In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-pytho
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all file

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserv
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside o
```

```
/kaggle/input/chatgpt-twitter-dataset/chatgpt1.csv
```

In [2]:

```
df = pd.read_csv('/kaggle/input/chatgpt-twitter-dataset/chatgpt1.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Datetime	Tweet Id	Text	Username	
0	2023-01-22 13:44:34+00:00	1617156270871699456	ChatGPTで遊ぶの忘れてた！！\n書類作るコード書いてみてほしいのと、\nどこまで思考整...	mochico0123	https://twitter.c
1	2023-01-22 13:44:39+00:00	1617156291046133761	@AlexandrovnaIng Prohibition of ChatGPT has be...	Caput_LupinumSG	https://twitter.cc
2	2023-01-22 13:44:44+00:00	1617156308926349312	Schaut Euch an, was @fobizz @DianaKnodel alles...	ciffi	https://twitte
3	2023-01-22 13:44:49+00:00	1617156332297256961	Bow down to chatGPT ..... https://t.co/ENTSzi...	Vishwasrisiri	https://twitte
4	2023-01-22 13:44:52+00:00	1617156345064570880	Profilinde vatan, Türkiye falan yazan bireyler...	0xGenetikciniz	https://twitte

In [4]:

```
df.tail()
```

Out[4]:

	Datetime	Tweet Id	Text	Username	
49996	2023-01-24 06:57:56+00:00	1617778712082096128	#ChatGPT ist ein #Chatbot, der durch künstlich...	HorstKrieger	https://twitter.com/H
49997	2023-01-24 06:57:59+00:00	1617778726393249792	@r8r Ich hab mal die AI dazu befragt (ChatGPT)...	werpu	https://twitter.com/wer
49998	2023-01-24 06:58:00+00:00	1617778728481992705	5 minuti di #chatGPT e ho capito che apprende ...	marcopiccinini	https://twitter.com/r
49999	2023-01-24 06:58:01+00:00	1617778731678044162	Portland Shop Uses ChatGPT To Tell Family Stor...	EuniceNyandat	https://twitter.com/Eu
50000	2023-01-24 06:58:01+00:00	1617778733355790342	Ahora sueño con el día en que Amazon integre u...	AmericoSD_69	https://twitter.com/Ame

In [5]:

```
df.shape
```

Out[5]:

```
(50001, 20)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['Datetime', 'Tweet Id', 'Text', 'Username', 'Permalink', 'User',
       'Outlinks', 'CountLinks', 'ReplyCount', 'RetweetCount', 'LikeCoun
t',
       'QuoteCount', 'ConversationId', 'Language', 'Source', 'Media',
       'QuotedTweet', 'MentionedUsers', 'hashtag', 'hastag_counts'],
      dtype='object')
```

In [7]:

```
df.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
Datetime                0
Tweet Id                0
Text                    0
Username                0
Permalink               0
User                    0
Outlinks            30059
CountLinks          30059
ReplyCount              0
RetweetCount            0
LikeCount               0
QuoteCount              0
ConversationId          0
Language                0
Source                  0
Media               40499
QuotedTweet         46438
MentionedUsers      32832
hashtag                 0
hastag_counts           0
dtype: int64
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50001 entries, 0 to 50000
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Datetime        50001 non-null  object
 1   Tweet Id        50001 non-null  int64 
 2   Text            50001 non-null  object
 3   Username        50001 non-null  object
 4   Permalink       50001 non-null  object
 5   User            50001 non-null  object
 6   Outlinks        19942 non-null  object
 7   CountLinks      19942 non-null  object
 8   ReplyCount      50001 non-null  int64 
 9   RetweetCount    50001 non-null  int64 
 10  LikeCount       50001 non-null  int64 
 11  QuoteCount      50001 non-null  int64 
 12  ConversationId  50001 non-null  int64 
 13  Language        50001 non-null  object
 14  Source          50001 non-null  object
 15  Media           9502 non-null   object
 16  QuotedTweet     3563 non-null   object
 17  MentionedUsers  17169 non-null  object
 18  hashtag         50001 non-null  object
 19  hastag_counts   50001 non-null  int64 
dtypes: int64(7), object(13)
memory usage: 7.6+ MB
```

In [10]:

```
df.describe()
```

Out[10]:

	Tweet Id	ReplyCount	RetweetCount	LikeCount	QuoteCount	ConversationI
count	5.000100e+04	50001.000000	50001.000000	50001.000000	50001.000000	5.000100e+(
mean	1.617493e+18	0.929141	1.498510	9.696326	0.219536	1.617205e+1
std	1.725682e+14	23.251710	46.030058	313.524215	10.356329	1.005075e+1
min	1.617156e+18	0.000000	0.000000	0.000000	0.000000	6.493609e+1
25%	1.617354e+18	0.000000	0.000000	0.000000	0.000000	1.617302e+1
50%	1.617525e+18	0.000000	0.000000	0.000000	0.000000	1.617504e+1
75%	1.617625e+18	1.000000	0.000000	2.000000	0.000000	1.617607e+1
max	1.617779e+18	3098.000000	6815.000000	56073.000000	1947.000000	1.617779e+1

In [11]:

```
df.nunique()
```

Out[11]:

```
Datetime          41559
Tweet Id          50001
Text              49555
Username          38433
Permalink         50001
User              38433
Outlinks          13769
CountLinks        19485
ReplyCount          108
RetweetCount        138
LikeCount           366
QuoteCount           51
ConversationId    41430
Language             61
Source              843
Media              9401
QuotedTweet        2040
MentionedUsers    10704
hashtag            7312
hastag_counts        27
dtype: int64
```

In [12]:

```
def most_frequent_values(data):
    total = data.count()
    tt = pd.DataFrame(total)
    tt.columns = ['Total']
    items = []
    vals = []
    for col in data.columns:
        try:
            itm = data[col].value_counts().index[0]
            val = data[col].value_counts().values[0]
            items.append(itm)
            vals.append(val)
        except Exception as ex:
            print(ex)
            items.append(0)
            vals.append(0)
            continue
    tt['Most frequent item'] = items
    tt['Frequence'] = vals
    tt['Percent from total'] = np.round(vals / total * 100, 3)
    return(np.transpose(tt))
```

In [13]:

```
most_frequent_values(df)
```

Out[13]:

	Datetime	Tweet Id	Text	Username	
Total	50001	50001	50001	50001	
Most frequent item	2023-01-23 17:11:13+00:00	1617156270871699456	@chatgpt_issac AI	translation_ja	https://twitter.co
Frequence	8	1	164	60	
Percent from total	0.016	0.002	0.328	0.12	

In [14]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [15]:

```
df['Username'].unique()
```

Out[15]:

```
array(['mochico0123', 'Caput_LupinumSG', 'ciffi', ..., 'marcopiccinini',
       'EuniceNyandat', 'AmericoSD_69'], dtype=object)
```

In [16]:

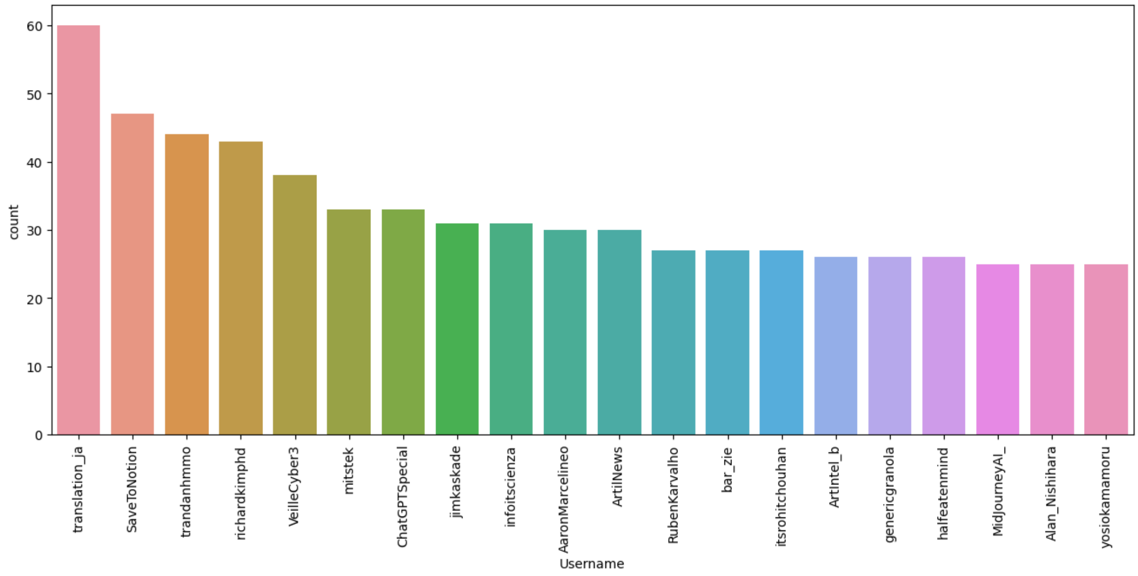
```
df['Username'].value_counts()
```

Out[16]:

```
translation_ja    60
SaveToNotion      47
trandanhmmo       44
richardkimphd     43
VeilleCyber3      38
                  ..
masayume_32        1
WRoughSketch       1
ayazfarooqui       1
Technology_GD      1
AmericoSD_69       1
Name: Username, Length: 38433, dtype: int64
```

