related to network bandwidth and system resources. The choice between serial and parallel execution should consider these factors to optimize performance effectively.

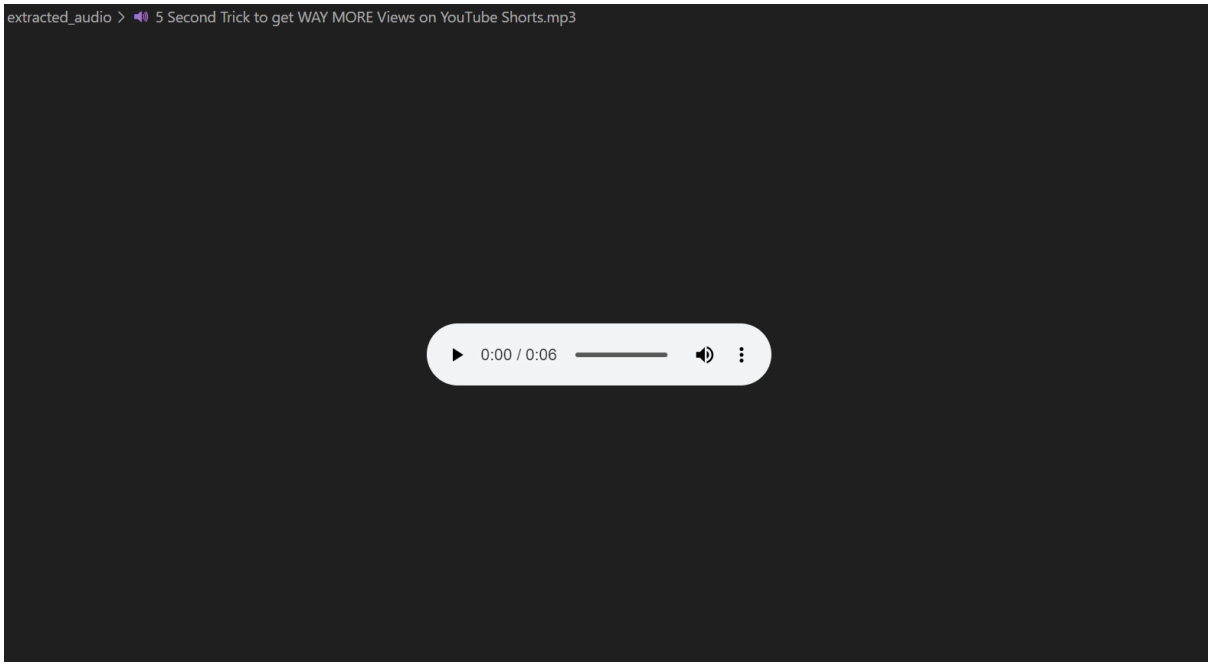
4. Develop a Python script to keep a log for each download. After downloading each video, create a logger to record which video was downloaded by which process or thread. Save the log entries to the same file, e.g., download_log.txt . For this script, you have to use threads and a mutex.


