EAST WEST UNIVERSITY

# *Lab 1*

## *Exploratory Data Analysis on YouTube Data*

***Course Title:*** *Data Mining*

***Course Code:*** *CSE 477*

***Semester:*** *Fall 2025*

***Section:*** *01*

## *Submitted by*

***Name:*** *Nushrat Jaben Aurnima*

***ID:*** *2022-2-60-146*

***Date:*** *17/10/2025*

## *Submitted to*

***Amit Mandal***

*Lecturer, Department of Computer Science and Engineering*

*East West University*

# Table of Contents

# Introduction

The lab ***"Exploratory Data Analysis on YouTube Data"*** aim was to observe to how people communicate on YouTube compared including how information is shown in captions. To fulfill this purpose, I had picked one educational video from **YouTube** and collected both its captions and comments. Through the analysis, it was seen was to look captions coming from speaker were more formal and clearer while comments were written by viewers in a more free and random way.

This lab is also about learning how to work with data in Python. By working on the lab, I learnt how to collect data from YouTube, clean it and perform visualization to understand what it meant, Although a small step into data analysis, it shows how simple text like YouTube comments can tell a lot about how people think and communicate online and become a contribution for NLP and LLMs.

# Tools and Setup

- **Python 3.10** for running all the code.
- Installed the following main libraries using pip:
    - **pandas** – for handling and cleaning data.
    - **matplotlib** – for creating plots and charts.
    - **numpy** – for basic calculations.
    - **webvtt-py** – for reading subtitle from videos.
    - **google-api-python-client** – to fetch YouTube comments using the **YouTube Data API v3**.
- **yt-dlp** to download the YouTube captions.
    - Installation done by running pip install yt-dlp in the terminal.
    - Checked it by typing yt-dlp --help to make sure it was working.
- The whole setup was done on **Google Colab**.

# Data Collection

For creating my own dataset, I selected a video from the ***"freeCodeCamp.org"*** YouTube channel. The video tile is: ***"APIs for Beginners – How to use an API (Full Course / Tutorial)"***. The video was released ***2 years ago*** having **3.7 million views** and over **1,000 comments**, which made it a good choice for analysis.

First, I downloaded the **auto-generated captions** using **yt-dlp** with this command:

```
# YouTube video URL
video_url = 'https://www.youtube.com/watch?v=WXsDOZgxjRw'
# Download captions
subprocess.run([
    'yt-dlp',
    '--skip-download',           # do not download the video itself
    '--write-auto-sub',          # fetch auto-generated captions
    '--sub-lang', 'en',          # limit to English
    '--convert-subs', 'vtt',     # convert subtitles to VTT format
    '-o', 'captions.%(ext)s',    # output filename pattern
(captions.en.vtt)
    video_url
], check=True)
```

This created a file named **captions.en.vtt** that contains all the subtitles from the video. After that I used the **YouTube Data API v3** with the **google-api-python-client** library to collect comments. The script helped to fetch each comment along real time data such as timestamp and number of likes on comments.

# Data Processing

## Loading Captions

- Reads every caption, keeps the text, and converts the **start time** to seconds for plotting purpose later.
- Result: **Loaded 9,934 captions.**

```
# Load captions from VTT file
caption_texts = []  # caption text list
caption_times = []  # start times of captions list

# Read each caption from the VTT file
for caption in webvtt.read('captions.en.vtt'):
```

```
        text = caption.text.strip()
        if text:
            caption_texts.append(text)
            # Convert the start timestamp (HH:MM:SS.mmm) to
seconds
            h, m, s = caption.start.split(':')
            seconds = int(h) * 3600 + int(m) * 60 + float(s)
            caption_times.append(seconds)

print(f'Loaded {len(caption_texts)} captions.')
```

