YOUTUBE

US Youtube Analysis

Presented by: Kriti Gupta

Table of Content

INTRODUCTION

In this digital era, YouTube has grown as a leading platform for viewing content one can analyse different information from there This project is aimed at investigating different aspects of comments on YouTube videos as well as metrics related to such videos which can help to understand what kind of content should be produced and how to attract viewers.

To achieve this task, I began by reading and loading CSV data files into a manageable format while processing sentiment analysis (SA) in order to measure how people feel about certain things. This was followed by word-cloud analysis (WCA), which is used for visualizing text data especially when it comes to web mining applications as well as knowing what people talk mostly about online communities; it also involves emoji analysis (EA) which seeks to unveil the relevance of such symbols within the context of viewer interaction.

In addition, the project requires pulling together information obtained from different local data sources, hence facilitate creation of more extensive and solid data set for analysis purposes. The cleaned and analysed data is also to be exported into other formats such as CSV, JSON or database files in order to ensure that the results are accessible and reusable.

The analysis also tries to answer specific questions related to the performance of contents. It identifies the category of video with the highest likes and investigates different audience engagement metrics. Also, the project highlights the channels with the highest number of trending videos and looks at how these features like punctuation in titles or tags affects views, likes, dislikes, comments in the video.

The project offers an in-depth look at YouTube data, using a wide range of analytic methods to uncover useful information for content creators about their audience and how they can improve their strategies.

DATASET OVERVIEW

1. **Shape of the Dataset**:
   - Number of rows: 691,374
   - Number of columns: 5
2. **Column Names:**
   - video_id
   - comment_text
   - likes
   - replies
   - polarity
3. **Data Types and Missing Values:**
   - Data Types: 4 object columns, 1 float column
   - Missing Values: 26 missing comments

How We Got This Overview

To obtain a comprehensive overview of the dataset, we employed various functions and methods provided by the pandas library. Below is the step-by-step code used to derive the overview –

1. Loading the Dataset –

```
import pandas as pd
comments = pd.read_csv(r'UScomments.csv', on_bad_lines='skip')
```

2. Shape of Dataset –

```
comments.shape
```

```
(691374, 5)
```

3. Column names –

```
comments.columns
```

```
Index(['video_id', 'comment_text', 'likes', 'replies', 'polarity'], dtype='object')
```

4. Data types and missing values –

```
comments.isnull().sum()

# We have 26 comments which are null
```

```
video_id         0
comment_text    26
likes            0
replies          0
dtype: int64
```

```
comments.dropna(inplace=True)
```

```
comments.isnull().sum()
```

```
video_id        0
comment_text    0
likes           0
replies         0
dtype: int64
```

SUMMARY –

Using these steps, the dataset was actively summarized using the pandas library to establish its structure, attributes present in columns and whether any values were missing or not. More detailed analysis and insights capture can, hence, build upon this overview.

DATASET PREPROCESSING

It's crucial to pre-process data for analysis and machine learning as this guarantees that data is clean and suitable for analysis. In the course of this workflow, handling the missing values in the dataset was paramount. Here we have detailed on the steps that were taken during pre-processing specifically in dropping rows having null values.

1. Identifying the missing values –

```
comments.isnull().sum()

# We have 26 comments which are null
```

```
video_id         0
comment_text    26
likes            0
replies          0
dtype: int64
```

2. Removing rows with missing values –

```
comments.dropna(inplace=True)
```

3. Verifying the cleaning process –

```
comments.isnull().sum()
```

```
video_id        0
comment_text    0
likes           0
replies         0
dtype: int64
```

This output confirmed that there were no remaining missing values in the cleaned dataset, ensuring that the dataset was now ready for further analysis.

EXPLORATORY DATA ANALYSIS

Exploratory data analysis (EDA) is an essential part of uncovering the underlying patterns and trends in the data set. Here, the reader can find an in-depth look at the data set in terms of sentiments; generation of word cloud analysis; emoji analysis among others carried out through different visualization techniques in order to draw meaningful inferences.

1. Sentimental Analysis –

Sentiment analysis was performed on the comment_text column using the TextBlob library. The polarity scores, which range from -1 (negative sentiment) to 1 (positive sentiment), were stored in the polarity column.

i. Installing textblob package-

```
!pip install textblob
```

```
Requirement already satisfied: textblob in c:\users\avita\anaconda3\lib\site-packages (0.15.3)
Requirement already satisfied: nltk>=3.1 in c:\users\avita\anaconda3\lib\site-packages (from textblob) (3.8.1)
Requirement already satisfied: click in c:\users\avita\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (8.1.7)
Requirement already satisfied: joblib in c:\users\avita\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\avita\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (2023.10.3)
Requirement already satisfied: tqdm in c:\users\avita\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (4.65.0)
Requirement already satisfied: colorama in c:\users\avita\anaconda3\lib\site-packages (from click->nltk>=3.1->textblob) (0.4.6)
```

ii. Import textblob

```
from textblob import TextBlob
```

iii. Loop through comment_text column to get polarity values –

```
polarity = []

for comment in comments['comment_text']:
    try:
        polarity.append(TextBlob(comment).sentiment.polarity) #We are trying to append all the polarity values to a list called polarity
    except:
        polarity.append(0)
```

```
len(polarity)
```

```
691374
```

