In [1]:

```
import glob
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
from os import path
import collections
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns
from scipy.stats import norm
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
import string
import re
import nltk
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
import pandas as pd
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from datetime import *
```

In [2]:

```
#cd excel
```

In [3]:

```
path = os.getcwd()
path
```

Out[3]:

```
'/Users/vishal/Desktop/DSPM Final Project'
```

In [4]:

```
#Inputting Data from all the excel files into one dataframe
all_data = pd.DataFrame()

for f in glob.glob(path+"/*.xlsx"):
    df = pd.read_excel(f)
    all_data = all_data.append(df,ignore_index=True)
```

In [5]:

```
#all_data.info()
```

Pre-processing the Data

In [6]:

```
all_data_1 = all_data.drop_duplicates()
##512711 - 511268 =1443
```

In [7]:

```
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 500)
all_data.head(3)
```

Out[7]:

	Post ID	Sound Bite Text	Ratings and Scores	Title	Source Type	Post Type	Media Type	
0	11456927580853042662	With the iPhone 8 and iPhone 8 Plus already in...	NaN	Apple iPhone X pre-order begins in India on Oc...	Blogs	Original	Link	http://tweeterb
1	3723388671848181876	The following list showcases our pick of the b...	NaN	Best free iPhone apps 2017	Blogs	Original	No Media	http://tweeterb
2	11827073999221105793	Apple iPhone 8 Plus 256GB Space Gray Factory ...	NaN	Apple iPhone 8 Plus 256GB Space Gray Factory U...	Blogs	Original	No Media	https://hellanor

In [8]:

```
all_data_2 = all_data_1.dropna(axis=0, subset=['Post ID' , 'Sound Bite Text' , '
Published Date (GMT-04:00) New York' , 'No. of Followers/Daily Unique Visitors']
)
#all_data_2.info()
#492518
```

In [9]:

```
all_data_2.isnull().sum()
```

Out[9]:

```
Post ID                                        0
Sound Bite Text                                0
Ratings and Scores                        492518
Title                                     189145
Source Type                                    0
Post Type                                      0
Media Type                                     0
URL                                            0
Domain                                         0
Published Date (GMT-04:00) New York            0
Author Gender                                  0
Author URL                                129371
Author Name                               113232
Author Handle                             286446
Author ID                                 286801
Author Location - Country 1               315659
Author Location - State/Province 1        453626
Author Location - City 1                  459038
Author Location - Country 2               492110
Author Location - State/Province 2        492167
Author Location - City 2                  492204
Author Location - Other                   492469
No. of Followers/Daily Unique Visitors         0
Professions                               485268
Interests                                 480333
Positive Objects                          385820
Negative Objects                          442919
Richness                                       0
Tags                                      492518
Quoted Post                               492209
Quoted Author Name                        492209
Quoted Author Handle                      492209
Total Engagements                         440044
Post Comments                             461899
Post Likes                                442265
Post Shares                               491350
Post Views                                492518
Post Dislikes                             492518
Product Name                              481120
Product Hierarchy                         492518
Rating                                    490328
dtype: int64
```

In [10]:

```
all_data_3= all_data_2[all_data_2['Post Type'] == 'Original']
#all_data_3.info()
## 492518 to 409755
```

In [11]:

```
all_data_4= all_data_3[all_data_3['Sound Bite Text'] != 'Post deleted by the aut
hor.']
#all_data_4.info()
#409755 to 409755
```

In [12]:

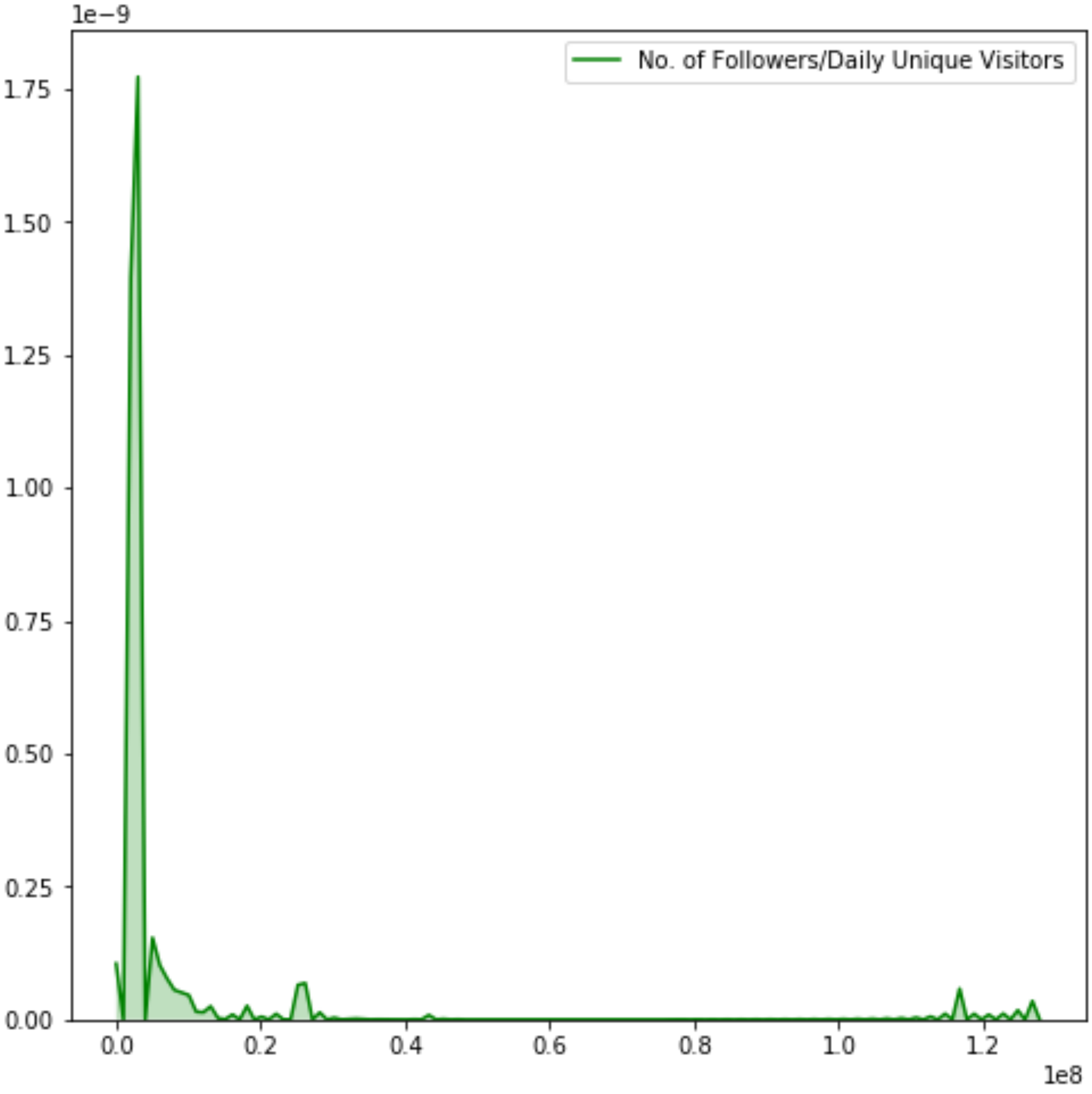
```
all_data_4['No. of Followers/Daily Unique Visitors'].describe()
```

