WEB AND SOCIAL MEDIA ANALYTICS 7BUIS025W

Coursework

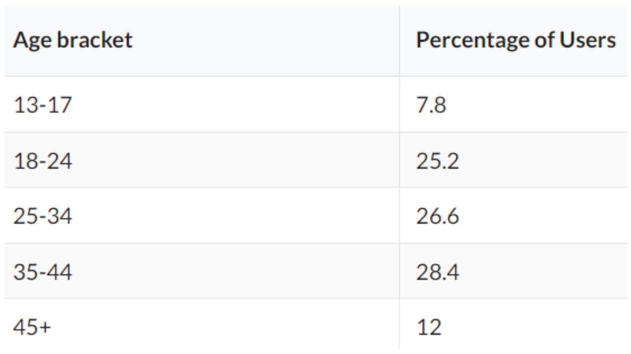
Nabeel-Osman Mohamed Osman Juma
W1929230
w1929230@my.westminster.ac.uk
Word count: 4213

<u>Introduction:</u>

Sentiment Analysis is a subset of the text mining field. It is the computation and treatment of comments, likes, opinions and sentiments to come up with an overview of public opinion on a particular topic, product or event (Hassan et al., 2014). The research in this area is still ongoing. The unprecedented emergence of new social media platforms and their resultant wide usage has led to an exponential growth in user generated data that is apt for sentiment mining and its subsequent analysis (Beigi et al., 1970). This analysis helps in making valid but nonobvious inferences which help people make informed decision keeping the overall sentiment in mind.

<u>The company:</u> Twitter

The rise of web 2.0 has led to the introduction and development of a huge number of social media platforms. Subsequently, microblogging has emerged as a convenient method of short communications that have been frequently used. A major player and, arguably the most successful one in this microblogging landscape, Twitter, amid major scandals has managed to accumulate a large user base spanning all over the world. It was founded in 2006 by Jack Dorsey, Noah Glass, Evan Williams and Biz stone. At last count, Twitter had around 2500 employees with an average monthly active user base of 450 million (Turner et al., 2023). The last known revenue details published by Twitter was in 2022 which showed a generation of $4.4 billion in revenue, which is 11% Y-o-Y decline. 90% of that revenue came from advertising.



Age bracket	Percentage of Users
13-17	7.8
18-24	25.2
25-34	26.6
35-44	28.4
45+	12

Fig(a)

Fig(a) shows the age distribution of Twitter. As you can see, it's very evenly distributed between the age groups of 18 to 44 (Iqbal, 2023).

The microblogs, or 'tweets' as they are commonly referred to, haven't been the only modes of communication or expression on the social media. It had briefly launched a short-video and live video feature which had subsequently been cancelled. It has recently been experimenting with Twitter space which is similar to Clubhouse where live audio conversations can take place. However, Twitter doesn't have a good track record on supporting products other than its base platform.

In April 2022, Elon Musk tweeted his interest to buy the microblogging social media platform for $44 billion. This offer was then finalised and completed on 27th October 2022 but not before he reneged on the offer multiple times citing the need for a full disclosure of data on bot accounts. The purchase of Twitter by Elon Musk was met with mixed reviews. Owing to the fact that Elon Musk had suggested unbanning all accounts that had a life-time ban, a lot of people branded him as the messiah of free speech. Alternatively, considering the fact that these people were banned for violating twitter terms and conditions, largely because of the use of inflammatory language and hate speech, another section of the public believed that he was propagating hate speech by this move. Financially speaking, some people believed him to be making a bad investment as Twitter was a failing company with a negative net profit. Similarly, a few others believed him to be capable of helping Twitter reach new heights.

<u>Data collection:</u>

I was concentrating on public opinion on **Elon Musk's successful takeover of Twitter**. Considering this topic, the following were my searches to help with data collection.

YouTube:

Elon Musk takes over Twitter: I used this phrase to find videos on news channels like BBC, Fox, MSNBC and CNN with a purpose to find public opinions posted in the form of comments on these videos.

Reddit:

Elon Musk Twitter: I used this phrase to look for pages about this topic. I then proceeded to choose pages with higher number of followers as I assumed that a higher number of followers would result in a higher number of interactions and comments. I then went on to search for relevant posts on and after the date 27th October 2022(the date Elon musk took over Twitter).

Elon Musk takes over Twitter good/bad: I also occasionally used the terms 'good' and 'bad' alternatively so as to get both positive as well as negative opinions on the matter.

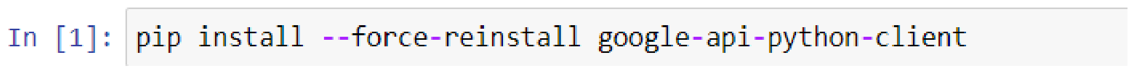
<u>Justifying the approach:</u>

I used the above-mentioned phrases so as to extract comments relevant to the specific topic, and the reasons are as follows:

1) Choosing specific key words/phrases helps in identifying posts and videos of a specific time during a specific change in the company. This helps in ensuring that the comments extracted are relevant to the topic chosen.
2) Using keywords like 'good' and 'bad' at the end help getting opposing views on the matter.
3) We use phrases instead of words as 'Elon Musk', 'Twitter' and 'Take over' may have a lot of results which may not be relevant to our required topic. Hence, when used in conjunction, these produced the desired results.

As a result of this, the method explained and finally employed here resulted in the desired posts and videos and hence the required comments for the dataset.

<u>The Dataset:</u>

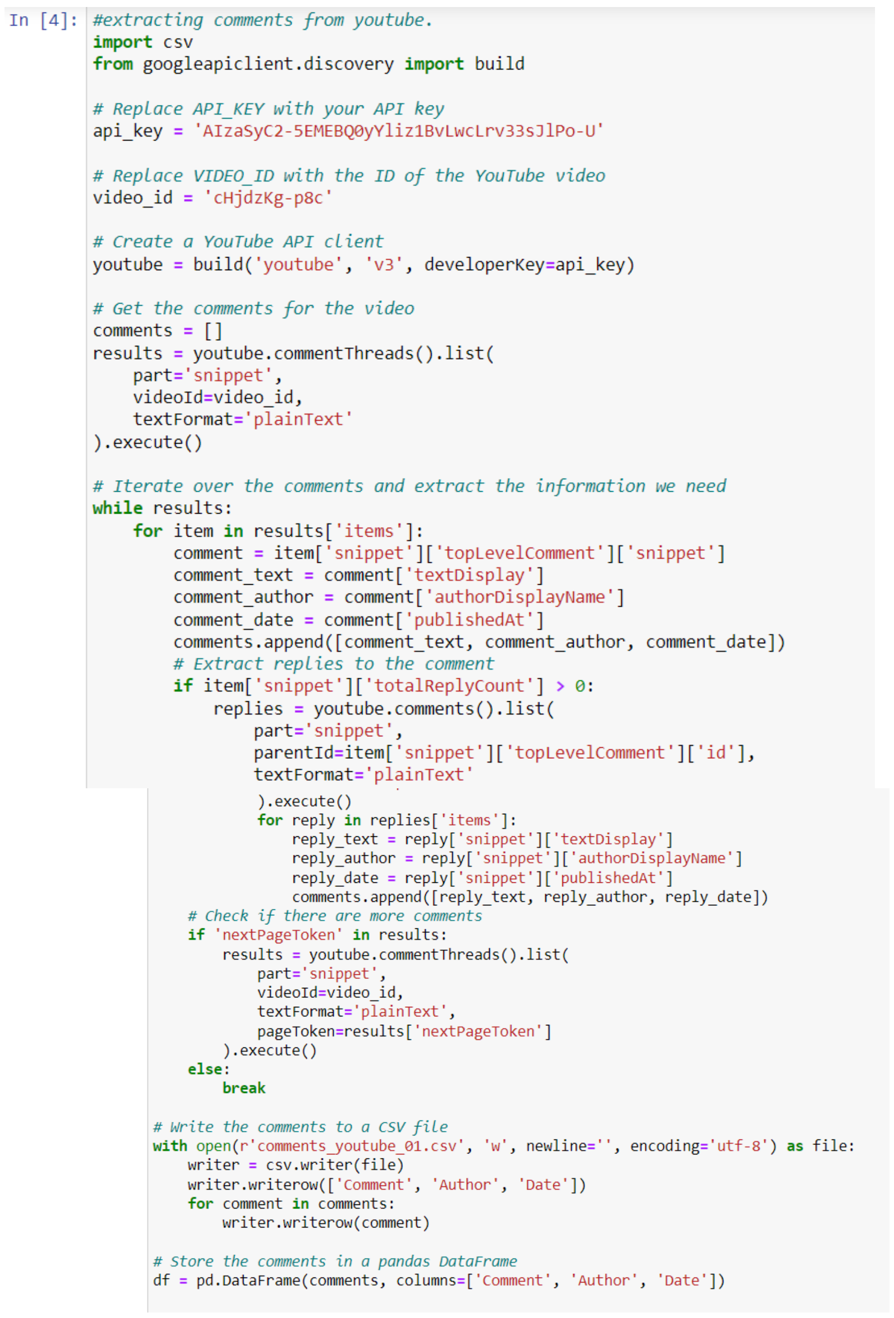


```
In [1]: pip install --force-reinstall google-api-python-client
```

Fig(b)

We start by reinstalling the 'google-api-client-package'. This is done to ensure that we have the latest bug fixes and features as the YouTube API is constantly being updated to enable improvement.