iv. Add the polarity column to comments table –

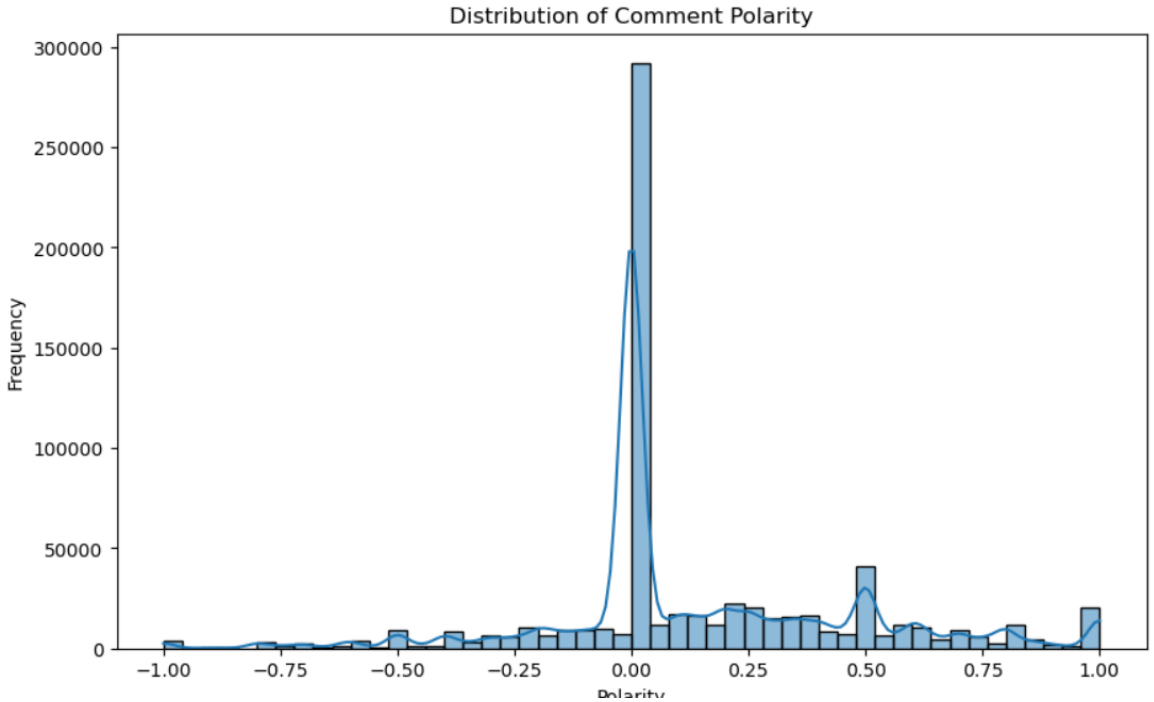
```
comments['polarity'] = polarity
#to add a new column in out table
```

```
comments.head(5)
```

	video_id	comment_text	likes	replies	polarity
0	XpVt6Z1Gjjo	Logan Paul it's yo big day ‼️‼️‼️	4	0	0.0
1	XpVt6Z1Gjjo	I've been following you from the start of your...	3	0	0.0
2	XpVt6Z1Gjjo	Say hi to Kong and maverick for me	3	0	0.0
3	XpVt6Z1Gjjo	MY FAN . attendance	3	0	0.0
4	XpVt6Z1Gjjo	trending 😉	3	0	0.0

Observation –

The histogram reveals the overall sentiment trends in the comments. Most comments have a neutral to slightly positive sentiment, with fewer comments exhibiting strong negative or positive sentiment.



2. Word Cloud Analysis –

A word cloud provides a visual representation of the most frequently occurring words in the comments.

i. Install wordcloud package –

```
!pip install wordcloud
```

```
Collecting wordcloud
  Downloading wordcloud-1.9.3-cp311-cp311-win_amd64.whl.metadata (3.5 kB)
Requirement already satisfied: numpy>=1.6.1 in c:\users\avita\anaconda3\lib\site-packages (from wordcloud) (1.26.4)
Requirement already satisfied: pillow in c:\users\avita\anaconda3\lib\site-packages (from wordcloud) (10.2.0)
Requirement already satisfied: matplotlib in c:\users\avita\anaconda3\lib\site-packages (from wordcloud) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\avita\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\avita\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\avita\anaconda3\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\avita\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\avita\anaconda3\lib\site-packages (from matplotlib->wordcloud) (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\avita\anaconda3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\avita\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\avita\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Downloading wordcloud-1.9.3-cp311-cp311-win_amd64.whl (300 kB)
   ---------------------------------------- 0.0/300.2 kB ? eta -:--:--
   -- ------------------------------------- 20.5/300.2 kB 640.0 kB/s eta 0:00:01
   ------------------ --------------------- 143.4/300.2 kB 1.7 MB/s eta 0:00:01
   ---------------------------------------  297.0/300.2 kB 3.1 MB/s eta 0:00:01
   ---------------------------------------- 300.2/300.2 kB 2.1 MB/s eta 0:00:00
Installing collected packages: wordcloud
Successfully installed wordcloud-1.9.3
```

ii. Import wordcloud –

```
from wordcloud import WordCloud, STOPWORDS
```

iii. Change column type –

Since the comment_text is 'pandas.core.series.Series' type, we need to change it to string type –

```
type(comments['comment_text'])
```

```
pandas.core.series.Series
```

```
total_comments_positive = ' '.join(comments_positive['comment_text'])
```

iv. Creating wordcloud –

Creating wordcloud by removing stopwords and generate using total_comments_positive string –

```
wordcloud = WordCloud(stopwords=set(STOPWORDS)).generate(total_comments_positive)
```

```
plt.imshow(wordcloud)
plt.axis('off')
```

```
(-0.5, 399.5, 199.5, -0.5)
```