5. Develop Python scripts to perform various video analysis tasks. After downloading a video, perform the following tasks. It is preferable to develop a separate script for each functionality. The five analysis subtasks are as follows.
    I. Extract audio from a video file.
    The code for extracting audio from video file is added in the extract_audio.py file and the extracted audio is saved under folder names ‘extracted_audio’

```python
def extract_audio(video_path, output_dir):
    try:
        video = mp.VideoFileClip(video_path)
        audio_path = os.path.join(output_dir, f"{os.path.basename(video_path).split('.')[0]}.mp3")
        video.audio.write_audiofile(audio_path)
        print(f"Audio extracted and saved to {audio_path}")
    except Exception as e:
        print(f"Failed to extract audio from {video_path}. Reason: {e}")

def main(video_paths):
    start_time = time.perf_counter()

    # Clear the results directory before processing new videos
    clear_results_directory(RESULTS_DIR)

    # Ensure the output directory exists
    os.makedirs(RESULTS_DIR, exist_ok=True)

    with ThreadPoolExecutor(max_workers=5) as executor:
        for video_path in video_paths:
            executor.submit(extract_audio, video_path, RESULTS_DIR)

    end_time = time.perf_counter()
    print(f"Audio extraction completed in {end_time - start_time:.2f} seconds")
```

```
(venv) C:\Users\wadev\OneDrive\Desktop\viplavi\BBK Folder\Semester-3\Big Data Analytics\BDA Project\BDA Project>python extract_audio.py
MoviePy - Writing audio in extracted_audio\Random Fact #youtubeshorts #shorts.mp3
MoviePy - Writing audio in extracted_audio\Dumb Things People Say (Part 25) #Shorts.mp3
MoviePy - Writing audio in extracted_audio\Mom Asks Son A Question About Texting #Shorts.mp3
MoviePy - Writing audio in extracted_audio\Naming Random NBA Players Until Someone Messes Up! 🏀 #shorts.mp3
MoviePy - Writing audio in extracted_audio\5 Second Trick to get WAY MORE Views on YouTube Shorts.mp3
MoviePy - Done.

                                                                                                                                                                Audio extracted and saved to extracted_audio\Random F
act #youtubeshorts #shorts.mp3                          | 102/323 [00:00<00:00, 509.61it/s, now=None]
MoviePy - Done.                                                                                                                                 | 117/333 [00:00<00:00, 508.35it/s, now=None]
                                                                                                                                                                Audio extracted and saved to extracted_audio\5 Second
 Trick to get WAY MORE Views on YouTube Shorts.mp3       | 153/323 [00:00<00:00, 509.61it/s, now=None]
MoviePy - Done.                                                                                                                                 | 177/333 [00:00<00:00, 508.35it/s, now=None]
                                                                                                                                                                Audio extracted and saved to extracted_audio\Dumb Thi
ngs People Say (Part 25) #Shorts.mp3                    | 307/323 [00:00<00:00, 685.79it/s, now=None]
MoviePy - Done.

                                                                                                                                                                Audio extracted and saved to extracted_audio\Mom Asks
 Son A Question About Texting #Shorts.mp3
MoviePy - Writing audio in extracted_audio\Success is not a comfortable procedure - Steve Harvey Motivational Speech.mp3
MoviePy - Writing audio in extracted_audio\Three rules for a happy life 🍀Beautiful Quotes🍀#shorts.mp3
MoviePy - Writing audio in extracted_audio\When You Never Get Any Credit !! #Shorts.mp3
MoviePy - Writing audio in extracted_audio\When Your Girlfriend Does Not Finish Her FOOD #shorts.mp3
MoviePy - Done.

                                                                                                                                                                Audio extracted and saved to extracted_audio\Success
is not a comfortable procedure - Steve Harvey Motivational Speech.mp300:00<00:00, 706.77it/s, now=None]
MoviePy - Writing audio in extracted_audio\You absolutely have to know this expression in English! 🧠🚫.mp3   | 115/551 [00:00<00:00, 490.19it/s, now=None]
MoviePy - Done.
chunk:   3%|                                                                                                                                          | 30/890 [00:00<00:03, 244.77it/s, now=None]Audio extracted and saved to extracted_audio\Naming R
andom NBA Players Until Someone Messes Up! 🏀 #shorts.mp3 | 611/729 [00:00<00:00, 679.58it/s, now=None]
MoviePy - Done.
chunk:  15%|                                                                                                                                          | 134/890 [00:00<00:01, 499.50it/s, now=None]Audio extracted and saved to extracted_audio\When You
 Never Get Any Credit !! #Shorts.mp3                    | 703/729 [00:01<00:00, 694.31it/s, now=None]
MoviePy - Done.
chunk:  20%|                                                                                                                                          | 174/890 [00:00<00:01, 499.50it/s, now=None]Audio extracted and saved to extracted_audio\Three ru
les for a happy life 🍀Beautiful Quotes🍀#shorts.mp3      | 524/717 [00:00<00:00, 647.00it/s, now=None]
MoviePy - Done.
chunk:  39%|                                                                                                                                          | 345/890 [00:00<00:00, 755.97it/s, now=None]Audio extracted and saved to extracted_audio\When You
r Girlfriend Does Not Finish Her FOOD #shorts.mp3        | 549/717 [00:00<00:00, 647.00it/s, now=None]
MoviePy - Done.
Audio extracted and saved to extracted_audio\You absolutely have to know this expression in English! 🧠🚫.mp3
Audio extraction completed in 2.45 seconds
```