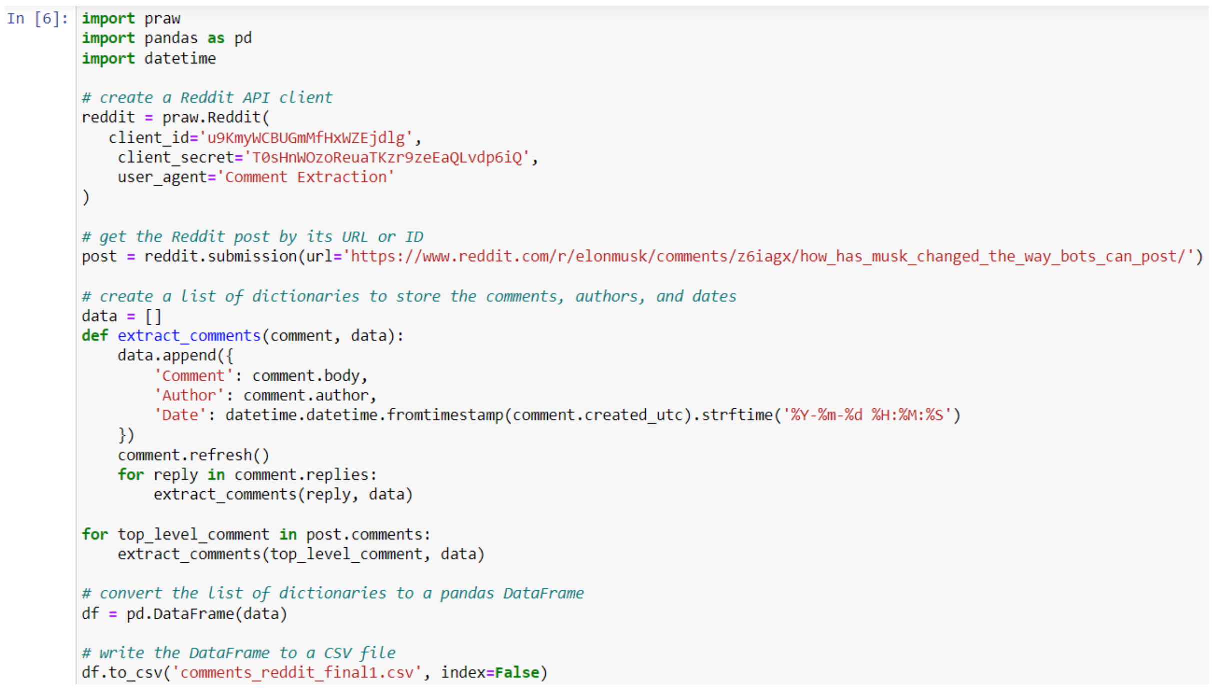
To perform our sentiment analysis on the public opinion regarding Elon Musk's takeover of Twitter, we scrap comments from Reddit and YouTube. We will then compare the results individually as well as together. This will give us a brief idea on demography as well as public perception individually and together.



```
In [4]: #extracting comments from youtube.
import csv
from googleapiclient.discovery import build

# Replace API_KEY with your API key
api_key = 'AIzaSyC2-5EMEBQ0yYliz1BvLwcLrv33sJlPo-U'

# Replace VIDEO_ID with the ID of the YouTube video
video_id = 'cHjdzKg-p8c'

# Create a YouTube API client
youtube = build('youtube', 'v3', developerKey=api_key)

# Get the comments for the video
comments = []
results = youtube.commentThreads().list(
    part='snippet',
    videoId=video_id,
    textFormat='plainText'
).execute()

# Iterate over the comments and extract the information we need
while results:
    for item in results['items']:
        comment = item['snippet']['topLevelComment']['snippet']
        comment_text = comment['textDisplay']
        comment_author = comment['authorDisplayName']
        comment_date = comment['publishedAt']
        comments.append([comment_text, comment_author, comment_date])
        # Extract replies to the comment
        if item['snippet']['totalReplyCount'] > 0:
            replies = youtube.comments().list(
                part='snippet',
                parentId=item['snippet']['topLevelComment']['id'],
                textFormat='plainText'
            ).execute()
            for reply in replies['items']:
                reply_text = reply['snippet']['textDisplay']
                reply_author = reply['snippet']['authorDisplayName']
                reply_date = reply['snippet']['publishedAt']
                comments.append([reply_text, reply_author, reply_date])
    # Check if there are more comments
    if 'nextPageToken' in results:
        results = youtube.commentThreads().list(
            part='snippet',
            videoId=video_id,
            textFormat='plainText',
            pageToken=results['nextPageToken']
        ).execute()
    else:
        break

# Write the comments to a CSV file
with open(r'comments_youtube_01.csv', 'w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(['Comment', 'Author', 'Date'])
    for comment in comments:
        writer.writerow(comment)

# Store the comments in a pandas DataFrame
df = pd.DataFrame(comments, columns=['Comment', 'Author', 'Date'])
```

Fig(c)

The code for the webscrapping of YouTube comments is given in Fig(c). The 'api_key' specifies our own API key. 'Video_id' showcases the id for the video whose comments need to be scrapped. This code uses a loop to extract all available comments and stores them as a csv.

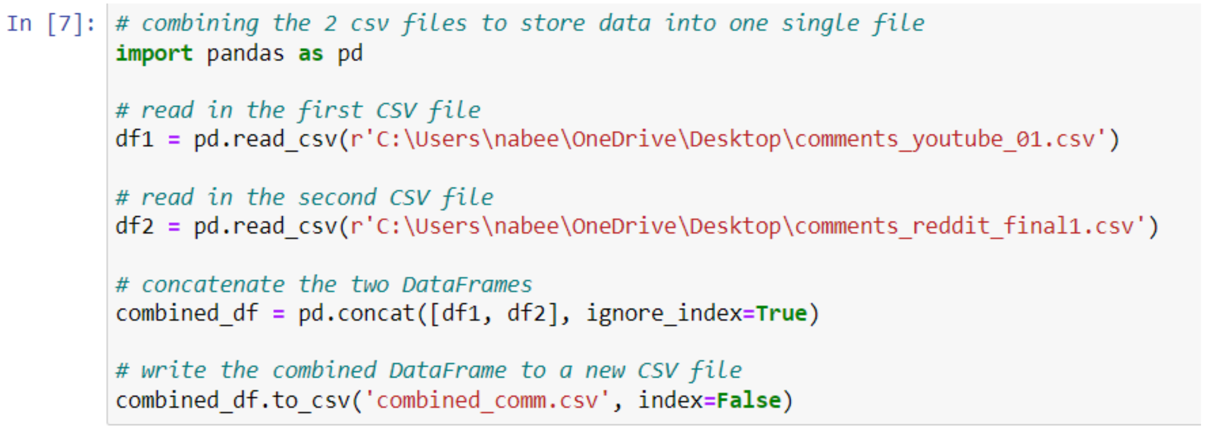


```
In [6]: import praw
        import pandas as pd
        import datetime

        # create a Reddit API client
        reddit = praw.Reddit(
           client_id='u9KmyWCBUGmMfHxWZEjdlg',
            client_secret='T0sHnWOzoReuaTKzr9zeEaQLvdp6iQ',
            user_agent='Comment Extraction'
        )

        # get the Reddit post by its URL or ID
        post = reddit.submission(url='https://www.reddit.com/r/elonmusk/comments/z6iagx/how_has_musk_changed_the_way_bots_can_post/')

        # create a list of dictionaries to store the comments, authors, and dates
        data = []
        def extract_comments(comment, data):
            data.append({
                'Comment': comment.body,
                'Author': comment.author,
                'Date': datetime.datetime.fromtimestamp(comment.created_utc).strftime('%Y-%m-%d %H:%M:%S')
            })
            comment.refresh()
            for reply in comment.replies:
                extract_comments(reply, data)

        for top_level_comment in post.comments:
            extract_comments(top_level_comment, data)

        # convert the list of dictionaries to a pandas DataFrame
        df = pd.DataFrame(data)

        # write the DataFrame to a CSV file
        df.to_csv('comments_reddit_final1.csv', index=False)
```

Fig(d)

The code starts by first importing PRAW(Python Reddit API Wrapper). It is used to extract comments from posts on Reddit. The 'client_id' specifies the client ID for reddit API. The 'client_secret' specifies the 'Secret key'. The 'post' specifies URL of the post whose comments need to be extracted. This data is converted to a pandas DataFrame which are in turn written to a csv file.

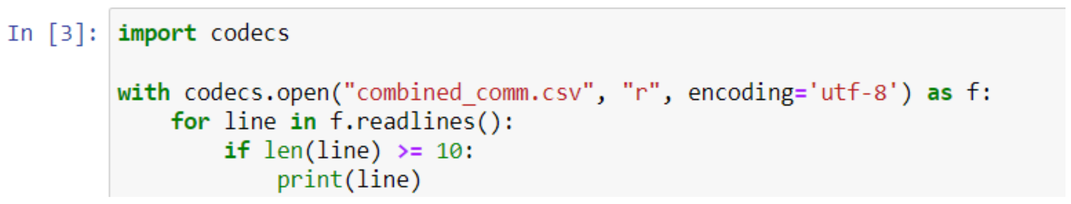
The above 2 steps of extracting shown in Fig(c) and Fig(d) are repeated multiple times to extract comments from multiple videos on YouTube and posts on Reddit so as to accumulate the desired number of responses from each platform. The comments are then combined into one YouTube comment sheet and one Reddit comment sheet containing all the comments extracted from the respective platforms.



```
In [7]: # combining the 2 csv files to store data into one single file
        import pandas as pd

        # read in the first CSV file
        df1 = pd.read_csv(r'C:\Users\nabee\OneDrive\Desktop\comments_youtube_01.csv')

        # read in the second CSV file
        df2 = pd.read_csv(r'C:\Users\nabee\OneDrive\Desktop\comments_reddit_final1.csv')

        # concatenate the two DataFrames
        combined_df = pd.concat([df1, df2], ignore_index=True)

        # write the combined DataFrame to a new CSV file
        combined_df.to_csv('combined_comm.csv', index=False)
```

Fig(e)

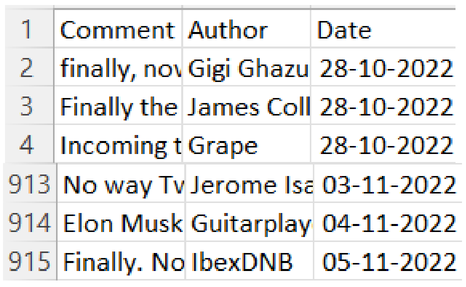
The above shown Fig(e) depicts the code to combine the data from the above specified csv files. These files are merged into one single file known as 'combined_comm.csv'.



```
In [3]: import codecs

        with codecs.open("combined_comm.csv", "r", encoding='utf-8') as f:
            for line in f.readlines():
                if len(line) >= 10:
                    print(line)
```

Fig(f)

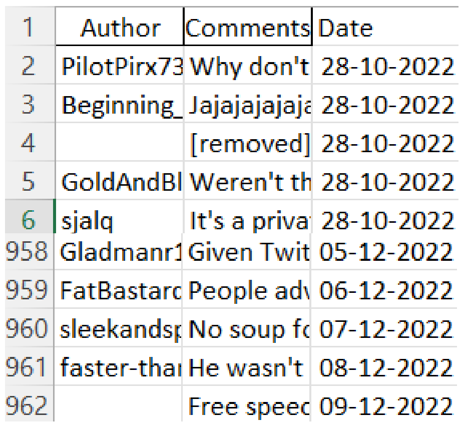
The above code is used to read the csv file which has all the data and filter out comments that have length smaller than 10. This is done to get rid of any useless comments that may not provide any sufficient opinion.