## Loading Comments (YouTube Data API v3)

- Pulles comments and replies per thread.
- Saves text, like counts, and timestamps for each comment.
- Result: **Fetched 980 total comments (including replies).**

```
# Fetch comments using the YouTube Data API.
api_key = 'AIzaSyC65SB6p_U5y6eXOjPOW3PriM1-_Iar6V4'  # Set
API key (rotate & use env var in practice)
video_id = video_url.split('v=')[-1]
youtube = build('youtube', 'v3', developerKey=api_key)

comments_api = []
next_page_token = None

while True:
    response = youtube.commentThreads().list(
        part='snippet,replies',
        videoId=video_id,
        pageToken=next_page_token,
        order='time',           # newest to oldest
        maxResults=100
    ).execute()

    for item in response['items']:
        # Top-level comment
        top_snippet =
item['snippet']['topLevelComment']['snippet']
```

```
            comments_api.append({
                'text': top_snippet['textDisplay'],
                'like_count': top_snippet.get('likeCount', 0),
                'timestamp': top_snippet['publishedAt']
            })
            # Replies (if any)
            if 'replies' in item:
                for reply in item['replies']['comments']:
                    rs = reply['snippet']
                    comments_api.append({
                        'text': rs['textDisplay'],
                        'like_count': rs.get('likeCount', 0),
                        'timestamp': rs['publishedAt']
                    })

        next_page_token = response.get('nextPageToken')
        if not next_page_token:
            break

    print(f'Fetched {len(comments_api)} total comments
    (including replies) via the API.')
```

## Cleaning and Removing Duplicate Comments

- Converts into lowercased, removes punctuation and emojis, trims spaces.
- Keeps one copy per cleaned text to avoids repeated texts.
- Result: **929 unique comments after cleaning.**

```
    # Clean comment and remove duplicate comments
    def clean_text(text):
        """Remove newlines, punctuation and non-ASCII
    characters, lowercase, strip."""
        text = text.replace('\n', ' ')
        text = text.translate(str.maketrans('', '',
    string.punctuation))
        text = re.sub(r'[^\x00-\x7F]+', '', text)  # remove
    emojis / non-ASCII
        text = text.lower().strip()
        return text
```

```
for c in comments_api:
    c['clean_text'] = clean_text(c['text'])

# Remove duplicate comments based on cleaned text
unique_comments = {}
for c in comments_api:
    key = c['clean_text']
    if key not in unique_comments:
        unique_comments[key] = c

unique_comments_list = list(unique_comments.values())
print(f'{len(unique_comments_list)} unique comments after
cleaning.')

# Plain list for later analysis
clean_comments = [c['clean_text'] for c in
unique_comments_list]
```

### Cleaning Captions

- Reuses `clean_text` function.
- Result: **Prepared 9,934 cleaned captions.**

```
# Prepare cleaned captions for analysis
clean_captions = [clean_text(t) for t in caption_texts]
print(f'Prepared {len(clean_captions)} cleaned captions.')
```

## Experiments, Results, and Discussion

In this I will analyze and discuss the experiments and visualization I performed on my created YouTube dataset. It will also include the discussion about the challenges I faced during the whole process.

