

Assignment – Text analytics

Problem Statement

Create a data story of the online / social media response of a movie or a TV series.

Time limit – **5 days**.

Instructions:

1. Select one recently launched movie or a TV show from Hotstar or Netflix.
2. Extract reviews, tweets, or any relevant text data from social media platforms/websites like Twitter, Facebook, Google etc.
3. Clean the data and create an appropriate schema to store it in a table format(s).
4. Perform EDA and apply relevant ML algorithms if required.
5. Highlight insights/relevant stats and conclude whether the movie/TV series has received a positive/negative or neutral response from the online community.
6. Record your outputs as a presentation or a dashboard.
7. Share the following outputs
 - a. PDF file of your presentation OR a dashboard link to an online public library (Example Tableau public or Power BI gallery).
 - b. Supporting documents in PDF format – code, data, approach etc.

You are being tested for the following:

1. Your ability to understand the problem statement correctly and define a scope of work for an open-ended problem.
2. Your ability to program.
3. Ability to use multiple tools to work on an analytics problem end to end.
4. Create an insightful story.

Tips:

1. Try to collect minimum 5 to 10k text data points.
2. You are encouraged to use Python or R for data cleaning tasks.
3. You can use plugins or open source/free tools for scraping data.
4. For presentations it is recommended (not mandatory) to use MS PPT, LO Impress, Keynote or Google slides. For dashboards, you may use Tableau or Power BI.

The Little Mermaid (2023 film)

The very first teaser trailer for The Little Mermaid was released in September 2022.

The Little Mermaid was released theatrically on May 26, 2023, in the United States by Walt Disney Studios.

It was made available to stream on Disney+ on Wednesday, September 6, 2023.

About the movie:

The Little Mermaid is a 2023 American musical romantic fantasy film directed by Rob Marshall from a screenplay written by David Magee. Co-produced by Walt Disney Pictures, Lucamar Productions, and Marc Platt Productions, it is a live-action adaptation of Disney's 1989 animated film of the same name, which itself is loosely based on the 1837 fairy tale by Hans Christian Andersen. The film stars singer Halle Bailey in the title role, alongside Jonah Hauer-King, Daveed Diggs, Awkwafina, Jacob Tremblay, Noma Dumezweni, Art Malik, Javier Bardem and Melissa McCarthy. The plot follows the mermaid princess Ariel, who is fascinated with the human world; after saving Prince Eric from a shipwreck, she makes a deal with the sea witch Ursula to walk on land.



'The Little Mermaid' breaks records on Disney+ as one of the most viewed Disney movie premieres ever, garnering 16 million views in its first five days streaming

source:

<https://thewaltdisneycompany.com/little-mermaid-disney-plus-numbers/>



Here are the important dates along with the associated activities for the movie The Little Mermaid:

Date	Activity
May 8, 2023	World premiere at the Dolby Theatre in Los Angeles
May 15, 2023	London premiere at the Odeon Luxe Leicester Square
May 26, 2023	Theatrical release in the United States
August 25, 2023	Sing-along edition released in theaters
September 9, 2022	Introduction of "Part of Your World" sequence
October 13, 2022	Release of the first poster
January 21, 2023	Casting call for character look-alike actresses
March 12, 2023	Official trailer shown during the 95th Academy Awards
July 25, 2023	Digital download release by Walt Disney Studios Home Entertainment
September 19, 2023	Release on Ultra HD Blu-ray, Blu-ray, and DVD
September 6, 2023	Release on Disney+

These dates and activities provide a timeline of significant events related to the movie "The Little Mermaid," including premieres, promotional activities, and home media releases.

Importing Required Libraries

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

In [2]: import warnings
warnings.simplefilter('ignore')
```

About YouTube

YouTube is an **online video sharing** and **social media platform** headquartered in San Bruno, California, United States. Accessible worldwide, it was launched on February 14, 2005, by Steve Chen, Chad Hurley, and Jawed Karim. It is owned by **Google** and is the second most visited website in the world, after Google Search. YouTube has more than 2.5 billion monthly users, who collectively watch more than one

billion hours of videos every day. As of May 2019, videos were being uploaded to the platform at a rate of more than 500 hours of content per minute.

Why I chose YouTube as the platform to collect data for the given assignment, which required extracting reviews, tweets, or relevant text data ?

1. **Wide Audience and Engagement:** YouTube is one of the **largest and most popular video-sharing platforms globally**, with a vast and diverse user base. This ensures that we can capture a wide range of opinions and perspectives about the movie, providing a comprehensive view of audience sentiment.
2. **Rich Multimedia Content:** YouTube allows users to express their thoughts through a combination of video content and comments, providing a rich source of textual data. This multimedia aspect can add depth and context to the collected text data.
3. **Accessible Public Data:** YouTube comments are often publicly accessible, making it a convenient and **ethical source of data**. This accessibility aligns with privacy and data usage regulations, ensuring that you collect data within legal and ethical boundaries.
4. **Structured Data:** YouTube comments are typically structured and well-organized, making it easier to extract and analyze compared to unstructured text data on other social media platforms.
5. **Consolidated Location:** YouTube comments for a particular video are centralized, allowing us to focus our data collection efforts on a single source. This streamlines the data extraction process and facilitates organization.
6. **Embedded Social Interaction:** YouTube comments often include interactions between users, such as replies, likes, and dislikes, providing insights into not only individual sentiments but also community sentiment.
7. **No API Restrictions:** Unlike some other social media platforms that may have strict API restrictions, YouTube often allows **web scraping for comments without significant limitations**, enabling us to gather a substantial amount of data.
8. **Duration of Comments:** YouTube comments remain accessible for a longer duration compared to tweets on Twitter, which can be fleeting. This ensures that we can collect and analyze comments over an extended period, capturing evolving opinions.
9. **Search and Filtering Options:** YouTube provides search and filtering options to find relevant comments easily, allowing us to focus on specific aspects of the movie, such as reviews, opinions on particular scenes, or audience engagement.
10. **Visual Content Alignment:** If the movie trailer is available on YouTube, it's a natural choice to collect data from the same platform, as it aligns with the visual content we are analyzing. This ensures that the context of the comments is closely related to the video content.

Source 1

The Little Mermaid | Official Trailer

Walt Disney Studios

Link: https://www.youtube.com/watch?v=kpGo2_d3oYE

```
In [6]: data = pd.DataFrame()

import json

def unnest_json(obj):
    """Unnests a JSON object into a flat dictionary.

    Args:
        obj: A JSON object.

    Returns:
        A flat dictionary containing the unnested JSON data.
    """

    result = {}
    for key, value in obj.items():
        if isinstance(value, dict):
            result.update(unnest_json(value))
        else:
            result[key] = value
    return result

import os
import googleapiclient.discovery

# Disable OAuthLib's HTTPS verification when running locally.
# *DO NOT* Leave this option enabled in production.
os.environ["OAUTHLIB_INSECURE_TRANSPORT"] = "1"

api_service_name = "youtube"
api_version = "v3"
DEVELOPER_KEY = "AIzaSyByZy3qAn_oil_NkwjYdjHzeIMK6oN1098"

youtube = googleapiclient.discovery.build(
    api_service_name, api_version, developerKey=DEVELOPER_KEY)

# You can adjust maxResults to the desired number of comments per request.
max_results_per_request = 50
total_results = 0 # Track the total number of results received

# Start with an empty page token to initiate the first request.
page_token = ""

while True:
    request = youtube.commentThreads().list(
        part="id, snippet",
        order="time",
        videoId="kpGo2_d3oYE",
        maxResults=max_results_per_request,
        pageToken=page_token
    )
    response = request.execute()

    text_original, published_at, author_display_name = [], [], []
    for item in response['items']:
        # Unnest the JSON object
        json_string = json.dumps(item)

        json_obj = json.loads(json_string)
        flat_json = unnest_json(json_obj)
        # Get the textOriginal, publishedAt, and authorDisplayName values
```