1	Comment	Author	Date
2	finally, nov	Gigi Ghazu	28-10-2022
3	Finally the	James Coll	28-10-2022
4	Incoming t	Grape	28-10-2022
913	No way Tv	Jerome Isa	03-11-2022
914	Elon Musk	Guitarplay	04-11-2022
915	Finally. No	IbexDNB	05-11-2022

This is the cropped dataset extracted from one of the YouTube videos showing that the comments have been extracted for atleast a period of 7 days.

Fig(g1)

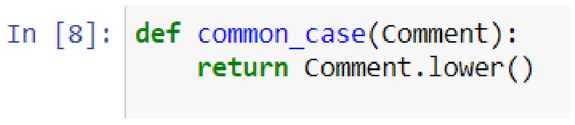


1	Author	Comments	Date
2	PilotPirx73	Why don't	28-10-2022
3	Beginning_	Jajajajajaja	28-10-2022
4		[removed]	28-10-2022
5	GoldAndBl	Weren't th	28-10-2022
6	sjalq	It's a priva	28-10-2022
958	Gladmanr1	Given Twit	05-12-2022
959	FatBastarc	People adv	06-12-2022
960	sleekandsp	No soup fc	07-12-2022
961	faster-thai	He wasn't	08-12-2022
962		Free speec	09-12-2022

This is the Reddit cropped dataset extracted from the responses posted to a reddit post relevant to the topic. The extracted comments have a time span a of atleast 7 days as shown in the screenshot.

Fig(g2)

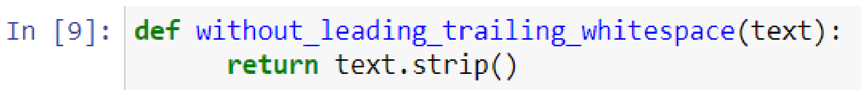
Text Processing:



```
In [8]: def common_case(Comment):
            return Comment.lower()
```

Fig(h)

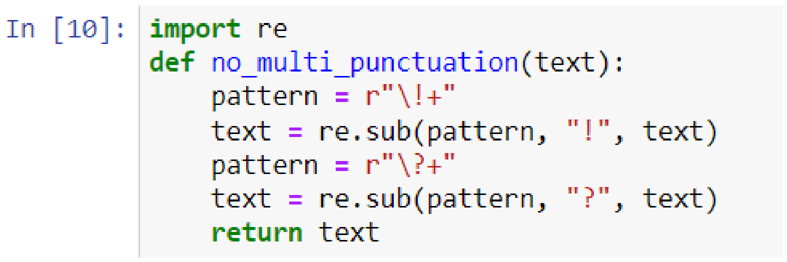
Fig(h) shows the first step in text processing. It converts all the comments to a lower case using the method lower(). This is done to avoid the same word being treated as two different words because of capitalisation.



```
In [9]: def without_leading_trailing_whitespace(text):
            return text.strip()
```

Fig(i)

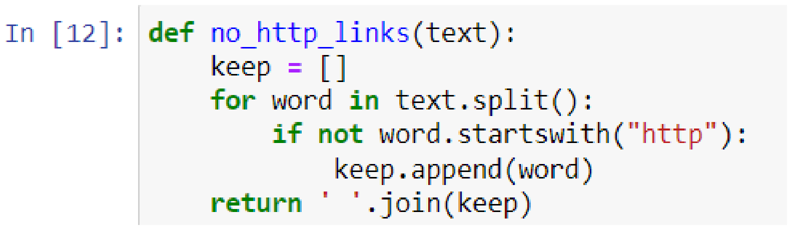
Code in Fig(i) is used to remove trailing and leading whitespaces in comments. This is done as comments could have these spaces inadvertently which may affect the analysis. A function ‘without_leading_trailing_whitespace’ is created and a method .strip() is used for this process.



```
In [10]: import re
         def no_multi_punctuation(text):
             pattern = r"\!+"
             text = re.sub(pattern, "!", text)
             pattern = r"\?+"
             text = re.sub(pattern, "?", text)
             return text
```

Fig(j)

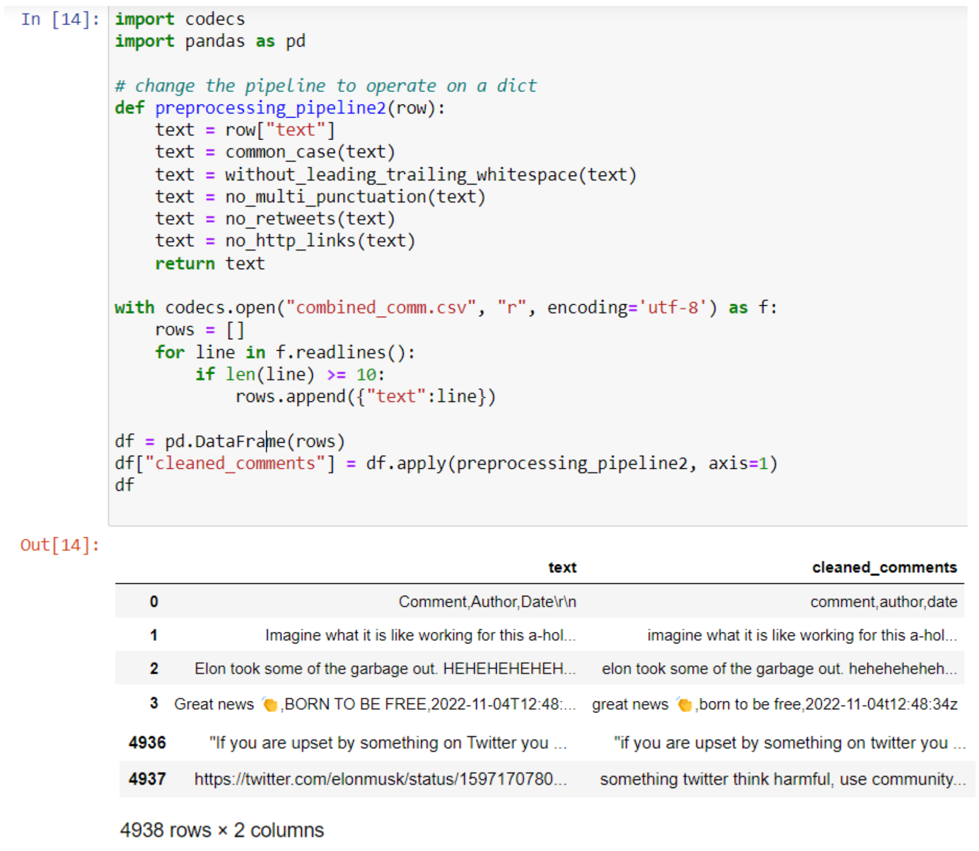
We see that very often; users use multiple punctuations like exclamation marks(!) and question marks(?) in a sequence while typing a comment. This may lead to issues during sentiment analysis. The code shown in Fig(j) is used to replace multiple punctuations with a single one.



```
In [12]: def no_http_links(text):
             keep = []
             for word in text.split():
                 if not word.startswith("http"):
                     keep.append(word)
             return ' '.join(keep)
```

Fig(k)

This code is used to remove hyperlinks from comments. It is done as these links may not be useful to the analysis and might just be adding noise to it. This is done by creating a function called ‘no_http_links’ and the method ‘split()’ to split the text.



```
In [14]: import codecs
         import pandas as pd

         # change the pipeline to operate on a dict
         def preprocessing_pipeline2(row):
             text = row["text"]
             text = common_case(text)
             text = without_leading_trailing_whitespace(text)
             text = no_multi_punctuation(text)
             text = no_retweets(text)
             text = no_http_links(text)
             return text

         with codecs.open("combined_comm.csv", "r", encoding='utf-8') as f:
             rows = []
             for line in f.readlines():
                 if len(line) >= 10:
                     rows.append({"text":line})

         df = pd.DataFrame(rows)
         df["cleaned_comments"] = df.apply(preprocessing_pipeline2, axis=1)
         df
```

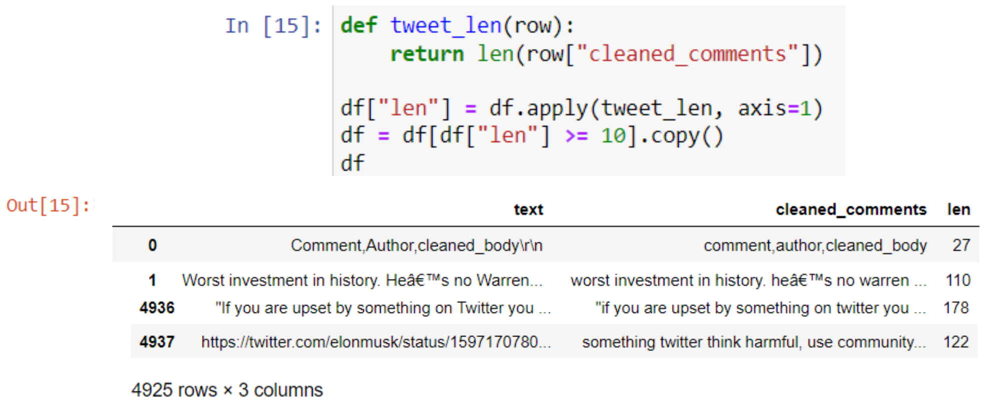
Out[14]:

	text	cleaned_comments
0	Comment,Author,Date\r\n	comment,author,date
1	Imagine what it is like working for this a-hol...	imagine what it is like working for this a-hol...
2	Elon took some of the garbage out. HEHEHEHEHEH...	elon took some of the garbage out. heheheheheh...
3	Great news 👏,BORN TO BE FREE,2022-11-04T12:48:...	great news 👏,born to be free,2022-11-04t12:48:34z
4936	"If you are upset by something on Twitter you ...	"if you are upset by something on twitter you ...
4937	https://twitter.com/elonmusk/status/1597170780...	something twitter think harmful, use community...