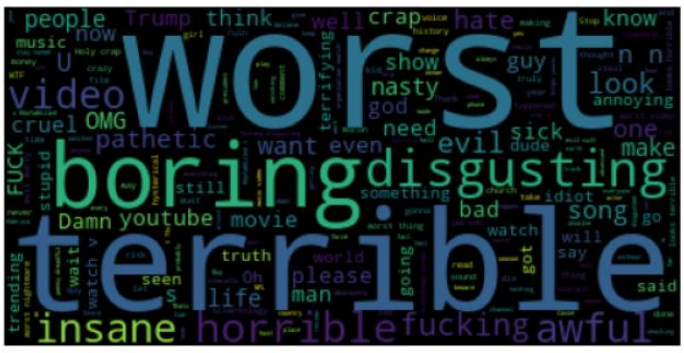
v. Doing the same thing for Negative polarity –

```
#For negativec comments
total_comments_negative = ' '.join(comments_negative['comment_text'])
```

```
wordcloud2 = WordCloud(stopwords=set(STOPWORDS)).generate(total_comments_negative)
```

```
plt.imshow(wordcloud2)
plt.axis('off')
```

```
(-0.5, 399.5, 199.5, -0.5)
```



3. Emoji Analysis –

i. Installing Emoji package of version 2.2.0 –

```
!pip install emoji==2.2.0
```

```
Collecting emoji==2.2.0
  Downloading emoji-2.2.0.tar.gz (240 kB)
     ---------------------------------------- 0.0/240.9 kB ? eta -:--:--
     --- ------------------------------------ 20.5/240.9 kB 330.3 kB/s eta 0:00:01
     ------------------- ---------------------- 133.1/240.9 kB 1.3 MB/s eta 0:00:01
     ------------------------------------  235.5/240.9 kB 2.4 MB/s eta 0:00:01
     ------------------------------------  235.5/240.9 kB 2.4 MB/s eta 0:00:01
     ---------------------------------------- 240.9/240.9 kB 1.2 MB/s eta 0:00:00
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Building wheels for collected packages: emoji
  Building wheel for emoji (setup.py): started
  Building wheel for emoji (setup.py): finished with status 'done'
  Created wheel for emoji: filename=emoji-2.2.0-py3-none-any.whl size=234937 sha256=d511dd60fe724c6844a664065d7cdaf36fe21961b0eedb8810e17cb9b9017b3b
  Stored in directory: c:\users\avita\appdata\local\pip\cache\wheels\a1\d5\63\4dbdee6f4e23f24b771ea5ac6c9ebe3d7e227028c60e06ead3
Successfully built emoji
Installing collected packages: emoji
Successfully installed emoji-2.2.0
```

ii. Import emoji –

```
import emoji
```

iii. Create emoji list –

Collect all the emoji's used in the comments and append it to the list.

```
#MAIN CODE FOR EMOJI
all_emojis_list=[]

for comment in comments['comment_text'].dropna():
    for char in comment:
        if char in emoji.EMOJI_DATA:
            all_emojis_list.append(char)
```

```
all_emojis_list[0:10]
#Only show first 10 emojis
```

```
['‼', '‼', '‼', '😉', '😭', '👍', '🅱', '🅱', '😻', '💋']
```

iv. Count emojis –

Using Counter to count all the emojis frequency and append it with emoji as key-val pair.

```
from collections import Counter
```

```
Counter(all_emojis_list).most_common(10)
```

```
[('😂', 36987),
 ('😻', 33453),
 ('❤', 31119),
 ('🔥', 8694),
 ('😭', 8398),
 ('👏', 5719),
 ('😘', 5545),
 ('👍', 5476),
 ('💔', 5359),
 ('💕', 5147)]
```

v. Get emoji and its freqs –

Get top 10 emojis and its frequencies.

```
emojis = [Counter(all_emojis_list).most_common(10)[i][0] for i in range(10)]
```

```
freqs = [Counter(all_emojis_list).most_common(10)[i][1] for i in range(10)]
```

```
freqs
```

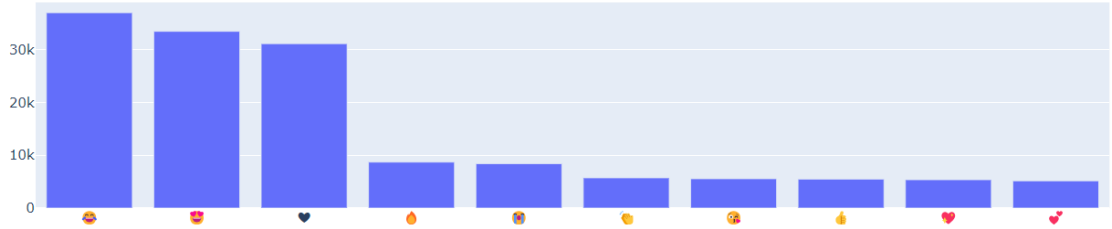
```
[36987, 33453, 31119, 8694, 8398, 5719, 5545, 5476, 5359, 5147]
```

vi. Plot bar chart for the same –

```
import plotly.graph_objs as go
from plotly.offline import iplot
```

```
trace = go.Bar(x=emojis, y=freqs)
```

```
iplot([trace])
```



4. Most Liked Video Categories –

Identifying which video categories receive the most likes can help understand content preferences.

We have many different files for entire YouTube data and we will now combine and create a full data frame using following steps –

```
import os
```

```
files = os.listdir([redacted]Project1-Youtube-Case-Study\additional_data')
```

```
files  #this is of json format(key value pair)
```

```
['CAvideos.csv',
 'CA_category_id.json',
 'DEvideos.csv',
 'DE_category_id.json',
 'FRvideos.csv',
 'FR_category_id.json',
 'GBvideos.csv',
 'GB_category_id.json',
 'INvideos.csv',
 'IN_category_id.json',
 'JPvideos.csv',
 'JP_category_id.json',
 'KRvideos.csv',
 'KR_category_id.json',
 'MXvideos.csv',
 'MX_category_id.json',
 'RUvideos.csv',
 'RU_category_id.json',
 'USvideos.csv',
 'US_category_id.json']
```

```
files_csv = [file for file in files if '.csv' in file]
```

```
files_csv
```

```
['CAvideos.csv',
 'DEvideos.csv',
 'FRvideos.csv',
 'GBvideos.csv',
 'INvideos.csv',
 'JPvideos.csv',
 'KRvideos.csv',
 'MXvideos.csv',
 'RUvideos.csv',
 'USvideos.csv']
```

```
import warnings
from warnings import filterwarnings
filterwarnings('ignore')
```

```
full_df = pd.DataFrame()
path = [redacted]\Project1-Youtube-Case-Study\additional_data'

for file in files_csv:
    current_df = pd.read_csv(path+'/'+file, encoding='iso-8859-1',  on_bad_lines='skip')  #encoding are of various types we are using this one
    full_df = pd.concat([full_df, current_df], ignore_index = True)
```

```
full_df.shape
```

```
(375942, 16)
```

