Sentiment Analysis - Open AI versus NLTK

Analysis of the Russia - Ukraine war TWEETS

GOOGLE COLAB NOTEBOOK

Zoia Panasenko

DOMAIN OF INTEREST

- Analysis of the Russia-Ukraine war tweets.

- Lets explore how sentiment analysis changed over time. Were tweets with negative sentiment liked or retweeted more vs positive ones. We are planning to implement Open AI API (gpt-3.5-turbo /text-davinci-003) and NLTK libraries for comparison analysis.

DATASET

✓ Original Dataset "Russia-Ukraine war - Tweets Dataset (65 days)", published during first 65 days of Russia-Ukraine war is 8.24GB.

✓ Author - Daria Purtova was analysing tweets based on the daily basis starting from 2022/01/01 till 2022/03/06. She was conducting an upload of the tweets (via Twitter API) based on the following search words - 'ukraine war', 'ukraine troops', 'ukraine border', 'ukraine NATO', 'StandwithUkraine', 'russian troops', 'russian border ukraine', 'russia invade'.

✓ As the total dataset is 8.24GB, below analysis is processed only on the subset data of 1.6GB, which comprises following searches: 'russia invade, 'StandwithUkraine','ukraine war'.

✓ Data was first downloaded in the form of separate csv files, which were combined together via Terminal into the combined dataset of 1.6GB (500k rows), downloaded on the Github with utilization of the git lfs - Tweets.csv.

Downloading Libraries:

```
[2]  import pandas as pd
     import seaborn as sns
     import numpy as np
     import matplotlib.pyplot as plt
```

Reading the Dataset from the GitHub:

```
[3]  url = 'https://media.githubusercontent.com/media/zoiia/project/main/tweets.csv'
     df = pd.read_csv(url)
     df.head()
```

```
<ipython-input-3-569123a603ea>:2: DtypeWarning: Columns (5,7,8,9,10,11,19,21) have mixed types. Specify dtype option on import or set low_
  df = pd.read_csv(url)
```

	_type	url	date	content	renderedContent	id	
0	snscrape.modules.twitter.Tweet	https://twitter.com/pat_ianni/status/150025982...	2022-03-05 23:59:50+00:00	JOE BIDEN SAYS HOW DO WE GET TO A PLACE WHERE ...	JOE BIDEN SAYS HOW DO WE GET TO A PLACE WHERE ...	1500259827154505728	'snscra
1	snscrape.modules.twitter.Tweet	https://twitter.com/luxeprogressive/status/150...	2022-03-05 23:59:05+00:00	@ProfPaulPoast He doesn't have to like it but ...	@ProfPaulPoast He doesn't have to like it but ...	1500259636863246336	'snscra

```
df.shape
```

```
(550606, 29)
```

As there were 3 large csv files combined together in a combined dataset we have to delete 2 extra headers that we got during this process (as one file went under another file we gathered extra headers). Naturally we can find those 2 extra headers by searching any of the column names inside of the dataset. This search shows that we have extra headers on the row #170835 and #318981.

```
print(df.loc[df['url'] == "url"])
```

```
        _type  url  date  content  renderedContent  id  user  replyCount  \
170835  _type  url  date  content  renderedContent  id  user  replyCount
318981  _type  url  date  content  renderedContent  id  user  replyCount

        retweetCount  likeCount  ...  retweetedTweet  quotedTweet  \
170835  retweetCount  likeCount  ...  retweetedTweet  quotedTweet
318981  retweetCount  likeCount  ...  retweetedTweet  quotedTweet

        inReplyToTweetId  inReplyToUser  mentionedUsers  coordinates  place  \
170835  inReplyToTweetId  inReplyToUser  mentionedUsers  coordinates  place
318981  inReplyToTweetId  inReplyToUser  mentionedUsers  coordinates  place

        hashtags  cashtags  Searh
170835  hashtags  cashtags  Searh
318981  hashtags  cashtags  Searh

[2 rows x 29 columns]
```

Dropping these rows from the dataset and creating new dataset - *df1* based on the original *df* for the sake of good order.

```
[5]  df1 = df.drop([170835, 318981])
```

0s

Checking for NA values:

```
[ ]  na_count = df1.isna().sum()
     print(na_count)
```

```
_type                  0
url                    0
date                   0
content                0
renderedContent        0
id                     0
user                   0
replyCount             0
retweetCount           0
likeCount              0
quoteCount             0
conversationId         0
lang                   0
source                 0
sourceUrl              0
sourceLabel            0
outlinks          372047
tcooutlinks       372047
media             482442
retweetedTweet    550604
quotedTweet       474599
inReplyToTweetId  308299
inReplyToUser     308299
mentionedUsers    272820
coordinates       541637
place             541637
hashtags          357628
cashtags          549207
Searh                  0
dtype: int64
```

Dropping EMPTY columns from the dataset:

header visibility

```
df1.drop(['outlinks', 'tcooutlinks', 'media', 'retweetedTweet', 'quotedTweet', 'inReplyToTweetId', 'inReplyToUser', 'mentionedUsers', 'c
```

Checking for NA values one more time:

```
[ ]  na_count1 = df1.isna().sum()
     print(na_count1)
```

```
_type             0
url               0
date              0
content           0
renderedContent   0
id                0
user              0
replyCount        0
retweetCount      0
likeCount         0
quoteCount        0
conversationId    0
lang              0
source            0
sourceUrl         0
sourceLabel       0
Searh             0
Time Stamp        0
Date              0
dtype: int64
```

Some columns have a lot of null values:

```
[11] df1['retweetCount'].value_counts()
```

```
0       388221
0        47383
1        43873
2        14671
1         7563
         ...
1062         1
3057         1
675          1
1326         1
328          1
Name: retweetCount, Length: 1275, dtype: int64
```

```
df1['likeCount'].value_counts()
```

```
0       248997
1        89685
2        38370
0        29542
3        20893
         ...
497          1
1956         1
3925         1
1226         1
1076         1
Name: likeCount, Length: 2657, dtype: int64
```

After dropping NA columns out dataset became much leaner (vs df.shape (550606, 29)):

```
[14] df1.shape
```

```
(550604, 17)
```

Adding new TIME STAMP column to the dataset - it is same date column, but trasformed to the date time format. Also adding DATE column with shorten Date format.

```
[15] df1['Time Stamp'] = pd.to_datetime(df1['date'], format = "%Y-%m-%d")
```

```
[16] df1['Date'] = df1['Time Stamp'].dt.date
```

```
df1.head(2)
```

int	quoteCount	conversationId	lang	source	sourceUrl	sourceLabel	Search	Time Stamp	Date
99	4	1500259827154505728	en	<a href="http://twitter.com/download/android" ...	http://twitter.com/download/android	Twitter for Android	Russia invade	2022-03-05 23:59:50+00:00	2022-03-05
2	0	1500097922788175879	en	<a href="http://twitter.com/download/iphone" r...	http://twitter.com/download/iphone	Twitter for iPhone	Russia invade	2022-03-05 23:59:05+00:00	2022-03-05

Converting following columns into INT (number format) data type:

```
[14] df1[['replyCount', 'retweetCount', 'likeCount','quoteCount']] = df1[['replyCount', 'retweetCount', 'likeCount','quoteCount']].astype(int)
```