Out[12]:

```
count    4.097550e+05
mean     4.089722e+04
std      9.111978e+05
min      0.000000e+00
25%      0.000000e+00
50%      0.000000e+00
75%      0.000000e+00
max      1.277749e+08
Name: No. of Followers/Daily Unique Visitors, dtype: float64
```

In [13]:

```
plt.figure(figsize = (8,8 ))
sns.kdeplot(all_data_4['No. of Followers/Daily Unique Visitors'], color="green",
shade=True)
plt.show()
```



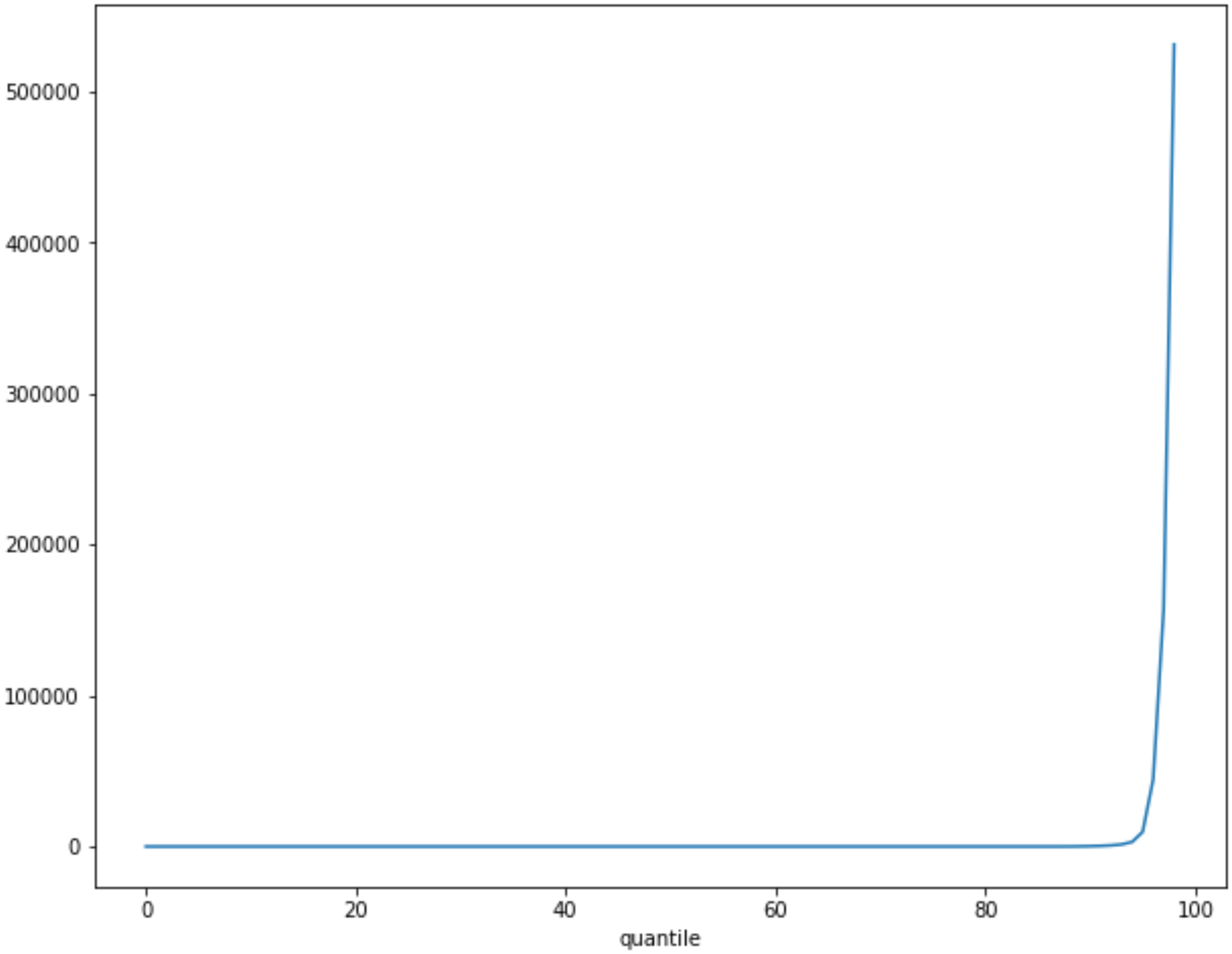
In [14]:

```
values = []
for a in range(0,99) :
    #print(a,"th quantile value :" , all_data_4['No. of Followers/Daily Unique Visitors'].quantile(a/100))
    values.append(all_data_4['No. of Followers/Daily Unique Visitors'].quantile(a/100))
print(values)
```

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0
.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 26.0, 96.0, 226.0, 420.0, 760.0, 139
2.0, 3016.0, 9688.0, 44330.0, 157039.0, 531195.0]
```

In [15]:

```
plt.figure(figsize=(10,8))
plt.plot(values)
plt.xlabel('quantile')
plt.show()
```



In [16]:

```
qcap = all_data_4['No. of Followers/Daily Unique Visitors'].quantile(0.95)
qcap
```

Out[16]:

```
9688.0
```

In [17]:

```
#all_data_4.info()
```

In [18]:

```
all_data_5 = all_data_4.loc[all_data_4['No. of Followers/Daily Unique Visitors']
< qcap]
#all_data_5.info()
#409755 to 389185
```

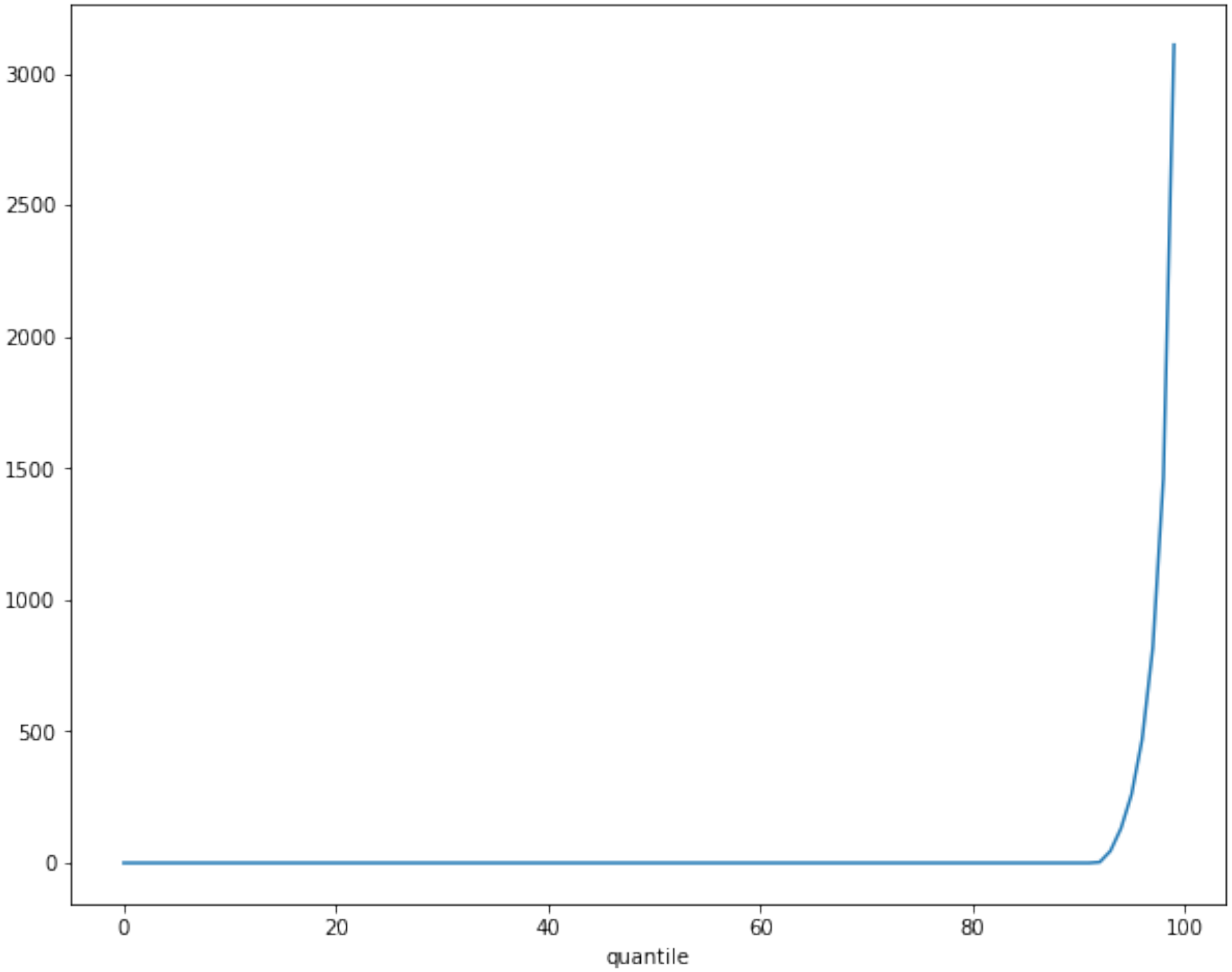
In [19]:

```
values = []
for a in range(0,100) :
    #print(a,"th quantile value :" , all_data_4['No. of Followers/Daily Unique V
isitors'].quantile(a/100))
    values.append(all_data_5['No. of Followers/Daily Unique Visitors'].quantile(
a/100))
print(values)
```

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0
.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.0, 45.0, 131.0
, 261.0, 470.0, 817.0, 1462.0, 3109.319999999949]
```

In [20]:

```
plt.figure(figsize=(10,8))
plt.plot(values)
plt.xlabel('quantile')
plt.show()
```



In [21]:

```
#all_data_5.info()
```

In [22]:

```
q95 = all_data_5['No. of Followers/Daily Unique Visitors'].quantile(0.95)
q95
```

Out[22]:

```
261.0
```

In [23]:

```
all_data_6 = all_data_5[all_data_5['No. of Followers/Daily Unique Visitors'] < q
95]
all_data_6['Type'] = 'Normal'
professional_df = all_data_5[all_data_5['No. of Followers/Daily Unique Visitors'
] >= q95]
professional_df['Type'] = 'Professional'

temp = [all_data_6 , professional_df]
all_data_7 = pd.concat(temp)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  after removing the cwd from sys.path.
```

In [24]:

```
all_data_7.groupby(['Type']).count()
```

Out[24]:

	Post ID	Sound Bite Text	Ratings and Scores	Title	Source Type	Post Type	Media Type	URL	Domain	Publ ((New
Type										
Normal	369719	369719	0	272861	369719	369719	369719	369719	369719	3(
Professional	19466	19466	0	206	19466	19466	19466	19466	19466	'