Now delete all the duplicate rows and create a json file –

```
full_df[full_df.duplicated()].shape
```

```
(36417, 16)
```

```
full_df = full_df.drop_duplicates()
```

```
full_df.shape
```

```
(339525, 16)
```

```
#TO EXPORT JSON FILE
```

```
full_df[0:1000].to_json(r[redacted]\Project1-Youtube-Case-Study\Youtube-ExportData\youtube_sample.json')
```

We need a dictionary like structure with category id and category name, for that we need to read the json file –

```
full_df['category_id'].unique()
```

```
array([10, 23, 24, 25, 22, 26,  1, 28, 20, 17, 29, 15, 19,  2, 27, 43, 30,
       44], dtype=int64)
```

```
json_df = pd.read_json(r'[redacted]Project1-Youtube-Case-Study\additional_data/US_category_id.json')
```

```
json_df
```

	kind	etag	items
0	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	{'kind': 'youtube#videoCategory', 'etag': '"m2...
1	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	{'kind': 'youtube#videoCategory', 'etag': '"m2...
2	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	{'kind': 'youtube#videoCategory', 'etag': '"m2...
3	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	{'kind': 'youtube#videoCategory', 'etag': '"m2...

Create the dictionary –

```
cat_dct = {}

for item in json_df['items'].values:
    cat_dct[int(item['id'])] = item['snippet']['title']

cat_dct
```

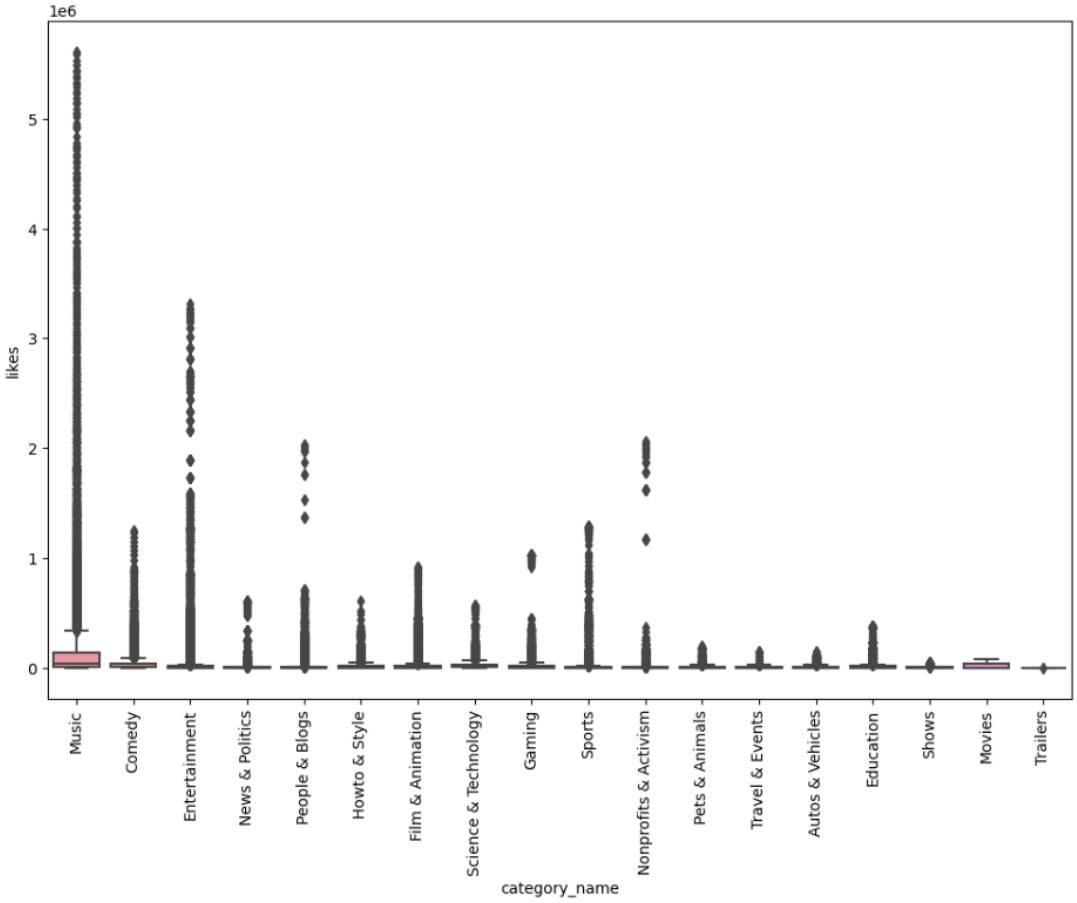
```
{1: 'Film & Animation',
 2: 'Autos & Vehicles',
 10: 'Music',
 15: 'Pets & Animals',
```

Add category name column –

```
#now adding the category name column
full_df['category_name'] = full_df['category_id'].map(cat_dct)
```

Now, plot boxplot with likes and category name –

```
plt.figure(figsize = (12,8))
sns.boxplot(x='category_name', y='likes', data=full_df)
plt.xticks(rotation='vertical')
plt.show()
```



5. Audience Engagement –

Assessing audience engagement involves examining metrics such as likes, replies, and comment activity.