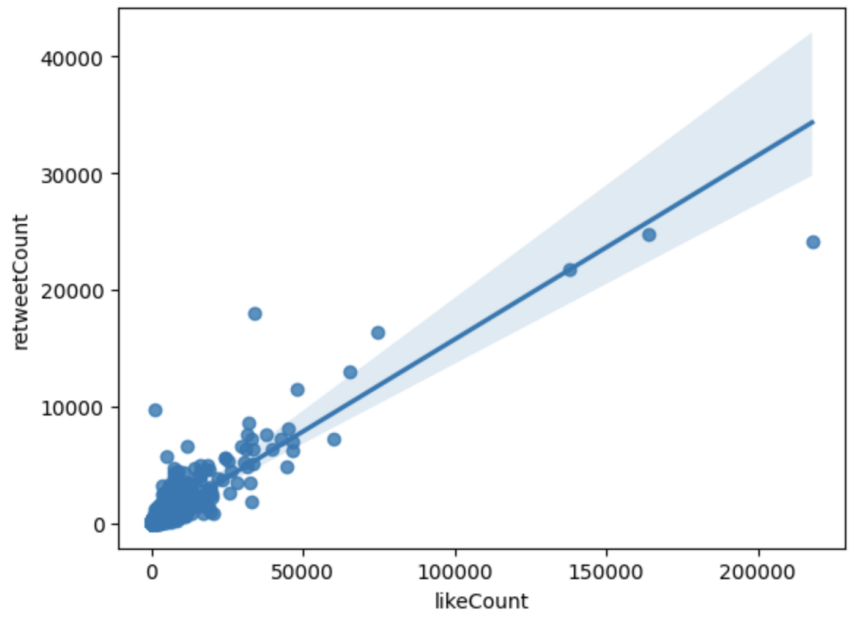
```
[26] df1.dtypes
```

```
_type                       object
url                         object
date                        object
content                     object
renderedContent             object
id                          object
user                        object
replyCount                   int64
retweetCount                 int64
likeCount                    int64
quoteCount                   int64
conversationId              object
lang                        object
source                      object
sourceUrl                   object
sourceLabel                 object
outlinks                    object
tcooutlinks                 object
media                       object
retweetedTweet              object
```

Plotting correlation between count of likes (likeCount) and count of retweets (retweetCount) - strong, positive correlation:

```
sns.regplot(x='likeCount', y='retweetCount', data = df1)
```

```
<Axes: xlabel='likeCount', ylabel='retweetCount'>
```



Checking statistics for the data set:

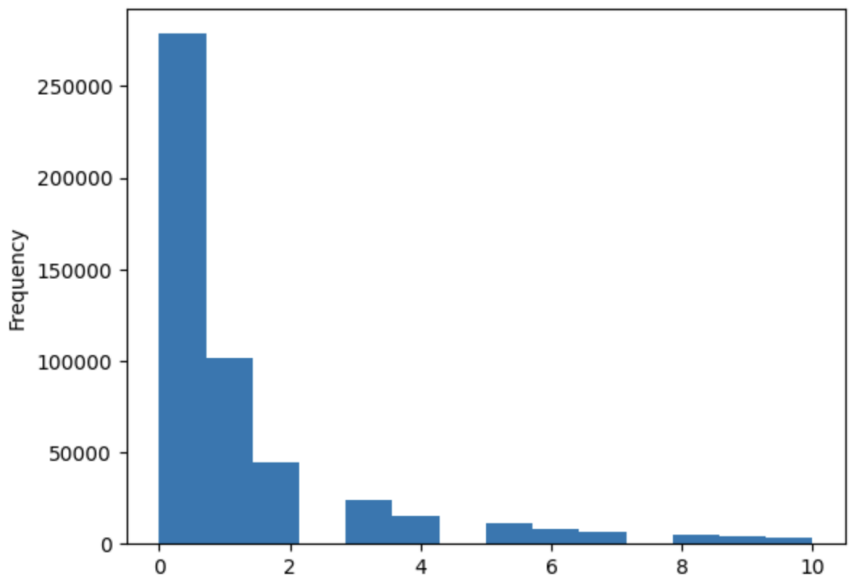
```
[16] df1.describe()
```

	replyCount	retweetCount	likeCount	quoteCount
count	550604.000000	550604.000000	550604.000000	550604.000000
mean	1.512581	3.860520	18.312646	0.412865
std	36.471867	95.741951	552.590509	15.504680
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000
75%	1.000000	0.000000	2.000000	0.000000
max	13603.000000	24732.000000	217883.000000	7454.000000

Plotting histogram for the # of likes:

```
[19] df1['likeCount'].plot(bins = np.linspace(0, 10, 15), kind = 'hist')
```

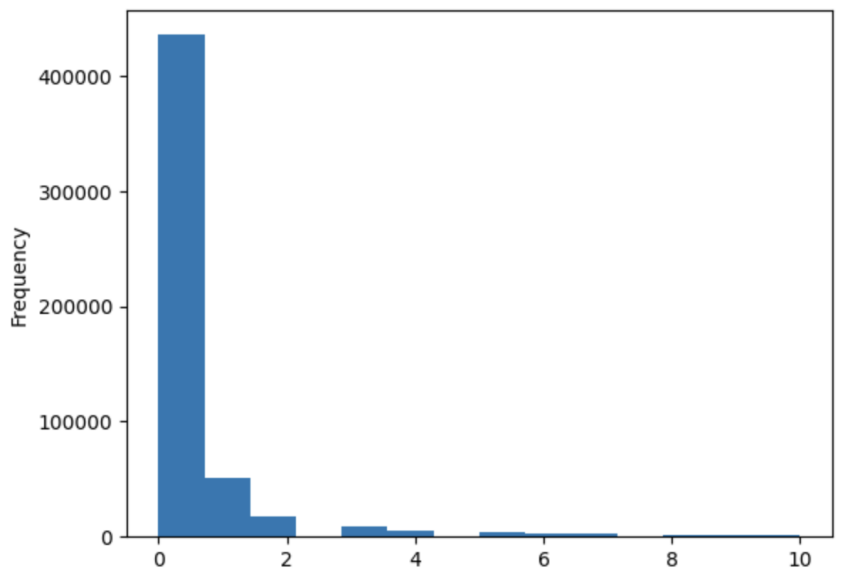
```
<Axes: ylabel='Frequency'>
```



Plotting histogram for the # of retweets:

```
[18] df1['retweetCount'].plot(bins = np.linspace(0, 10, 15), kind = 'hist')
```

```
<Axes: ylabel='Frequency'>
```



Same without zeros:

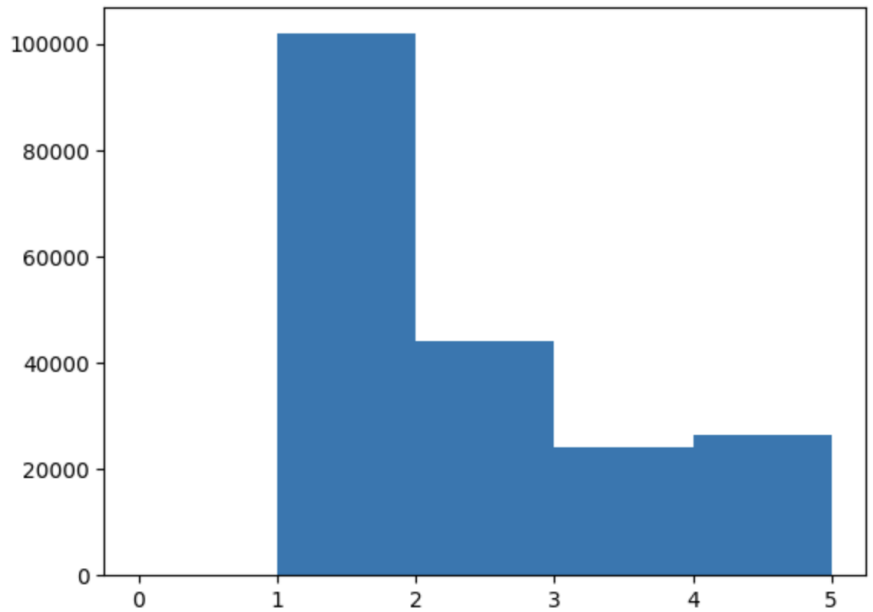
Plotting histogram for the # of likes [without zeros].

```
[20] no_zero = df1['likeCount'].values
```

```
[21] no_zero = no_zero[no_zero != 0]
```

```
plt.hist(no_zero, bins = np.linspace(0,5,6))
```

```
(array([     0., 101832.,  44141.,  24043.,  26494.]),
 array([0., 1., 2., 3., 4., 5.]),
 <BarContainer object of 5 artists>)
```