In [17]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='Username', order=df["Username"].value_counts().index[:20])
plt.xticks(rotation=90)
plt.show()
```



In [18]:

```
df['User'].unique()
```

Out[18]:

```
array(['https://twitter.com/mochico0123',
       'https://twitter.com/Caput_LupinumSG', 'https://twitter.com/ciffi',
       ..., 'https://twitter.com/marcopiccinini',
       'https://twitter.com/EuniceNyandat',
       'https://twitter.com/AmericoSD_69'], dtype=object)
```

In [19]:

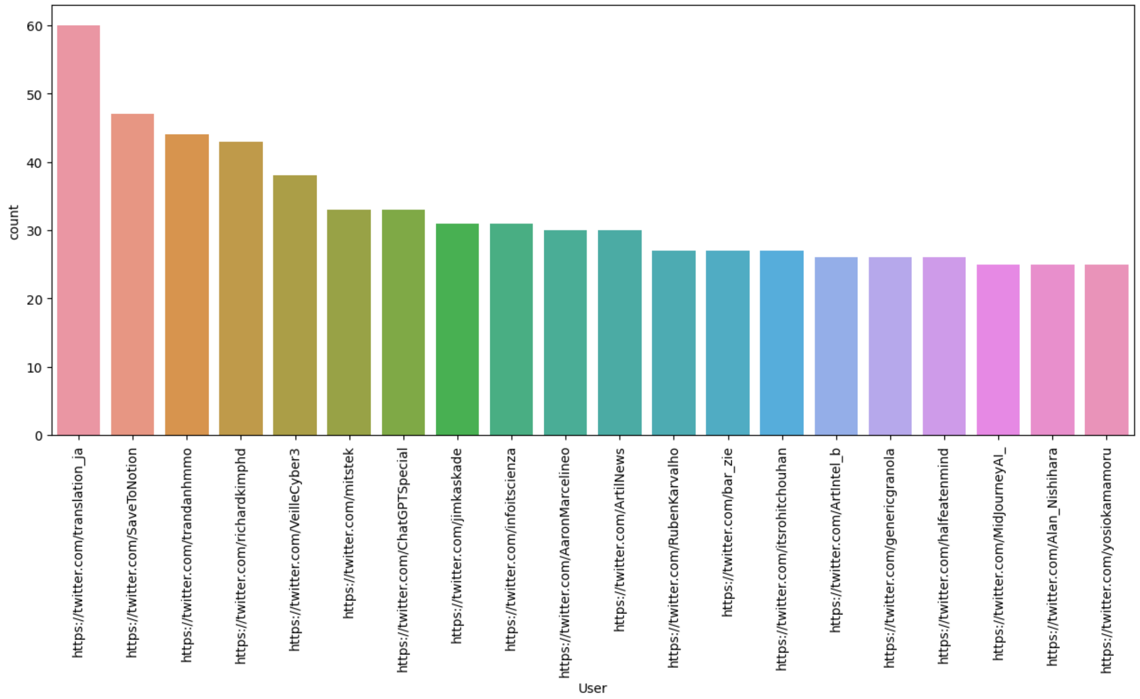
```
df['User'].value_counts()
```

Out[19]:

```
https://twitter.com/translation_ja (https://twitter.com/translation_ja)
60
https://twitter.com/SaveToNotion (https://twitter.com/SaveToNotion)      4
7
https://twitter.com/trandanhmmo (https://twitter.com/trandanhmmo)       44
https://twitter.com/richardkimphd (https://twitter.com/richardkimphd)
43
https://twitter.com/VeilleCyber3 (https://twitter.com/VeilleCyber3)      3
8
                                      ..
https://twitter.com/masayume_32 (https://twitter.com/masayume_32)        1
https://twitter.com/WRoughSketch (https://twitter.com/WRoughSketch)
1
https://twitter.com/ayazfarooqui (https://twitter.com/ayazfarooqui)
1
https://twitter.com/Technology_GD (https://twitter.com/Technology_GD)
1
https://twitter.com/AmericoSD_69 (https://twitter.com/AmericoSD_69)
1
Name: User, Length: 38433, dtype: int64
```

In [20]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='User', order=df["User"].value_counts().index[:20])
plt.xticks(rotation=90)
plt.show()
```



In [21]:

```
df['ReplyCount'].unique()
```

Out[21]:

```
array([   1,    0,   23,    4,   37,    2,    3,  209,  149,  126,   17,
         22,    5,   54,   34,   21,   13, 3098,   69, 1421,   10,    8,
          6,    9,    7,   43,   15,   11,   12,  111,   29,  164,  286,
         28,   20,  496,   14,   31,   40,   47,  374,   33,  338,   26,
        100,   39,   18,   59,  154,   16,   55,  114,  476,   44,   24,
        119,   99,   42,   36,   27,   68,   92,  159,   19,   88,   35,
        446,   52,   45,  147,  106,  166,   32,   89,  165,   50,  130,
         51,   95, 1455, 3044,   96,  490,  194,   48,   74,  248,   72,
         80,   57,   25,   93,   38, 1110,  161,  183,   41,  135,   79,
        103,   71,   49,   30,   63,  176,   67,  777,  331])
```

In [22]:

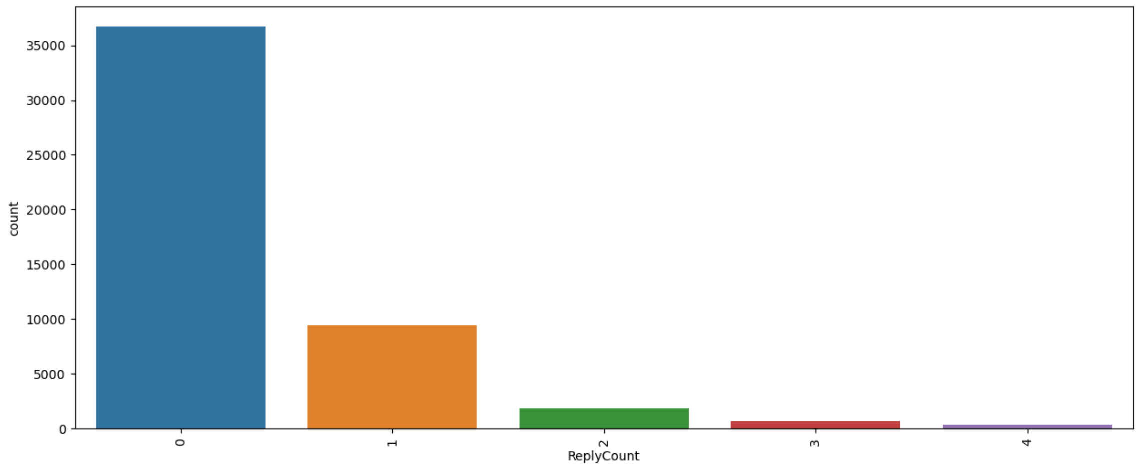
```
df['ReplyCount'].value_counts()
```

Out[22]:

```
0      36736
1       9430
2       1801
3        653
4        333
       ...  
45         1
147        1
106        1
166        1
331        1
Name: ReplyCount, Length: 108, dtype: int64
```

In [23]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='ReplyCount', order=df["ReplyCount"].value_counts().index[:5])
plt.xticks(rotation=90)
plt.show()
```



In [24]:

```
df['RetweetCount'].unique()
```

Out[24]:

```
array([   0,    1,    5,    6,  542,   40,   49,   18,    3,    2,   16,
          4,  112,   39,   36, 1094,   12, 6815,   10,   20,    7,   58,
         23,   11,   15,   45,   37,    8,   31,   19,  160,   27,    9,
         38,   34,   14,  125,   17,   74,  461,   68,   26,   13, 2627,
         47,   53,   88,   21,   25,  221,  334,   50,   30,   22,  252,
         33,  713,  597,   76,   43,   28,  227, 1732,   66,   98,  136,
         92,  114,  730,   64,   24,  202, 1874,   67,   29,  186,  118,
         97,   52,  108, 2463,   59,   32,   93,   35,   56,  564, 3987,
        170,   55,  257, 1307,   95,   62,   44,   57,   89,   54,   83,
        222,  337,  236,  376,  121, 1533,  458,   78,   61,   51, 2203,
         70,  428,   42,  140,   46, 1534,  164,  162,  213,  139,  289,
       1113,   48,  418,  206,  107,  148,   99,  516,   41,  452,  630,
        176,   79,   69,  679, 2082,  248])
```

In [25]:

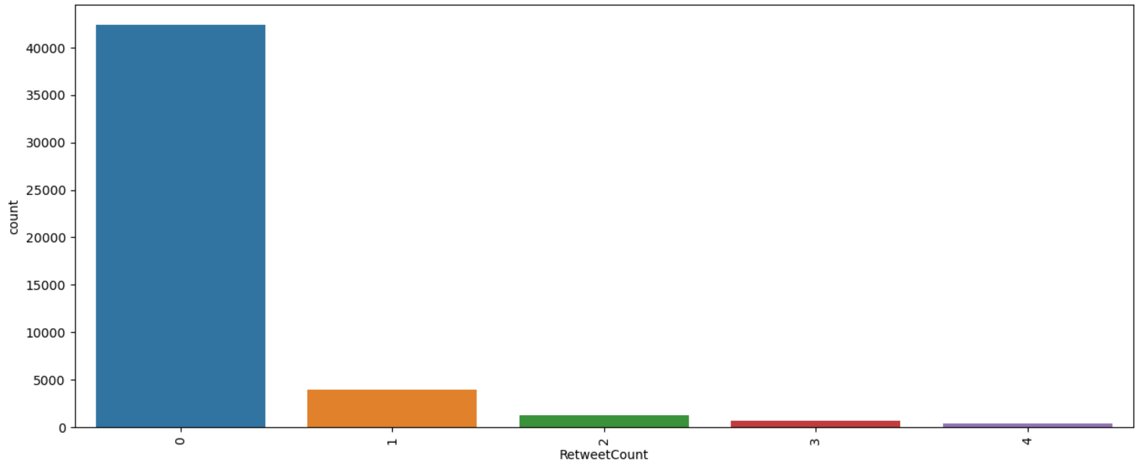
```
df['RetweetCount'].value_counts()
```

Out[25]:

```
0      42416
1       3925
2       1204
3        620
4        365
       ...  
221        1
222        1
337        1
236        1
248        1
Name: RetweetCount, Length: 138, dtype: int64
```

In [26]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='RetweetCount', order=df["RetweetCount"].value_counts().index[:
plt.xticks(rotation=90)
plt.show()
```



In [27]:

```
df['LikeCount'].unique()
```

Out[27]:

```
array([    5,     4,     2,     1,     0,     3,    66,    20,  9125,
           7,   348,   607,    11,     9,   404,     8,     6,    68,
         329,    10,    17,    15,  1905,   211,    16,    22,    14,
         381,    13,  5682,    42,   190,    57, 56073,    26,    38,
          30,    97,    24,    19,    12,    50,    18,    29,    51,
          33,    41,    55,    44,   663,    75,    54,    39,    21,
          37,   252,    40,    47,   390,    87,   177,    23,    84,
          28,    31,   131,   251,    86,  1055,   171,    65,   144,
          58,   111,    71,    74,   127,    36,    32,   234,   654,
         222,   711,  3952,    43,    46,   273,   168,   349,   239,
         118,   112,    64,    27,   228,    25,    56,   186,    93,
       12557,   733,    98,    61,    35,   100,   147,   322,    49,
         383,   189,    67,    45,    34,   135,   132,   113,    76,
         759,   165,   164,    48,    72,   220,   107,  1608,  2250,
         302,    73,   149,   289,    82,   429,   108,   834,   106,
         114,    62,   176,   123,   858,    90,   128,   185,    91,
        4413,  9677,   368,    59,   415,   104,   334,   103,    92,
          94,   297,   121,    88,   208,   236,   153,    70,  1466,
          69,    60,   102,  9946,    53,   262,    85,  1182,   279,
         187,   162,    78,    77,   443,   284,   116,   673,   589,
         701,   110,   274,    89,   254,   355,   109,   122,    52,
        1732,  6979,   292,   140,   326,   226,   130,   247,   859,
         372, 16856,   835,   115,   138,    81,   708,   642,   160,
         338,   174,   503,   396,    63,   656,   481,   169,   126,
         154,   117,   196,  1239,   419,   311, 10153,   191,   137,
         161,   276,   492,   242,   202,   347,    83,   214,  2297,
       17150,   158,   505,   324,  1238,  1517,  5513,   151,   200,
         178,   789,   767,   125,    96,   167,   163,   248,   134,
         188,   263,   235,    79,   458,    99,   430,   343,   150,
         166,   210,  1213,  1707,   277,   124,   215,   245,   152,
         218,  2353,   143,  3529,   170,   339,   244,   342,   954,
          80,   509,   206,   270,   173, 11520,   373,   394,   253,
         139,   209,   728,  1189,   386,   133,  1082,   497,   360,
         267,   105,  5911,   364,  1077,   384,   194,   213,  1935,
         643,   316,   421,   225,   385,   588,   233,   148,   318,
         354,   195,  1850,   616,   303,  3732,  2712,  1199,   641,
        4643,   451,   203,   644,  2770,   320,  2223,   181,  3915,
         327,   129,   145,   249,   436,   480,   119,   455,   101,
         752,   180,   309,  1118,   428,   238,   490,   281,   156,
        4492,   280,   219,    95,   374,   136,   282,   216, 12158,
         217,  2184,   506,   142,   398,   424])
```

In [28]:

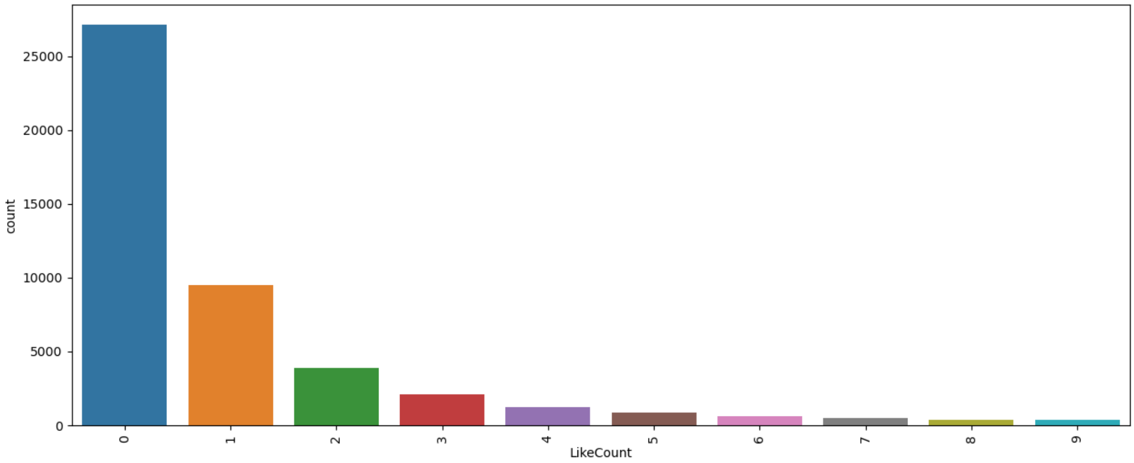
```
df['LikeCount'].value_counts()
```

Out[28]:

```
0      27141
1       9490
2       3873
3       2086
4       1241
       ...  
242        1
202        1
347        1
404        1
424        1
Name: LikeCount, Length: 366, dtype: int64
```

In [29]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='LikeCount', order=df["LikeCount"].value_counts().index[:10])
plt.xticks(rotation=90)
plt.show()
```



In [30]:

```
df['QuoteCount'].unique()
```

Out[30]:

```
array([   0,    1,    2,   15,    6,    8,    3,   14,   45, 1947,    9,
          4,    5,   23,  110,  726,   11,   27,    7,   87,   10,  126,
         16,   22,  205,  298,   13,   80,   38,   81,   29,   24,   12,
        495,  216,   25,   37,   57,   48,   19,  413,   18,  374,   20,
         60,   46,   21,   42,   17,   55,  456])
```

In [31]:

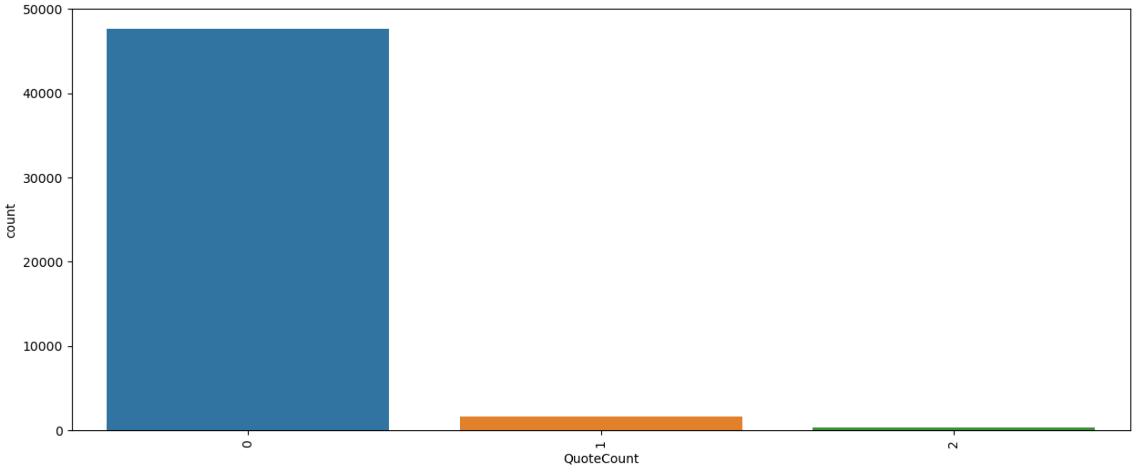
```
df['QuoteCount'].value_counts()
```

Out[31]:

```
0       47665
1        1575
2         337
3         147
4          70
5          34
7          28
6          23
9          16
8          13
10         13
11         10
12          8
27          6
14          5
25          4
15          4
24          3
16          3
37          2
19          2
60          2
29          2
38          2
57          1
48          1
21          1
413         1
46          1
374         1
20          1
55          1
17          1
42          1
18          1
298         1
216         1
495         1
81          1
80          1
13          1
205         1
22          1
126         1
87          1
726         1
110         1
23          1
1947        1
45          1
456         1
Name: QuoteCount, dtype: int64
```

In [32]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='QuoteCount', order=df["QuoteCount"].value_counts().index[:3])
plt.xticks(rotation=90)
plt.show()
```



In [33]:

```
df['Language'].unique()
```

Out[33]:

```
array(['ja', 'en', 'de', 'tr', 'pl', 'fr', 'es', 'pt', 'lo', 'no', 'ca',
       'zh', 'qme', 'th', 'ne', 'ko', 'und', 'nl', 'fa', 'it', 'da', 'fi',
       'eu', 'hi', 'ar', 'sv', 'in', 'ru', 'qht', 'tl', 'hu', 'cs', 'uk',
       'iw', 'et', 'cy', 'bg', 'ht', 'el', 'vi', 'sl', 'kn', 'ro', 'lt',
       'ur', 'zxx', 'ml', 'mr', 'lv', 'gu', 'qam', 'is', 'ta', 'te', 'pa',
       'sd', 'am', 'sr', 'hy', 'or', 'bn'], dtype=object)
```

In [34]:

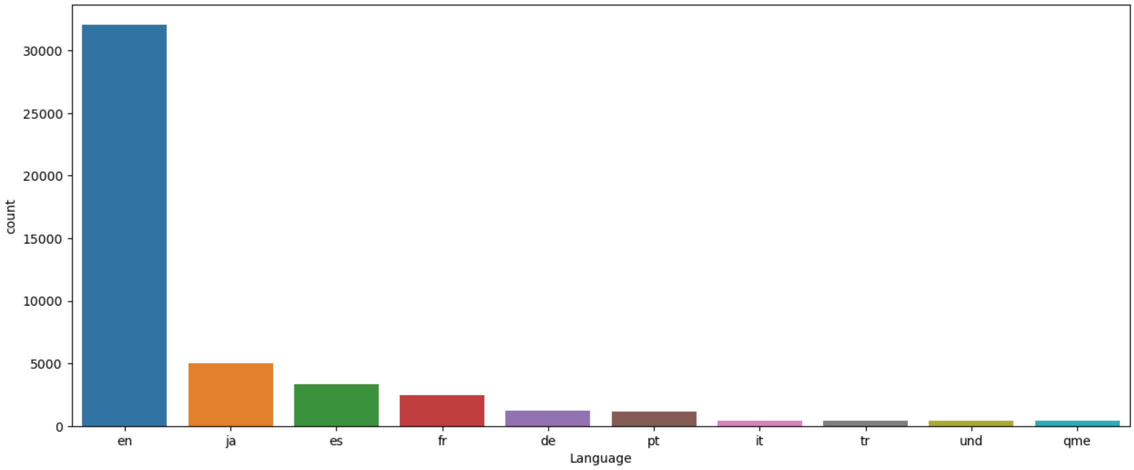
```
df['Language'].value_counts()
```

Out[34]:

```
en    32076
ja     5046
es     3315
fr     2492
de     1207
      ...  
sd        1
am        1
hy        1
or        1
bn        1
Name: Language, Length: 61, dtype: int64
```

In [35]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='Language', order=df["Language"].value_counts().index[:10])
plt.show()
```



In [36]:

```
df['Source'].unique()
```

Out[36]:

```
array(['<a href="http://twitter.com/download/iphone" rel="nofollow">Twi
tter for iPhone</a>',
       '<a href="http://twitter.com/#!/download/ipad" rel="nofollow">Tw
itter for iPad</a>',
       '<a href="http://twitter.com/download/android" rel="nofollow">Tw
itter for Android</a>',
       '<a href="https://about.twitter.com/products/tweetdeck" rel="nof
ollow">TweetDeck</a>',
       '<a href="https://mobile.twitter.com" rel="nofollow">Twitter Web
App</a>',
       '<a href="https://nowtice.net/" rel="nofollow">nowtice_news</a
>',
       '<a href="https://smarterqueue.com" rel="nofollow">SmarterQueue
</a>',
       '<a href="https://github.com/M157q/py-feedr" rel="nofollow">py-f
eedr-M157q</a>',
       '<a href="http://www.linkedin.com/" rel="nofollow">LinkedIn</a
>',
```

In [37]:

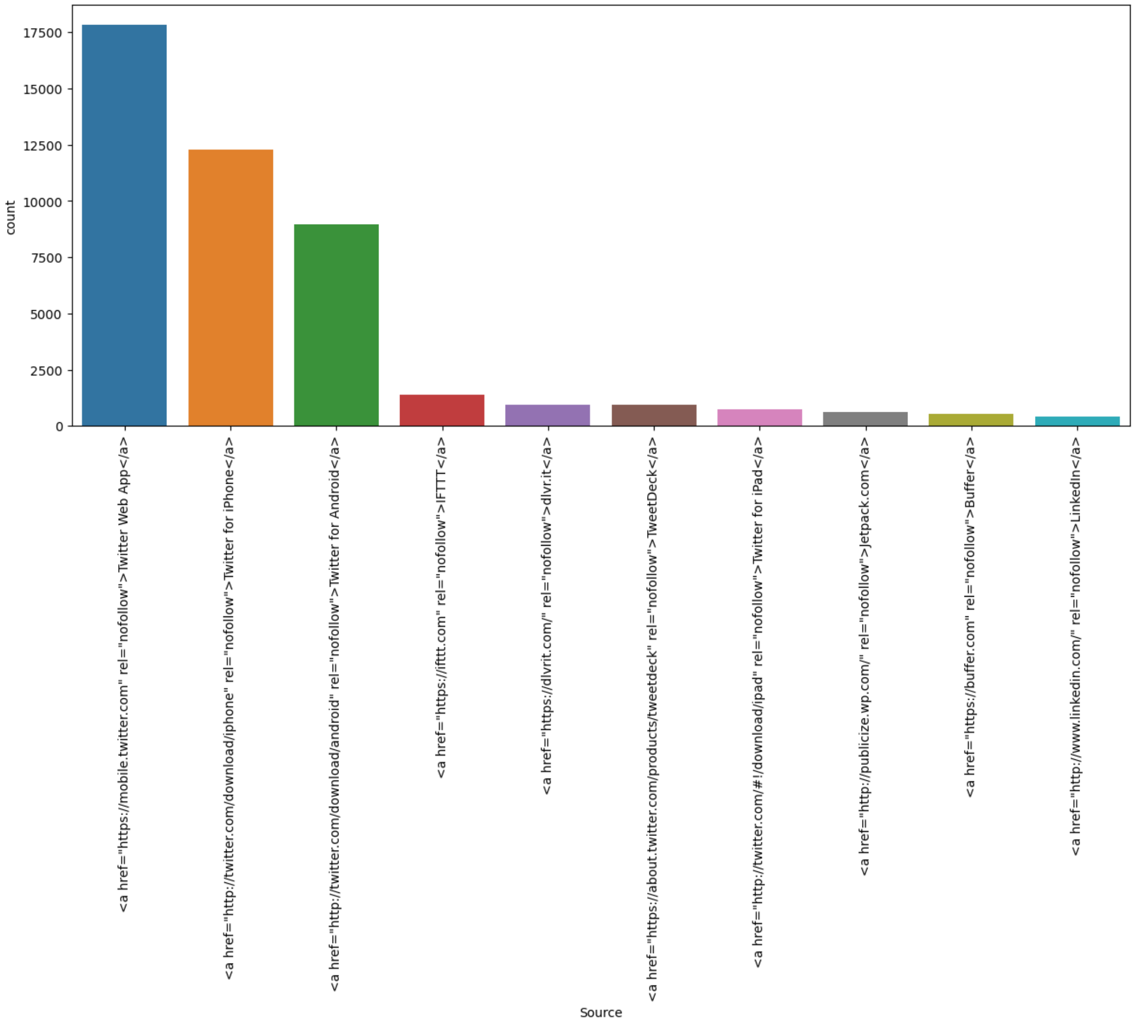
```
df['Source'].value_counts()
```

Out[37]:

```
<a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a>
17814
<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iP
hone</a>      12281
<a href="http://twitter.com/download/android" rel="nofollow">Twitter for A
ndroid</a>     8972
<a href="https://ifttt.com" rel="nofollow">IFTTT</a>
1383
<a href="https://dlvrit.com/" rel="nofollow">dlvr.it</a>
959