4938 rows × 2 columns

Fig(l)

The above code uses the preprocessing_pipeline2() function on the comments. This function takes it as a dictionary and the cleaned data is then saved in the DataFrame under the column 'cleaned_comments'. You can also see that there are a total of 4938 rows, meaning that there are a total of 4398 comments.



```
In [15]: def tweet_len(row):
             return len(row["cleaned_comments"])

         df["len"] = df.apply(tweet_len, axis=1)
         df = df[df["len"] >= 10].copy()
         df
```

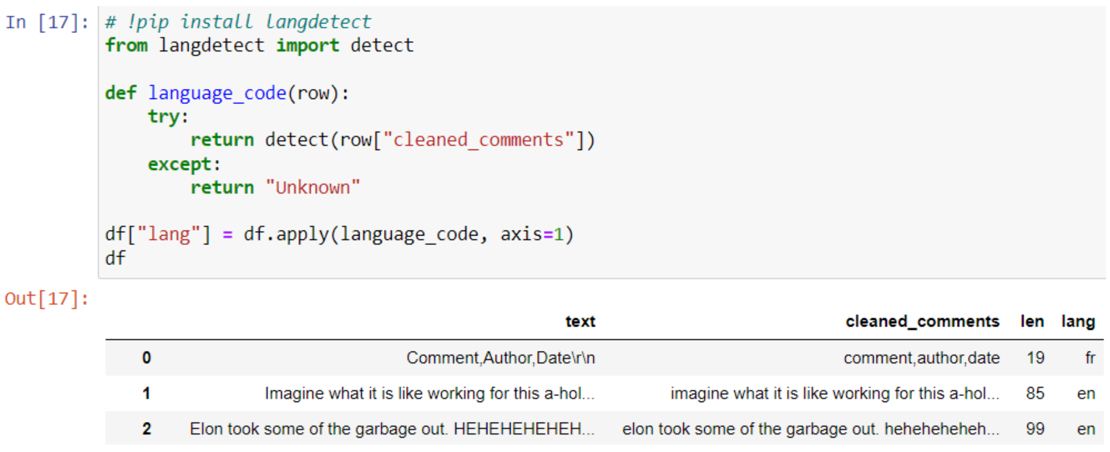
Out[15]:

	text	cleaned_comments	len
0	Comment,Author,cleaned_body\r\n	comment,author,cleaned_body	27
1	Worst investment in history. Heâ€™s no Warren...	worst investment in history. heâ€™s no warren ...	110
4936	"If you are upset by something on Twitter you ...	"if you are upset by something on twitter you ...	178
4937	https://twitter.com/elonmusk/status/1597170780...	something twitter think harmful, use community...	122

4925 rows × 3 columns

Fig(m)

The above code creates a new column in the DataFrame known as length. It stored the number of characters in each comment. Following this, it deleted comments that have a length less than 10. As you can now see, the number of rows has been reduced from 4938 in Fig(l) to 4925 in Fig(m).



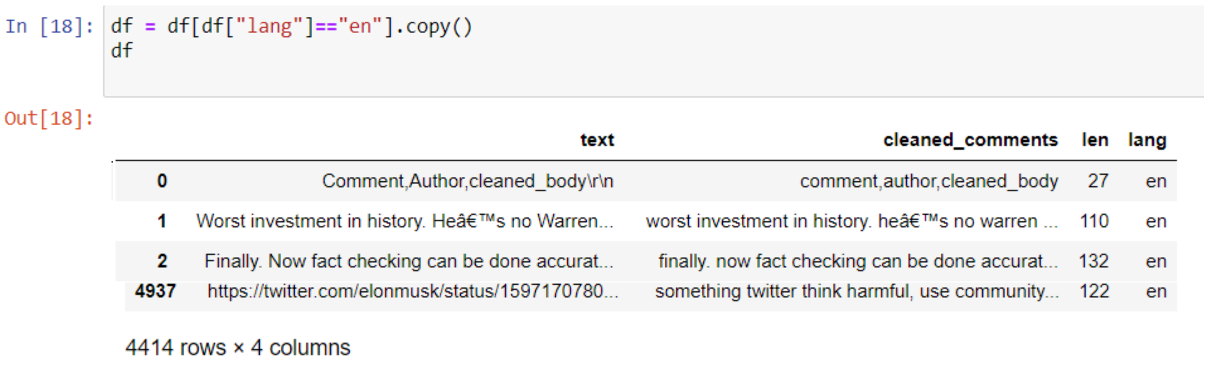
```
In [17]: # !pip install langdetect
         from langdetect import detect

         def language_code(row):
             try:
                 return detect(row["cleaned_comments"])
             except:
                 return "Unknown"

         df["lang"] = df.apply(language_code, axis=1)
         df
```

Out[17]:

	text	cleaned_comments	len	lang
0	Comment,Author,Date\r\n	comment,author,date	19	fr
1	Imagine what it is like working for this a-hol...	imagine what it is like working for this a-hol...	85	en
2	Elon took some of the garbage out. HEHEHEHEHEH...	elon took some of the garbage out. heheheheheh...	99	en

Fig(n)

Creates a new column in the Dataframe as shown in Fig(n). This DataFrame column contains the language the comment is in.



```
In [18]: df = df[df["lang"]=="en"].copy()
         df
```

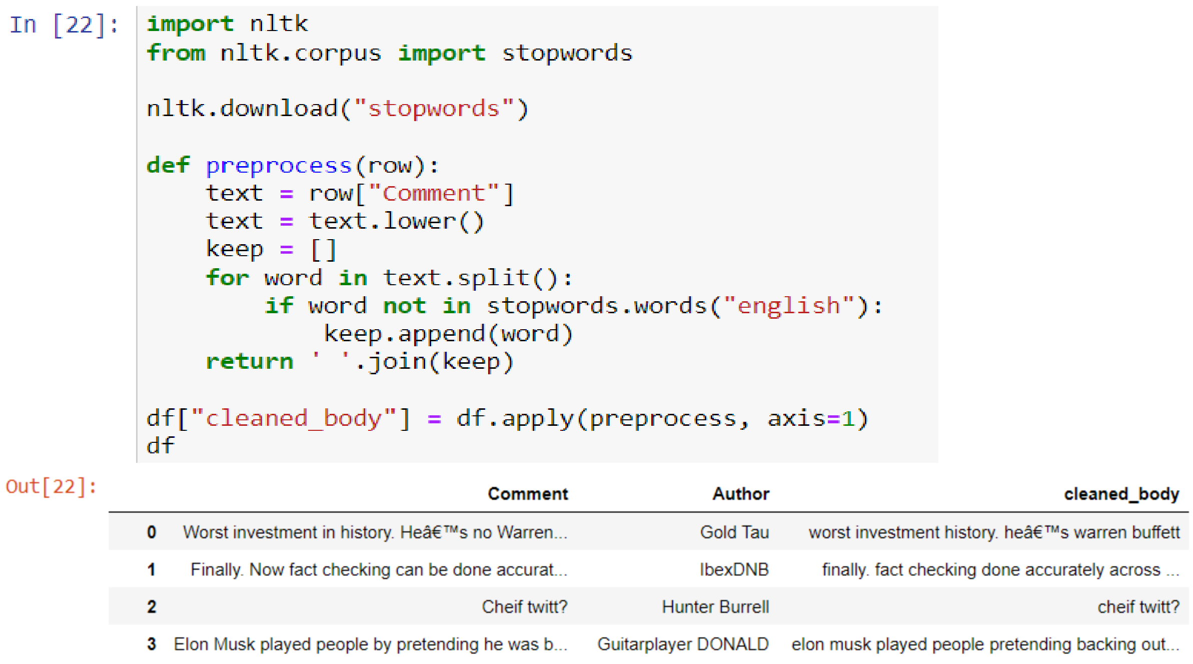
Out[18]:

	text	cleaned_comments	len	lang
0	Comment,Author,cleaned_body\r\n	comment,author,cleaned_body	27	en
1	Worst investment in history. Heâ€™s no Warren...	worst investment in history. heâ€™s no warren ...	110	en
2	Finally. Now fact checking can be done accurat...	finally. now fact checking can be done accurat...	132	en
4937	https://twitter.com/elonmusk/status/1597170780...	something twitter think harmful, use community...	122	en

4414 rows × 4 columns

Fig(o)

Removes any comment that's not in English to help with the processing. Thus, the number of rows has been reduced from 4925 in Fig(m) to 4414 in Fig(o).

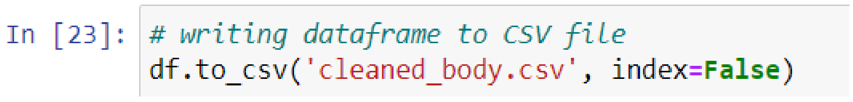


```
In [22]: import nltk
         from nltk.corpus import stopwords

         nltk.download("stopwords")

         def preprocess(row):
             text = row["Comment"]
             text = text.lower()
             keep = []
             for word in text.split():
                 if word not in stopwords.words("english"):
                     keep.append(word)
             return ' '.join(keep)

         df["cleaned_body"] = df.apply(preprocess, axis=1)
         df
```

Out[22]:

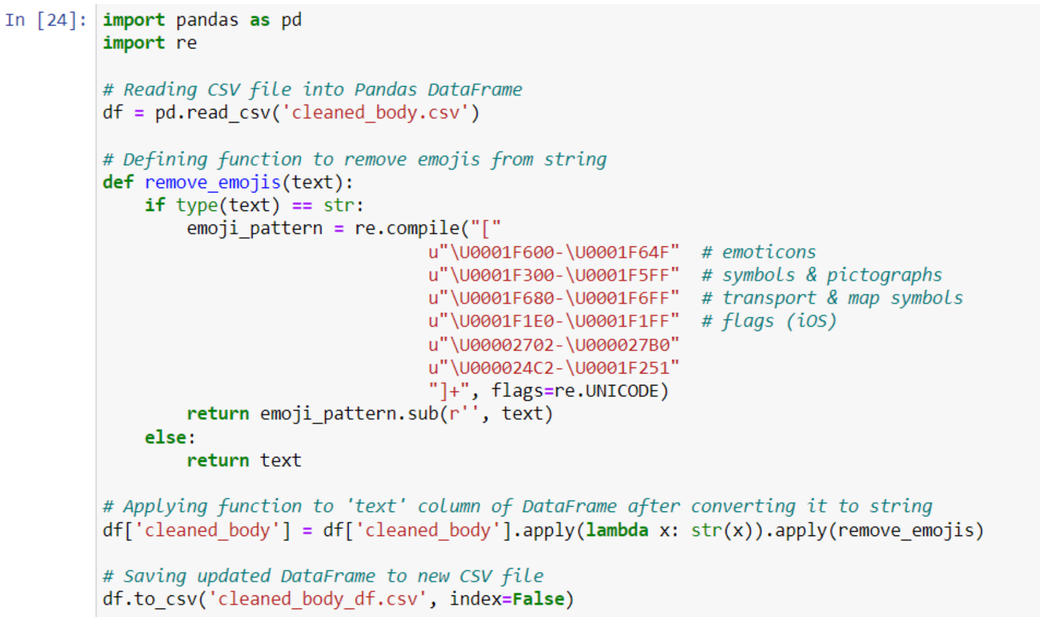
	Comment	Author	cleaned_body
0	Worst investment in history. Heâ€™s no Warren...	Gold Tau	worst investment history. heâ€™s warren buffett
1	Finally. Now fact checking can be done accurat...	IbexDNB	finally. fact checking done accurately across ...
2	Cheif twitt?	Hunter Burrell	cheif twitt?
3	Elon Musk played people by pretending he was b...	Guitarplayer DONALD	elon musk played people pretending backing out...