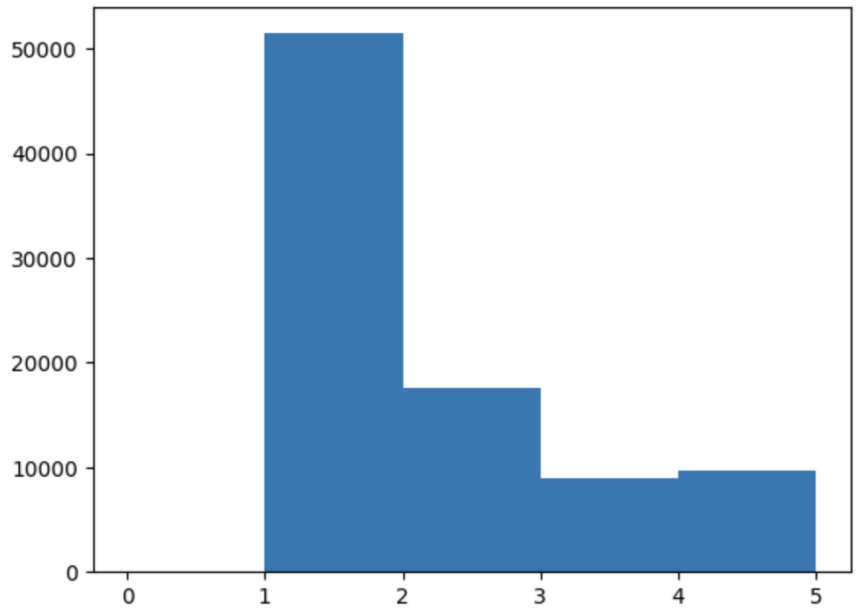
Plotting histogram for the # retweets [without zeros].

```
[27] no_zero1 = df1['retweetCount'].values
```

```
[28] no_zero1 = no_zero1[no_zero1 != 0]
```

```
plt.hist(no_zero1, bins = np.linspace(0,5,6))
```

```
(array([    0., 51436., 17514.,  8949.,  9680.]),
 array([0., 1., 2., 3., 4., 5.]),
 <BarContainer object of 5 artists>)
```



Checking for the distribution of the Search topics:

```
[30] df1.Searh.value_counts()
```

```
Ukraine war          231624
Russia invade        170835
StandWithUkraine     148145
Name: Searh, dtype: int64
```

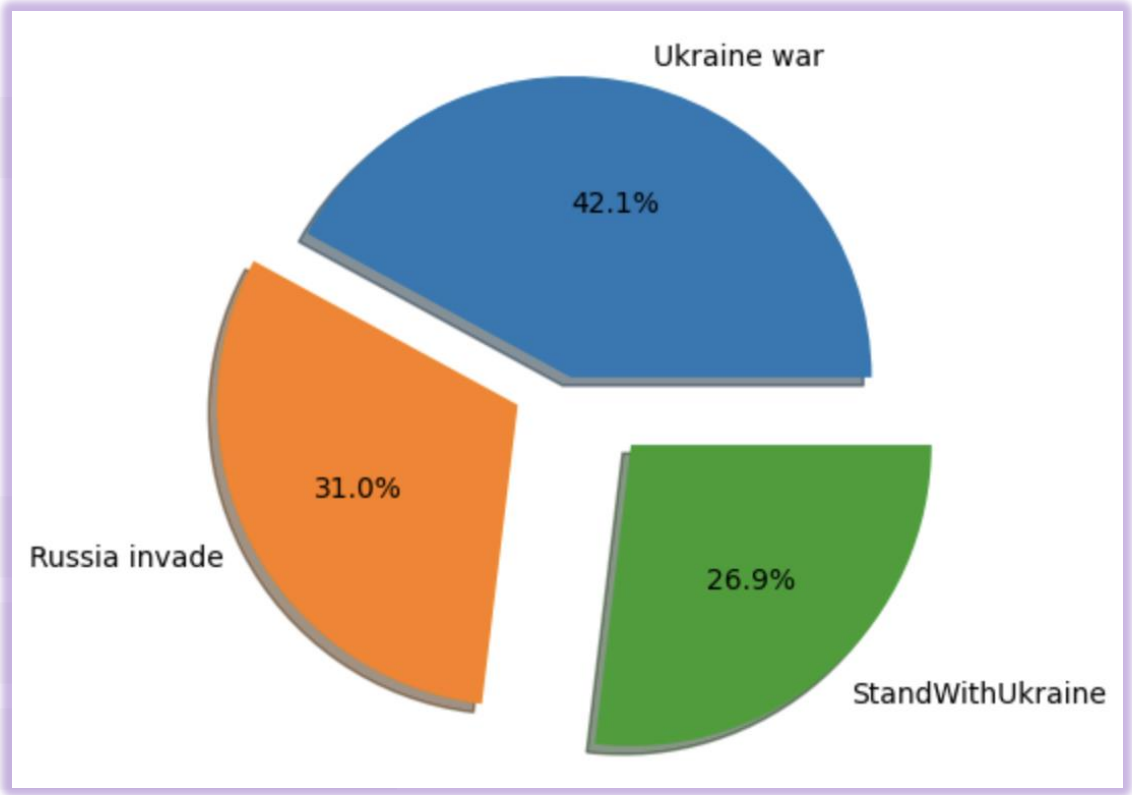
Plotting share of each Search topic:

```
[31] label = df1.Searh.value_counts().index
```

```
[32] sizes = df1.Searh.value_counts().values
```

```
[33] explode = [0, 0.2, 0.3]
```

```
[34] fig, ax = plt.subplots()
     ax.pie(sizes, explode = explode, labels = label, autopct='%1.1f%%', shadow = True)
     plt.show()
```

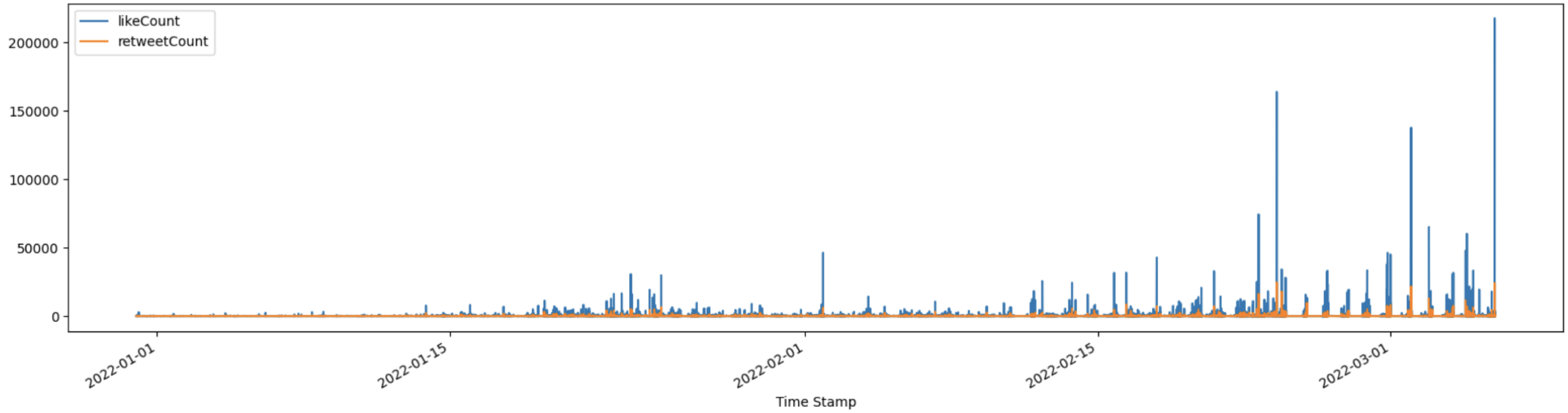


Plotting # of retweets and # of likes time series.

