EAST WEST UNIVERSITY

CSE477

Section: 02

Lab: 01 Report

Topic: Exploratory Data Analysis on YouTube Data

Submitted By:

Name: Md Sifatullah Sheikh

ID: 2022-1-60-029

Submitted To:

Amit Mandal

Lecturer

Department of Computer Science & Engineering

Date: 5 July 2025

PART 1: Getting Ready (Tools, Setup, and Verified References)

A. Install Python: I already have Python, and the version is **Python 3.11.9**

Command: **python --version**

```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>python --version
Python 3.11.9
```

B. Install yt-dlp (Both Methods, All Verified):

Method 1 (Install with pip): I already have yt-dlp on my computer environment.

Command: pip install yt-dlp

```
C:\Users\Admin>pip install yt-dlp
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: yt-dlp in c:\users\admin\appdata\local\packages\pythonsoftwarefoundation.python.3
```

Method 2: Download Standalone Executable: Download **yt-dlp.exe** from the official GitHub Releases page: https://github.com/yt-dlp/yt-dlp/releases
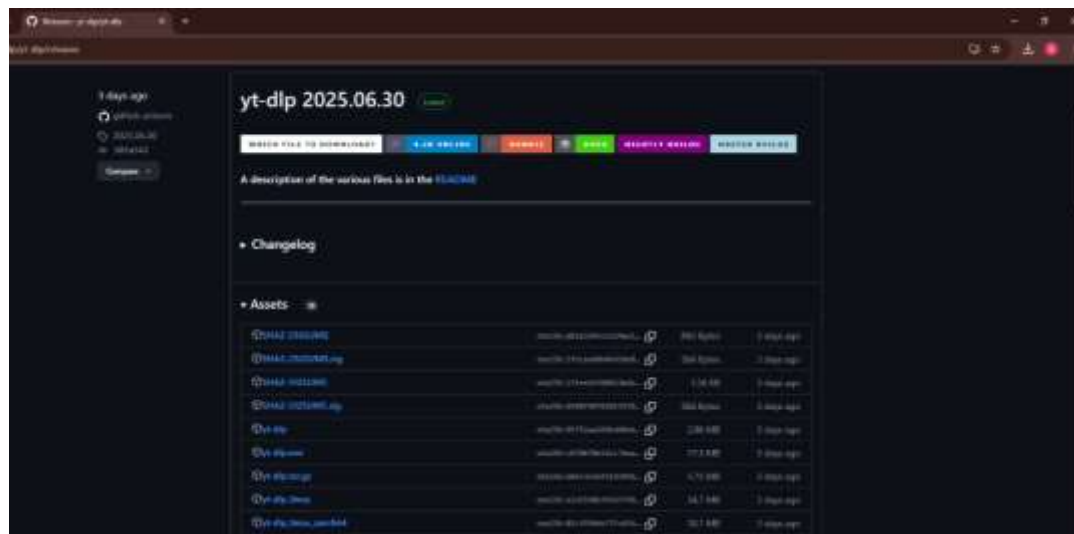
➔ Download it to a particular workable folder.

C. Install Python Libraries: Change to the particular folder disk path, which is **D:**

Command: pip install pandas matplotlib webvtt-py

# PART 2: Data Collection & Selection (Channels, Ethics, and Procedures)

A. Selecting a YouTube Channel/Video (All Channels Verified Neutral) Criteria: I chose a video, and it is from **BBC Earth**

YouTube link: https://www.youtube.com/watch?v=T7oExc711xE

Video Description:

"Explore The Secret Areas of Earth Where Nature Remains Untouched | 4K UHD | BBC Earth"