## Distribution of Caption vs. Comment Lengths

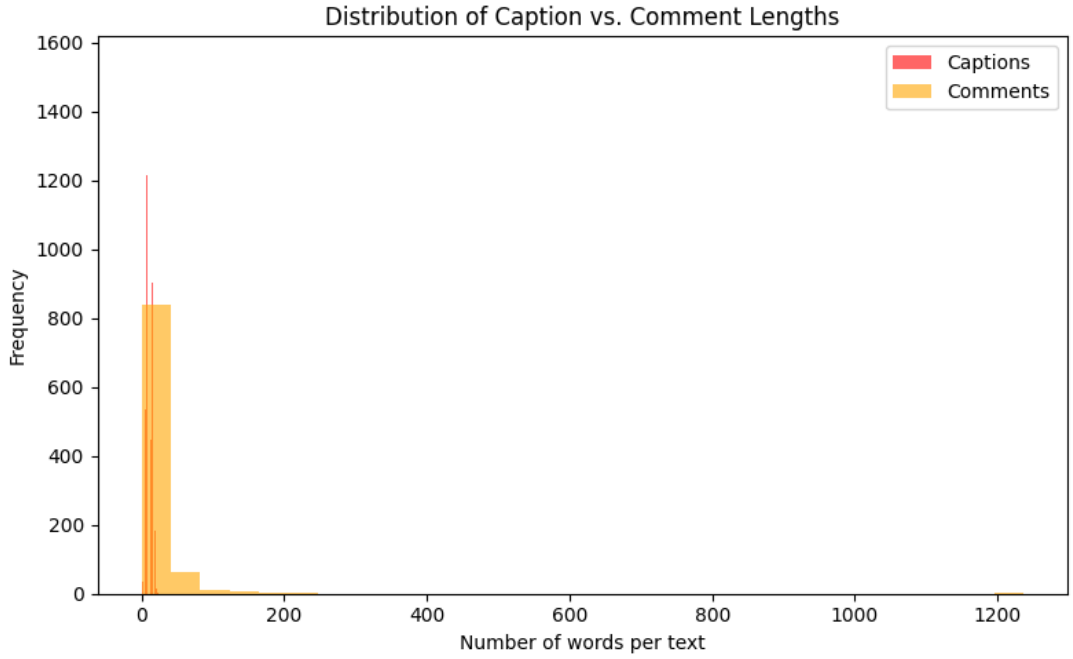


***Figure 1***

We can easily observe that most of the captions and comments are short. However, comments have a wider range despite a few extend to hundreds of words. On the other hand, captions remain tightly clustered. This suggests that captions are kept concise for readability, whereas comments vary widely depending on how the audiences feel.

## Vocabulary Diversity (Type–Token Ratio)

A higher TTR score represents more diverse vocabulary which is very clear in the words of comments having a core 18%. Compared to those, words in captions achieved a 2% score which determines that it relies on repetition in terms of clarity.

| Text | Type–Token Ratio |
|---|---|
| Captions | 0.020 |
| Comments | 0.180 |

***Table 1***

## Top 20-word Frequency

From Table 2 it is cleared that, captions are dominated by natural speech words like “so,” “uh,” and “I,” whereas comments contain more content-related terms such as “API,” “video,” and “Spotify,” reflecting the topics viewers discuss.

| Rank | Caption word | Frequency | Comment word | Frequency |
|---|---|---|---|---|
| 1 | so | 2347 | i | 542 |
| 2 | uh | 2266 | api | 167 |
| 3 | i | 2133 | video | 129 |
| 4 | going | 1934 | not | 124 |
| 5 | we | 1755 | course | 109 |
| 6 | im | 1290 | have | 104 |
| 7 | here | 1290 | spotify | 104 |
| 8 | can | 984 | like | 100 |
| 9 | do | 959 | me | 97 |
| 10 | its | 862 | twilio | 94 |
| 11 | were | 788 | so | 93 |
| 12 | what | 768 | how | 92 |
| 13 | right | 761 | my | 91 |
| 14 | if | 760 | can | 89 |
| 15 | my | 716 | what | 88 |
| 16 | um | 682 | from | 88 |
| 17 | see | 661 | apis | 85 |
| 18 | have | 626 | at | 83 |
| 19 | get | 619 | your | 79 |
| 20 | all | 615 | thank | 78 |