Notable that retweets and likes are overlapping, naturally those posts which are being liked are usually retweeted.

```
[ ]  df1.plot(x= 'Time Stamp', y=['likeCount', 'retweetCount'], figsize = (20,5), kind = 'line')
```

```
<Axes: xlabel='Time Stamp'>
```



DATAFRAME PRE-PROCESSING

```
[42] # Dropping all the columns that are not important for this analysis

     df_filtered = df1.drop(['_type', 'url', 'old_date', 'id', 'conversationId', 'source', 'sourceUrl', 'sourceLabel', 'Searh'], axis = 1, inp
```

```
[43] df_filtered.head(1)
```

	content	renderedContent	user	replyCount	retweetCount	likeCount	quoteCount	lang	Time Stamp	Date
0	JOE BIDEN SAYS HOW DO WE GET TO A PLACE WHERE ...	JOE BIDEN SAYS HOW DO WE GET TO A PLACE WHERE ...	{'_type': 'snscrape.modules.twitter.User', 'us...	14	26	99	4	en	2022-03-05 23:59:50+00:00	2022-03-05

```
[45] df_filtered = df_filtered[df_filtered['lang'] == 'en']
```

```
[46] df_filtered.shape

     (473334, 10)
```

```
[47] duplicates = df_filtered.duplicated()

     # get the count of duplicates
     num_duplicates = duplicates.sum()

     print("Number of duplicate rows: ", num_duplicates)

     Number of duplicate rows:  1399
```

```
[48] df_filtered.shape

     (473334, 10)
```

```
[49] df_filtered = df_filtered.drop_duplicates()
```

```
[50] df_filtered.shape

     (471935, 10)
```

1. Filtering out unnecessary columns
2. Leaving only English tweets
3. Dropping duplicates

ANALYZING THE SAME SAMPLE DATASET IN BOTH SCENARIOS (5000 rows, 10 columns):

```
[51] sample_size = 5000
     seed = 42
     df2 = df_filtered.sample(n=sample_size, random_state=seed).reset_index()
     df2.drop(['index'], axis = 1, inplace = True)
```

```
[52] df2.head(2)
```

	content	renderedContent	user	replyCount	retweetCount	likeCount	quoteCount	lang	Time Stamp	Date
0	Everyone is annoyed by Putin's delusions.\n#NO...	Everyone is annoyed by Putin's delusions.\n#NO...	{'_type': 'snscrape.modules.twitter.User', 'us...	0	0	0	0	en	2022-02-26 23:42:46+00:00	2022-02-26
1	Russia’s deputy ambassador to the United Natio...	Russia’s deputy ambassador to the United Natio...	{'_type': 'snscrape.modules.twitter.User', 'us...	1	1	7	0	en	2022-02-16 04:25:33+00:00	2022-02-16

```
[53] df2.shape
```

```
(5000, 10)
```

… LOADING of the necessary packages and libraries

gpt-3.5-turbo

```
!pip install pandas openai requests
!pip install tqdm
!pip install python-docx
```

```
[55] import pandas as pd
     import openai
     import requests
     from tqdm import tqdm
     import time
     import docx
```

▸ OPEN AI KEY - hidden cell

```
[56]  Show code
```

```
[57] pip install backoff
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting backoff
  Downloading backoff-2.2.1-py3-none-any.whl (15 kB)
Installing collected packages: backoff
Successfully installed backoff-2.2.1
```

```
[62] from openai.error import RateLimitError, OpenAIError
     from requests.exceptions import RequestException
     import backoff
```

NLTK Library

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop = stopwords.words('english')
```

```
from textblob import TextBlob
nltk.download('punkt')
```

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

from nltk.sentiment import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

gpt-3.5-turbo

1. Paid Account with Open AI to connect via API
2. Connect to the OPEN AI API with the KEY
3. Write function which will utilize gpt-3.5-turbo model to deliver results of the sentiment Analysis
4. As most of the time model is overloaded backoff concept
5. Be prepared for the long processing time (buy COLAB extra CPU units if necessary)

```
[64] sentiments = []

     for content in tqdm(df2["content"], desc="tweets analysis"):
         sentiment = tweet_analyzer(content)
         sentiments.append(sentiment)