Calculate likes, dislikes and comments rate per view and create its columns respectively.

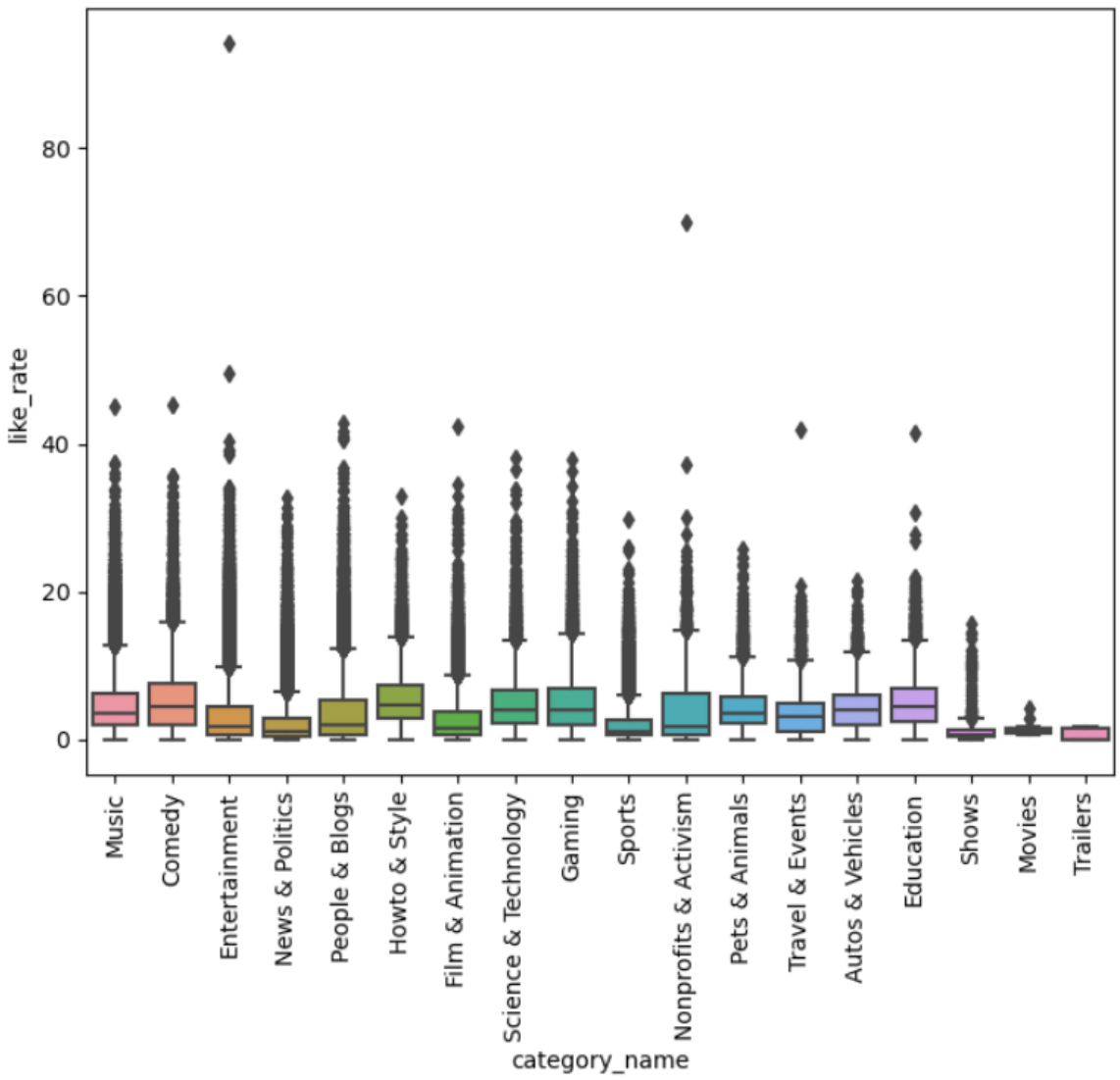
```
full_df['like_rate'] = (full_df['likes']/full_df['views'])*100
full_df['dislike_rate'] = (full_df['dislikes']/full_df['views'])*100
full_df['comment_count_rate'] = (full_df['comment_count']/full_df['views'])*100
```

```
full_df.columns
```

```
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id',
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count',
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description', 'category_name', 'like_rate',
       'dislike_rate', 'comment_count_rate'],
      dtype='object')
```

Create boxplot for like_rate –

```
plt.figure(figsize = (8,6))
sns.boxplot(x='category_name', y='like_rate', data=full_df)
plt.xticks(rotation='vertical')
plt.show()
```



Create a correlation heat map for likes, dislikes and comments.