The above code is used to remove stop words. This is done by using the stopwords library from the Natural Language Toolkit(nltk). Stop words like 'and', 'the', 'etc' do not provide much meaning and just add noise to the processing. Removing them also helps in reduction of the dimensionality of the data thereby letting it concentrate on more important words. The output shown above also shows the removal of these stop words in the newly created column of 'cleaned_body'.



```
In [23]: # writing dataframe to csv file
         df.to_csv('cleaned_body.csv', index=False)
```

Writing the DataFrame to the newly created csv file called 'cleaned_body'.

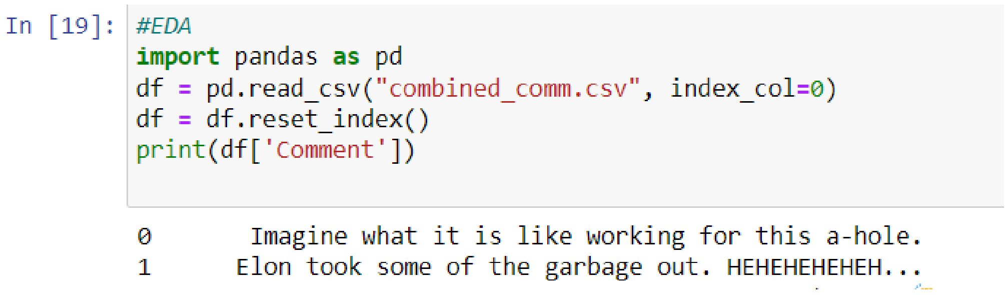


```
In [24]: import pandas as pd
         import re

         # Reading CSV file into Pandas DataFrame
         df = pd.read_csv('cleaned_body.csv')

         # Defining function to remove emojis from string
         def remove_emojis(text):
             if type(text) == str:
                 emoji_pattern = re.compile("["
                                            u"\U0001F600-\U0001F64F"  # emoticons
                                            u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                                            u"\U0001F680-\U0001F6FF"  # transport & map symbols
                                            u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                                            u"\U00002702-\U000027B0"
                                            u"\U000024C2-\U0001F251"
                                            "]+", flags=re.UNICODE)
                 return emoji_pattern.sub(r'', text)
             else:
                 return text

         # Applying function to 'text' column of DataFrame after converting it to string
         df['cleaned_body'] = df['cleaned_body'].apply(lambda x: str(x)).apply(remove_emojis)

         # Saving updated DataFrame to new CSV file
         df.to_csv('cleaned_body_df.csv', index=False)
```

The above code is used to remove emojis from the comments. Considering the fact that emojis are Unicode characters, they may cause issues with certain algorithms related to text analysis and processing.

Exploratory Data Analysis:

I'll be showing a comparative analysis of visuals for the Reddit dataset, YouTube dataset and the dataset created by combining both of them.

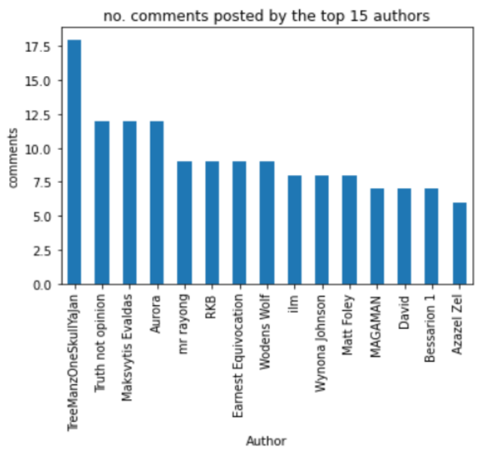


```
In [19]: #EDA
         import pandas as pd
         df = pd.read_csv("combined_comm.csv", index_col=0)
         df = df.reset_index()
         print(df['Comment'])

         0        Imagine what it is like working for this a-hole.
         1       Elon took some of the garbage out. HEHEHEHEHEH...
```

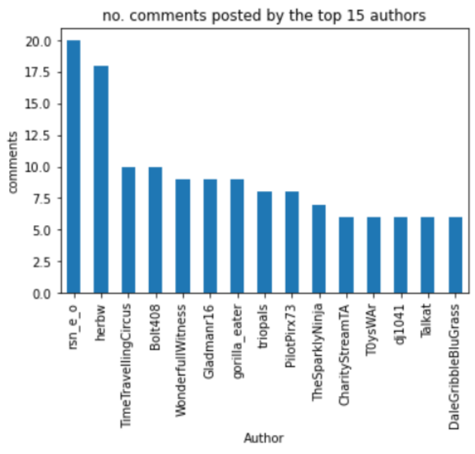
Fig(p)

We start our process of EDA by first indexing the comments as shown above. This helps in plotting the values in later steps.

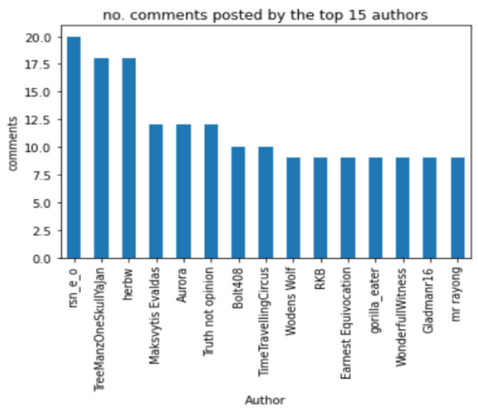
```
In [20]: df.groupby(df["Author"]).size().sort_values(ascending=False).iloc[0:15].plot(kind="bar", ylabel="comments",
                title="no. comments posted by the top 15 authors")
```



Fig(q1)



Fig(q2)



Fig(q3)

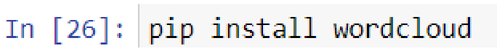
The 3 charts show the display of 15 top comments for only the YouTube dataset in Fig(q1), only the reddit database in Fig(q2) and for the combined dataset for both Reddit and YouTube in Fig(q3).

```
In [25]: from collections import Counter

         word_counter = Counter()
         for row in df.to_dict("records"):
             word_counter.update(row["cleaned_body"].split())
         df_tf = pd.DataFrame(word_counter.most_common(10))
         df_tf.columns = ["term", "frequency"]
         df_tf
```

Out[25]:

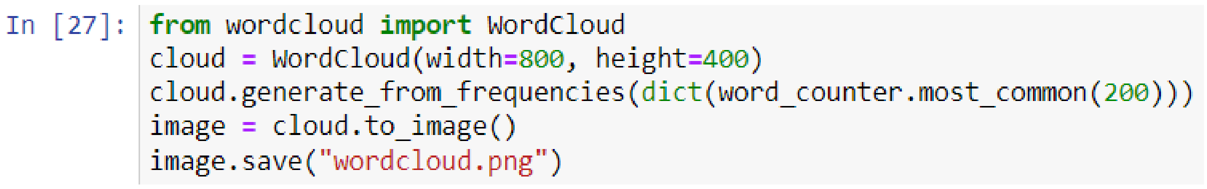
	term	frequency
0	twitter	603
1	elon	529
2	musk	376
3	people	329
4	like	324
5	free	306
6	speech	230
7	would	212
8	get	205
9	one	184

The above code splits each comment row into multiple words. A counter is associated with each word. The code lists the top 10 most frequently used words.



```
In [26]: pip install wordcloud
```

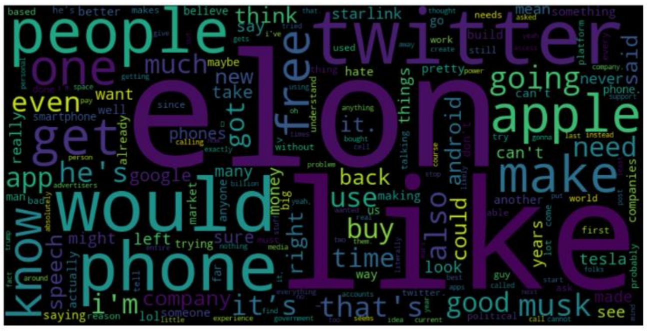
Installing python package called 'wordcloud'.



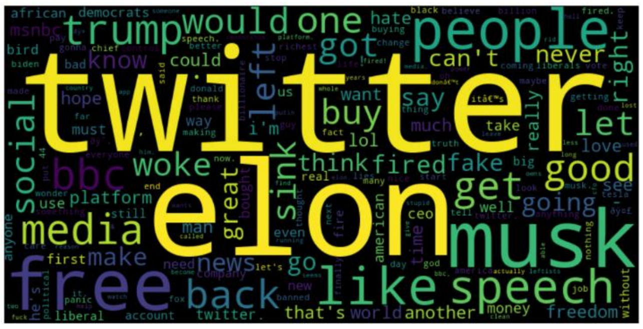
```
In [27]: from wordcloud import WordCloud
         cloud = WordCloud(width=800, height=400)
         cloud.generate_from_frequencies(dict(word_counter.most_common(200)))
         image = cloud.to_image()
         image.save("wordcloud.png")
```

Generating word cloud for the given dataset. This is very helpful in exploratory data analysis. Visualising the dataset helps in identifying various important words that are repeated regularly in the dataset. Additionally, they are appealing to look at and easy to understand.

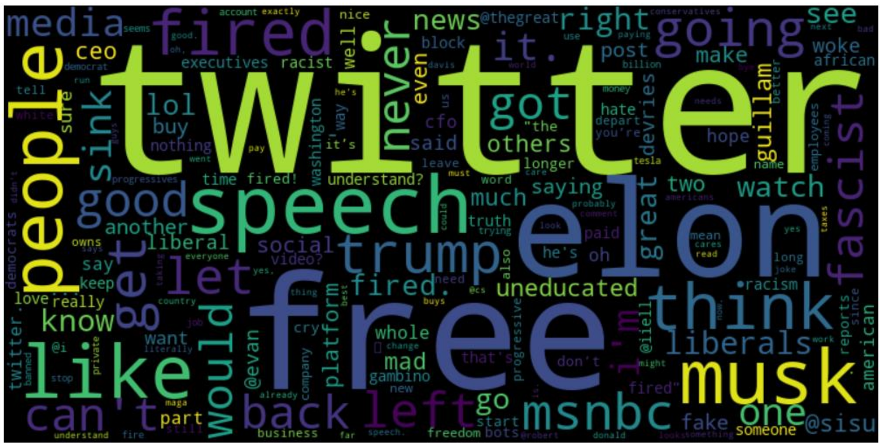
Word Cloud:



Fig(r1)



Fig(r2)



Fig(r3)

In the above three word clouds, we can see Fig(r1), the word cloud depicting the Reddit dataset, the words 'Elon', 'Twitter', 'Like' and 'Free' are of prominence. Similarly, in the Fig(r2), the word cloud depicting the YouTube dataset, 'Elon', 'Musk', 'Twitter', 'Speech', etc are given importance. The word cloud depicted in Fig(r3) is created by the merged dataset of both Reddit and Twitter and shows us a joint result. It points towards the importance of words like 'Elon', 'Twitter', 'Speech', 'Free' and 'Like'. Not surprisingly, another word also showcased here is 'Trump' which has resulted from people's opinion regarding Elon Musk's controversial decision of unbanning Donald Trump's account.