     df2["sentiment"] = sentiments
```

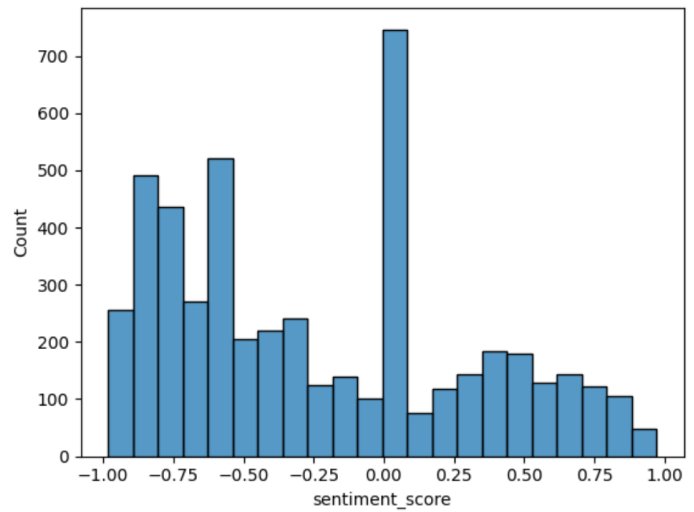
```
tweets analysis:  99%|▉| 4941/5000 [1:00:37<00:36,  1.64it/s]NEGATIVE
tweets analysis:  99%|▉| 4942/5000 [1:00:38<00:35,  1.65it/s]NEGATIVE
tweets analysis:  99%|▉| 4943/5000 [1:00:39<00:46,  1.22it/s]NEUTRAL
tweets analysis:  99%|▉| 4944/5000 [1:00:40<00:42,  1.33it/s]NEGATIVE
tweets analysis:  99%|▉| 4945/5000 [1:00:40<00:38,  1.41it/s]NEUTRAL
tweets analysis:  99%|▉| 4946/5000 [1:00:41<00:35,  1.52it/s]NEGATIVE
tweets analysis:  99%|▉| 4947/5000 [1:00:41<00:33,  1.56it/s]NEUTRAL
tweets analysis:  99%|▉| 4948/5000 [1:00:42<00:32,  1.61it/s]NEGATIVE
tweets analysis:  99%|▉| 4949/5000 [1:00:43<00:30,  1.68it/s]NEUTRAL
tweets analysis:  99%|▉| 4950/5000 [1:00:43<00:28,  1.75it/s]NEUTRAL
tweets analysis:  99%|▉| 4951/5000 [1:00:44<00:29,  1.69it/s]NEGATIVE
tweets analysis:  99%|▉| 4952/5000 [1:00:44<00:28,  1.68it/s]NEGATIVE
tweets analysis:  99%|▉| 4953/5000 [1:00:45<00:27,  1.70it/s]NEGATIVE
tweets analysis:  99%|▉| 4954/5000 [1:00:46<00:29,  1.57it/s]NEGATIVE
tweets analysis:  99%|▉| 4955/5000 [1:00:46<00:27,  1.62it/s]NEUTRAL
tweets analysis:  99%|▉| 4956/5000 [1:00:47<00:26,  1.65it/s]NEGATIVE
tweets analysis:  99%|▉| 4957/5000 [1:00:47<00:26,  1.62it/s]NEGATIVE
tweets analysis:  99%|▉| 4958/5000 [1:00:48<00:25,  1.67it/s]NEGATIVE
tweets analysis:  99%|▉| 4959/5000 [1:00:49<00:23,  1.71it/s]NEGATIVE
tweets analysis:  99%|▉| 4960/5000 [1:00:49<00:22,  1.74it/s]NEGATIVE
tweets analysis:  99%|▉| 4961/5000 [1:00:50<00:22,  1.70it/s]POSITIVE
tweets analysis:  99%|▉| 4962/5000 [1:00:50<00:23,  1.65it/s]NEUTRAL
tweets analysis:  99%|▉| 4963/5000 [1:00:51<00:22,  1.66it/s]NEUTRAL
tweets analysis:  99%|▉| 4964/5000 [1:00:52<00:21,  1.68it/s]NEGATIVE
tweets analysis:  99%|▉| 4965/5000 [1:00:52<00:20,  1.73it/s]NEUTRAL
tweets analysis:  99%|▉| 4966/5000 [1:00:53<00:20,  1.66it/s]NEGATIVE
tweets analysis:  99%|▉| 4967/5000 [1:00:53<00:19,  1.67it/s]NEUTRAL
tweets analysis:  99%|▉| 4968/5000 [1:00:54<00:18,  1.69it/s]NEGATIVE
tweets analysis:  99%|▉| 4969/5000 [1:00:54<00:18,  1.71it/s]NEUTRAL
tweets analysis:  99%|▉| 4970/5000 [1:00:55<00:17,  1.68it/s]NEGATIVE
tweets analysis:  99%|▉| 4971/5000 [1:00:56<00:17,  1.70it/s]NEGATIVE
```

NLTK Library

1. Basic Text Preprocessing:
   a) Lower case text
   b) Punctuation Removal
   c) Stop words Removal
2. Spelling Correction (TextBlob)
3. Tokenization
4. Lemmatization
5. Initialize the sentiment intensity analyzer

```
[120] sns.histplot(df3['sentiment_score'])
```

```
<Axes: xlabel='sentiment_score', ylabel='Count'>
```



STANDARTIZATION OF RESULTS

Create numeric equivalent for the “sentiment” column of the gpt-3.5-turbo analysis:

```
[75] def label_to_numerical(label):
        if label == 'NEGATIVE':
            return -1
        elif label == 'NEUTRAL':
            return 0
        else:  # label == 'POSITIVE'
            return 1

    # Create a new column 'sentiment_numeric' based on the 'sentiment' column
    df2['sentiment_numeric'] = df2['sentiment'].apply(label_to_numerical)

    df2.head(2)
```

Create label equivalent (Neutral/Positive/Negative) for the “sentiment_score” column of the NLTK analysis:

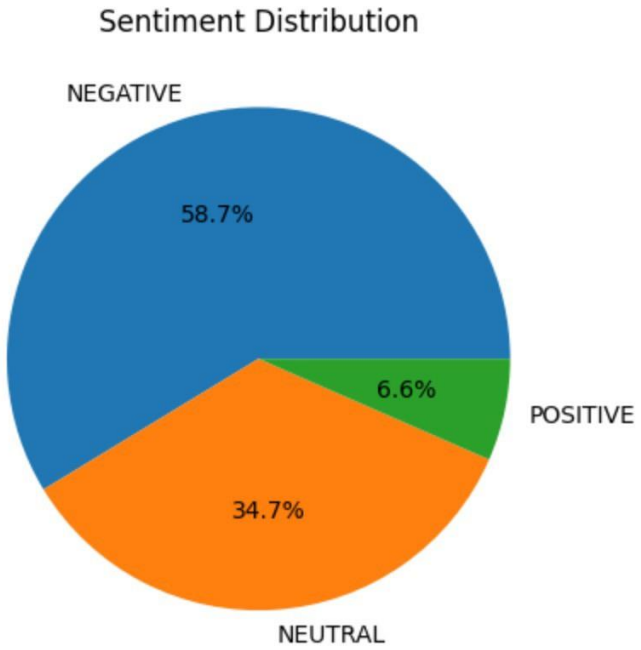
```
def score_to_label(score):
    if score < -0.05:
        return "NEGATIVE"
    elif score > 0.05:
        return "POSITIVE"
    else:
        return "NEUTRAL"

df3['sentiment_label'] = df3['sentiment_score'].apply(score_to_label)
df3.head(2)
```

	user	replyCount	retweetCount	likeCount	quoteCount	lang	Time Stamp	Date	sentiment	sentiment_numeric	tokens	sentiment_score	sentiment_label
	{'_type': 'odules.twitter.User', 'us...	0	0	0	0	en	2022-02-26 23:42:46+00:00	2022-02-26	NEGATIVE	-1	[everyone, duties, nowarinukraine, standwithuk...	0.00	NEUTRAL
	{'_type': 'odules.twitter.User', 'us...	1	1	7	0	en	2022-02-16 04:25:33+00:00	2022-02-16	NEGATIVE	-1	[russian, deputy, ambassador, united, nations,...	-0.34	NEGATIVE

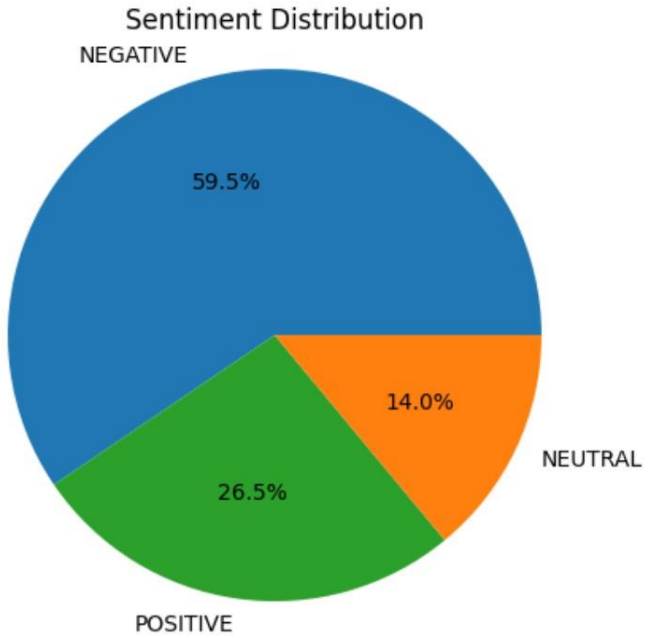
gpt-3.5-turbo

```
[71] df2['sentiment'].value_counts().plot(kind='pie', autopct='%1.1f%%')
     plt.title('Sentiment Distribution')
     plt.ylabel('')
     plt.show()
```



NLTK Library