```

text_original.append(flat_json['textOriginal'])
published_at.append(flat_json['publishedAt'])
author_display_name.append(flat_json['authorDisplayName'])

data = data.append(pd.DataFrame({'comment': text_original,
                                'published_at': published_at,
                                'user_name': author_display_name}), ignore_index=True)

total_results = len(data)

# Check if there is a next page of results.
page_token = response.get('nextPageToken')

# Check if we have reached the desired total number of results.
if total_results >= 5000:
    break

# Check if there are no more pages to retrieve.
if page_token is None:
    break

```

In [7]: data

Out[7]:

	comment	published_at	user_name
0	Favorite film of the year and my mother just w...	2023-10-15T04:54:36Z	P
1	BEAUTIFUL ♥♥♥♥♥♥♥♥♥♥	2023-10-14T18:30:32Z	AJ Harris
2	What are these comments	2023-10-14T18:09:56Z	♡QueerShaneGlines Fan♡
3	I loved the part where triton jumps up on a cr...	2023-10-14T16:50:05Z	Pranay Vasala
4	Everyone in this comment section:\n" I love th...	2023-10-14T12:30:38Z	Oywiththepoodlesalready
...
5043	I think I'll pass. Terrible reviews.	2023-05-25T12:31:51Z	Christopher Adams
5044	Oh hell nahhh not the black ariel	2023-05-25T12:29:09Z	Adee Enamno
5045	Only one day left before showtime!! I can alr...	2023-05-25T12:13:46Z	sweetangel550
5046	1:30	2023-05-25T12:13:18Z	Laras Andrea
5047	Anyone have any information on Universal's ver...	2023-05-25T11:22:44Z	SakuraandSesshomaru

5048 rows × 3 columns

Source 2:

The Little Mermaid | Part Of Your World

Walt Disney Studios

Link: <https://www.youtube.com/watch?v=qcxr2xuVLUs>

In [8]: data2 = pd.DataFrame()

```

import json

def unnest_json(obj):
    """Unnests a JSON object into a flat dictionary.

    Args:
        obj: A JSON object.

    Returns:
        A flat dictionary containing the unnested JSON data.
    """

```

```

result = {}
for key, value in obj.items():
    if isinstance(value, dict):
        result.update(unnest_json(value))
    else:
        result[key] = value
return result

import os
import googleapiclient.discovery

# Disable OAuthLib's HTTPS verification when running Locally.
# *DO NOT* Leave this option enabled in production.
os.environ["OAUTHLIB_INSECURE_TRANSPORT"] = "1"

api_service_name = "youtube"
api_version = "v3"
DEVELOPER_KEY = "AIzaSyByZy3qAn_oil_NkwjYdjHze1MK6oN1098"

youtube = googleapiclient.discovery.build(
    api_service_name, api_version, developerKey=DEVELOPER_KEY)

# You can adjust maxResults to the desired number of comments per request.
max_results_per_request = 30
total_results = 0 # Track the total number of results received

# Start with an empty page token to initiate the first request.
page_token = ""

while True:
    request = youtube.commentThreads().list(
        part="id, snippet",
        order="relevance",
        videoId="qcxr2xuVLU",
        maxResults=max_results_per_request,
        pageToken=page_token
    )
    response = request.execute()

    text_original, published_at, author_display_name = [], [], []
    for item in response['items']:
        # Unnest the JSON object
        json_string = json.dumps(item)

        json_obj = json.loads(json_string)
        flat_json = unnest_json(json_obj)
        # Get the textOriginal, publishedAt, and authorDisplayName values

        text_original.append(flat_json['textOriginal'])
        published_at.append(flat_json['publishedAt'])
        author_display_name.append(flat_json['authorDisplayName'])

    data2 = data2.append(pd.DataFrame({'comment': text_original,
                                      'published_at': published_at,
                                      'user_name': author_display_name}), ignore_index=True)

    total_results = len(data2)

    # Check if there is a next page of results.
    page_token = response.get('nextPageToken')


    # Check if we have reached the desired total number of results.
    if total_results >= 5000:
        break

    # Check if there are no more pages to retrieve.
    if page_token is None:
        break

```

In [9]: data2

Out[9]:

	comment	published_at	user_name
0	I know this movie has gotten a lot of flack... b...	2023-05-30T23:59:16Z	R. Fazor
1	You cannot deny that she has an amazing voice....	2023-05-12T20:23:19Z	Matt Thornton
2	The way her tail moves and the iridescent colo...	2023-05-11T09:04:08Z	shady boots
3	I love hearing her "swish" through the water. ...	2023-05-16T14:36:24Z	Anastasia's Secret
4	Watching this in the theater literally gave me...	2023-05-28T16:02:36Z	ashley Apolinar
...
1239		2023-05-10T00:14:29Z	Reverend Kush
1240	so bad	2023-05-24T00:22:05Z	Marta Recuero Arias
1241	FLOPPED	2023-05-31T22:15:21Z	red velvet
1242	Absolutely awfull	2023-05-09T05:14:46Z	Grzegorz Piechowski
1243	Crap	2023-06-05T15:48:59Z	Baba Yaga

1244 rows × 3 columns


In [10]: *# Combining the records collected from the two sources*

```
df = data.append(data2)
```

In [11]: *# Glance on the combined data*

```
df
```

Out[11]:

	comment	published_at	user_name
0	Favorite film of the year and my mother just w...	2023-10-15T04:54:36Z	P
1	BEAUTIFUL ♥♥♥♥♥♥♥♥♥♥	2023-10-14T18:30:32Z	AJ Harris
2	What are these comments	2023-10-14T18:09:56Z	♥QueerShaneGlines Fan♥
3	I loved the part where triton jumps up on a cr...	2023-10-14T16:50:05Z	Pranay Vasala
4	Everyone in this comment section:\n" I love th...	2023-10-14T12:30:38Z	Oywiththepoodlesalready
...
1239		2023-05-10T00:14:29Z	Reverend Kush
1240	so bad	2023-05-24T00:22:05Z	Marta Recuero Arias
1241	FLOPPED	2023-05-31T22:15:21Z	red velvet
1242	Absolutely awfull	2023-05-09T05:14:46Z	Grzegorz Piechowski
1243	Crap	2023-06-05T15:48:59Z	Baba Yaga

6292 rows × 3 columns


In [13]: *# The records which was collected and stored the previous day*

```
df1 = pd.read_csv(r"C:\Users\Karan\Desktop\GroupM\GroupM_data_2636.csv").drop(columns = ['Unna
```

In [16]: *# Viewing the data which was collected the previous day*

```
df1
```


Out[16]:

	comment	published_at	user_name
0	I love the part where Vin Diesel says "Fish an...	2023-03-15T07:32:48Z	Mighty Car Mods
1	I love the part when Ariel came close to Eric ...	2023-09-22T01:11:33Z	FucXYou
2	I love the part where Eric goes back inside an...	2023-10-05T11:01:11Z	TakeALiPi
3	I love the part when Ariel says"I am the storm...	2023-10-05T07:30:14Z	tea_dealer
4	I love the part where Ariel says "With great p...	2023-09-06T18:34:36Z	Lerato Moroalo
...
2631		2023-05-10T00:14:29Z	Reverend Kush
2632	so bad	2023-05-24T00:22:05Z	Marta Recuero Arias
2633	FLOPPED	2023-05-31T22:15:21Z	red velvet
2634	Absolutely awfull	2023-05-09T05:14:46Z	Grzegorz Piechowski
2635	Crap	2023-06-05T15:48:59Z	Baba Yaga

2636 rows × 3 columns

In [14]:

```
# Merging both the data and dropping the duplicates

df = df.append(df1).drop_duplicates(keep = 'first', ignore_index = True)
```

In [15]:

```
# Viewing the data after merging the data and dropping the duplicates

df
```

Out[15]:

	comment	published_at	user_name
0	Favorite film of the year and my mother just w...	2023-10-15T04:54:36Z	P
1	BEAUTIFUL ♥♥♥♥♥♥♥♥♥♥	2023-10-14T18:30:32Z	AJ Harris
2	What are these comments	2023-10-14T18:09:56Z	♥QueerShaneGlines Fan♥
3	I loved the part where triton jumps up on a cr...	2023-10-14T16:50:05Z	Pranay Vasala
4	Everyone in this comment section:\n" I love th...	2023-10-14T12:30:38Z	Oywiththepoodlesalready
...
6557	0:11 HOMEBOY IS ON FIRE YOO	2023-03-13T21:40:11Z	MIKE HAWK
6558	1:16 Ariel Mermaid Turns Into A Human Legs	2023-03-13T01:52:58Z	Toon Disney & 4KIDS TV
6559	0:55 flounder sounds like Luca	2023-03-15T02:49:25Z	David galindo
6560	2 million dislikes lol	2023-05-18T15:49:18Z	David Zuniga
6561	1:57 it seemed so cringe and awkward to me. Ev...	2023-03-23T18:07:26Z	Saadia Satter Sathi

6562 rows × 3 columns

In [17]:

```
# Storing the current data so that it can be merged with the future collected data

df.to_csv(r"C:\Users\Karan\Desktop\GroupM\GroupM_data6562.csv")
```

Now that we've amassed a substantial dataset from YouTube trailer videos, it's time to embark on some preliminary data exploration.

Data Dictionary:

Column Name	Information
comment	The original, raw text of the comment as it was initially posted or last updated
published_at	The date and time when the comment was originally published. The value is specified in ISO 8601 format.
user_name	The display name of the user who posted the comment

Shape of the Dataset

```
In [18]: # shape of the dataset
df.shape
```

```
Out[18]: (6562, 3)
```

Dataset info

```
In [19]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6562 entries, 0 to 6561
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype  
---  -
 0   comment       6562 non-null   object  
 1   published_at   6562 non-null   object  
 2   user_name     6562 non-null   object  
dtypes: object(3)
memory usage: 153.9+ KB
```

- There are no missing values in the dataset

```
In [21]: # Since the published_at feature is not in the desired datetime format

df['published_at'] = pd.to_datetime(df['published_at'])
```

Statistical Description of the data

```
In [22]: df.describe()
```

```
Out[22]:
```

	comment	published_at	user_name
count	6562	6562	6562
unique	6395	6549	5772
top	👉	2023-05-28 10:24:25+00:00	sweetangel550
freq	41	2	78
first	NaN	2023-03-13 01:52:58+00:00	NaN
last	NaN	2023-10-15 04:54:36+00:00	NaN

Conclusions that can be drawn from the statistical description of the "comment," "published_at," and "user_name" features:

1. **Count:** There are a total of 6,562 rows of data in the dataset, which indicates the number of comments, published times, and user names available for analysis.

2. Unique Values:

- For the 'comment' column, there are 6,395 unique comments. This suggests that there is some degree of repetition in the comments, with several comments being repeated multiple times.
- In the 'published_at' column, there are 6,549 unique timestamps, which suggests that there are some duplicated timestamps in the dataset.
- For the 'user_name' column, there are 5,772 unique user names, indicating that multiple users share the same name or that some user names are repeated.

3. Top Values:

- The most frequent comment in the 'comment' column is "😏," which appears 41 times in the dataset. This emoji seems to be a commonly used comment.
- The most frequent timestamp in the 'published_at' column is "2023-05-28 10:24:25+00:00," which appears twice in the dataset.
- The most common user name in the 'user_name' column is "sweetangel550," occurring 78 times, indicating that this user is quite active.

4. First and Last Timestamps:

- The first timestamp in the 'published_at' column is "2023-03-13 01:52:58+00:00," suggesting that the dataset starts from this date.
- The last timestamp in the 'published_at' column is "2023-10-15 04:54:36+00:00," indicating that the dataset ends on this date.

Data Cleaning

Replacing emojis in the comment feature with their textual representation

```
In [26]: # Sample DataFrame with a column containing text with emojis
data = df

# Function to replace emojis with their textual representations
def replace_emojis_with_text(text):
    return emoji.demojize(text)

# Apply the function to the DataFrame column
data['comment'] = data['comment'].apply(replace_emojis_with_text)

# Print the updated feature
data['comment'].head()
```

```
Out[26]: 0    Favorite film of the year and my mother just w...
1    BEAUTIFUL :heart_suit::heart_suit::heart_suit:...
2                What are these comments
3    I loved the part where triton jumps up on a cr...
4    Everyone in this comment section:\n" I love th...
Name: comment, dtype: object
```

Is there any missing value in the dataset?

```
In [28]: df.isna().any()
```

```
Out[28]: comment          False
published_at        False
user_name           False
dtype: bool
```

Is there any duplicate value in the dataset?

```
In [29]: df.duplicated().any()
```

Out[29]: False

```
In [27]: df.head()
```

Out[27]:

	comment	published_at	user_name
0	Favorite film of the year and my mother just w...	2023-10-15 04:54:36+00:00	P
1	BEAUTIFUL :heart_suit::heart_suit::heart_suit:...	2023-10-14 18:30:32+00:00	AJ Harris
2	What are these comments	2023-10-14 18:09:56+00:00	♡QueerShaneGlines Fan♡
3	I loved the part where triton jumps up on a cr...	2023-10-14 16:50:05+00:00	Pranay Vasala
4	Everyone in this comment section:\n" I love th...	2023-10-14 12:30:38+00:00	Oywiththepoodlesalready

```
In [17]: # df = pd.read_csv(r"C:\Users\Karan\Desktop\GroupM\GroupM_data6562_cleaned.csv").drop(columns
# df['published_at'] = pd.to_datetime(df['published_at'])
```

Let's segment the data based on time intervals.

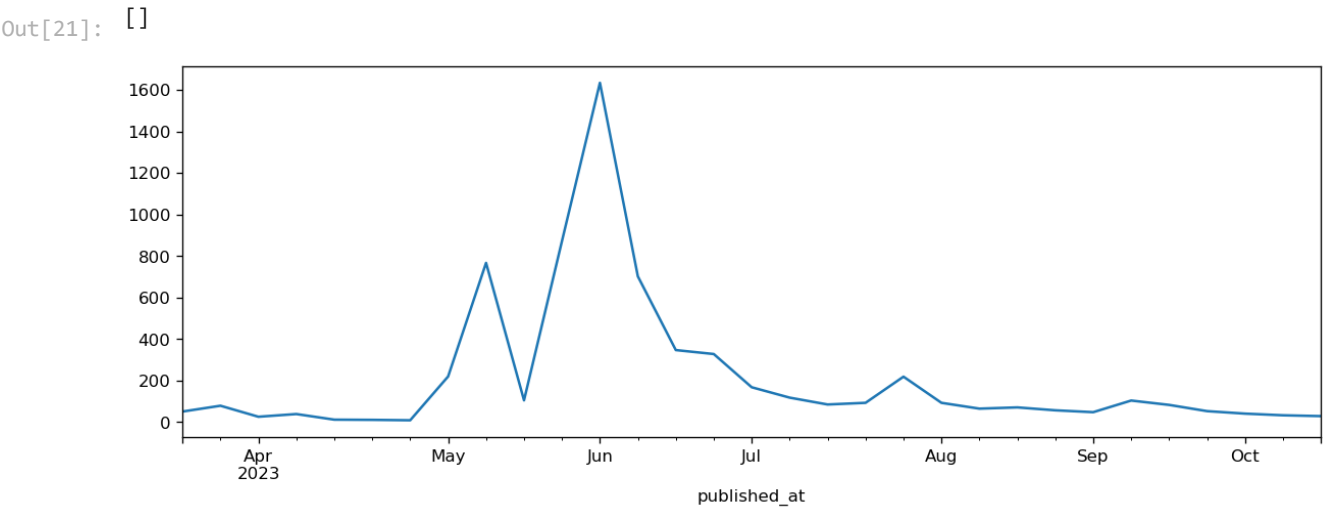
```
In [18]: df1 = df.set_index('published_at')
```