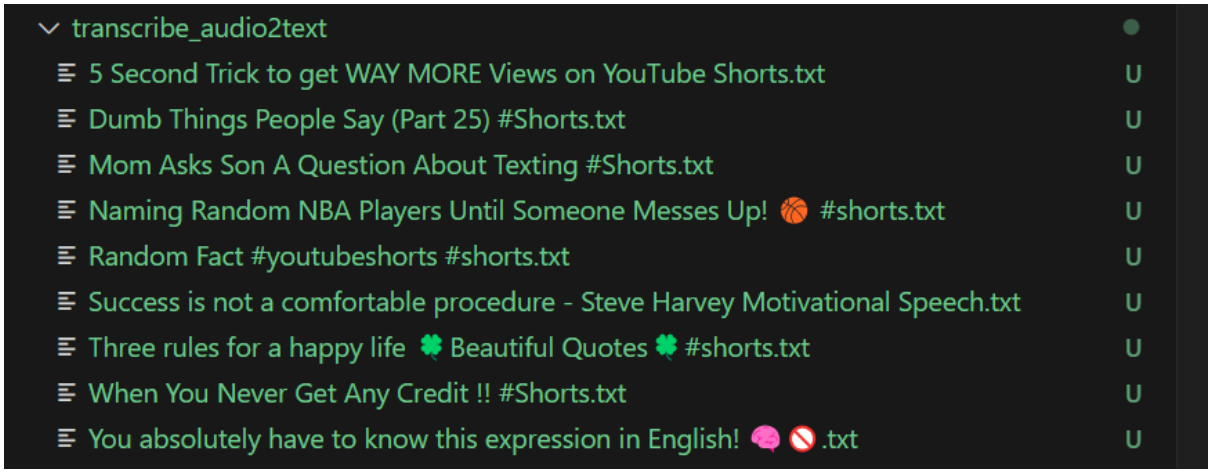


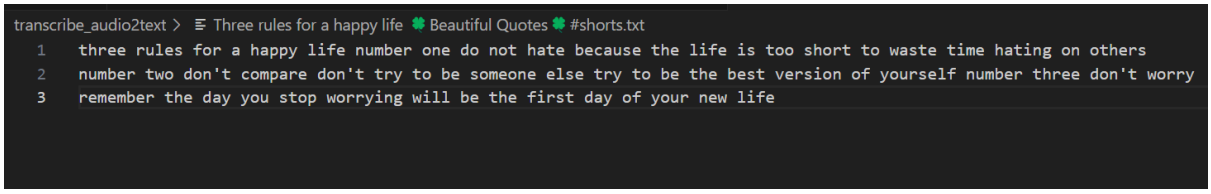



II. Transcribe audio to text.
The transcription of audio files to text file is completed in the file transcribe_audio.py file and the transcribed text files are stored under the transcribe_audio2text folder

```python
def transcribe_audio(wav_path, text_path):
    recognizer = sr.Recognizer()
    try:
        with sr.AudioFile(wav_path) as source:
            audio_data = recognizer.record(source)
            text = recognizer.recognize_google(audio_data)
            with open(text_path, 'w') as text_file:
                text_file.write(text)
        print(f'Transcribed audio for {wav_path}')
    except Exception as e:
        print(f'Could not transcribe audio for {wav_path}: {e}')
```

```
(venv) C:\Users\wadev\OneDrive\Desktop\viplavi\BBK Folder\Semester-3\Big Data Analytics\BDA Project\BDA Project>python transcribe_audio.py
Transcribed audio for extracted_audio\5 Second Trick to get WAY MORE Views on YouTube Shorts.wav
Transcribed audio for extracted_audio\Dumb Things People Say (Part 25) #Shorts.wav
Transcribed audio for extracted_audio\Mom Asks Son A Question About Texting #Shorts.wav
Transcribed audio for extracted_audio\Naming Random NBA Players Until Someone Messes Up! 🏀 #shorts.wav
Transcribed audio for extracted_audio\Random Fact #youtubeshorts #shorts.wav
Transcribed audio for extracted_audio\Success is not a comfortable procedure - Steve Harvey Motivational Speech.wav
Transcribed audio for extracted_audio\Three rules for a happy life 🍀Beautiful Quotes🍀#shorts.wav
Transcribed audio for extracted_audio\When You Never Get Any Credit !! #Shorts.wav
Could not transcribe audio for extracted_audio\When Your Girlfriend Does Not Finish Her FOOD #shorts.wav:
Transcribed audio for extracted_audio\You absolutely have to know this expression in English! 🧠🚫.wav
```