Evaluation of methods used:

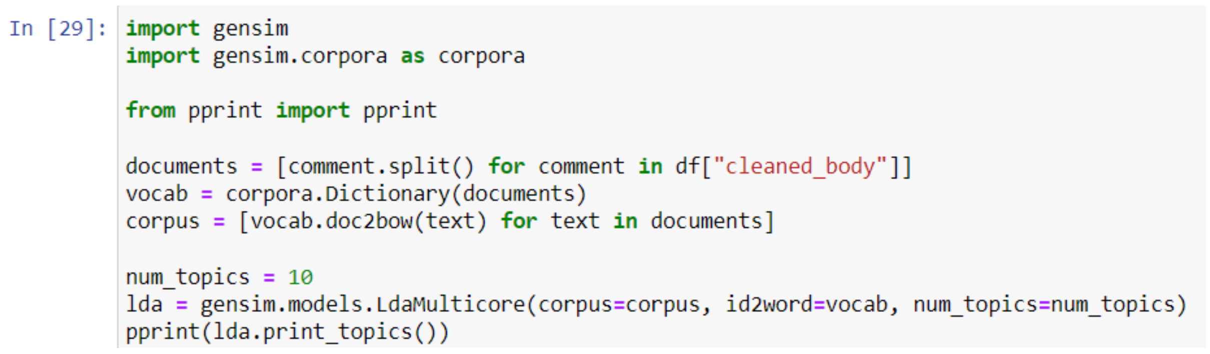
The various steps employed by me during the EDA like indexing, removing stopwords and emojis and using a counter to find the top 10 repeated words in the comments list are standard. Visually displaying the top 15 comments to compare their occurrence is also done in the usual manner. However, the word cloud, inspite of being able to provide useful results has still ended up giving some degree of prominence to words like ‘use’, ‘got’, ‘get’ and ‘let’. These words aren’t stopwords. However, they do not provide any context or importance to topic.

Reflection of initial results:

From having a look at the results, we get from the word clouds, we can say that the data processing worked satisfactorily in removing various stopwords like ‘a’, ‘the’, etc and preventing them from coming up in the word cloud. Similarly, it helped in emphasising the importance of the words like ‘elon’, ‘musk’, ‘twitter’, ‘free’, ‘speech’, ‘trump’, etc. and making sure that this emphasis is shown in the final word cloud. This helps us understand that these are words of importance in the topic of discussion. However, the words ‘free’ and ‘speech’ on their own do not hold much context in our topic. Maybe a word cloud for displaying phrases might have given importance to the phrase ‘free speech’ which depicts a point of view on one of Elon Musk’s controversial decision of unbanning certain accounts that have been handed lifetime bans for making inflammatory comments.

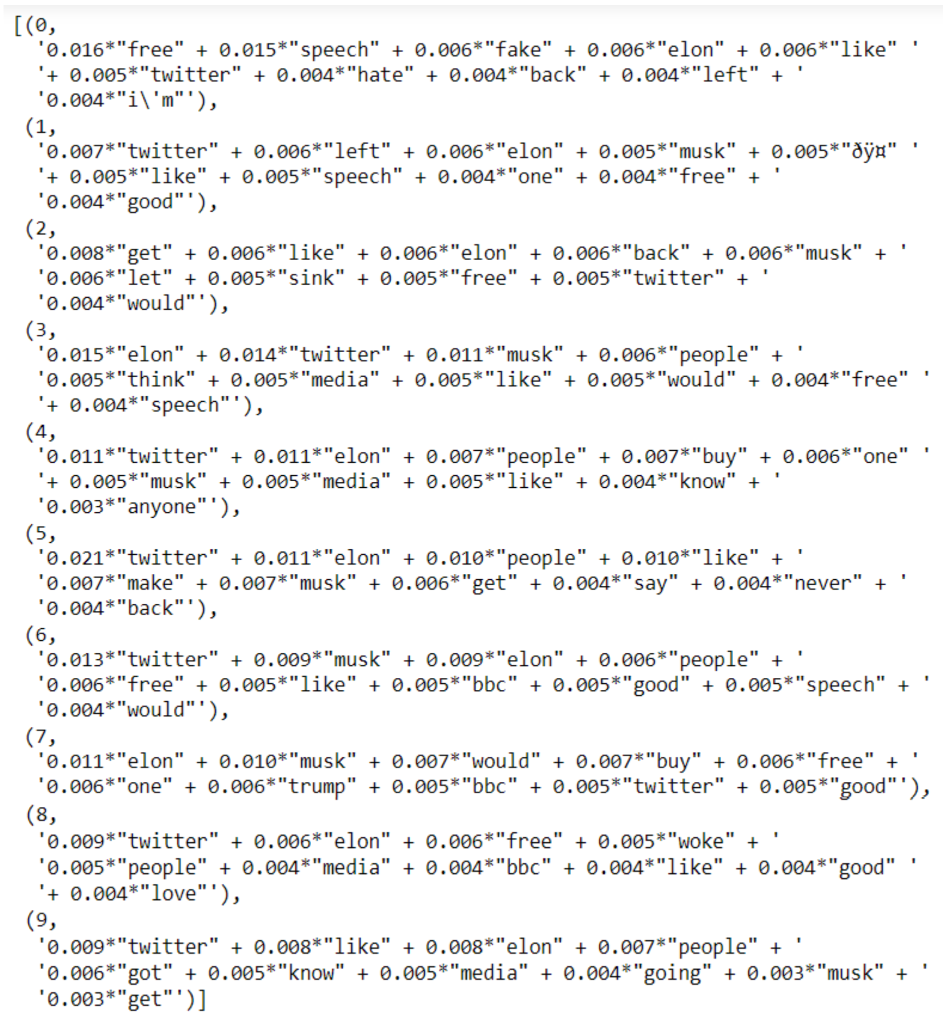
Topic Model Analysis:

Topic modelling is basically the method used to identify themes and topics in a collection of documents. It is useful for tasks like classification of texts and clustering documents. These topics are a list of words that occur in a corpus. We will be using the LDA method for topic modelling here.



```
In [29]: import gensim
         import gensim.corpora as corpora

         from pprint import pprint

         documents = [comment.split() for comment in df["cleaned_body"]]
         vocab = corpora.Dictionary(documents)
         corpus = [vocab.doc2bow(text) for text in documents]

         num_topics = 10
         lda = gensim.models.LdaMulticore(corpus=corpus, id2word=vocab, num_topics=num_topics)
         pprint(lda.print_topics())
```

The Latent Dirichlet Allocation (LDA) is an algorithm used for topic modelling available in the Gensim library. The LDA is probabilistic model used to collect data that are discrete in nature like a collection of documents or corpus. It helps in finding the hidden topic in a corpus and to decide the various topics a particular document can belong to. The grammatical role and the word order aren’t given importance in this method as each document is considered to be only a bag of words. The stop words are eliminated (Blei et al.,2003). The output generated by this model are some words which can be used to find the meaning of the corpus content. It is very efficient in finding trends in huge datasets.



```
[(0,
  '0.016*"free" + 0.015*"speech" + 0.006*"fake" + 0.006*"elon" + 0.006*"like" '
  '+ 0.005*"twitter" + 0.004*"hate" + 0.004*"back" + 0.004*"left" + '
  '0.004*"i\'m"'),
 (1,
  '0.007*"twitter" + 0.006*"left" + 0.006*"elon" + 0.005*"musk" + 0.005*"ðÿ¤" '
  '+ 0.005*"like" + 0.005*"speech" + 0.004*"one" + 0.004*"free" + '
  '0.004*"good"'),
 (2,
  '0.008*"get" + 0.006*"like" + 0.006*"elon" + 0.006*"back" + 0.006*"musk" + '
  '0.006*"let" + 0.005*"sink" + 0.005*"free" + 0.005*"twitter" + '
  '0.004*"would"'),
 (3,
  '0.015*"elon" + 0.014*"twitter" + 0.011*"musk" + 0.006*"people" + '
  '0.005*"think" + 0.005*"media" + 0.005*"like" + 0.005*"would" + 0.004*"free" '
  '+ 0.004*"speech"'),
 (4,
  '0.011*"twitter" + 0.011*"elon" + 0.007*"people" + 0.007*"buy" + 0.006*"one" '
  '+ 0.005*"musk" + 0.005*"media" + 0.005*"like" + 0.004*"know" + '
  '0.003*"anyone"'),
 (5,
  '0.021*"twitter" + 0.011*"elon" + 0.010*"people" + 0.010*"like" + '
  '0.007*"make" + 0.007*"musk" + 0.006*"get" + 0.004*"say" + 0.004*"never" + '
  '0.004*"back"'),
 (6,
  '0.013*"twitter" + 0.009*"musk" + 0.009*"elon" + 0.006*"people" + '
  '0.006*"free" + 0.005*"like" + 0.005*"bbc" + 0.005*"good" + 0.005*"speech" + '
  '0.004*"would"'),
 (7,
  '0.011*"elon" + 0.010*"musk" + 0.007*"would" + 0.007*"buy" + 0.006*"free" + '
  '0.006*"one" + 0.006*"trump" + 0.005*"bbc" + 0.005*"twitter" + 0.005*"good"'),
 (8,
  '0.009*"twitter" + 0.006*"elon" + 0.006*"free" + 0.005*"woke" + '
  '0.005*"people" + 0.004*"media" + 0.004*"bbc" + 0.004*"like" + 0.004*"good" '
  '+ 0.004*"love"'),
 (9,
  '0.009*"twitter" + 0.008*"like" + 0.008*"elon" + 0.007*"people" + '
  '0.006*"got" + 0.005*"know" + 0.005*"media" + 0.004*"going" + 0.003*"musk" + '
  '0.003*"get"')]
```

Fig(s)

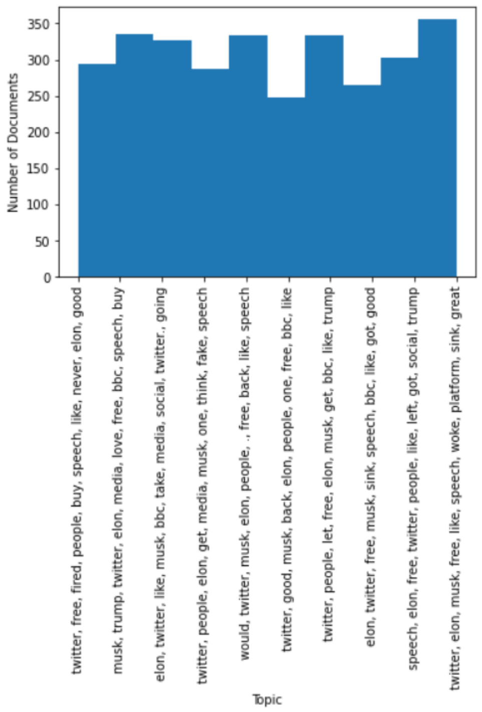
The above screenshot shows the output generated by the topic modelling done by the LDA algorithm when run on the dataset of comments provided to it. It shows 10 topics that the algorithm has identified containing 10 different words that are highly representative of the topic and its associated probabilities. It also contains the indexing of the topics from 0-9 with the 10 different topics that the algorithm has predicted. They are of no particular significance and are just numbers that are arbitrary in nature solely used for the purpose of identification.