1. Time length is 15.57 mins
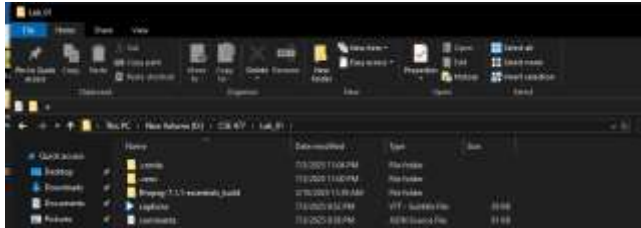
2. 258 comments

3. CC English generated
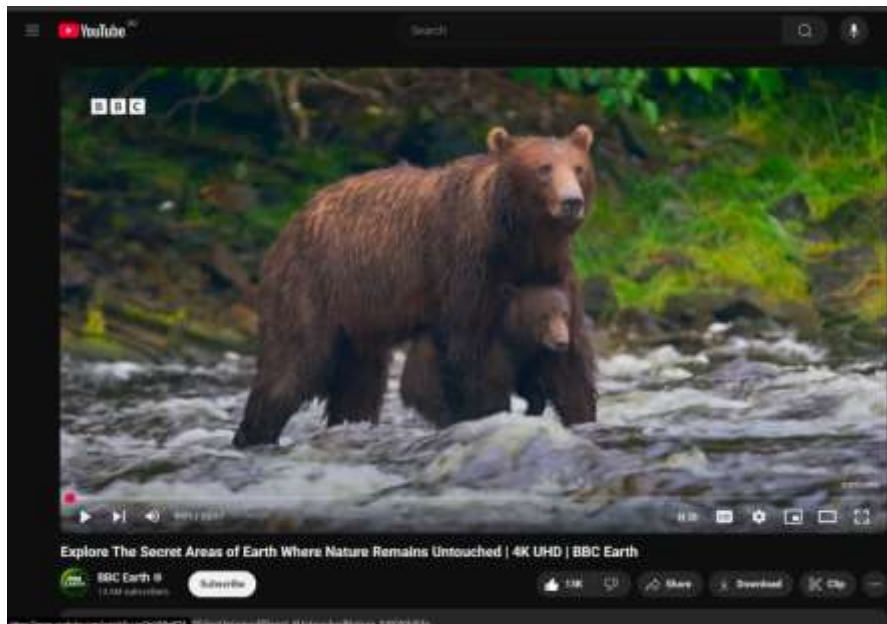
B. Download Captions (Subtitles) Using yt-dlp:

Command: yt-dlp --write-auto-subs --sub-lang en --skip-download
https://www.youtube.com/watch?v=T7oExc711xE

➔ A VTT file is downloaded. Which I named **captions.VTT**

C. Downloading Comments Using a Script:

Commands: pip install youtube-comment-downloader

PART 3: Data Processing & Lab Experiments (Python Skeletons & Analysis)

A. Loading Comments and Captions in Python:

Comments:

```
import json

def load_json_comments(filepath='comments.json'):

    comments = []
```

```python
    with open(filepath, 'r', encoding='utf-8') as f:

        for line in f:

            try:

                obj = json.loads(line)

                comments.append(obj['text'])  # You can also access other fields like obj['time']

            except json.JSONDecodeError:

                continue  # Skip lines that aren't valid JSON

    return comments

# Load and print

raw_comments = load_json_comments()

print(f"Loaded {len(raw_comments)} comment lines.")

print(raw_comments[:5])
```


Comments Output :

Captions:

```python
def load_vtt_captions(filepath='captions.vtt'):

    captions = []
```

```python
    with open(filepath, 'r', encoding='utf-8') as f:

        for line in f:

            line = line.strip()

            # Ignore metadata and timestamps

            if '-->' not in line and line and not line.isdigit() and 'WEBVTT' not in line:

                captions.append(line)

    return captions

# Load and print

raw_captions = load_vtt_captions()

print(f"Loaded {len(raw_captions)} caption lines.")

print(raw_captions[:5])
```

Captions Output:

B. Required Lab Experiments

1. Histogram of Lengths (Captions vs. Comments):

```python
import json

import matplotlib.pyplot as plt


# ----------------------------
# Load captions from .vtt file
# ----------------------------
def load_vtt_captions(filepath='captions.vtt'):

    captions = []

    with open(filepath, 'r', encoding='utf-8') as f:

        for line in f:
```

```python
        line = line.strip()

        # Ignore metadata and timestamps

        if '-->' not in line and line and not line.isdigit() and 'WEBVTT' not in line:

            captions.append(line)

    return captions


# ----------------------------
# Load comments from .json file
# ----------------------------
def load_json_comments(filepath='comments.json'):

    comments = []

    with open(filepath, 'r', encoding='utf-8') as f:

        for line in f:

            try:

                obj = json.loads(line)

                comments.append(obj['text'])

            except json.JSONDecodeError:

                continue

    return comments


# Load data

raw_captions = load_vtt_captions()

raw_comments = load_json_comments()


# Print previews

print(f"Loaded {len(raw_captions)} caption lines.")
```

```python
print(raw_captions[:5])

print(f"Loaded {len(raw_comments)} comment lines.")

print(raw_comments[:5])


# Plot Histogram of Lengths

caption_lengths = [len(x) for x in raw_captions]

comment_lengths = [len(x) for x in raw_comments]


plt.hist(caption_lengths, bins=20, alpha=0.7, label='Captions')

plt.hist(comment_lengths, bins=20, alpha=0.7, label='Comments')


plt.legend()

plt.xlabel('Length (characters)')

plt.ylabel('Frequency')

plt.title('Caption vs. Comment Lengths')

plt.grid(True)

plt.tight_layout()

plt.show()
```
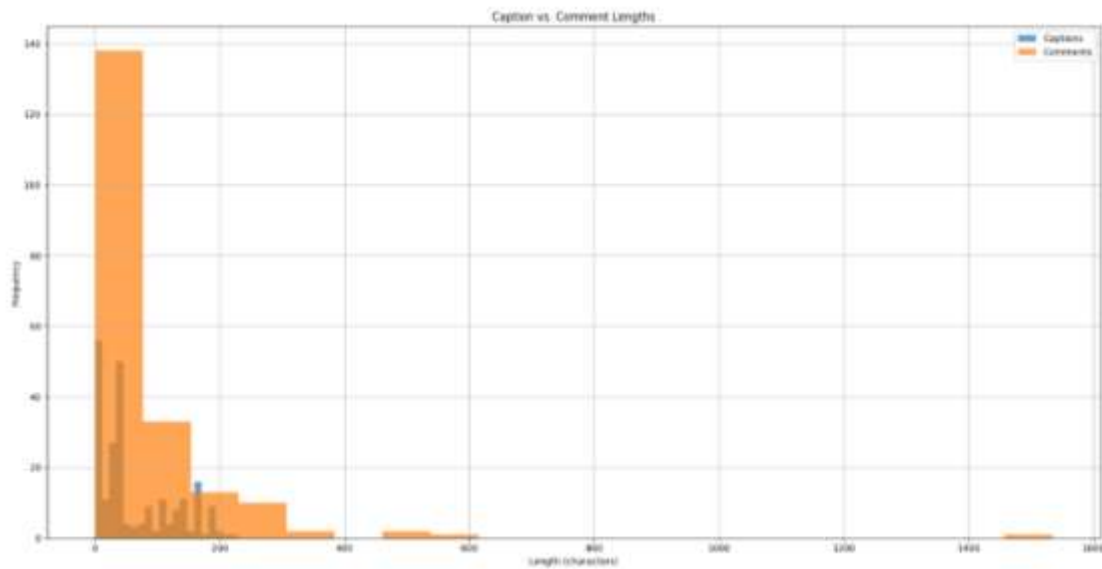
Output :

```
def type_token_ratio(lines):

    words = [word.lower() for line in lines for word in line.split()]

    unique = set(words)

    return len(unique) / len(words) if words else 0

# Print TTR for captions and comments

print("Caption TTR:", type_token_ratio(raw_captions))

print("Comment TTR:", type_token_ratio(raw_comments))

print("Total Captions:", len(raw_captions))

print("Total Comments:", len(raw_comments))
```

Output :

3. Top-N Word Frequency (After Stopword Removal):

```python
import matplotlib.pyplot as plt

plt.bar(['Captions', 'Comments'], [len(raw_captions), len(raw_comments)], color=['skyblue', 'orange'])

plt.ylabel('Count')

plt.title('Number of Captions vs Comments')

plt.show()

# Print first few captions and comments

import pandas as pd

df_caps = pd.DataFrame({'text': raw_captions})

df_caps['length'] = df_caps['text'].apply(len)

print(df_caps.describe())

plt.hist(df_caps['length'], bins=20, alpha=0.7, label='Caption Lengths')

plt.xlabel('Caption Length (characters)')

plt.ylabel('Frequency')

plt.title('Caption Length Distribution')

plt.legend()

plt.grid(True)

plt.show()

# Assuming you've extracted timestamps already

df_caps['timestamp'] = [...]  # e.g., in seconds or minutes

plt.scatter(df_caps['timestamp'], df_caps['length'])

plt.xlabel('Time (sec)')

plt.ylabel('Caption Length')

plt.title('Caption Length vs Timestamp')

plt.show()

# Example: parsing comment JSON into a DataFrame
```

```python
import json

comments = []
with open('comments.json', 'r', encoding='utf-8') as f:
    for line in f:
        obj = json.loads(line)
        comments.append({
            'text': obj['text'],
            'likes': obj.get('likes', 0),
            'length': len(obj['text'])
        })

df_comments = pd.DataFrame(comments)
plt.scatter(df_comments['length'], df_comments['likes'])
plt.xlabel('Comment Length')
plt.ylabel('Likes')
plt.title('Comment Length vs Likes')
plt.grid(True)
plt.show()
plt.scatter(df_caps['timestamp'], df_caps['length'])
plt.title('Caption Lengths Over Time')
plt.xlabel('Time (sec)')
plt.ylabel('Length')
plt.grid(True)
plt.show()
plt.hist(df_comments['length'], bins=30, color='orange')
```

```
plt.title('Comment Length Distribution')

plt.xlabel('Comment Length')

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

# Assuming `df_comments['minute']` is the parsed minute mark

df_by_minute = df_comments.groupby('minute').size()

df_by_minute.plot(kind='line')

plt.xlabel('Minute')

plt.ylabel('Number of Comments')

plt.title('Comments per Minute')

plt.grid(True)

plt.show()
```

Output :

Caption Length Distribution