```
[175] df3['sentiment_label'].value_counts().plot(kind='pie', autopct='%1.1f%%')
      plt.title('Sentiment Distribution')
      plt.ylabel('')
      plt.show()
```

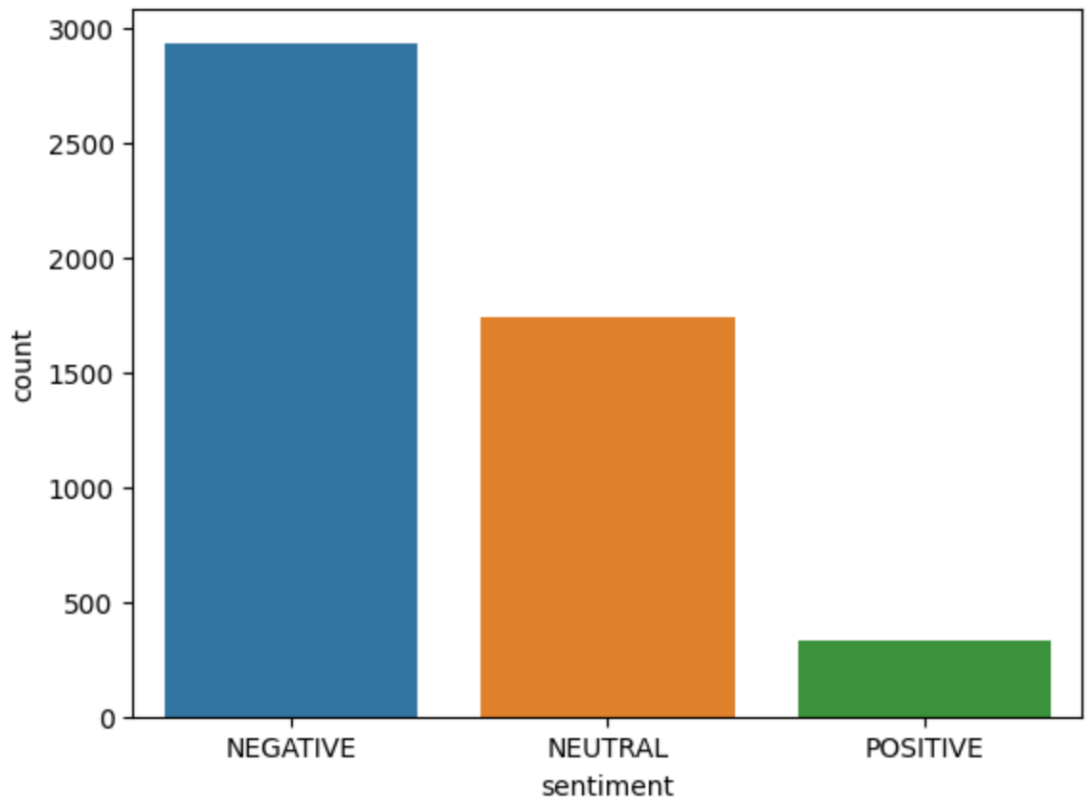


Both gpt-3.5-turbo and NLTK underlines that there are almost 60% of negative tweets.

Same in numbers:

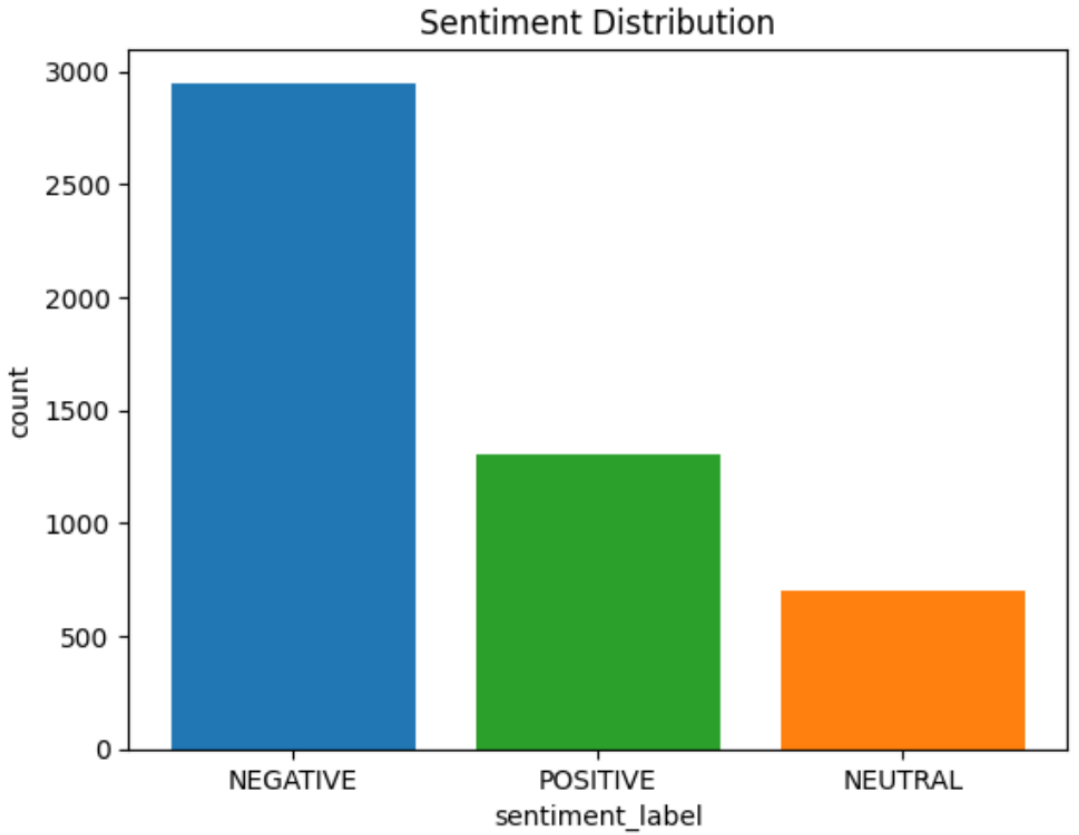
gpt-3.5-turbo

```
[139] sorted_order = ['NEGATIVE', 'NEUTRAL', 'POSITIVE']
      sns.countplot(x=df2['sentiment'], order=sorted_order)
      plt.show()
```



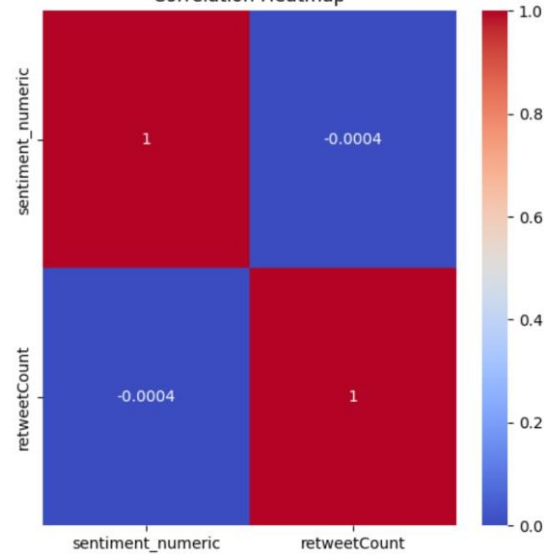
NLTK Library