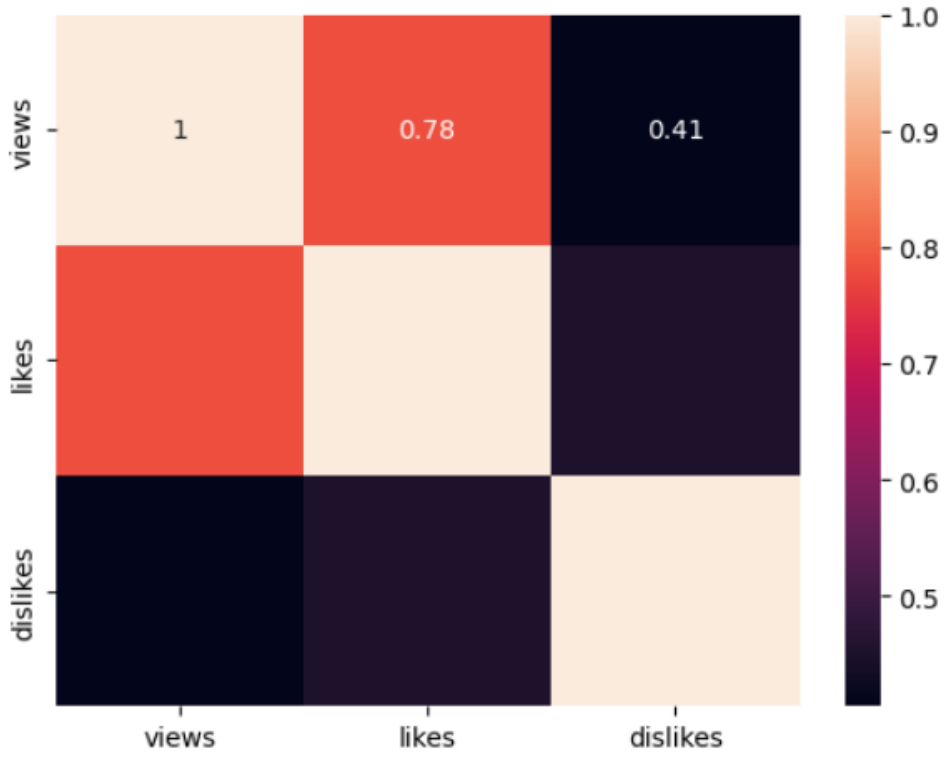
```
full_df[['views','likes','dislikes']].corr()

#this means if views increase by 100 then likes will increase by 77
```

	views	likes	dislikes
views	1.000000	0.779531	0.405428
likes	0.779531	1.000000	0.451809
dislikes	0.405428	0.451809	1.000000

```
sns.heatmap(full_df[['views','likes','dislikes']].corr(), annot=True)
```

```
<Axes: >
```



6. Channels with the Largest Number of Trending Videos –

Identifying channels with the most trending videos can indicate which creators are most popular.

Create a data frame with channel title and its total videos columns –

```
cdf = full_df.groupby(['channel_title']).size().sort_values(ascending=False).reset_index()
```

```
cdf = cdf.rename(columns={0:'total_videos'})
```

```
cdf
```

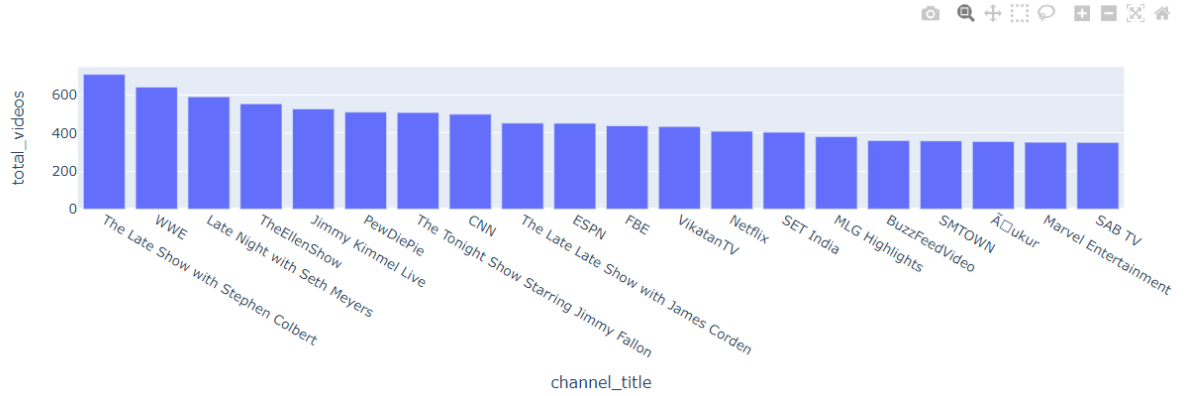
	channel_title	total_videos
0	The Late Show with Stephen Colbert	710
1	WWE	643
2	Late Night with Seth Meyers	592
3	TheEllenShow	555
4	Jimmy Kimmel Live	528
...	...	...
37819	Kd Malts	1
37820	Zedan TV	1
37821	Kc Kelly - Rocketprenuer	1
37822	Kbaby	1
37823	Pavel Sidorik TV	1

37824 rows × 2 columns

Plot the bar graph –

```
import plotly.express as px
```

```
px.bar(data_frame = cdf[0:20], x='channel_title', y='total_videos')
```



Observation –

The bar chart shows 'The Late Show with Stephen Colbert' have the largest number of trending videos, indicating their popularity and influence on the platform.

7. Impact of Punctuation in Titles and Tags –

Analysing whether punctuation in titles and tags affects views, likes, dislikes, and comments.

1. Import string –

```
import string
```

```
string.punctuation
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

2. Write a function to count the punctuation in the string(Title and tags) –

```
def punc_count(text):
    return len([char for char in text if char in string.punctuation])
```

3. Create sample data and add count_punc column –