For example, in the topic indexed as 0, the words, ‘free’, ‘speech’, ‘fake’ elon’, ‘like’ have the highest probabilities. This can show us that the topic can be related to free speech and Elon.

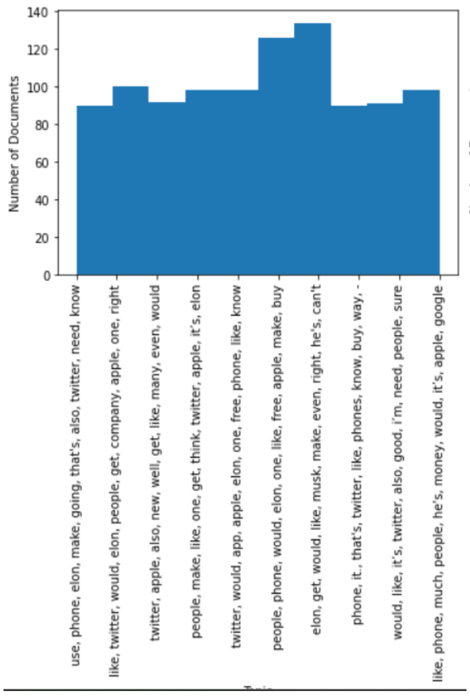
Similarly, for the topic indexed as 1, the important words are ‘twitter’, ‘left’, ‘elon’, ‘musk. This shows us most of the comments are related to elon musk and twitter. The term ‘left’ is very vague in this scenario. It could either mean the left-wing supporters who weren’t very pleased with Elon Musk taking over Twitter or it could be referring to someone who ‘left’ Twitter.

<u>Visualising the topic distribution for each document:</u>

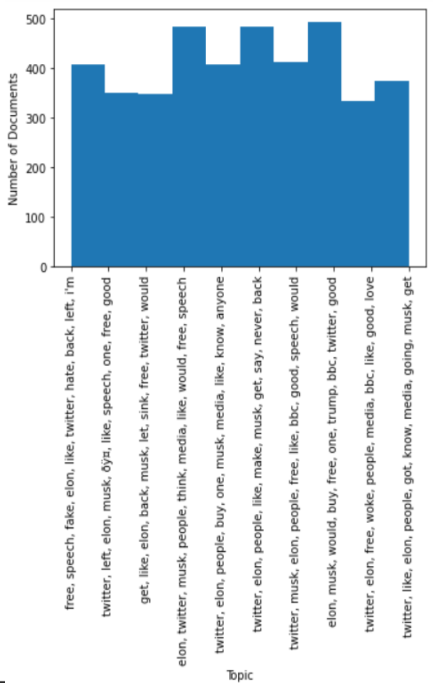
```
In [31]: import matplotlib.pyplot as plt

         # Get topic distribution for each document
         topic_distribution = []
         for doc in corpus:
             topic_distribution.append(lda[doc])

         # Get the dominant topic for each document
         dominant_topics = []
         for dist in topic_distribution:
             dominant_topics.append(max(dist, key=lambda item: item[1])[0])

         # Get topic names
         topic_names = [', '.join([word[0] for word in lda.show_topic(topic)]) for topic in range(num_topics)]

         # Create histogram with topic names on x-axis
         plt.hist(dominant_topics, bins=num_topics)
         plt.xticks(range(num_topics), topic_names, rotation=90)
         plt.xlabel("Topic")
         plt.ylabel("Number of Documents")
         plt.show()
```



Fig(t1)



Fig(t2)

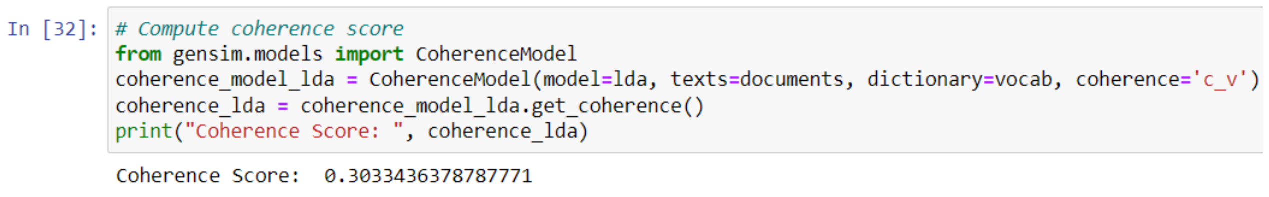


Fig(t3)

Fig(t1) is used to display the topic distribution for the Reddit comments. Fig(t2) shows the topic distribution for the YouTube comments and Fig(t3) is used to show the topic distribution of the dataset containing the combined comments data.

Goodness Of Fit:

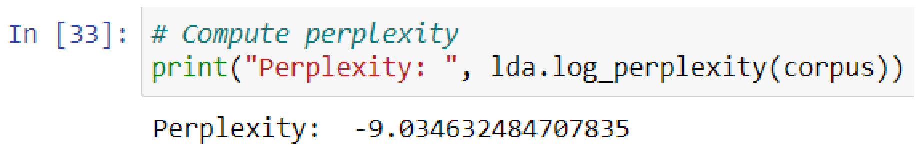
Coherence score:



```
In [32]: # Compute coherence score
         from gensim.models import CoherenceModel
         coherence_model_lda = CoherenceModel(model=lda, texts=documents, dictionary=vocab, coherence='c_v')
         coherence_lda = coherence_model_lda.get_coherence()
         print("Coherence Score: ", coherence_lda)

         Coherence Score:  0.3033436378787771
```

The coherence score is the degree of similarity between the high scoring words of each topic. It has a range of 0 to 1. A higher score depicts a better coherence score and that the topics can be interpreted better. An acceptable coherence score is between 0.3 to 0.4. However, this also largely depends on the dataset and specific cases.

Perplexity score:



```
In [33]: # Compute perplexity
         print("Perplexity: ", lda.log_perplexity(corpus))

         Perplexity:  -9.034632484707835
```

It is the value indicating how well a model can predict unseen or held out data. A higher perplexity score means that the model isn't able to accurately predict unseen data for a given model. Hence, a lower score means the model can accurately predict unseen data for a given model. A score of -9.03 means that the model can accurately predict unseen data based on the given data.

Critical evaluation of the approach and interpretation of the results:

There are many topic modelling algorithms like LDA, NMF, PLSA, HDP, etc. each having different uses.

The LDA is an algorithm that has been generally used in topic modelling of such data. It has some limitations like:

- Being sensitive to hyperparameters, any changes in them can lead to significant changes in results.
- If the dataset is sparce or has a lot of noise, it has problems identifying topics.
- Designed to handle large documents, it often faces issues while analysis headlines and similar short texts.
- The positioning of words is of no importance to LDA. However, while considering social media data, the word positions do hold a lot of value.

However, inspite of its drawbacks, the LDA model has proved successful in giving a satisfactory performance for the given dataset. This has been depicted in the explanation of the LDA result and the goodness of fit scores associated with coherence and perplexity explained previously. Additionally, the visualisation depicting the importance and distribution of certain topic words also shows that a lot of these words are similar and hence, it can be concluded that most of these topics are a part of the same topic of discussion.

Sentiment Analysis:

The advent of technology and connectivity and the low requirements to post opinions online has made social media platforms easily accessible to the masses. These opinions that are commonly referred to as comments on these platforms largely depict the emotions or feelings of a person surrounding products, events and decisions on varied range of topics. It has become important for companies to not only keep a track of their mentions but to also identify the positive, negative or neutral undertones that a particular post related to them contains. This is fuelled the need for progress in the sentiment analysis area (Yue et al., 2019). To help with the improved acceptance of their products and to gauge the public response to an event, sentiment analysis of social media posts has proved to be of great importance.

Sentiment Analysis using VADER:

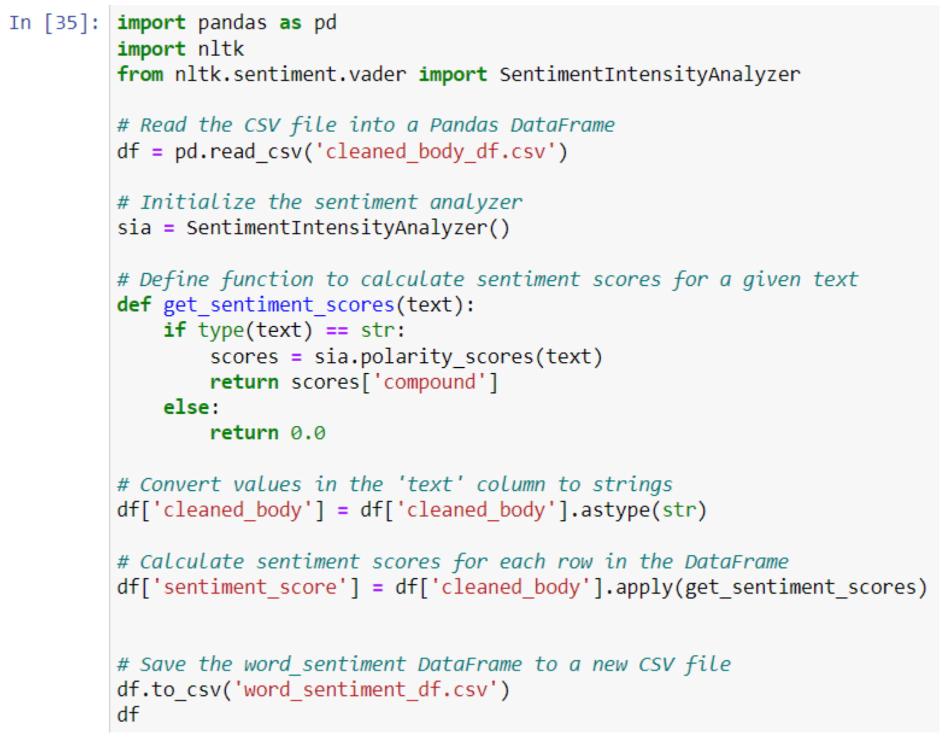
Vader:

The Valence Aware Dictionary sEntiment Reasoner (VADER) is a tool specifically designed to work with social media posts. It is rule based and has an in-built dictionary of lexical features such as emotions and its ability to identify the sentiment of a text. It owes it's importance to social media because of its ability to handle slang and informal language that is widely used on social media.