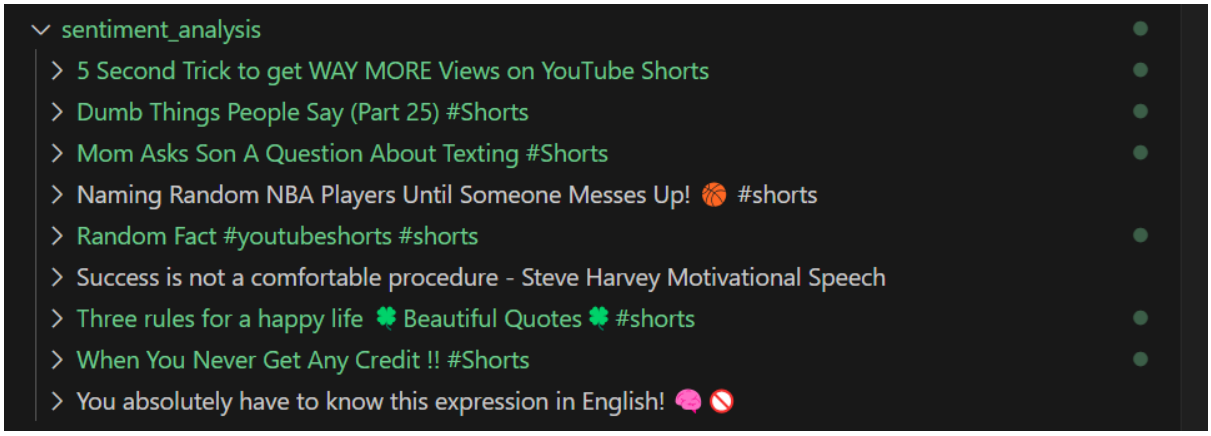


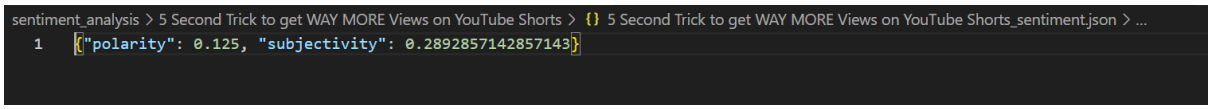




III. Perform the sentiment analysis on a video's content, extracting its polarity and sensitivity.
The sentiment analysis is completed in sentiment_analysis.py file and stored under sentiment_analysis folder. In this example the sentiment analysis is completed in 0.12 seconds. And the sentiment shows the polarity and subjectivity of the video. And it is stored under a json file format for every youtube video.

```python
def analyze_sentiment(text_path, output_dir):
    with open(text_path, "r") as file:
        text = file.read()

    blob = TextBlob(text)
    sentiment = {
        "polarity": blob.sentiment.polarity,
        "subjectivity": blob.sentiment.subjectivity
    }
    sentiment_path = os.path.join(output_dir, f"{os.path.splitext(os.path.basename(text_path))[0]}_sentiment.json")
    with open(sentiment_path, "w") as file:
        json.dump(sentiment, file)
    print(f"Sentiment analysis saved to {sentiment_path}")

def main(text_paths):
    start_time = time.perf_counter()
    with ThreadPoolExecutor(max_workers=5) as executor:
        for text_path in text_paths:
            output_dir = os.path.join("sentiment_analysis", os.path.splitext(os.path.basename(text_path))[0])
            os.makedirs(output_dir, exist_ok=True)
            executor.submit(analyze_sentiment, text_path, output_dir)
    end_time = time.perf_counter()
    print(f"Sentiment analysis completed in {end_time - start_time:.2f} seconds")
```