```
In [19]: df1.head()
```

Out[19]:

	comment	user_name
published_at		
2023-10-15 04:54:36+00:00	Favorite film of the year and my mother just w...	P
2023-10-14 18:30:32+00:00	BEAUTIFUL :heart_suit::heart_suit::heart_suit:...	AJ Harris
2023-10-14 18:09:56+00:00	What are these comments	♡QueerShaneGlines Fan♡
2023-10-14 16:50:05+00:00	I loved the part where triton jumps up on a cr...	Pranay Vasala
2023-10-14 12:30:38+00:00	Everyone in this comment section:\n" I love th...	Oywiththepoodlesalready

```
In [21]: plt.figure(figsize = (12, 4), dpi = 120)
df1.resample('W')['comment'].count().plot()
plt.plot()
```



Insights from the above plot:

- 1. **Peak Engagement Pre-Theatrical Launch:** We observed a significant surge in user engagement, with a notably high number of comments being posted just prior to the theatrical launch of the

movie on May 26, 2023. This heightened activity suggests that anticipation and excitement were building up among the audience in the lead-up to the release.

- The surge in comments before the theatrical launch indicates that users were actively discussing the movie, sharing their expectations, and generating buzz on the platform.
2. **Post-Launch Engagement Trend:** Following the theatrical launch, user engagement continued to rise, reaching its peak shortly after the movie's release. However, we observed a gradual and steady decline in the number of comments over the course of approximately one month.
- The sustained increase in comments after the movie's release suggests that users were actively sharing their thoughts, reviews, and reactions to the movie. The subsequent decline may be attributed to the initial burst of user activity gradually tapering off as the movie garnered initial feedback and attention.
3. **Intermittent Peaks Coinciding with Release Dates:** Throughout our analysis, we observed two more distinct peaks in user engagement that correlate with specific release dates associated with "The Little Mermaid."
- **July Peak:** A notable peak in user engagement was observed at the end of July. This surge aligns with the digital download release of "The Little Mermaid" by Walt Disney Studios Home Entertainment on July 25, 2023.
 - The sharp increase in user comments during this period suggests a heightened level of user activity, likely driven by the excitement surrounding the availability of the movie in a digital format. Users were actively discussing and sharing their reactions to the digital release.
 - **September Peak:** Another relatively smaller peak was observed in September, corresponding to the release of "The Little Mermaid" on Ultra HD Blu-ray, Blu-ray, and DVD, which occurred on September 19, 2023.
 - The presence the peak in user engagement during this timeframe indicates a renewed wave of user interactions, possibly due to the physical media release of the movie. This demonstrates ongoing interest and engagement with the film as different formats became accessible to the audience.

In conclusion, the intermittent peaks in user engagement aligning with specific release dates demonstrate the enduring and multifaceted appeal of "The Little Mermaid" among audiences. These insights shed light on the impact of different release formats and the sustained interest in the film. These insights help us understand the dynamics of user engagement before and after the movie's theatrical launch, highlighting the periods of heightened activity and the subsequent trends in user comments on the platform.

From the above insights, several conclusions can be drawn regarding the user engagement with the movie "The Little Mermaid" on the platform:

1. **Anticipation and Excitement Pre-Theatrical Launch:**

- The surge in user engagement, with a notably high number of comments, just before the theatrical launch of the movie on May 26, 2023, indicates a strong sense of anticipation and excitement among the audience.
- Users were actively discussing the movie, sharing their expectations, and generating buzz on the platform. This heightened activity suggests that the movie generated significant pre-release attention and enthusiasm.

2. **Sustained Post-Launch Engagement:**

- Following the theatrical launch, user engagement continued to rise, reaching its peak shortly after the movie's release. This reflects the immediate impact and initial response to the film's

availability.

- The sustained increase in comments after the movie's release demonstrates that users were actively sharing their thoughts, reviews, and reactions to the movie. It suggests ongoing engagement and the film's ability to maintain user interest post-launch.

3. Release Date Impact:

- The analysis of user engagement revealed two additional distinct peaks that correlate with specific release dates associated with "The Little Mermaid."
- The notable peak at the end of July aligns with the digital download release, emphasizing the impact of different release formats on user activity. The excitement surrounding the availability of the movie in a digital format drove a surge in user comments.
- The relatively smaller peak in September coincided with the release of the movie on physical media, indicating that different release formats contributed to renewed waves of user interactions. This demonstrates the enduring appeal of the film across various platforms and formats.

In summary, the insights underscore the movie's ability to generate anticipation, maintain user engagement post-launch, and leverage different release formats to sustain interest. The analysis provides valuable information about the movie's impact on the platform and its enduring popularity among the audience.

Creating a new feature comment_length

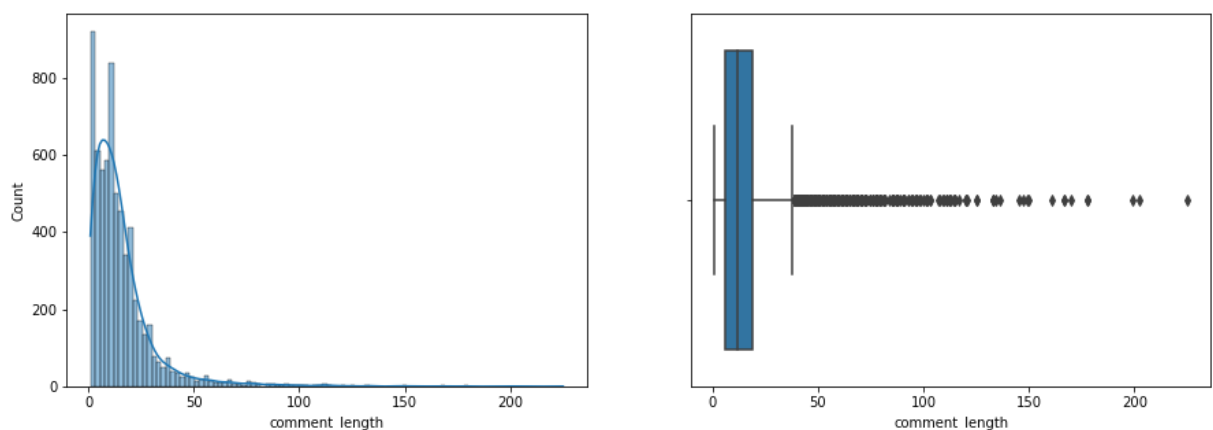
```
In [31]: df['comment_length'] = df['comment'].apply(lambda x: len(x.split()))
df['comment_length'].head()
```

```
Out[31]: 0    14
1     2
2     4
3    27
4    11
Name: comment_length, dtype: int64
```

Let us see how the original distribution of comment length looks like:

```
In [36]: plt.figure(figsize = (15, 5))
plt.subplot(1, 2, 1)
sns.histplot(df['comment_length'], kde = True, bins = 100)
plt.subplot(1, 2, 2)
sns.boxplot(df['comment_length'])
plt.plot()
```

```
Out[36]: []
```



- The distribution of comment length exhibits a significant right skew.

- In simpler terms, the majority of user comments fall within the range of 6 (25th percentile) to 19 (75th percentile) words.
- Conversely, only a small number of users have posted considerably lengthy comments.

It can be seen from the above plots that the sample does not come from normal distribution.

Checking if the distribution follows a log normal distribution

```
In [43]: log_transform = np.log(df['comment_length'])
```

- Applying Shapiro-Wilk test for normality

H_0 : The sample **follows normal distribution**

H_1 : The sample **does not follow normal distribution**

alpha = 0.05

Test Statistics : **Shapiro-Wilk test for normality**

```
In [41]: import scipy.stats as spy
```

```
In [44]: test_stat, p_value = spy.shapiro(log_transform)
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

p-value 2.4254452269856836e-36

The sample does not follow normal distribution

Determining the mean number of words in the comment made by each user

```
In [26]: # The code snippet performs a loop to calculate the mean number of words for different
# sample sizes of users
```