...
<a href="https://www.oliberal.com/" rel="nofollow">bot_twitter_oliberal</a
>                  1
<a href="http://www.google.com" rel="nofollow">hogeeee</a>
1
<a href="https://google.com" rel="nofollow">bdtw</a>
1
<a href="https://euwatch.live" rel="nofollow">EUwatch</a>
1
<a href="http://twmode.sf.net/" rel="nofollow">twmode</a>
1
Name: Source, Length: 843, dtype: int64
```

In [38]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='Source', order=df["Source"].value_counts().index[:10])
plt.xticks(rotation = 90)
plt.show()
```



In [39]:

```
df['hastag_counts'].unique()
```

Out[39]:

```
array([ 0,  1,  2,  4, 11,  5,  3, 10, 15,  8, 12,  9,  6,  7, 13, 14, 16,
       23, 20, 18, 21, 24, 22, 25, 17, 28, 19])
```

In [40]:

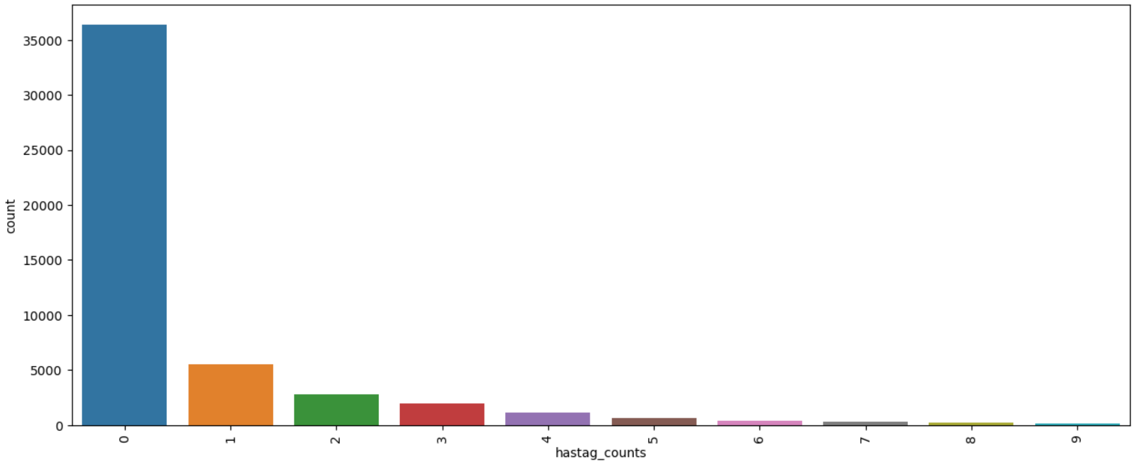
```
df['hastag_counts'].value_counts()
```

Out[40]:

```
0     36414
1      5516
2      2772
3      1944
4      1150
5       651
6       396
7       295
8       229
9       124
10       98
13       76
11       69
12       51
15       44
16       43
14       41
18       25
17       14
23        9
19        9
25        8
21        7
20        7
24        6
22        2
28        1
Name: hastag_counts, dtype: int64
```

In [41]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='hastag_counts', order=df["hastag_counts"].value_counts().index
plt.xticks(rotation = 90)
plt.show()
```



In [42]:

```
from wordcloud import WordCloud, STOPWORDS
from sklearn.decomposition import LatentDirichletAllocation
from collections import Counter
from nltk.sentiment import SentimentIntensityAnalyzer
from textblob import TextBlob
```

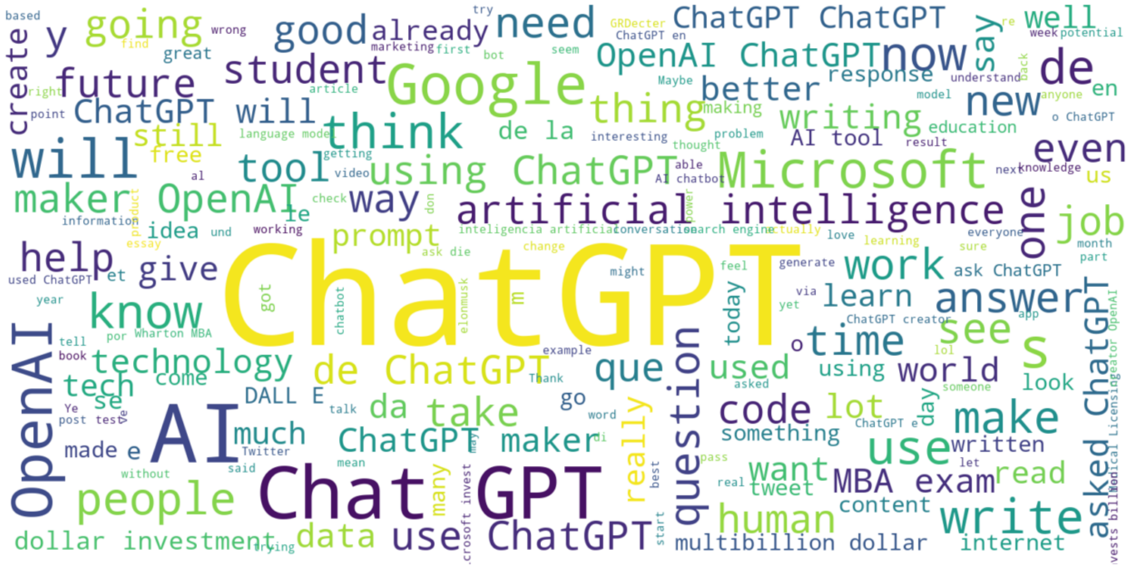
```
/opt/conda/lib/python3.7/site-packages/nltk/twitter/__init__.py:20: UserWa
rning: The twython library has not been installed. Some functionality from
the twitter package will not be available.
  warnings.warn("The twython library has not been installed. "
```

In [43]:

```
stopwords = set(STOPWORDS)

def show_wordcloud(data, mask=None, title=""):
    text = " ".join(t for t in data.dropna())
    stopwords = set(STOPWORDS)
    stopwords.update(["t", "co", "https", "amp", "U", "Comment", "text", "attr", "object
    wordcloud = WordCloud(stopwords=stopwords, scale=4, max_font_size=50, max_words=500,
    fig = plt.figure(1, figsize=(16,16))
    plt.axis('off')
    fig.suptitle(title, fontsize=20)
    fig.subplots_adjust(top=2.3)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.show()
```

In [44]:

```
show_wordcloud(df['Text'], title = 'Prevalent words in tweets')
```



Prevalent words in tweets

In [45]:

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

In [46]:

```
df['year'] = df['Datetime'].dt.year
df['month'] = df['Datetime'].dt.month
df['day'] = df['Datetime'].dt.day
df['dayofweek'] = df['Datetime'].dt.dayofweek
df['hour'] = df['Datetime'].dt.hour
df['minute'] = df['Datetime'].dt.minute
df['dayofyear'] = df['Datetime'].dt.dayofyear
df['date_only'] = df['Datetime'].dt.date
```

In [47]:

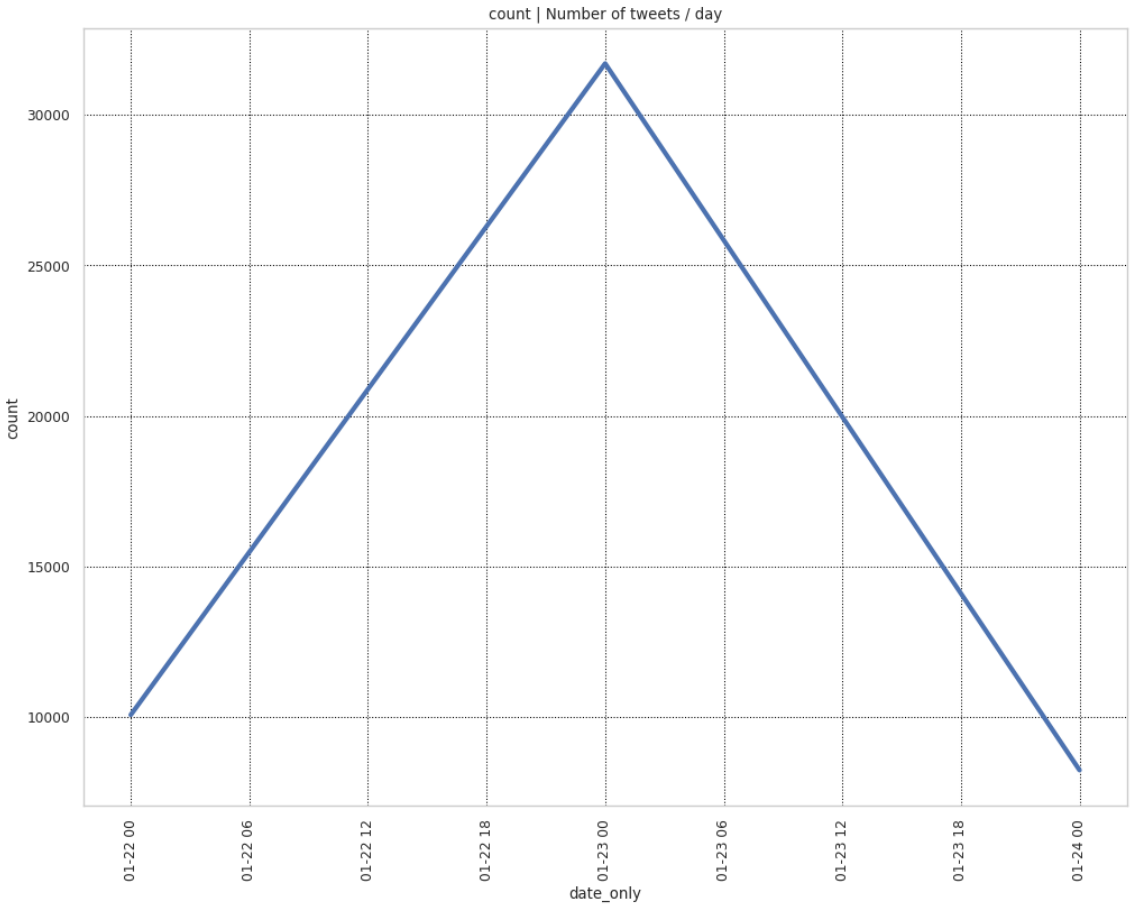
```
tweets_agg_df = df.groupby(["date_only"])["Text"].count().reset_index()
tweets_agg_df.columns = ["date_only", "count"]
```

In [48]:

```
def plot_time_variation(df, x='date_only', y='count', hue=None, size=1, title="", is_log
    sns.set(style="whitegrid")
    paper_rc = {'lines.linewidth': 3, 'lines.markersize': 20}
    sns.set_context("paper", rc = paper_rc)
    f, ax = plt.subplots(1,1, figsize=(4*size,3*size))
    g = sns.lineplot(x=x, y=y, hue=hue, data=df)
    plt.xticks(rotation=90)
    if hue:
        plt.title(f'{y} grouped by {hue} | {title}')
    else:
        plt.title(f'{y} | {title}')
    if(is_log):
        ax.set(yscale="log")
    ax.grid(color='black', linestyle='dotted', linewidth=0.75)
    plt.show()
```

In [49]:

```
plot_time_variation(tweets_agg_df, x='date_only', title="Number of tweets / day",size=3)
```



In [50]:

```
df['dayofweek'].unique()
```

Out[50]:

```
array([6, 0, 1])
```

In [51]:

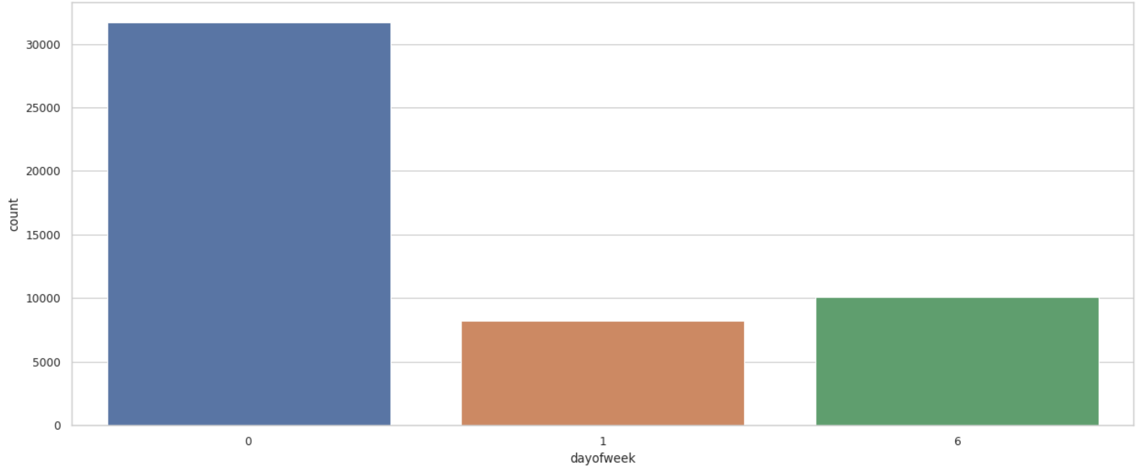
```
df['dayofweek'].value_counts()
```

Out[51]:

```
0    31700
6    10068
1     8233
Name: dayofweek, dtype: int64
```

In [52]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='dayofweek')
plt.show()
```



In [53]:

```
df['dayofyear'].unique()
```

Out[53]:

```
array([22, 23, 24])
```

In [54]:

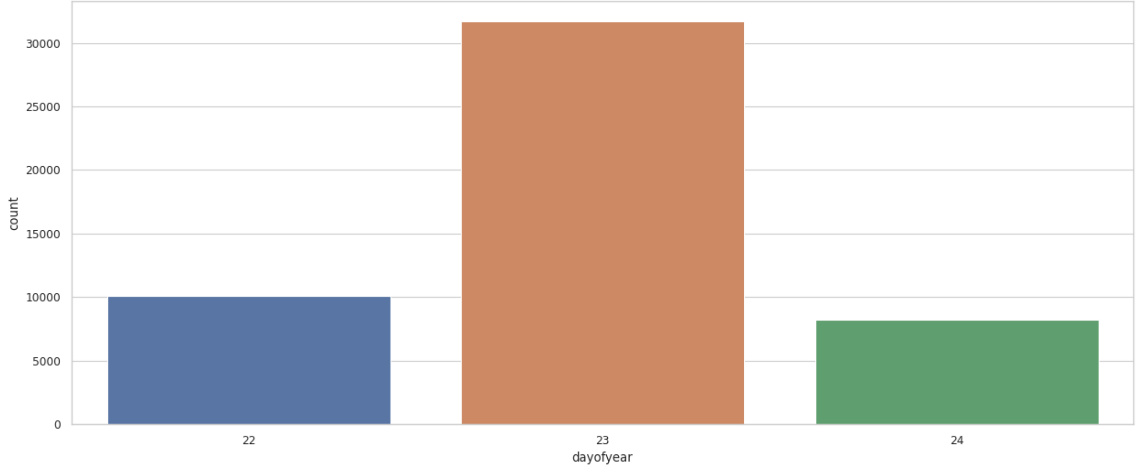
```
df['dayofyear'].value_counts()
```

Out[54]:

```
23    31700
22    10068
24     8233
Name: dayofyear, dtype: int64
```

In [55]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='dayofyear')
plt.show()
```



In [56]:

```
df['date_only'].unique()
```

Out[56]:

```
array([datetime.date(2023, 1, 22), datetime.date(2023, 1, 23),
       datetime.date(2023, 1, 24)], dtype=object)
```

In [57]:

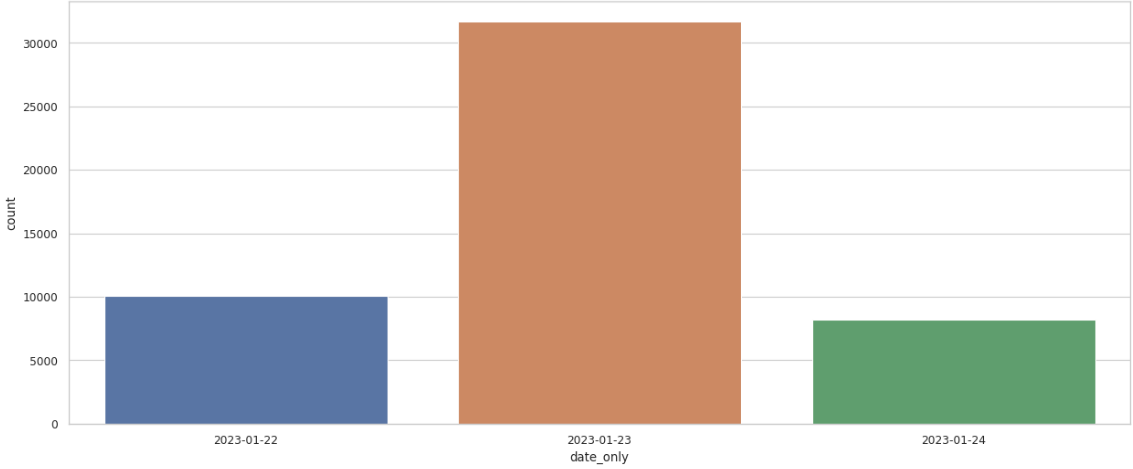
```
df['date_only'].value_counts()
```

Out[57]:

```
2023-01-23    31700
2023-01-22    10068
2023-01-24     8233
Name: date_only, dtype: int64
```

In [58]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='date_only')
plt.show()
```



In [59]:

```
df['hour'].unique()
```

Out[59]:

```
array([13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,  0,  1,  2,  3,  4,  5,
        6,  7,  8,  9, 10, 11, 12])
```

In [60]:

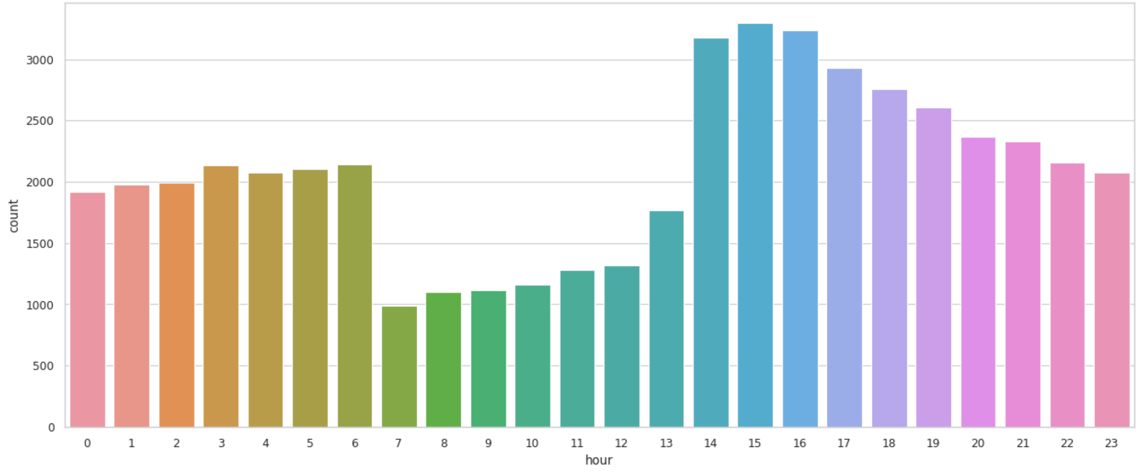
```
df['hour'].value_counts()
```

Out[60]:

```
15    3297
16    3237
14    3178
17    2927
18    2756
19    2604
20    2370
21    2330
22    2158
6     2140
3     2134
5     2105
23    2078
4     2077
2     1994
1     1978
0     1914
13    1765
12    1319
11    1277
10    1161
9     1113
8     1103
7      986
Name: hour, dtype: int64
```

In [61]:

```
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='hour')
plt.show()
```



In [62]:

```
df['minute'].unique()
```

Out[62]:

```
array([44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,  0,
        1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43])
```

In [63]:

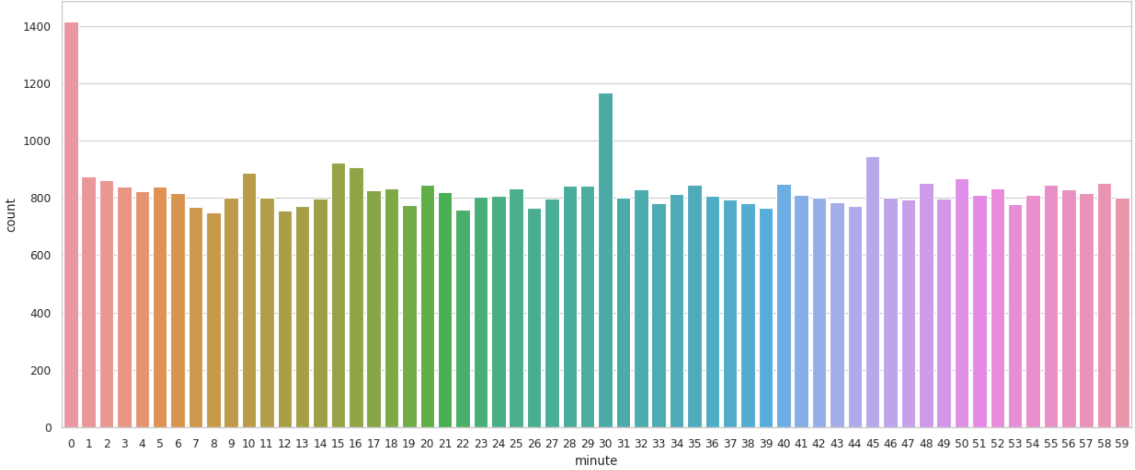
```
df['minute'].value_counts()
```

```
Out[63]:
```

```
0     1415
30    1168
45     945
15     923
16     907
10     886
1      873
50     868
2      863
52     830
56     830
17     826
4      822
21     821
6      816
57     815
34     813
41     811
51     811
54     810
36     807
24     806
23     804
59     802
42     801
11     800
9      800
31     799
46     799
27     798
49     797
14     797
47     795
37     794
43     783
33     780
38     780
53     778
19     774
44     772
13     770
7      769
39     766
26     765
22     759
12     755
```

In [64]:

```python
plt.figure(figsize=(15,6))
sns.countplot(data=df, x='minute')
plt.show()
```



In [65]:

```python
sia = SentimentIntensityAnalyzer()
def find_sentiment(post):
    try:
        if sia.polarity_scores(post)["compound"] > 0:
            return "Positive"
        elif sia.polarity_scores(post)["compound"] < 0:
            return "Negative"
        else:
            return "Neutral"
    except:
        return "Neutral"
```

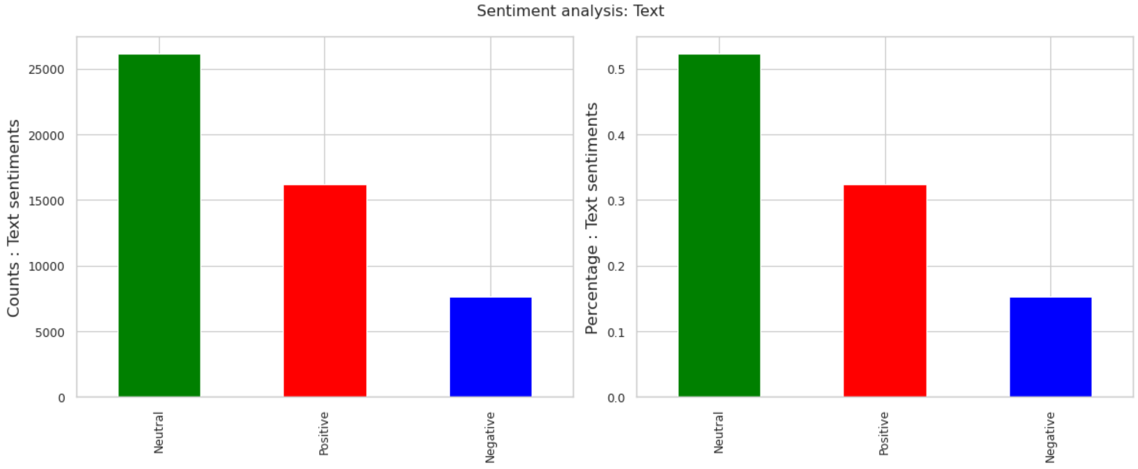
In [66]:

```python
def plot_sentiment(df, feature, title):
    counts = df[feature].value_counts()
    percent = counts/sum(counts)

    fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 5))

    colors = ["green", "red", "blue"]
    counts.plot(kind='bar', ax=ax1, color=colors)
    percent.plot(kind='bar', ax=ax2, color=colors)
    ax1.set_ylabel(f'Counts : {title} sentiments', size=12)
    ax2.set_ylabel(f'Percentage : {title} sentiments', size=12)
    plt.suptitle(f"Sentiment analysis: {title}")
    plt.tight_layout()
    plt.show()
```

```
8       749
Name: minute, dtype: int64
```

In [67]:

```
df['text_sentiment'] = df['Text'].apply(lambda x: find_sentiment(x))
plot_sentiment(df, 'text_sentiment', 'Text')
```