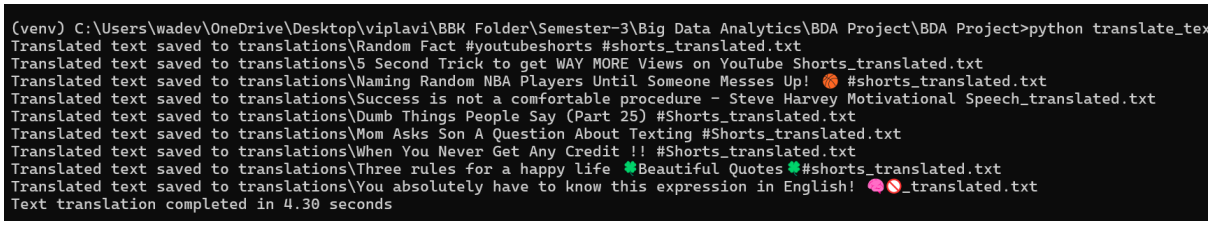
```
(venv) C:\Users\wadev\OneDrive\Desktop\viplavi\BBK Folder\Semester-3\Big Data Analytics\BDA Project\BDA Project>python sentiment_analysis.py
Sentiment analysis saved to sentiiment_analysis\Dumb Things People Say (Part 25) #Shorts\Dumb Things People Say (Part 25) #Shorts_sentiment.json
Sentiment analysis saved to sentiiment_analysis\Mom Asks Son A Question About Texting #Shorts\Mom Asks Son A Question About Texting #Shorts_sentiment.json
Sentiment analysis saved to sentiiment_analysis\Random Fact #youtubeshorts #shorts\Random Fact #youtubeshorts #shorts_sentiment.json
Sentiment analysis saved to sentiiment_analysis\Three rules for a happy life 🍀Beautiful Quotes🍀#shorts\Three rules for a happy life 🍀Beautiful Quotes🍀#shorts_sentiment.json
Sentiment analysis saved to sentiiment_analysis\5 Second Trick to get WAY MORE Views on YouTube Shorts\5 Second Trick to get WAY MORE Views on YouTube Shorts_sentiment.json
Sentiment analysis saved to sentiiment_analysis\When You Never Get Any Credit !! #Shorts\When You Never Get Any Credit !! #Shorts_sentiment.json
Sentiment analysis completed in 0.12 seconds
```