```
[177] sorted_order = ['NEGATIVE','POSITIVE', 'NEUTRAL']
      sns.countplot(x=df3['sentiment_label'], order=sorted_order)
      plt.show()
```

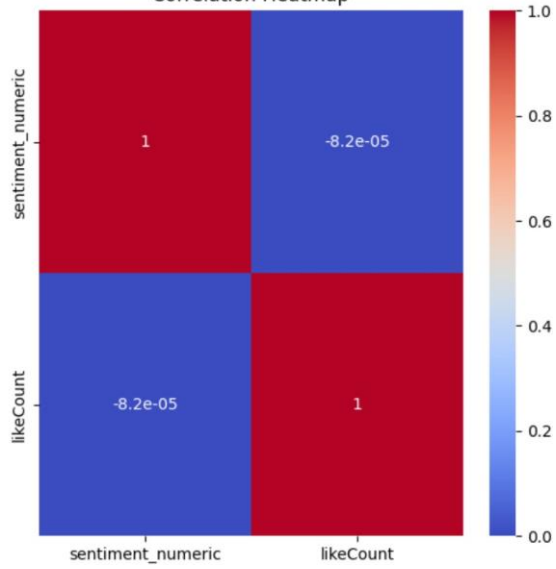


gpt-3.5-turbo RESULTS

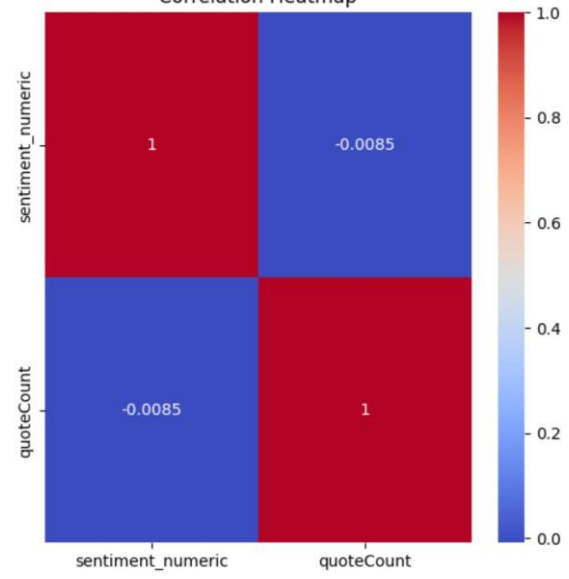
	sentiment_numeric	retweetCount
sentiment_numeric	1.000000	-0.000402
retweetCount	-0.000402	1.000000



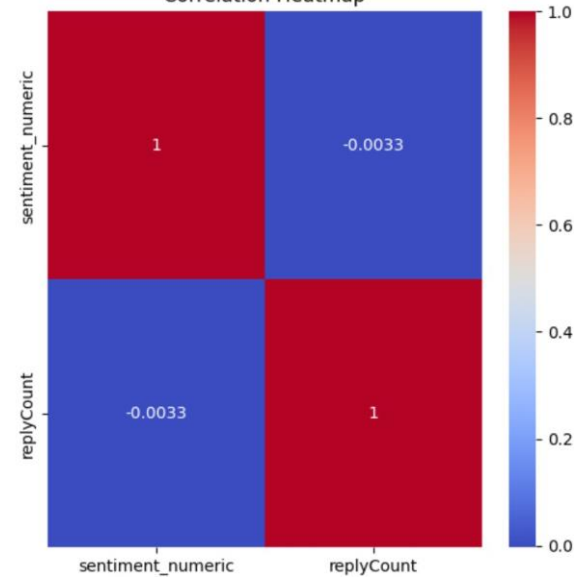
	sentiment_numeric	likeCount
sentiment_numeric	1.000000	-0.000082
likeCount	-0.000082	1.000000



	sentiment_numeric	quoteCount
sentiment_numeric	1.000000	-0.008523
quoteCount	-0.008523	1.000000

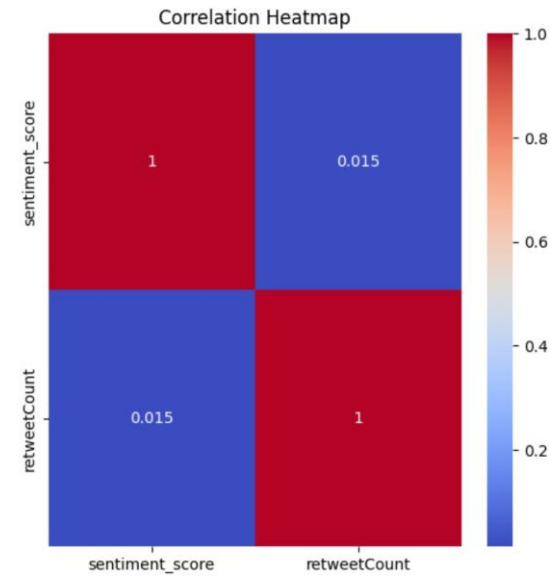


	sentiment_numeric	replyCount
sentiment_numeric	1.000000	-0.003341
replyCount	-0.003341	1.000000

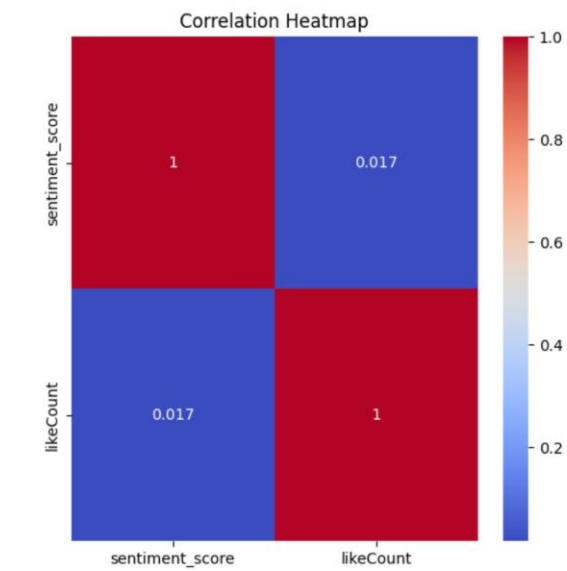


NLKT LIBRARY RESULTS

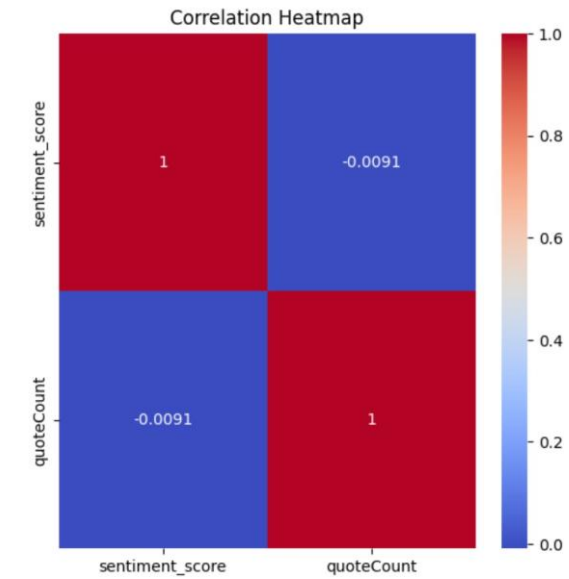
	sentiment_score	retweetCount
sentiment_score	1.000000	0.014607
retweetCount	0.014607	1.000000



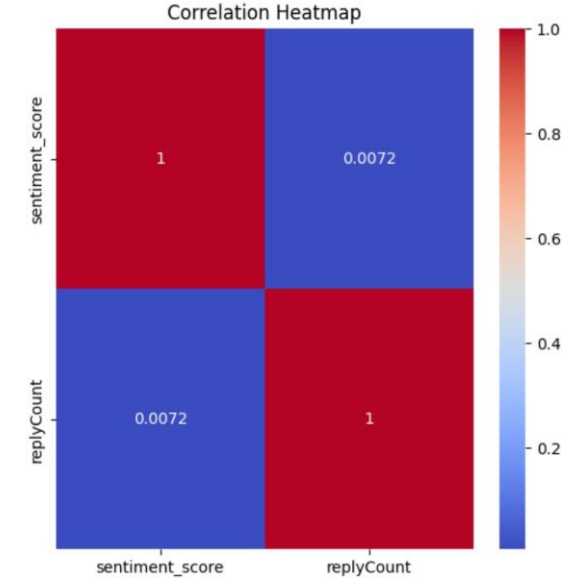
	sentiment_score	likeCount
sentiment_score	1.000000	0.016748
likeCount	0.016748	1.000000



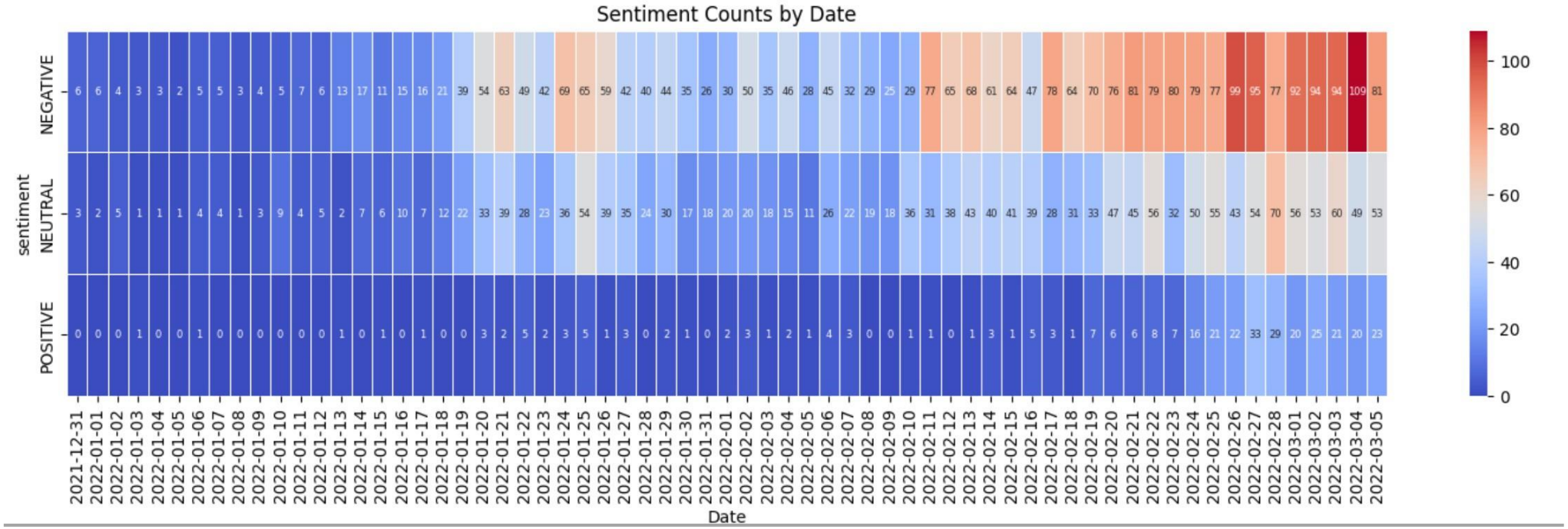
	sentiment_score	quoteCount
sentiment_score	1.000000	-0.009143
quoteCount	-0.009143	1.000000



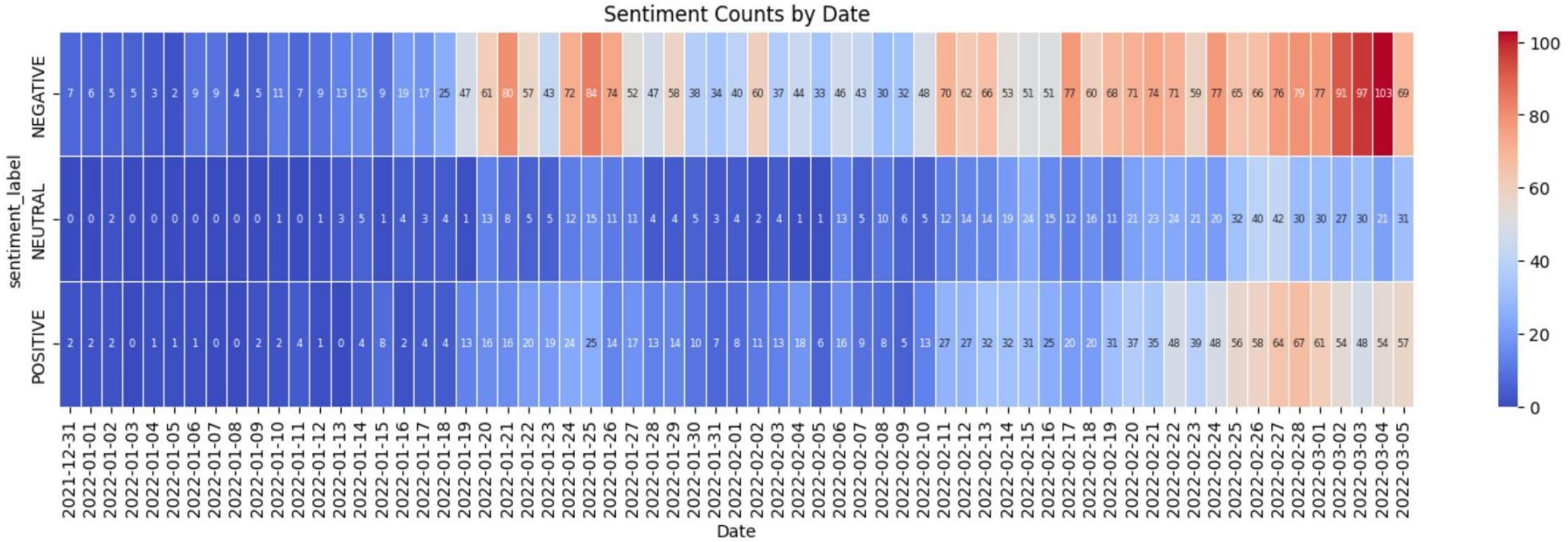
	sentiment_score	replyCount
sentiment_score	1.000000	0.007161