***Table 2***

## Caption Length vs. Timestamp

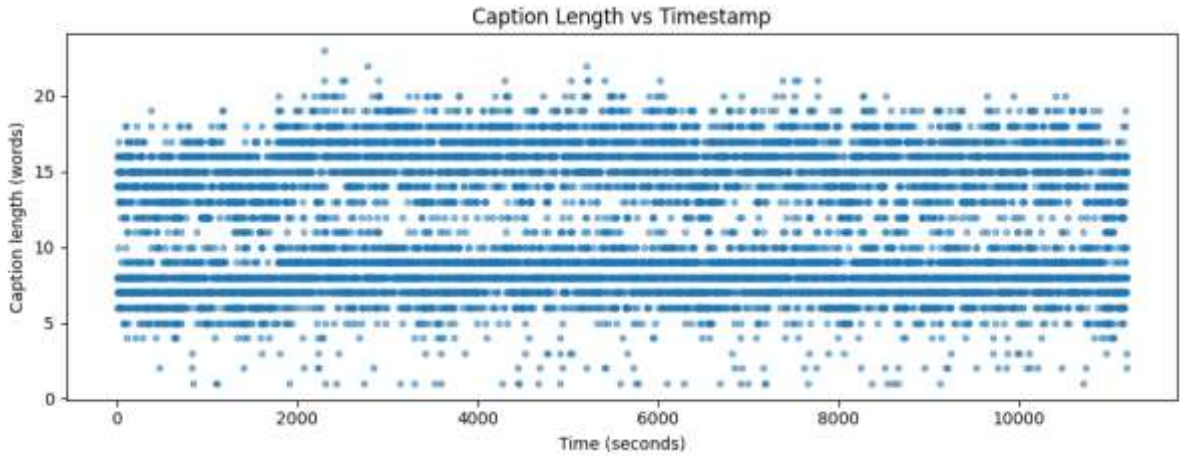


***Figure 2***

The graph clearly shows that caption lengths remain steady throughout the video, generally between 5 and 20 words. There is no clear upward or downward trend, indicating a consistent and well-paced captioning style. So, longer captions are not concentrated near the start or end rather caption length is consistent across the entire timeline

## Comment Activity by Hour

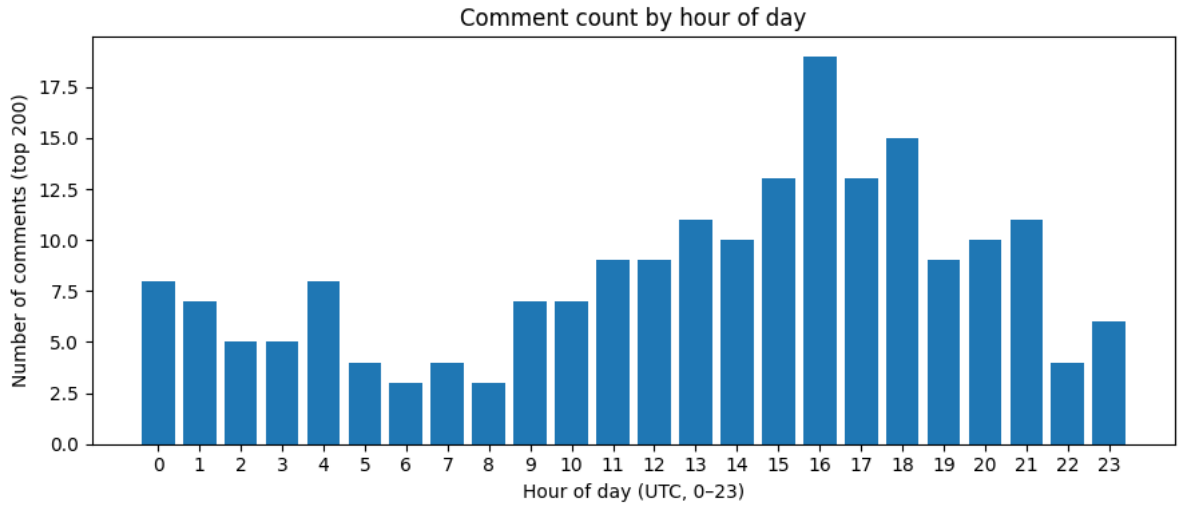


***Figure 3***

The bar plot displays number of comments posted per hour using UTC as the global time reference. Although there us and downs in the comment activity throughout the day, the peak is observed at between 16:00 and 18:00 UTC. This likely corresponds to their afternoon or evening hours depending on viewer's location. It highlights the times when people are generally more active online.