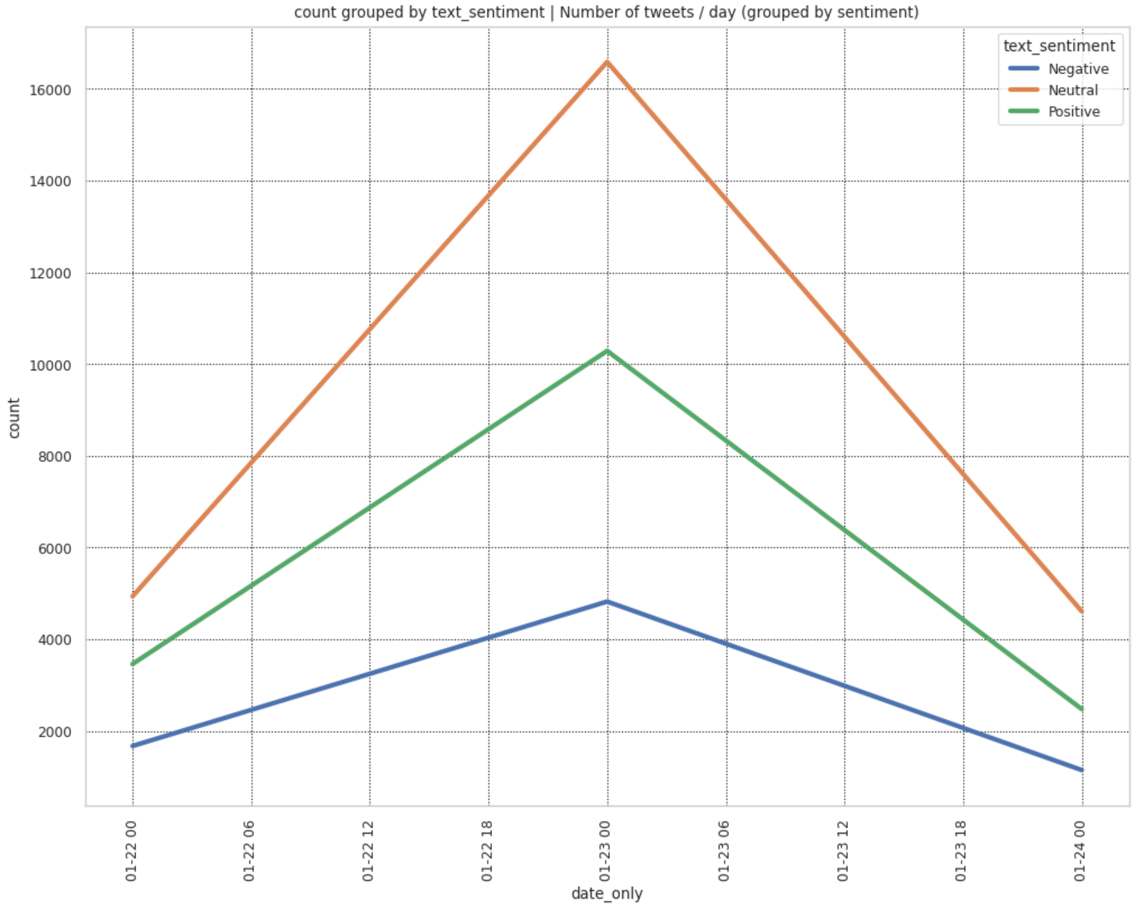


In [68]:

```
tweets_agg_df = df.groupby(["date_only", "text_sentiment"])["Text"].count().reset_index(
tweets_agg_df.columns = ["date_only", "text_sentiment", "count"]
```

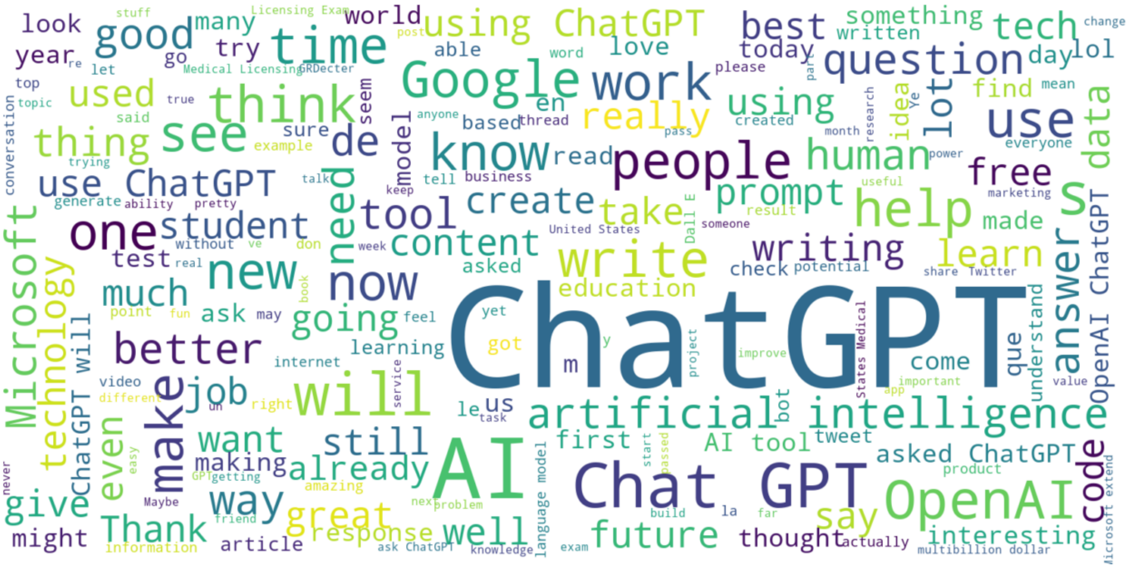
In [69]:

```
plot_time_variation(tweets_agg_df, x='date_only', hue="text_sentiment", title="Number of
```



In [70]:

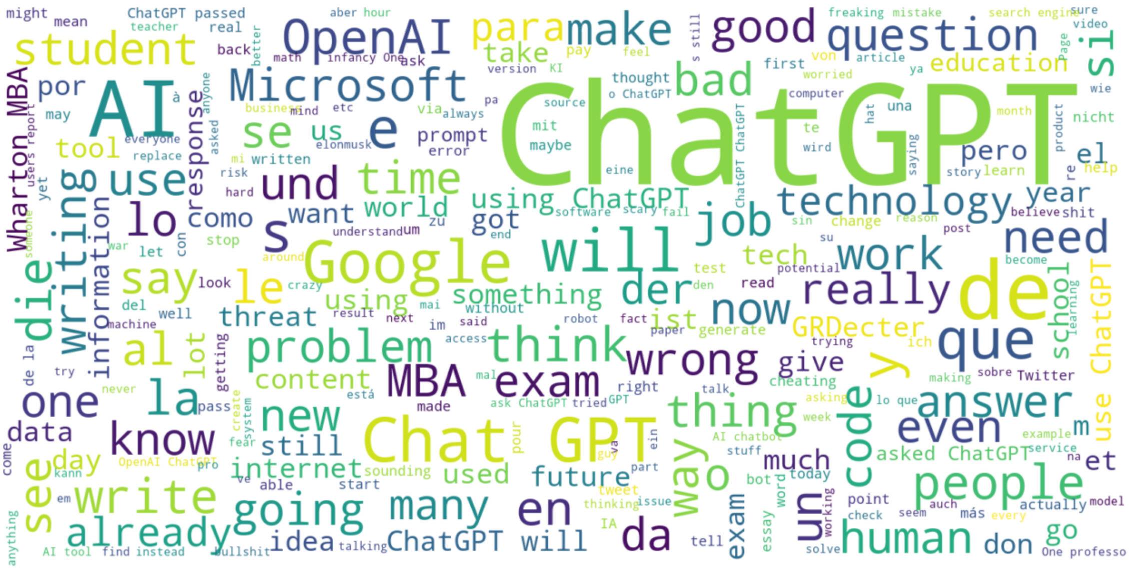
```
show_wordcloud(df.loc[df['text_sentiment']=="Positive", 'Text'], title = 'Prevalent word
```



Prevalent words in Text with Positive sentiment

In [71]:

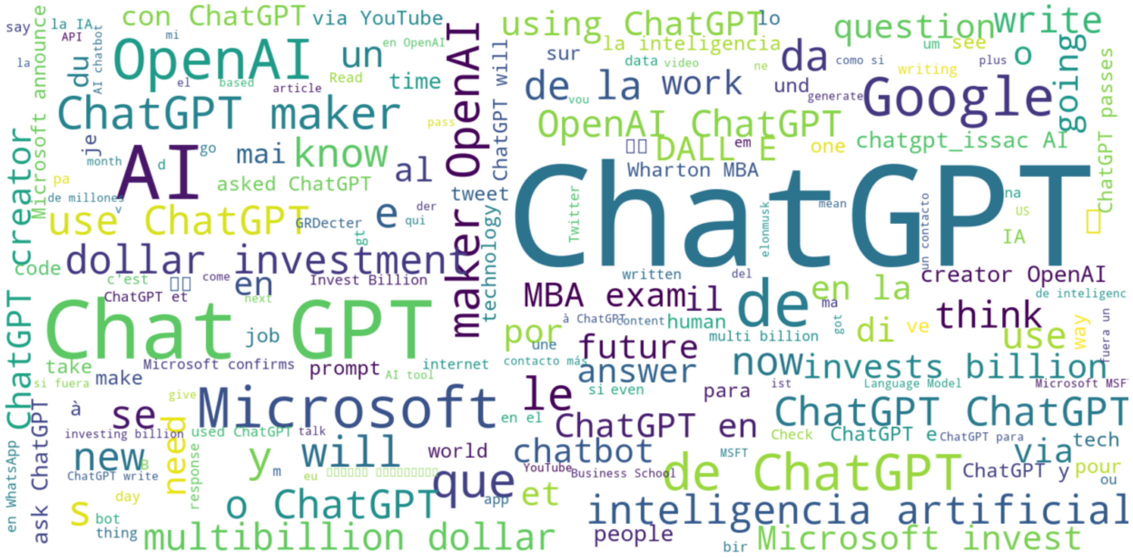
```
show_wordcloud(df.loc[df['text_sentiment']=="Negative", 'Text'], title = 'Prevalent word
```



Prevalent words in Text with Negative sentiment

In [72]:

```
show_wordcloud(df.loc[df['text_sentiment']=="Neutral", 'Text'], title = 'Prevalent words
```



Prevalent words in Text with Neutral sentiment