However, it has its own drawbacks:

- Its unable to understand sarcasm.
- It lacks domain specific knowledge.
- Supports only English language.

Inspite of these drawbacks, VADER still is one of the most widely used sentiment analysis tools for social media platforms.



```
In [35]: import pandas as pd
         import nltk
         from nltk.sentiment.vader import SentimentIntensityAnalyzer

         # Read the CSV file into a Pandas DataFrame
         df = pd.read_csv('cleaned_body_df.csv')

         # Initialize the sentiment analyzer
         sia = SentimentIntensityAnalyzer()

         # Define function to calculate sentiment scores for a given text
         def get_sentiment_scores(text):
             if type(text) == str:
                 scores = sia.polarity_scores(text)
                 return scores['compound']
             else:
                 return 0.0

         # Convert values in the 'text' column to strings
         df['cleaned_body'] = df['cleaned_body'].astype(str)

         # Calculate sentiment scores for each row in the DataFrame
         df['sentiment_score'] = df['cleaned_body'].apply(get_sentiment_scores)


         # Save the word_sentiment DataFrame to a new CSV file
         df.to_csv('word_sentiment_df.csv')
         df
```

Out[35]:

	Comment	Author	cleaned_body	sentiment_score
0	Worst investment in history. Heâ€™s no Warren...	Gold Tau	worst investment history. heâ€™s warren buffett	-0.6249
1	Finally. Now fact checking can be done accurat...	IbexDNB	finally. fact checking done accurately across ...	0.0000
2	Cheif twitt?	Hunter Burrell	cheif twitt?	0.0000
3	Elon Musk played people by pretending he was b...	Guitarplayer DONALD	elon musk played people pretending backing out...	0.4404

We use the above code for analyse the sentiment of our dataset using VADER. We start by initialising the SentimentIntensityAnalyzer from the library nltk.sentiment.vader. We then get the sentiment scores for each row using the function named ‘get_sentiment_scores’. Overall, we used VADER to calculate the sentiment score of each row and input it into a new column in the DataFrame known as ‘sentiment_score’ as shown in the above screenshot.

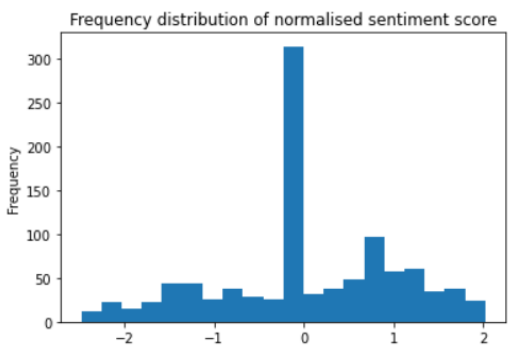
```
In [36]: import pandas as pd
         import matplotlib.pyplot as plt

         # Read the CSV file into a Pandas DataFrame
         df = pd.read_csv('word_sentiment_df.csv')

         # Calculate the normalised sentiment score
         df['normalised_score'] = (df['sentiment_score'] - df['sentiment_score'].mean()) / df['sentiment_score'].std()
         df['normalised_score']
```

```
Out[36]: 0       -1.472051
         1       -0.169787
         2       -0.169787
         3        0.747987
         4       -1.521440
```

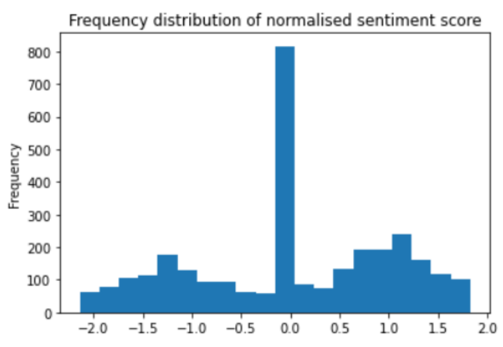
Normalising the sentiment score as a method of preparation to plot their frequency distribution in the next step.

Visualising the result:

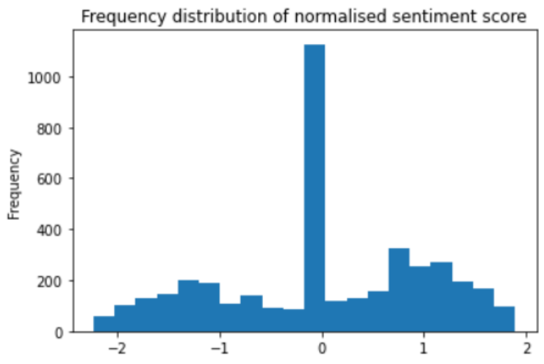
```
In [30]: # Plot a histogram of the normalised score
         df['normalised_score'].plot(kind='hist', bins=20,title="Frequency distribution of normalised sentiment score")
         plt.show()
```



Fig(u1)



Fig(u2)



Fig(u3)

Fig(u1) shows the frequency distribution of the normalised sentiment score for the Reddit comments. It shows a spike very close to 0 but on the negative side. However, the frequency of the words on the positive side is noticeably higher compared to the words with a negative normalised sentiment score. Similarly, Fig(u2) shows the frequency distribution for comments extracted from YouTube videos. As we can see, the frequency of the words having a positive and negative sentiment score is pretty close. Finally, fig(u3) displays the normalised sentiment score distribution for the dataset containing the combined comments from both YouTube and Reddit.

Goodness of fit:

There are multiple ways to calculate the goodness of fit for the sentiment scores generated by VADER for social media dataset. Some examples for this are precision, recall and accuracy. However, to calculate any of these, we need to first have a labelled dataset. That is, a dataset wherein the comments have their sentiments labelled as positive, negative or neutral. This is then compared with the output generated by running the same dataset through VADER. The similarity is then compared to find the Accuracy of the model on that particular dataset. Owing to the absence of these ground-truth sentiment labels in the dataset, we cannot predict the goodness of fit for the sentiment analysis.

Interpretation of results:

There can be multiple algorithms to analyse social media datasets like VADER, textBlob, IBM Watson, etc. However, VADER is the most commonly used algorithm for this purpose. Inspite of its drawbacks, it has satisfactorily provided us with a sentiment score for each comment. On plotting these scores after their normalisation, we see that there is an almost even distribution between the sentiments, with a little bias towards the positive side. There is a higher frequency of comments having positive sentiments as displayed in the screenshots in Fig(u1), Fig(u2) and Fig(u3).

Critical Evaluation of methods used:

We discussed the various drawbacks of VADER which include it being oblivious to sarcasm. It also cannot understand the context of a comment if it is in response to another post. This can't lead to comments like 'Great investment genius' being interpreted as positive when it was sarcastic in nature aimed to be negative. Similarly, while replying to another comment, the users positive or negative response could be to a previous comment and not to the original intended post.

Conclusion:

Having a look at the results obtained in the above steps we can conclude that this practical experience of running the dataset through LDA for the purpose of topic modelling and VADER for its subsequent sentiment analysis has been successful. During topic modelling, LDA has provided 10 different topics and they contain words similar to each other. This has been reinforced with a coherence score of 0.30334. This means that there is a high chance that most of these comments are of the same topic. Similarly, using VADER has helped provide a satisfactory sentiment score for the comments. The plotting of this score shows that there are a larger number of positive comments but the comments on the negative side have a higher level of negativity.

Inspite of this combination giving a satisfactory output, we could have used other algorithms that could have provided a better result. For example, IBM Watson is usually used for social media sentiment analysis. This may have provided us with a more accurate result.

In terms of the response to the event of Elon Musk taking over Twitter, our result shows that there is a large number of people on these platforms supporting the move. Reddit users support this move in a larger proportion compared to YouTube. We can see that by having a look at the frequency distribution of the sentiment scores of the datasets. However, we can also see that the fewer people who oppose this move have a lot of negativity in their comments.

Potential steps for future work:

We can run this dataset on other algorithms to see if they can provide a different sentiment analysis. Alternatively, we can collect additional data from different platforms like ‘Twitter’ and ‘Facebook’ to provide a larger range of demographic opinion regarding the event. Additionally, we can also classify which age groups largely use which platforms. This will also help us identify which age groups have what overall bias in their opinions.

Companies can use these sorts of sentiment analysis to help identify whether a particular decision or product has been well received by their target audience or not.

Reference:

1. Hassan , A. et al. (2014) Sentiment analysis algorithms and applications: A survey, Ain Shams Engineering Journal. Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S2090447914000550 (Accessed: April 27, 2023).

2. Beigi, G. et al. (1970) An overview of sentiment analysis in social media and its applications in disaster relief, SpringerLink. Springer International Publishing. Available at: https://link.springer.com/chapter/10.1007/978-3-319-30319-2_13 (Accessed: April 27, 2023).

3. Turner, A. et al. (2023) How many users does Twitter have?, BankMyCell. Available at: https://www.bankmycell.com/blog/how-many-users-does-twitter-have (Accessed: April 27, 2023).

4. Iqbal, M. (2023) Twitter revenue and Usage Statistics (2023), Business of Apps. Available at: https://www.businessofapps.com/data/twitter-statistics/ (Accessed: April 27, 2023).

5. Blei, D.M. et al. (2003) Latent dirichlet allocation - Journal of Machine Learning Research. Available at: https://jmlr.org/papers/volume3/blei03a/blei03a.pdf (Accessed: April 27, 2023).

6. Yue, L. et al. (2019) A survey of sentiment analysis in social media, Knowledge and Information Systems. Available at: https://dl.acm.org/doi/10.1007/s10115-018-1236-4 (Accessed: April 27, 2023).