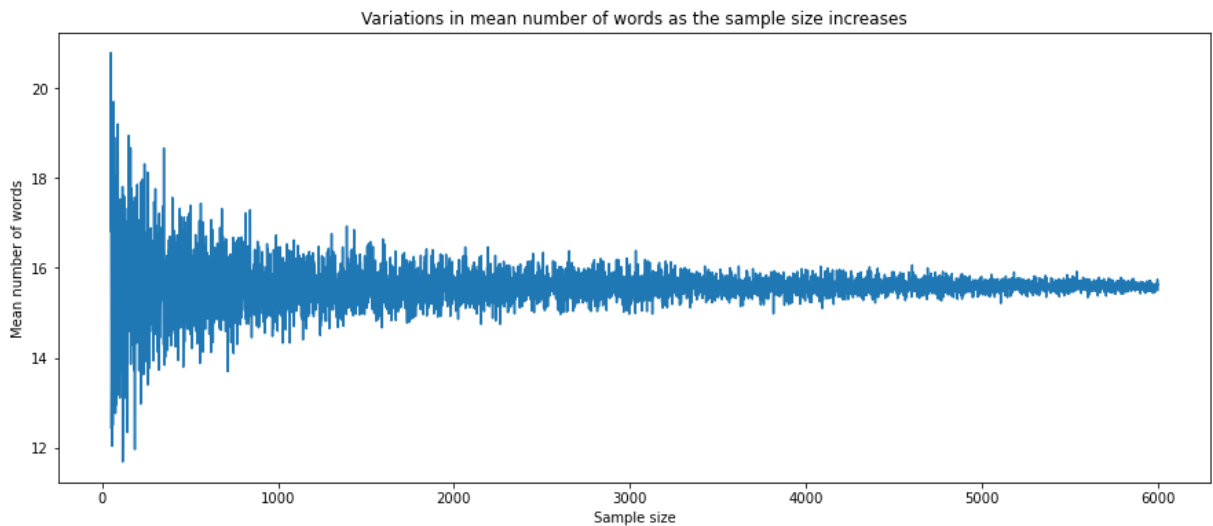
```
mean_words = []
for sample_size in range(50, 6000):
    sample_mean = df['comment_length'].sample(sample_size).mean()
    mean_words.append(sample_mean)

# It iterates over a range of sample sizes from 50 to 6000, and for each iteration,
# it takes a random sample of the specified size from the 'comment_length' column
# of the DataFrame and calculates the mean of the sampled values.
# The calculated mean values are then stored in the 'mean_purchases' List.
```

```
In [27]: # Creating a plot using matplotlib to visualize the trend of the mean number of words
# as the sample size increases
```

```
plt.figure(figsize = (15, 6))
plt.title('Variations in mean number of words as the sample size increases')
plt.plot(np.arange(50, 6000), mean_words)
plt.xlabel('Sample size')
plt.ylabel('Mean number of words')
plt.plot()
```

```
Out[27]: []
```



It can be inferred from the above plot that as the sample size is small the deviations are fairly high. As the sample size increases, the deviation becomes smaller and smaller.

Finding the confidence interval of average number of words in each comment

```
In [28]: means_comment = []
size = df['comment_length'].shape[0]
for bootstrapped_sample in range(10000):
    sample_mean = df['comment_length'].sample(size, replace = True).mean()
    means_comment.append(sample_mean)
```

```
In [29]: # The below code generates a histogram plot with kernel density estimation and
# adds vertical lines to represent confidence intervals at 90%, 95%, and 99% Level

plt.figure(figsize = (15, 6)) # setting the figure size of the plot

sns.histplot(means_comment, kde = True, bins = 100, fill = True, element = 'step')

# Above line plots a histogram of the data contained in the `means_comment` variable.
# The `kde=True` argument adds a kernel density estimation line to the plot.
# The `bins=100` argument sets the number of bins for the histogram

# Above line calculates the z-score corresponding to the 90% confidence level using the
# inverse of the cumulative distribution function (CDF) of a standard normal distribution

ll_90 = np.percentile(means_comment, 5)
# calculating the lower limit of the 90% confidence interval
ul_90 = np.percentile(means_comment, 95)
# calculating the upper limit of the 90% confidence interval
plt.axvline(ll_90, label = f'11_90 : {round(ll_90, 2)}', linestyle = '--')
# adding a vertical line at the lower limit of the 90% confidence interval
plt.axvline(ul_90, label = f'u1_90 : {round(ul_90, 2)}', linestyle = '--')
# adding a vertical line at the upper limit of the 90% confidence interval

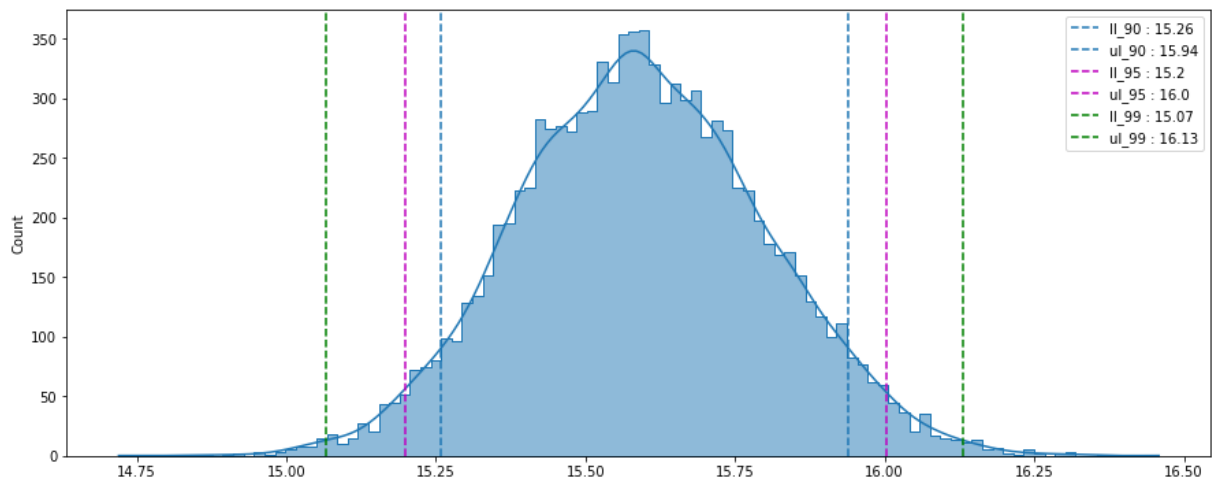
# Similar steps are repeated for calculating and plotting the 95% and 99% confidence intervals
# with different line colors (`color='m'` for 95% and `color='g'` for 99%)

ll_95 = np.percentile(means_comment, 2.5)
ul_95 = np.percentile(means_comment, 97.5)
plt.axvline(ll_95, label = f'11_95 : {round(ll_95, 2)}', linestyle = '--', color = 'm')
plt.axvline(ul_95, label = f'u1_95 : {round(ul_95, 2)}', linestyle = '--', color = 'm')

ll_99 = np.percentile(means_comment, 0.5)
ul_99 = np.percentile(means_comment, 99.5)
plt.axvline(ll_99, label = f'11_99 : {round(ll_99, 2)}', linestyle = '--', color = 'g')
plt.axvline(ul_99, label = f'u1_99 : {round(ul_99, 2)}', linestyle = '--', color = 'g')

plt.legend() # displaying a legend for the plotted lines.
plt.plot() # displaying the plot.
```


Out[29]: []



- Through the bootstrapping method, we have been able to estimate the average number of words in each comment made by each user, despite having data for only 6562 users. This provides us with a reasonable approximation of the range within which the average number of words in each comment made by each user falls, with a certain level of confidence

```
In [31]: print(f"The Mean of total number of words in each comment for each user will be approximately
The Mean of total number of words in each comment for each user will be approximately = 16.0
```