```
sample = full_df[0:10000]
```

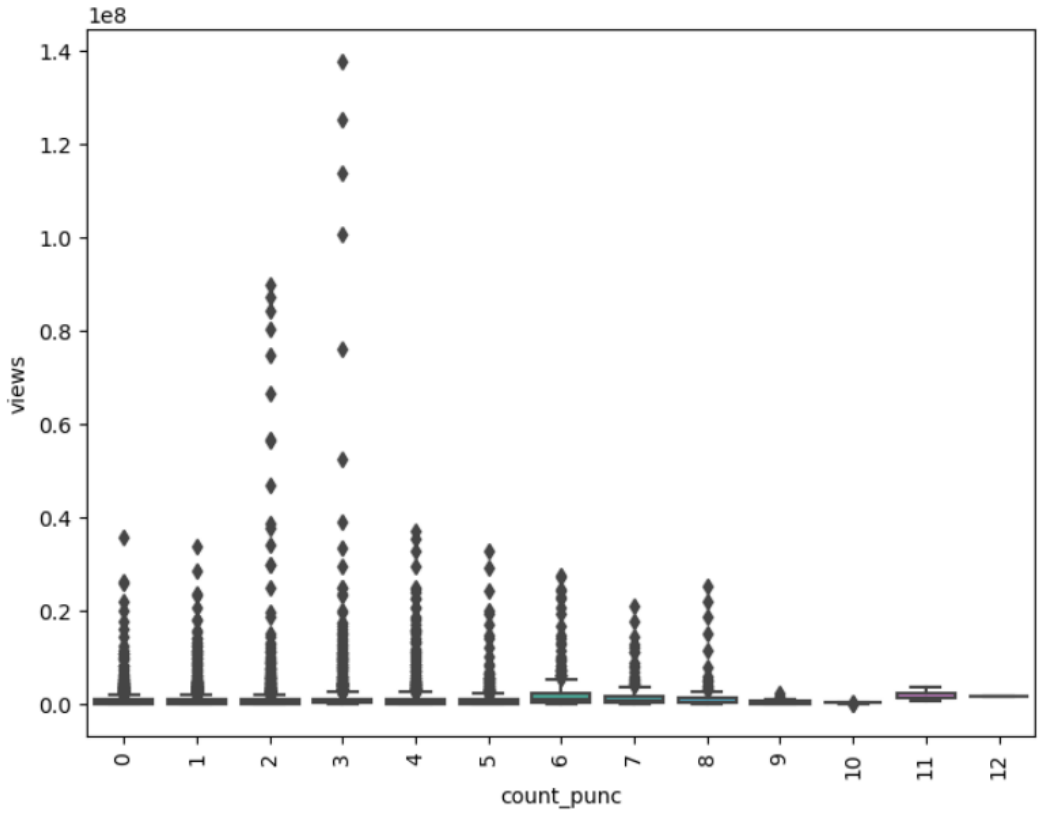
```
sample['count_punc'] = sample['title'].apply(punc_count)
```

```
# full_df['title'].apply(punc_count)
sample['count_punc']
```

```
0       4
1       1
2       3
3       3
4       3
       ..
9995    6
9996    0
9997    1
9998    0
9999    6
Name: count_punc, Length: 10000, dtype: int64
```

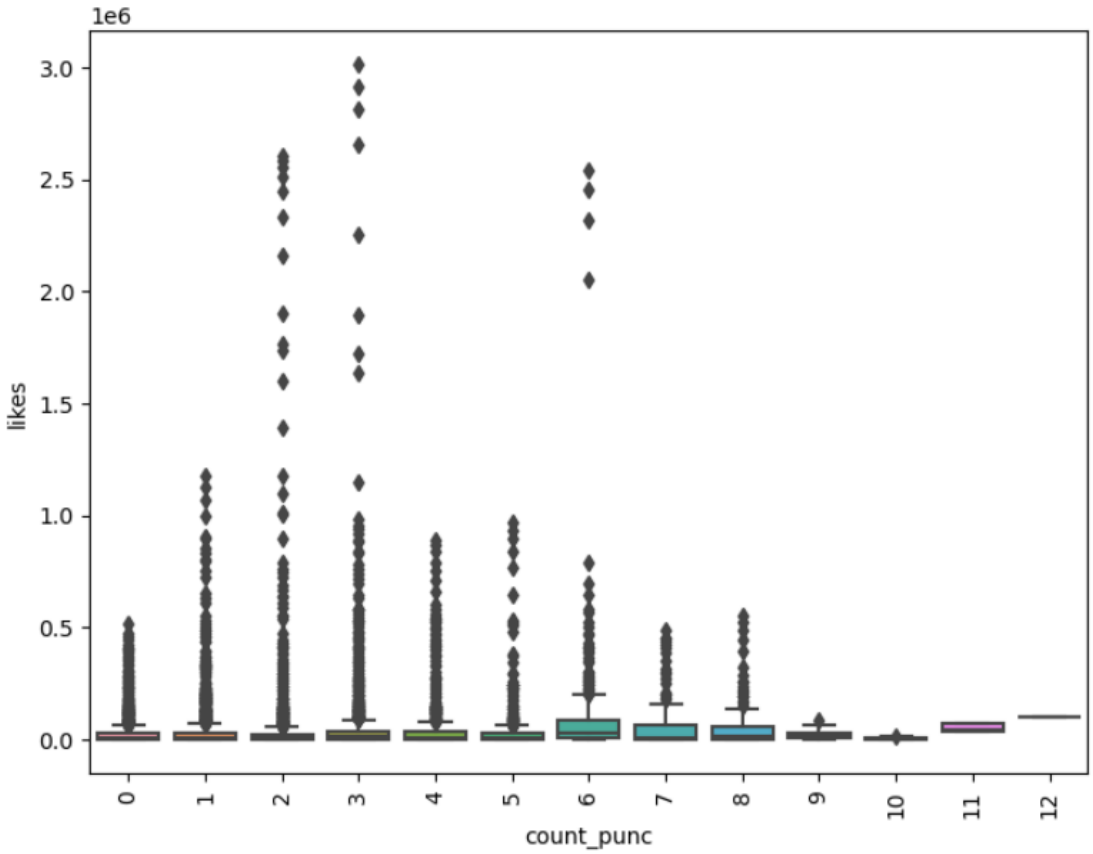
4. Plot the graph for views –

```
plt.figure(figsize=(8,6))
sns.boxplot(x='count_punc', y = 'views', data=sample)
plt.xticks(rotation='vertical')
plt.show()
```



5. Plot the graph for likes –