replyCount	0.007161	1.000000



gpt-3.5-turbo RESULTS



NLKT LIBRARY RESULTS



CONCLUSIONS:

```
[168] df3.to_csv('overall-results.csv')
```

```
[169] comparison = df3['sentiment'] == df3['sentiment_label']
```

```
[170] print(comparison)
```

```
0       False
1        True
2        True
3       False
4        True
        ...
4995    False
4996     True
4997    False
4998     True
4999     True
Length: 5000, dtype: bool
```

```
[171] comparison.value_counts()
```

```
True     2507
False    2493
dtype: int64
```

```
[174] print(f'{comparison.sum()/comparison.size: .2%}')
```

```
 50.14%
```

1. As per Correlation Analysis (slide #10) we can conclude that there is no significant correlation between sentiment of the tweets and such features as count of likes, count of retweets, count of replies and count of quotes.
In other words there is no correlation between sentiment and these factors.
This has been shown by both gpt-3.5 turbo and by the NLKT analysis.

2. Previous slide (slide #11) presents a heatmap of sentiment / tweet count per day.
Despite of the fact that only 51% of the gpt-3.5 turbo and NLKT analysis overlap (see hereby on the left) – “mood” heatmap is pretty similar for both cases. The war started on February 24, 2022 and both maps reflect spike in the amount of negative tweets.