Sentiment Analysis Using Textblob

```
In [33]: # Import the necessary libraries

from textblob import TextBlob
```

```
In [34]: df2 = df.copy()

# Define a function for sentiment analysis with custom thresholds
def analyze_sentiment(comment):
    # Analyze the sentiment of the comment using TextBlob
    analysis = TextBlob(comment)

    # Set custom thresholds for sentiment classification
    if analysis.sentiment.polarity > 0.1:
        return "Positive"
    elif analysis.sentiment.polarity < -0.1:
        return "Negative"
    else:
        return "Neutral"

# Apply sentiment analysis to the DataFrame and create a new 'sentiment' column
df2['sentiment'] = df2['comment'].apply(analyze_sentiment)

# Display the DataFrame with the 'comment' and 'sentiment' columns
df2[['comment', 'sentiment']].head()
```

```
Out[34]:
```

	comment	sentiment
0	Favorite film of the year and my mother just w...	Positive
1	BEAUTIFUL :heart_suit::heart_suit::heart_suit:...	Positive
2	What are these comments	Neutral
3	I loved the part where triton jumps up on a cr...	Positive
4	Everyone in this comment section:\n" I love th...	Positive

```
In [58]: # Create a DataFrame 'df_emp_length' to hold the counts of each 'emp_length' category
df2_sentiment = df2['sentiment'].value_counts().to_frame().reset_index().sort_values(by = 'index')

# Calculate the percentage contribution of each "sentiment" category
df2_sentiment['perc_contribution'] = np.round(df2_sentiment['sentiment'] * 100 / df2_sentiment['index'])

# Create a bar plot to visualize the percentage contribution of each "emp Length" category
plt.figure(figsize=(15, 8), dpi = 150)
plt.title('Distribution of sentiment types',
          weight = 'semibold', family = 'serif', size = 14, color='darkgreen')

color_map = sns.color_palette('Greens', as_cmap = True)

bar_plot = sns.barplot(data=df2_sentiment,
                       x=df2_sentiment['index'],
                       y=df2_sentiment['perc_contribution'],
                       palette = color_map(df2_sentiment['perc_contribution'] * 0.02))

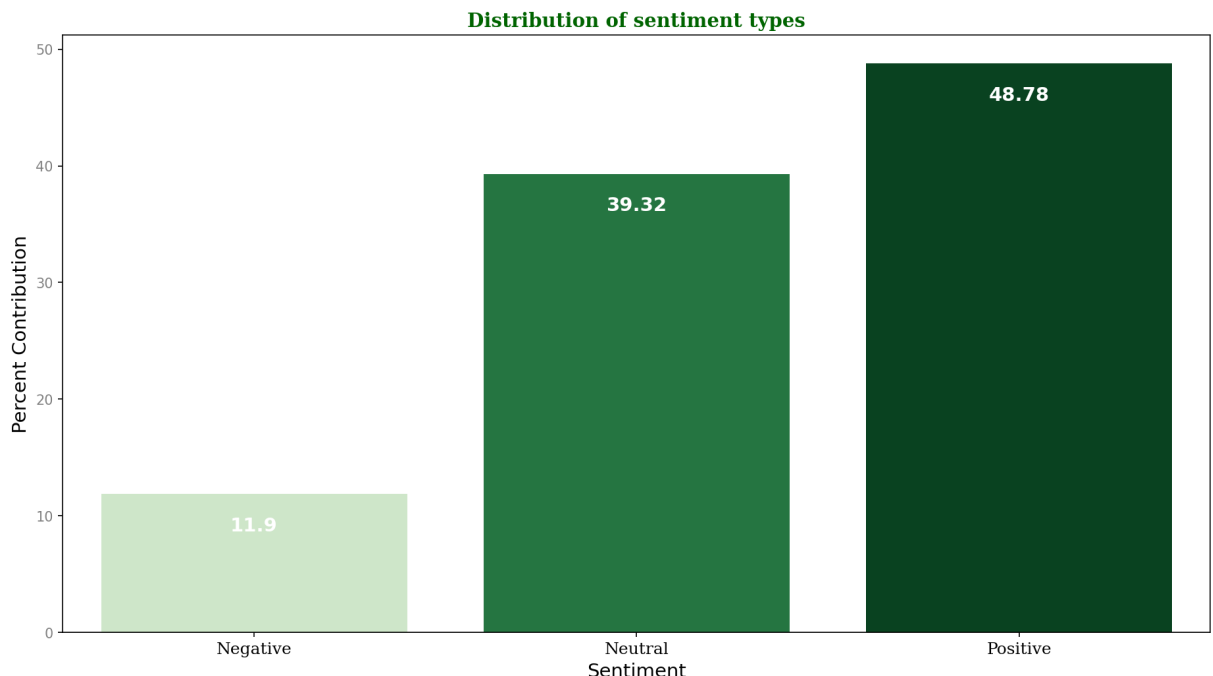
# Add percentage labels to the bars on the bar plot
plt.bar_label(container=bar_plot.containers[0],
              labels=df2_sentiment['perc_contribution'],
              color='white', size=14, padding = - 30, weight = 'semibold')

# Set Labels for the x-axis and y-axis
plt.ylabel('Percent Contribution', size=14)
plt.xlabel('Sentiment', size=14)

# Set font sizes for y-axis ticks
plt.xticks(size=12, family='serif')

# De-emphasizing values in y axis
plt.yticks(color = 'gray')

# Display the bar plot
plt.show()
```



Sentiment Analysis Using NLTK's SentimentIntensityAnalyzer

```
In [27]: from nltk.sentiment.vader import SentimentIntensityAnalyzer

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize
```

```
from nltk.stem import WordNetLemmatizer
```

```
In [35]: df2 = df.copy()

# Source: https://www.datacamp.com/tutorial/text-analytics-beginners-nltk
# Author: Moez Ali (Data Scientist, Founder & Creator of PyCaret)
# This code was adapted from the example provided on the website.

# create preprocess_text function
def preprocess_text(text):

    # Tokenize the text

    tokens = word_tokenize(text.lower())

    # Remove stop words

    filtered_tokens = [token for token in tokens if token not in stopwords.words('english')]

    # Lemmatize the tokens

    lemmatizer = WordNetLemmatizer()

    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in filtered_tokens]

    # Join the tokens back into a string

    processed_text = ' '.join(lemmatized_tokens)

    return processed_text

# apply the function df
df2['comment'] = df2['comment'].apply(preprocess_text)

df2
```

Out[35]:

	comment	published_at	user_name	comment_length
0	favorite film year mother watched love .	2023-10-15 04:54:36+00:00	P	14
1	beautiful : heart_suit : :heart_suit : :heart_...	2023-10-14 18:30:32+00:00	AJ Harris	2
2	comment	2023-10-14 18:09:56+00:00	♡QueerShaneGlines Fan♡	4
3	loved part triton jump cruise ship , stand rai...	2023-10-14 16:50:05+00:00	Pranay Vasala	27
4	everyone comment section : " love part "	2023-10-14 12:30:38+00:00	Oywiththepoodlesalready	11
...
6557	0:11 homeboy fire yoo	2023-03-13 21:40:11+00:00	MIKE HAWK	6
6558	1:16 ariel mermaid turn human leg	2023-03-13 01:52:58+00:00	Toon Disney & 4K!DS TV	8
6559	0:55 flounder sound like luca	2023-03-15 02:49:25+00:00	David galindo	5
6560	2 million dislike lol	2023-05-18 15:49:18+00:00	David Zuniga	4
6561	1:57 seemed cringe awkward . even hair ai n't ...	2023-03-23 18:07:26+00:00	Saadia Satter Sathi	14

6562 rows × 4 columns

In [36]:

```

# Create a SentimentIntensityAnalyzer object
analyzer = SentimentIntensityAnalyzer()

# Define a function to get sentiment
def get_sentiment(text):
    scores = analyzer.polarity_scores(text)

    if scores['compound'] > 0.1:
        sentiment = 1 # Positive sentiment
    elif scores['compound'] < -0.1:
        sentiment = -1 # Negative sentiment
    else:
        sentiment = 0 # Neutral sentiment

    return sentiment

# Apply get_sentiment function to your DataFrame
df2['sentiment'] = df2['comment'].apply(get_sentiment)

# View the DataFrame with sentiment values
df2.head()