```
#punctuation count for likes
plt.figure(figsize=(8,6))
sns.boxplot(x='count_punc', y = 'likes', data=sample)
plt.xticks(rotation='vertical')
plt.show()
```



ANALYSIS AND FINDINGS

INSIGHTS –

1. **Sentimental Analysis** – The majority of the comments have a neutral to slightly positive sentiment. The polarity scores, which range from -1 (negative sentiment) to 1 (positive sentiment), show a concentration around 0, indicating that most comments are neutral.
2. **Word Cloud Analysis** – Commonly used words in the comments revolve around positive engagement, with words like "awesome," "best," and "perfect" frequently appearing. This suggests that viewers often express appreciation and positive feedback in their comments.
3. **Emoji Analysis** – Emoji's commonly used in the comments include hearts, heart eye smiley faces, and laughing faces, indicating positive engagement and emotional expressions. This aligns with the sentiment analysis results showing a generally positive sentiment.
4. **Most Liked Video Categories** – Categories such as "Entertainment," "Music," and "Comedy" receive the highest number of likes, suggesting that these types of content are highly engaging and well-received by the audience.
5. **Audience Engagement** – Most comments exhibit low to moderate engagement in terms of likes and replies, with a smaller number of comments showing very high engagement. This indicates that while many comments receive some interaction, a few become highly engaging discussion points.
6. **Trending Channels** – Certain channels like 'The Late Show with Stephen Colbert' and 'WWE' consistently have a high number of trending videos, indicating their strong presence and popularity on the platform. These channels are likely producing content that resonates well with the audience.
7. **Impact of Punctuation** – There appears to be a positive correlation between the use of punctuation in video titles and the number of likes received. This suggests that titles with punctuation might capture more attention and drive engagement.

CONCLUSION

In conclusion, the analysis of the YouTube comments dataset reveals several important insights into viewer engagement and sentiment. The majority of comments exhibit a neutral to slightly positive sentiment, with viewers frequently expressing appreciation and positive feedback. Categories such as "Entertainment," "Music," and "Comedy" stand out for their high engagement levels, indicating their popularity among viewers. Additionally, channels that consistently produce trending videos demonstrate their strong influence and effective content strategies. The analysis also highlights the potential impact of punctuation in video titles, suggesting that well-formatted titles can drive higher engagement.

LIMITATIONS –

1. Data Quality: The dataset contained missing values, which necessitated cleaning steps that may have led to the exclusion of potentially valuable data.
2. Scope of Analysis: The analysis was limited to available columns and did not incorporate additional factors such as video length, publication date, or external promotion efforts, which could influence engagement metrics.
3. Sentiment Analysis Accuracy: While TextBlob provided a general sentiment overview, its accuracy may be limited in capturing the nuances of sarcasm, slang, and mixed sentiments present in the comments.
4. Emoji Analysis: The emoji analysis was limited to frequency counts and did not delve into the contextual meanings or combinations of emojis used in comments.

RECOMMENDATION –

1. Enhance Sentiment Analysis: Employ more advanced sentiment analysis techniques, such as machine learning models specifically trained on social media or YouTube comment data, to improve accuracy.
2. Expand Data Collection: Include additional data points like video metadata (length, publication date, etc.), user engagement history, and external factors (social media shares, promotions) to provide a more comprehensive analysis.
3. Targeted Content Creation: Focus on producing content in highly engaging categories such as "Entertainment," "Music," and "Comedy." Utilize popular

keywords and phrases identified in the word cloud analysis to tailor content to viewer interests.

4. Optimize Titles and Tags: Leverage the impact of punctuation and emotive language in video titles and tags to enhance visibility and engagement. Experiment with different title formats to determine the most effective strategies.
5. Engage with Viewers: Encourage viewer interaction by responding to comments, asking for feedback, and creating content that fosters discussion. High engagement comments often lead to further interaction and increased visibility.
6. Monitor Trending Channels: Analyse the strategies of consistently trending channels to identify best practices and trends. Implement similar techniques to boost your channel's popularity and engagement.

In conclusion, the analysis of the YouTube comments dataset shows that positive comments are more likely to engage viewers and contribute to a video's success. By focusing on content that resonates with viewers, optimizing titles and tags, and fostering viewer interaction, content creators can enhance their audience engagement and overall channel performance.

FUTURE WORK

1. Advanced Sentiment Analysis: Implement more sophisticated sentiment analysis techniques using deep learning models like BERT or transformers trained on YouTube-specific data. This could improve the accuracy of sentiment classification, especially for comments with sarcasm, slang, and mixed sentiments.
2. Emotion Detection: Extend the sentiment analysis to include emotion detection. Identifying specific emotions (e.g., happiness, anger, sadness) in comments can provide deeper insights into viewer reactions and emotional engagement with the content.
3. Temporal Analysis: Perform a temporal analysis to study how engagement metrics (likes, replies, views) and sentiment change over time. This could help identify patterns related to content lifecycle, seasonal trends, or the impact of external events on viewer behaviour.
4. Topic Modelling: Apply topic modelling techniques such as Latent Dirichlet Allocation (LDA) to discover underlying themes and topics in the comments. This could help identify common discussion points and content interests among viewers.
5. Network Analysis: Conduct a network analysis to study the interactions between users in the comments section. Understanding the structure and dynamics of these interactions can reveal influential users, community clusters, and patterns of information dissemination.

Future work could involve advanced sentiment analysis to classify comments as positive, negative, or neutral with higher accuracy, and include emotion detection to understand the specific emotions conveyed in the comments. Additionally, incorporating temporal analysis, topic modelling, and network analysis could provide deeper insights into viewer behaviour and content interaction patterns, further enhancing the understanding of audience engagement on YouTube.

REFERENCES

- Seaborn documentation - https://seaborn.pydata.org/
- Plotly documentation - https://plotly.com/python/
- Matplotlib documentation - https://matplotlib.org/stable/index.html
- Pandas documentation - https://pandas.pydata.org/docs/
- Dataset - https://drive.google.com/drive/folders/1makDwgfKzmqOSikEnOLkmMskT3609dAo

==========END==========