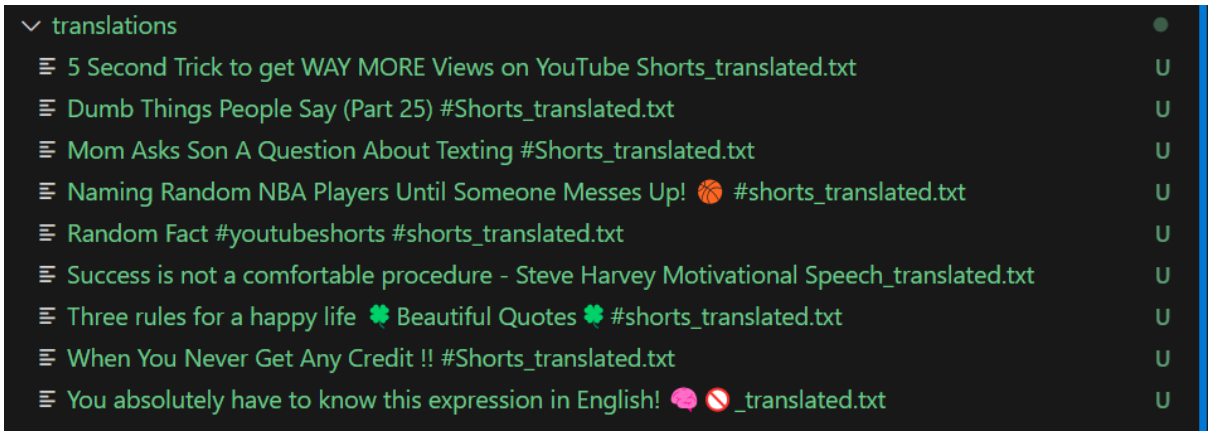


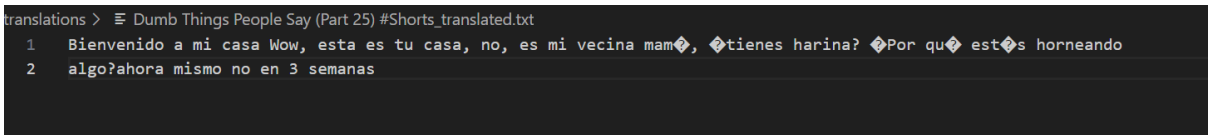




IV. Translate the text into another language, e.g. Spanish.
The code for translating text from English to any other language. Here, default set as 'Spanish' is completed in 'translate_text.py'. The execution for translating the English text to Spanish took around 4.30 seconds to complete. The translated text is stored under the folder named 'translations'.

```python
def translate_text(text_path, output_dir, target_language="es"):
    try:
        with open(text_path, "r") as file:
            text = file.read()

        # Use GoogleTrans Translator explicitly
        translator = Translator()
        translated_text = translator.translate(text, dest=target_language).text

        filename = os.path.basename(text_path).split('.')[0]
        translation_path = os.path.join(output_dir, f"{filename}_translated.txt")

        with open(translation_path, "w") as file:
            file.write(translated_text)

        print(f"Translated text saved to {translation_path}")
    except Exception as e:
        print(f"Error translating {text_path}: {e}")
```




V. Extract the emotions of a text.

The emotion of the text is determined under the extract_emotions.py. The emotions are stored as a json file where the json file contains the values i.e. the emotions of the text as a form in Happy, Angry, Surprise, Sad, and Fear. These json files are stored under a folder named extracted_emotions. The execution of this python script is completed in 1.92 seconds.

```python
def extract_emotions(text_path, output_dir):
    try:
        with open(text_path, "r") as file:
            text = file.read()

        # Extract emotions using the text2emotion library
        emotions = te.get_emotion(text)

        # Create the path for saving the extracted emotions
        filename = os.path.basename(text_path).split('.')[0]
        emotions_path = os.path.join(output_dir, f"{filename}_emotions.json")

        # Write the extracted emotions to a JSON file
        with open(emotions_path, "w") as file:
            json.dump(emotions, file)

        print(f"Emotions extracted and saved to {emotions_path}")
    except Exception as e:
        print(f"Error extracting emotions from {text_path}: {e}")
```

```
(venv) C:\Users\wadev\OneDrive\Desktop\viplavi\BBK Folder\Semester-3\Big Data Analytics\BDA Project\BDA Project>python extract_emotions.py
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\wadev\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\wadev\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\wadev\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Error extracting emotions from transcribe_audio2text\Dumb Things People Say (Part 25) #Shorts.txt: 'WordListCorpusReader' object has no attribut
e '_LazyCorpusLoader__reader_cls'
Error extracting emotions from transcribe_audio2text\Naming Random NBA Players Until Someone Messes Up! 🏀 #shorts.txt: 'WordListCorpusReader' o
bject has no attribute '_LazyCorpusLoader__reader_cls'
Error extracting emotions from transcribe_audio2text\Mom Asks Son A Question About Texting #Shorts.txt: 'WordListCorpusReader' object has no att
ribute '_LazyCorpusLoader__args'
Emotions extracted and saved to extracted_emotions\Random Fact #youtubeshorts #shorts_emotions.json
Emotions extracted and saved to extracted_emotions\5 Second Trick to get WAY MORE Views on YouTube Shorts_emotions.json
Emotions extracted and saved to extracted_emotions\When You Never Get Any Credit !! #Shorts_emotions.json
Emotions extracted and saved to extracted_emotions\Success is not a comfortable procedure - Steve Harvey Motivational Speech_emotions.json
Emotions extracted and saved to extracted_emotions\Three rules for a happy life 🍀Beautiful Quotes🍀#shorts_emotions.json
Emotions extracted and saved to extracted_emotions\You absolutely have to know this expression in English! 🧠🚫_emotions.json
Emotions extraction completed in 1.92 seconds
```