```

Out[36]:

	comment	published_at	user_name	comment_length	sentiment
0	favorite film year mother watched love .	2023-10-15 04:54:36+00:00	P	14	1
1	beautiful : heart_suit : :heart_suit : :heart_...	2023-10-14 18:30:32+00:00	AJ Harris	2	1
2	comment	2023-10-14 18:09:56+00:00	♡QueerShaneGlines Fan♡	4	0
3	loved part triton jump cruise ship , stand rai...	2023-10-14 16:50:05+00:00	Pranay Vasala	27	1
4	everyone comment section : " love part "	2023-10-14 12:30:38+00:00	Oywiththepoodlesalready	11	1

In [37]:

```
# Create a DataFrame 'df_emp_Length' to hold the counts of each 'emp_Length' category
df2_sentiment = df2['sentiment'].value_counts().to_frame().reset_index().sort_values(by = 'index')

# Calculate the percentage contribution of each "sentiment" category
df2_sentiment['perc_contribution'] = np.round(df2_sentiment['sentiment'] * 100 / df2_sentiment['sentiment'].sum(), 2)

# Create a bar plot to visualize the percentage contribution of each "emp Length" category
plt.figure(figsize=(15, 8), dpi = 150)
plt.title('Distribution of sentiment types',
          weight = 'semibold', family = 'serif', size = 14, color='darkgreen')

color_map = sns.color_palette('Greens', as_cmap = True)

bar_plot = sns.barplot(data=df2_sentiment,
                       x=df2_sentiment['index'],
                       y=df2_sentiment['perc_contribution'],
                       palette = color_map(df2_sentiment['perc_contribution'] * 0.02))

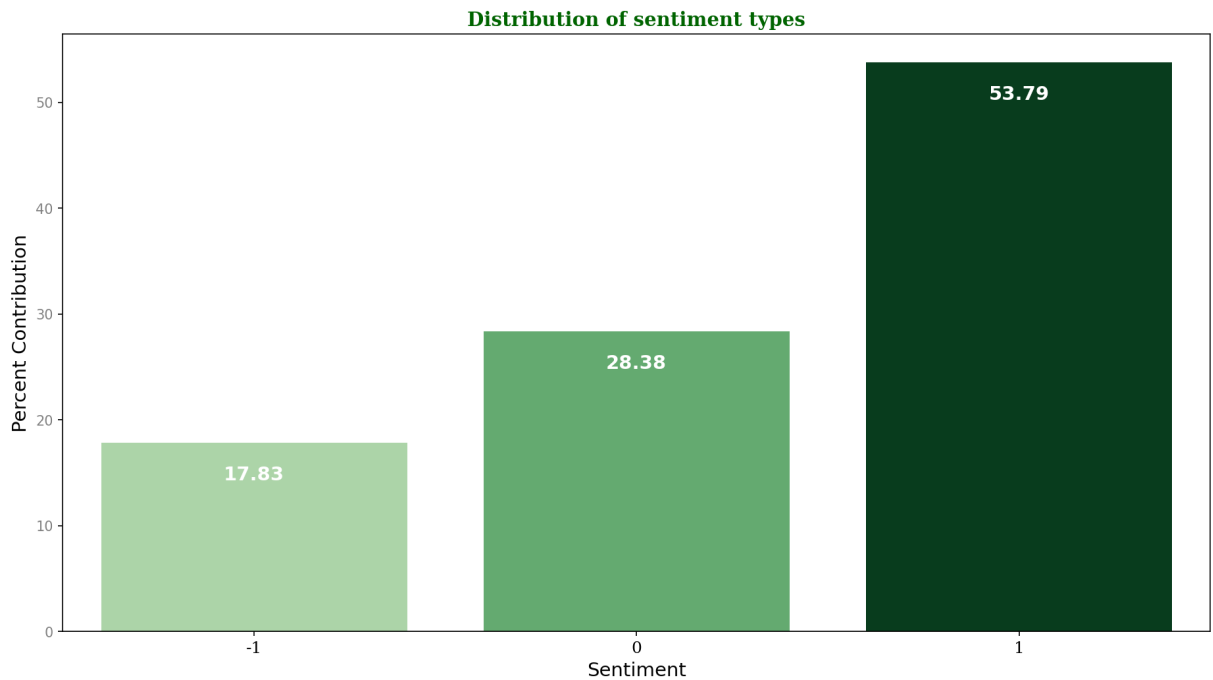
# Add percentage Labels to the bars on the bar plot
plt.bar_label(container=bar_plot.containers[0],
              labels=df2_sentiment['perc_contribution'],
              color='white', size=14, padding = - 30, weight = 'semibold')

# Set Labels for the x-axis and y-axis
plt.ylabel('Percent Contribution', size=14)
plt.xlabel('Sentiment', size=14)

# Set font sizes for y-axis ticks
plt.xticks(size=12, family='serif')

# De-emphasizing values in y axis
plt.yticks(color = 'gray')

# Display the bar plot
plt.show()
```



From the above sentiment plot for the comments on the movie "The Little Mermaid," several conclusions can be drawn:

1. Sentiment Distribution:

- The sentiment data is distributed across three categories: Negative (-1), Neutral (0), and Positive (1).

2. Predominantly Positive Sentiment:

- A significant portion of the comments (53.79%) fall under the "Positive" sentiment category (1). This suggests that a majority of users expressed positive sentiment in their comments about the movie.

3. Substantial Neutral Sentiment:

- A substantial proportion of comments (28.38%) are classified as "Neutral" (0). This indicates that a considerable number of users provided comments that did not strongly express either positive or negative sentiment.

4. Relatively Lower Negative Sentiment:

- A smaller fraction of comments (17.83%) are categorized as "Negative" sentiment (-1). While negative sentiment is present, it is less prevalent compared to positive and neutral sentiments.

5. Overall Positive Tone:

- The distribution of sentiment data suggests an overall positive tone and reception of the movie among the online community. The combined percentage of positive and neutral sentiments (82.17%) significantly outweighs the negative sentiment (17.83%).

6. User Engagement:

- The predominance of positive sentiment may indicate that users were generally pleased, excited, or satisfied with the movie, which could contribute to higher user engagement, positive reviews, and word-of-mouth recommendations.

7. Potential Improvement Opportunities:

- The presence of negative sentiment, albeit lower in percentage, indicates areas where improvements or feedback may be valuable. Analyzing negative comments can provide insights into aspects of the movie that could be enhanced.

In summary, the sentiment data suggests that "The Little Mermaid" movie generated a predominantly positive response among the online community, with a significant number of users expressing positive or neutral sentiments. This positive sentiment can contribute to the movie's success and popularity. However, it's essential to consider negative sentiment for potential areas of improvement and feedback.

```
In [20]: # Import necessary libraries
import pandas as pd
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from gensim.summarization import keywords

# Assuming you have a DataFrame 'df' with a 'comment' column
comments = df['comment']

# Tokenize the comments
tokenized_comments = comments.apply(word_tokenize)

# Convert tokenized comments to text
tokenized_comments = tokenized_comments.apply(' '.join)

# TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=100, stop_words='english')
tfidf_matrix = vectorizer.fit_transform(tokenized_comments)

# LDA Topic Modeling
lda = LatentDirichletAllocation(n_components=5, random_state=42)
lda.fit(tfidf_matrix)

# Get the common keywords for each topic
common_keywords = []

for topic in lda.components_:
    top_keyword_indices = topic.argsort()[-5:][::-1] # Adjust the number of top keywords
    top_keywords = [vectorizer.get_feature_names_out()[i] for i in top_keyword_indices]
    common_keywords.append(top_keywords)

# Display common keywords for each topic
for i, keywords in enumerate(common_keywords):
    print(f"Topic {i+1}: {' '.join(keywords)}")

# Create word cloud for common keywords
all_keywords = [keyword for keywords in common_keywords for keyword in keywords]

print("All Keywords: ", all_keywords)
```

```
Topic 1: ariel, loved, said, love, hair
Topic 2: disney, movie, really, wait, make
Topic 3: movie, red_heart, face_with_tears_of_joy, dislikes, looks
Topic 4: mermaid, little, comments, like, voice
Topic 5: love, says, ariel, crying_face, woke
All Keywords: ['ariel', 'loved', 'said', 'love', 'hair', 'disney', 'movie', 'really', 'wait', 'make', 'movie', 'red_heart', 'face_with_tears_of_joy', 'dislikes', 'looks', 'mermaid', 'little', 'comments', 'like', 'voice', 'love', 'says', 'ariel', 'crying_face', 'woke']
```

Based on the common keywords extracted from the comments related to the movie "The Little Mermaid," we can draw several conclusions and insights:

1. Topic 1: Ariel's Character and Appearance:

- The keywords in Topic 1, including "ariel," "loved," "love," and "hair," suggest that users are discussing and expressing affection for the character Ariel. This could indicate that Ariel's character and her visual appearance in the movie were significant points of discussion.