## Likes vs. Comment Length

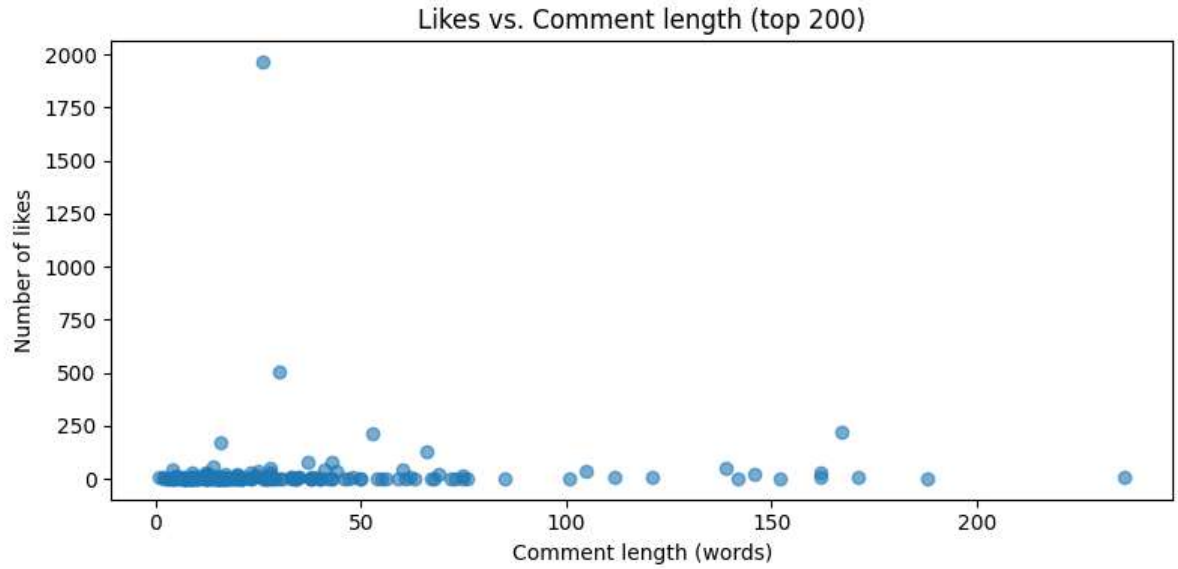


***Figure 4***

In the scatter plot above, most of the comments are short, which only a few likes around 1-60. However, a handful of long comments attract high like counts such as170, 220, 505 including having an outlier Comment having nearly 2000 likes. Overall, there is no strong relationship between length and likes.

## Challenges Faced and Solutions

At first, I tried to fetch YouTube data without using an API. using normal python logic code. However, I could only get the data that was publicly shown, but not the exact timestamp of when each comment was published nor the correct number of likes. For example, the top comment had "1.9K" likes. The code interpreted it as 1900 instead of 1965, which is the original values. Later, I solved this issue by using **YouTube Data API v3**. First, I created a project on **Google Cloud**. Then I enabled the API and generated a key.

Then second problem came after using the API as it only fetched **980 comments**, while the video is showing **1017**. But after intensive searching I found out that the problem was most probably due to **API restrictions** and **regional visibility access**, which means some comments are not available for public availability. Therefore, I kept the 980 comments for analysis since they still represented my experiments and results well.

# Conclusion

This lab helped me understand how captions and comments on YouTube show two different ways of communication. Whereas captions were short and clear, staying consistent throughout the video, the comments were more random and expressive. The results showed that people write in many different styles depending on how they feel about the topic. I also noticed that most comments were made during the afternoon or evening, and longer comments did not get more likes. While working on this project, I learned data analysis on regular texts, numeric data but complete new represented data. It taught me to use APIs and why it is necessary in data analysis. Overall, this lab gave me a good start in understanding how people interact online using data.