2. Topic 2: Disney and Movie Anticipation:

- Keywords like "disney," "movie," "really," and "wait" in Topic 2 indicate that users were discussing their excitement and anticipation for the Disney movie. This suggests that the Disney brand and the movie's release generated considerable buzz.

3. Topic 3: Mixed Sentiments and Dislikes:

- Topic 3 includes keywords like "movie," "dislikes," and "face_with_tears_of_joy," indicating that there might be mixed sentiments about the movie. Users could be discussing both positive and negative aspects, with some expressing humor.

4. Topic 4: "Little Mermaid" References:

- Keywords such as "mermaid," "little," and "voice" in Topic 4 are closely related to the movie's title, "The Little Mermaid." Users might be discussing elements specific to the movie's storyline and characters.

5. Topic 5: Love for Ariel:

- Keywords in Topic 5, including "love," "ariel," and "crying_face," reveal users' strong emotional connections with Ariel's character. This topic suggests that many viewers have a deep affection for the character.

Overall Insights:

- The keywords reflect a mixture of sentiments, including love and anticipation, as well as potential areas of discussion and criticism.
- The strong presence of Ariel's character and references to the Disney brand highlight their significance in generating user engagement.
- The keywords are consistent with the context of a movie release, and they align with the expected topics related to "The Little Mermaid."
- The presence of both positive and negative sentiment-related keywords suggests that users are discussing a variety of aspects, contributing to a well-rounded online conversation about the movie.

In summary, the common keywords extracted from the comments indicate that discussions around "The Little Mermaid" movie are multifaceted, covering aspects such as character appreciation, Disney's influence, and mixed sentiments. These insights provide a valuable understanding of the topics generating the most discussion and engagement among the online community.

Insights

- There are a total of 6,562 rows of data in the dataset, which indicates the number of comments, published times, and user names available for analysis.
- For the 'comment' column, there are 6,395 unique comments. This suggests that there is some degree of repetition in the comments, with several comments being repeated multiple times.
- In the 'published_at' column, there are 6,549 unique timestamps, which suggests that there are some duplicated timestamps in the dataset.
- For the 'user_name' column, there are 5,772 unique user names, indicating that multiple users share the same name or that some user names are repeated.
- The most frequent comment in the 'comment' column is "😭," which appears 41 times in the dataset. This emoji seems to be a commonly used comment.

- The most common user name in the 'user_name' column is "sweetangel550," occurring 78 times, indicating that this user is quite active.
- The first timestamp in the 'published_at' column is "2023-03-13 01:52:58+00:00," suggesting that the dataset starts from this date.
- The last timestamp in the 'published_at' column is "2023-10-15 04:54:36+00:00," indicating that the dataset ends on this date.
- We observed a significant surge in user engagement, with a notably high number of comments being posted just prior to the theatrical launch of the movie on May 26, 2023. This heightened activity suggests that anticipation and excitement were building up among the audience in the lead-up to the release.
- A notable peak in user engagement was observed at the end of July. This surge aligns with the digital download release of "The Little Mermaid" by Walt Disney Studios Home Entertainment on July 25, 2023.
- Another relatively smaller peak was observed in September, corresponding to the release of "The Little Mermaid" on Ultra HD Blu-ray, Blu-ray, and DVD, which occurred on September 19, 2023.
- The majority of user comments fall within the range of 6 (25th percentile) to 19 (75th percentile) words.
- The average word count per comment for each user is approximately 16 words.
- A significant portion of the comments (53.79%) fall under the "Positive" sentiment category (1). This suggests that a majority of users expressed positive sentiment in their comments about the movie.
- A substantial proportion of comments (28.38%) are classified as "Neutral" (0). This indicates that a considerable number of users provided comments that did not strongly express either positive or negative sentiment.
- A smaller fraction of comments (17.83%) are categorized as "Negative" sentiment (-1). While negative sentiment is present, it is less prevalent compared to positive and neutral sentiments.
- The distribution of sentiment data suggests an overall positive tone and reception of the movie among the online community. The combined percentage of positive and neutral sentiments (82.17%) significantly outweighs the negative sentiment (17.83%).
- The strong presence of Ariel's character and references to the Disney brand highlight their significance in generating user engagement.

- The presence of both positive and negative sentiment-related keywords suggests that users are discussing a variety of aspects, contributing to a well-rounded online conversation about the movie.

Recommendations

Based on the insights generated from the analysis of the dataset related to user comments on the movie "The Little Mermaid," here are some recommendations for a media investment company:

1. Audience Engagement and Anticipation:

- The significant surge in user engagement just before the theatrical launch on May 26, 2023, indicates that there is a strong interest and anticipation among the audience. This presents an opportunity for the media investment company to consider marketing campaigns and promotional activities leading up to a movie release.

2. Digital Release Strategy:

- The peak in user engagement observed in July, coinciding with the digital download release of the movie, suggests that digital releases can be effective in generating user interest and engagement. The company might explore digital distribution and streaming platforms as a key distribution strategy.

3. Physical Media Releases:

- The smaller peak in September, aligned with the release of physical media formats (Ultra HD Blu-ray, Blu-ray, DVD), indicates ongoing interest in different formats. The company could leverage physical media releases as part of their distribution and revenue strategy.

4. User Sentiment and Content Creation:

- The prevalence of positive sentiment (53.79%) in user comments suggests a positive reception. This sentiment can be harnessed for marketing and content creation, such as user-generated content and testimonials.

5. Variety of Sentiments:

- The presence of both positive and negative sentiment-related keywords indicates a well-rounded online conversation about the movie. The company can use this information to understand user preferences and potential areas of improvement in future projects.

6. User Activity and Behavior:

- The dataset's characteristics, such as the most frequent comment and active user names, provide insights into user behavior and preferences. The company can tailor marketing efforts to engage with these active users.

7. Content Timing:

- The dataset's start and end dates reveal the time frame of user engagement. The company can plan its marketing and promotional activities to align with key milestones and events related to the movie.

8. User Name Variability:

- The variety of user names (5,772 unique names) might suggest that a diverse user base is engaging with the content. The company can leverage this diversity for audience targeting and engagement strategies.

9. Word Count Insights:

- The average word count per comment (approximately 16 words) provides an understanding of user engagement depth. The company can create content and engagement strategies that cater to users' preferences for shorter or longer comments.

10. Positive Sentiment Dominance:

- The dominance of positive sentiment suggests a favorable perception of the movie. The company can use this positive sentiment to strengthen its brand and reputation in the media industry.

11. Sentiment Distribution:

- The distribution of sentiment data, with a combined percentage of positive and neutral sentiments outweighing negative sentiment, indicates an overall positive tone. The company can capitalize on this positivity in its marketing and branding efforts.

In conclusion, the insights derived from the analysis provide valuable information to the media investment company for strategizing marketing campaigns, distribution methods, and content creation that align with user preferences and engagement patterns. Leveraging these insights can lead to a more successful and effective media investment strategy for future projects.