In [25]:

```
normal_df=all_data_6
```

In [26]:

```
normal_df['Published Date (GMT-04:00) New York']= pd.to_datetime\
(normal_df['Published Date (GMT-04:00) New York']).dt.date
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```

In [27]:

```
normal_df.sort_values(inplace=True,ascending=True,by='Published Date (GMT-04:00)
New York')
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  """Entry point for launching an IPython kernel.
```

In [28]:

```
normal_df.drop(normal_df[normal_df["Author Name"].str.contains("News|news",na=Fa
lse)].index, inplace = True)
```

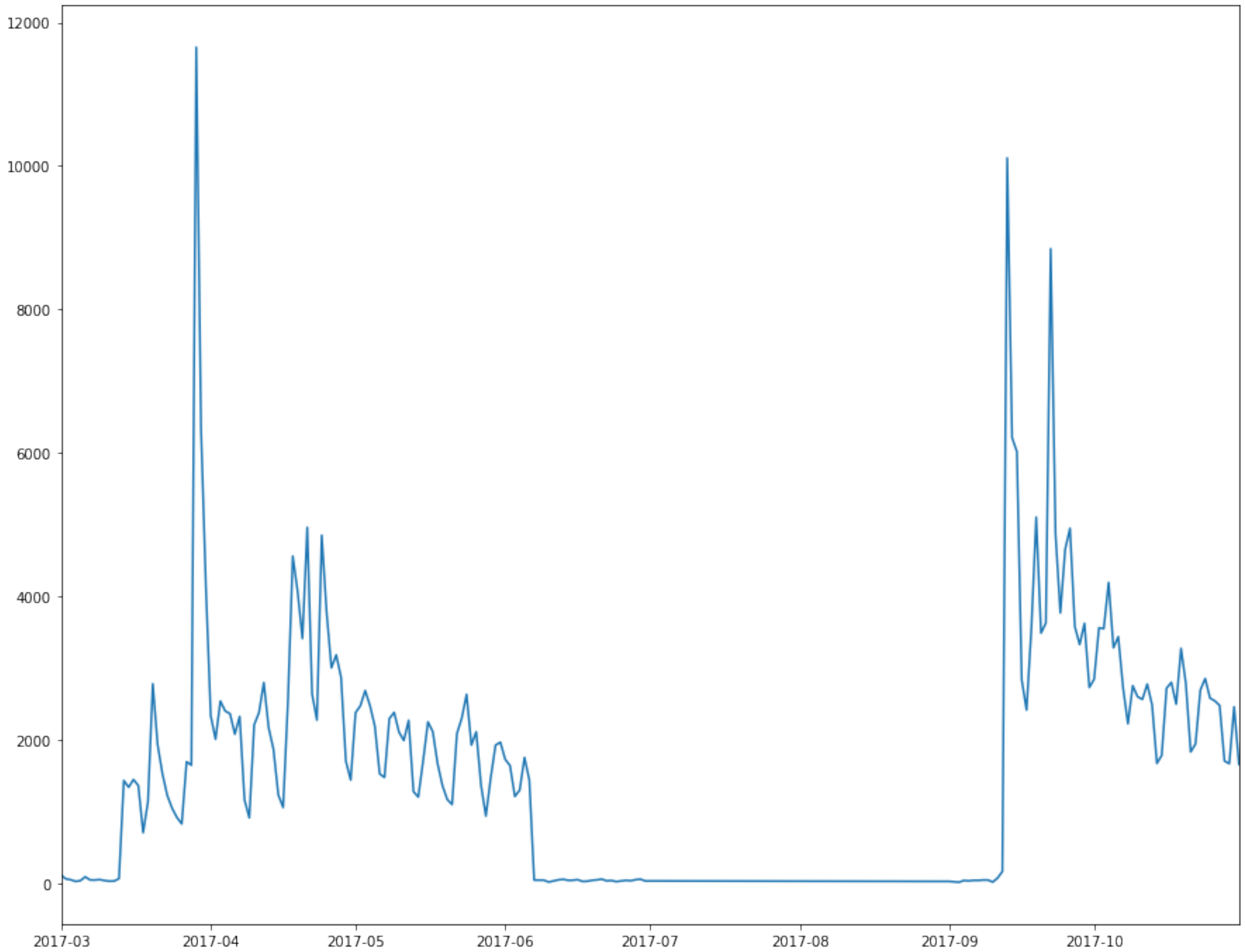
```
/opt/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:4102
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  errors=errors,
```

In [29]:

```
#Tweet Counts for all the products
s = normal_df['Published Date (GMT-04:00) New York'].value_counts()
plt.figure(figsize=(15,12))
s.plot()
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2ef80e90>
```



Extracting Information for the Products

In [30]:

```
df_iphone_8 = normal_df[(normal_df['Sound Bite Text'].str.contains('iPhone 8'))
| (normal_df['Sound Bite Text'].str.contains('iphone 8'))]
df_iphone_8['Sound Bite Text'] = df_iphone_8['Sound Bite Text'].str.lower()
df_iphone_8 = df_iphone_8[  (~df_iphone_8['Sound Bite Text'].str.contains('Galax
y S8')) & (~df_iphone_8['Sound Bite Text'].str.contains('iphone X'))
                        & (~df_iphone_8['Sound Bite Text'].str.contains('iPhone X
'))]
df_iphone_8.info()
df_iphone_8.head()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 167632 entries, 30033 to 12950
Data columns (total 42 columns):
Post ID                                   167632 non-null object
Sound Bite Text                           167632 non-null object
Ratings and Scores                        0 non-null float64
Title                                     123722 non-null object
Source Type                               167632 non-null object
Post Type                                 167632 non-null object
Media Type                                167632 non-null object
URL                                       167632 non-null object
Domain                                    167632 non-null object
Published Date (GMT-04:00) New York       167632 non-null object
Author Gender                             167632 non-null object
Author URL                                110640 non-null object
Author Name                               117413 non-null object
Author Handle                             74428 non-null object
Author ID                                 69285 non-null object
Author Location - Country 1               54972 non-null object
Author Location - State/Province 1        6864 non-null object
Author Location - City 1                  6581 non-null object
Author Location - Country 2               8 non-null object
Author Location - State/Province 2        6 non-null object
Author Location - City 2                  6 non-null object
Author Location - Other                   0 non-null object
No. of Followers/Daily Unique Visitors    167632 non-null float64
Professions                               432 non-null object
Interests                                 712 non-null object
Positive Objects                          35977 non-null object
Negative Objects                          16905 non-null object
Richness                                  167632 non-null float64
Tags                                      0 non-null float64
Quoted Post                               38 non-null object
Quoted Author Name                        38 non-null object
Quoted Author Handle                      38 non-null object
Total Engagements                         21292 non-null float64
Post Comments                             12057 non-null float64
Post Likes                                21042 non-null float64
Post Shares                               12 non-null float64
Post Views                                0 non-null float64
Post Dislikes                             0 non-null float64
Product Name                              2697 non-null object
Product Hierarchy                         0 non-null float64
Rating                                    732 non-null float64
Type                                      167632 non-null object
dtypes: float64(12), object(30)
memory usage: 55.0+ MB
```

Out[30]:

	Post ID	Sound Bite Text	Ratings and Scores	Title	Source Type	Post Type	Media Type	
30033	8.36905e+17	another small teaser (a) iphone 8 #iphone8 #ip...	NaN	NaN	Twitter	Original	Image	http://twitter.com/C
62726	8.3694e+17	instagram media from: the.luxurygram, new ipho...	NaN	NaN	Twitter	Original	No Media	http://twitter.com/ibr
62752	8.36941e+17	iphone 8 to ditch lightning port in favor of u...	NaN	NaN	Twitter	Original	Link	http://twitter.com/lt
35346	8.36929e+17	iphone 8 to sport fingerprint scanner underneа...	NaN	NaN	Twitter	Original	Image; Link	http://twitter.com/ga
63207	8.36927e+17	iphone 8 to use usb-c? xbox to subscription ga...	NaN	NaN	Twitter	Original	No Media	http://twitter.com/h

In [31]:

```
df_iphone_X = normal_df[(normal_df['Sound Bite Text'].str.contains('iPhone X'))
| (normal_df['Sound Bite Text'].str.contains('iphone X'))
                               | (normal_df['Sound Bite Text'].str.contains('iPhone 10')
) | (normal_df['Sound Bite Text'].str.contains('iphone 10'))]
df_iphone_X['Sound Bite Text'] = df_iphone_X['Sound Bite Text'].str.lower()
df_iphone_X = df_iphone_X[(~df_iphone_X['Sound Bite Text'].str.contains('iPhone
8'))]
df_iphone_X = df_iphone_X[(~df_iphone_X['Sound Bite Text'].str.contains('Galaxy'
))]
df_iphone_X.info()
df_iphone_X.head()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  This is separate from the ipykernel package so we can avoid doing
imports until
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41856 entries, 24242 to 12950
Data columns (total 42 columns):
Post ID                                   41856 non-null object
Sound Bite Text                           41856 non-null object
Ratings and Scores                        0 non-null float64
Title                                     36927 non-null object
Source Type                               41856 non-null object
Post Type                                 41856 non-null object
Media Type                                41856 non-null object
URL                                       41856 non-null object
Domain                                    41856 non-null object
Published Date (GMT-04:00) New York       41856 non-null object
Author Gender                             41856 non-null object
Author URL                                26107 non-null object
Author Name                               29866 non-null object
Author Handle                             14876 non-null object
Author ID                                 12395 non-null object
Author Location - Country 1               13713 non-null object
Author Location - State/Province 1        421 non-null object
Author Location - City 1                  405 non-null object
Author Location - Country 2               0 non-null object
Author Location - State/Province 2        0 non-null object
Author Location - City 2                  0 non-null object
Author Location - Other                   0 non-null object
No. of Followers/Daily Unique Visitors    41856 non-null float64
Professions                               30 non-null object
Interests                                 56 non-null object
Positive Objects                          9078 non-null object
Negative Objects                          4168 non-null object
Richness                                  41856 non-null float64
Tags                                      0 non-null float64
Quoted Post                               2 non-null object
Quoted Author Name                        2 non-null object
Quoted Author Handle                      2 non-null object
Total Engagements                         1921 non-null float64
Post Comments                             925 non-null float64
Post Likes                                1912 non-null float64
Post Shares                               0 non-null float64
Post Views                                0 non-null float64
Post Dislikes                             0 non-null float64
Product Name                              38 non-null object
Product Hierarchy                         0 non-null float64
Rating                                    200 non-null float64
Type                                      41856 non-null object
dtypes: float64(12), object(30)
memory usage: 13.7+ MB
```

Out[31]:

	Post ID	Sound Bite Text	Ratings and Scores	Titl
24242	8.39871e+17	i liked a @youtube video youtu.be/3cumfuwnsti?...	NaN	Nal
71039	8.4015e+17	i liked a @youtube video youtu.be/yzbfed0gwgm?...	NaN	Nal
492602	5481249823234012751	even die-hard iphone fans have to admit that t...	NaN	These ar the 1 mos believabl iPhone rumo.
503517	3b277e8dbc8af598a49033f99bd33dd0	with the iphone 8 set to receive a major redes...	NaN	iPhone Concept an Mockup – Part
503449	https://forums.macrumors.com/threads/iphone-8-...	#1 with the iphone 8 set to receive a major re...	NaN	iPhone Concept an Mockup - Part

In [32]:

```
df_galaxy = normal_df[(normal_df['Sound Bite Text'].str.contains('Galaxy')) | (normal_df['Sound Bite Text'].str.contains('galaxy'))]
df_galaxy['Sound Bite Text'] = df_galaxy['Sound Bite Text'].str.lower()
df_galaxy = df_galaxy[(~df_galaxy['Sound Bite Text'].str.contains('iPhone'))]
df_galaxy = df_galaxy[(~df_galaxy['Sound Bite Text'].str.contains('iphone'))]
df_galaxy.info()
df_galaxy.head()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 160320 entries, 63256 to 8670
Data columns (total 42 columns):
Post ID                                   160320 non-null object
Sound Bite Text                           160320 non-null object
Ratings and Scores                        0 non-null float64
Title                                     118133 non-null object
Source Type                               160320 non-null object
Post Type                                 160320 non-null object
Media Type                                160320 non-null object
URL                                       160320 non-null object
Domain                                    160320 non-null object
Published Date (GMT-04:00) New York       160320 non-null object
Author Gender                             160320 non-null object
Author URL                                112126 non-null object
Author Name                               114649 non-null object
Author Handle                             69492 non-null object
Author ID                                 62336 non-null object
Author Location - Country 1               49429 non-null object
Author Location - State/Province 1        7038 non-null object
Author Location - City 1                  6445 non-null object
Author Location - Country 2               22 non-null object
Author Location - State/Province 2        18 non-null object
Author Location - City 2                  15 non-null object
Author Location - Other                   1 non-null object
No. of Followers/Daily Unique Visitors    160320 non-null float64
Professions                               509 non-null object
Interests                                 957 non-null object
Positive Objects                          34937 non-null object
Negative Objects                          15079 non-null object
Richness                                  160320 non-null float64
Tags                                      0 non-null float64
Quoted Post                               36 non-null object
Quoted Author Name                        36 non-null object
Quoted Author Handle                      36 non-null object
Total Engagements                         12677 non-null float64
Post Comments                             7475 non-null float64
Post Likes                                12137 non-null float64
Post Shares                               19 non-null float64
Post Views                                0 non-null float64
Post Dislikes                             0 non-null float64
Product Name                              8122 non-null object
Product Hierarchy                         0 non-null float64
Rating                                    1220 non-null float64
Type                                      160320 non-null object
dtypes: float64(12), object(30)
memory usage: 52.6+ MB
```

Out[32]:

	Post ID	Sound Bite Text	Ratings and Scores	Title	Source Type	Post Type	Media Type	
63256	8.36958e+17	the samsung galaxy s8 probably looks exactly l...	NaN	NaN	Twitter	Original	Link	http://twitter.com/br
62751	8.36937e+17	latest leak 'confirms' galaxy s8's aggressive ...	NaN	NaN	Twitter	Original	Link	http://twitter.com/itj
62749	8.36936e+17	more galaxy s8 leaks mound up ahead of #samsun...	NaN	NaN	Twitter	Original	Link	http://twitter.com/GTV
62682	8.37019e+17	everything we think we know about the galaxy s...	NaN	NaN	Twitter	Original	Image; Link	http://twitter.com/Ri
62639	8.37037e+17	i liked a @youtube video from @androidauth you...	NaN	NaN	Twitter	Original	Link	http://twitter.com/c

In [33]:

```
df_both = normal_df[(normal_df['Sound Bite Text'].str.contains('Galaxy')) | (nor
mal_df['Sound Bite Text'].str.contains('galaxy')) |
                   (normal_df['Sound Bite Text'].str.contains('iphone')) | (norm
al_df['Sound Bite Text'].str.contains('iPhone'))]
df_both.info()
df_both.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 348735 entries, 63256 to 12950
Data columns (total 42 columns):
Post ID                                   348735 non-null object
Sound Bite Text                           348735 non-null object
Ratings and Scores                        0 non-null float64
Title                                     256407 non-null object
Source Type                               348735 non-null object
Post Type                                 348735 non-null object
Media Type                                348735 non-null object
URL                                       348735 non-null object
Domain                                    348735 non-null object
Published Date (GMT-04:00) New York       348735 non-null object
Author Gender                             348735 non-null object
Author URL                                237393 non-null object
Author Name                               247250 non-null object
Author Handle                             153873 non-null object
Author ID                                 140863 non-null object
Author Location - Country 1               110560 non-null object
Author Location - State/Province 1        14861 non-null object
Author Location - City 1                  13952 non-null object
Author Location - Country 2               32 non-null object
Author Location - State/Province 2        25 non-null object
Author Location - City 2                  21 non-null object
Author Location - Other                   2 non-null object
No. of Followers/Daily Unique Visitors    348735 non-null float64
Professions                               976 non-null object
Interests                                 1723 non-null object
Positive Objects                          76162 non-null object
Negative Objects                          34172 non-null object
Richness                                  348735 non-null float64
Tags                                      0 non-null float64
Quoted Post                               77 non-null object
Quoted Author Name                        77 non-null object
Quoted Author Handle                      77 non-null object
Total Engagements                         37746 non-null float64
Post Comments                             21602 non-null float64
Post Likes                                36935 non-null float64
Post Shares                               32 non-null float64
Post Views                                0 non-null float64
Post Dislikes                             0 non-null float64
Product Name                              11086 non-null object
Product Hierarchy                         0 non-null float64
Rating                                    2069 non-null float64
Type                                      348735 non-null object
dtypes: float64(12), object(30)
memory usage: 114.4+ MB
```

Out[33]:

	Post ID	Sound Bite Text	Ratings and Scores	Title	Source Type	Post Type	Media Type	
63256	8.36958e+17	The Samsung Galaxy S8 probably looks exactly l...	NaN	NaN	Twitter	Original	Link	http://twitter.com/I
62751	8.36937e+17	Latest Leak 'Confirms' Galaxy S8's Aggressive ...	NaN	NaN	Twitter	Original	Link	http://twitter.com/
30033	8.36905e+17	Another small teaser (A) iPhone 8 #iPhone8 #iP...	NaN	NaN	Twitter	Original	Image	http://twitter.com/C
62749	8.36936e+17	More Galaxy S8 leaks mound up ahead of #Samsun...	NaN	NaN	Twitter	Original	Link	http://twitter.com/GT
62726	8.3694e+17	Instagram Media from: the.luxurygram, New iPho...	NaN	NaN	Twitter	Original	No Media	http://twitter.com/ibr

In [34]:

```
#iPhone 8 Before and After Splitting
df_before = df_iphone_8[df_iphone_8['Published Date (GMT-04:00) New York'] < dat
e(2017, 9, 22)]
df_after = df_iphone_8[df_iphone_8['Published Date (GMT-04:00) New York'] >= dat
e(2017, 9, 22)]
df_before['Before/After'] = "Before"
df_after["Before/After"] = "After"
temp = [df_before , df_after]
df_iphone_8 = pd.concat(temp)
df_iphone_8.info()
```

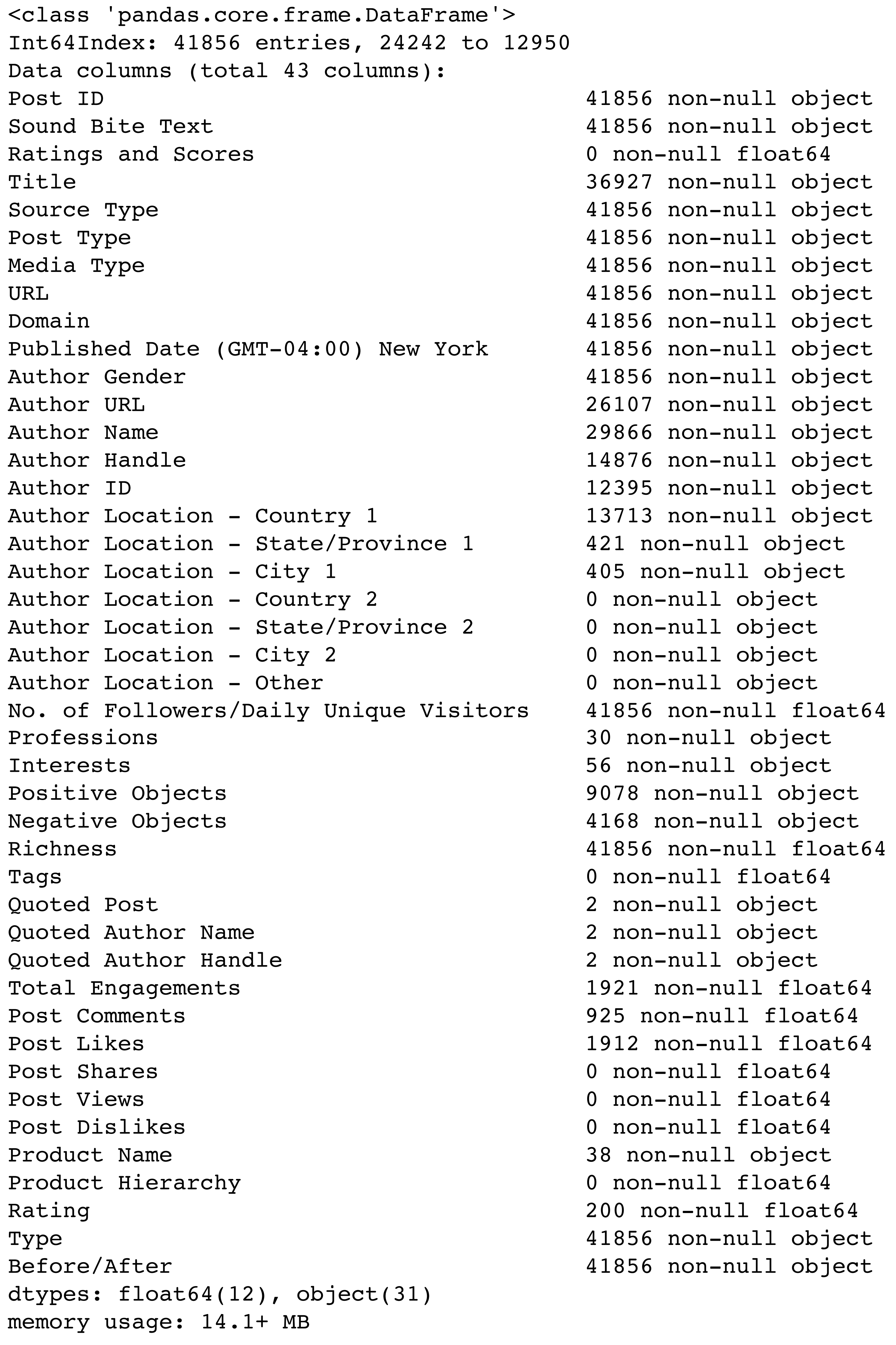
```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  after removing the cwd from sys.path.
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  """
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 167632 entries, 30033 to 12950
Data columns (total 43 columns):
Post ID                                   167632 non-null object
Sound Bite Text                           167632 non-null object
Ratings and Scores                        0 non-null float64
Title                                     123722 non-null object
Source Type                               167632 non-null object
Post Type                                 167632 non-null object
Media Type                                167632 non-null object
URL                                       167632 non-null object
Domain                                    167632 non-null object
Published Date (GMT-04:00) New York       167632 non-null object
Author Gender                             167632 non-null object
Author URL                                110640 non-null object
Author Name                               117413 non-null object
Author Handle                             74428 non-null object
Author ID                                 69285 non-null object
Author Location - Country 1               54972 non-null object
Author Location - State/Province 1        6864 non-null object
Author Location - City 1                  6581 non-null object
Author Location - Country 2               8 non-null object
Author Location - State/Province 2        6 non-null object
Author Location - City 2                  6 non-null object
Author Location - Other                   0 non-null object
No. of Followers/Daily Unique Visitors    167632 non-null float64
Professions                               432 non-null object
Interests                                 712 non-null object
Positive Objects                          35977 non-null object
Negative Objects                          16905 non-null object
Richness                                  167632 non-null float64
Tags                                      0 non-null float64
Quoted Post                               38 non-null object
Quoted Author Name                        38 non-null object
Quoted Author Handle                      38 non-null object
Total Engagements                         21292 non-null float64
Post Comments                             12057 non-null float64
Post Likes                                21042 non-null float64
Post Shares                               12 non-null float64
Post Views                                0 non-null float64
Post Dislikes                             0 non-null float64
Product Name                              2697 non-null object
Product Hierarchy                         0 non-null float64
Rating                                    732 non-null float64
Type                                      167632 non-null object
Before/After                              167632 non-null object
dtypes: float64(12), object(31)
memory usage: 56.3+ MB
```

In [35]:

```
df_before = df_iphone_X[df_iphone_X['Published Date (GMT-04:00) New York'] < dat
e(2017, 9, 22)]
df_after = df_iphone_X[df_iphone_X['Published Date (GMT-04:00) New York'] >= dat
e(2017, 9, 22)]
df_before['Before/After'] = "Before"
df_after["Before/After"] = "After"
temp = [df_before , df_after]
df_iphone_X = pd.concat(temp)
df_iphone_X.info()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  This is separate from the ipykernel package so we can avoid doing
imports until
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  after removing the cwd from sys.path.
```



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41856 entries, 24242 to 12950
Data columns (total 43 columns):
Post ID                                   41856 non-null object
Sound Bite Text                           41856 non-null object
Ratings and Scores                        0 non-null float64
Title                                     36927 non-null object
Source Type                               41856 non-null object
Post Type                                 41856 non-null object
Media Type                                41856 non-null object
URL                                       41856 non-null object
Domain                                    41856 non-null object
Published Date (GMT-04:00) New York       41856 non-null object
Author Gender                             41856 non-null object
Author URL                                26107 non-null object
Author Name                               29866 non-null object
Author Handle                             14876 non-null object
Author ID                                 12395 non-null object
Author Location - Country 1               13713 non-null object
Author Location - State/Province 1        421 non-null object
Author Location - City 1                  405 non-null object
Author Location - Country 2               0 non-null object
Author Location - State/Province 2        0 non-null object
Author Location - City 2                  0 non-null object
Author Location - Other                   0 non-null object
No. of Followers/Daily Unique Visitors    41856 non-null float64
Professions                               30 non-null object
Interests                                 56 non-null object
Positive Objects                          9078 non-null object
Negative Objects                          4168 non-null object
Richness                                  41856 non-null float64
Tags                                      0 non-null float64
Quoted Post                               2 non-null object
Quoted Author Name                        2 non-null object
Quoted Author Handle                      2 non-null object
Total Engagements                         1921 non-null float64
Post Comments                             925 non-null float64
Post Likes                                1912 non-null float64
Post Shares                               0 non-null float64
Post Views                                0 non-null float64
Post Dislikes                             0 non-null float64
Product Name                              38 non-null object
Product Hierarchy                         0 non-null float64
Rating                                    200 non-null float64
Type                                      41856 non-null object
Before/After                              41856 non-null object
dtypes: float64(12), object(31)
memory usage: 14.1+ MB
```

In [36]:

```
df_before = df_galaxy[df_galaxy['Published Date (GMT-04:00) New York'] < date(20
17, 3, 29)]
df_after = df_galaxy[df_galaxy['Published Date (GMT-04:00) New York'] >= date(20
17, 3, 29)]
df_before['Before/After'] = "Before"
df_after["Before/After"] = "After"
temp = [df_before , df_after]
df_galaxy = pd.concat(temp)
df_galaxy.info()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  This is separate from the ipykernel package so we can avoid doing
imports until
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  after removing the cwd from sys.path.
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 160320 entries, 63256 to 8670
Data columns (total 43 columns):
Post ID                                   160320 non-null object
Sound Bite Text                           160320 non-null object
Ratings and Scores                        0 non-null float64
Title                                     118133 non-null object
Source Type                               160320 non-null object
Post Type                                 160320 non-null object
Media Type                                160320 non-null object
URL                                       160320 non-null object
Domain                                    160320 non-null object
Published Date (GMT-04:00) New York       160320 non-null object
Author Gender                             160320 non-null object
Author URL                                112126 non-null object
Author Name                               114649 non-null object
Author Handle                             69492 non-null object
Author ID                                 62336 non-null object
Author Location - Country 1               49429 non-null object
Author Location - State/Province 1        7038 non-null object
Author Location - City 1                  6445 non-null object
Author Location - Country 2               22 non-null object
Author Location - State/Province 2        18 non-null object
Author Location - City 2                  15 non-null object
Author Location - Other                   1 non-null object
No. of Followers/Daily Unique Visitors    160320 non-null float64
Professions                               509 non-null object
Interests                                 957 non-null object
Positive Objects                          34937 non-null object
Negative Objects                          15079 non-null object
Richness                                  160320 non-null float64
Tags                                      0 non-null float64
Quoted Post                               36 non-null object
Quoted Author Name                        36 non-null object
Quoted Author Handle                      36 non-null object
Total Engagements                         12677 non-null float64
Post Comments                             7475 non-null float64
Post Likes                                12137 non-null float64
Post Shares                               19 non-null float64
Post Views                                0 non-null float64
Post Dislikes                             0 non-null float64
Product Name                              8122 non-null object
Product Hierarchy                         0 non-null float64
Rating                                    1220 non-null float64
Type                                      160320 non-null object
Before/After                              160320 non-null object
dtypes: float64(12), object(31)
memory usage: 53.8+ MB
```

Most Important Attributes

In [37]:

```python
from collections import Counter
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
```

In [38]:

```python
#Splitting the Dataframe into before and after for iPhone 8
df_iphone_8_before = pd.DataFrame(df_iphone_8[df_iphone_8['Before/After'] == 'Before'])
df_iphone_8_after = pd.DataFrame(df_iphone_8[df_iphone_8['Before/After'] == 'After'])



df = df_iphone_8_before
stop = stopwords.words('english')
df['Sound Bite Text'].apply(lambda x: x.lower())
df['tokenize'] = df['Sound Bite Text'].apply(word_tokenize)
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [word for word in x if word.isalpha()])
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [lemmatizer.lemmatize(y) for y in x])
#stemmer = PorterStemmer()
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [stemmer.stem(y) for y in x])
df['tokenize'] = df['tokenize'].apply(lambda x: [item.lower() for item in x])
df['sentence'] = df['tokenize'].apply(' '.join)
df['attr'] = df['tokenize'].apply(lambda x: nltk.pos_tag(x))
att = []
for values in df['attr']:
    for pair in values :
        if(pair[1] == 'NN' or pair[1] == 'NNS' or pair[1] == 'NNP' or pair[1]=='NNPS'):
            att.append(pair[0])
        if(pair[1] == 'JJ' or pair[1] == 'JJS' or pair[1] == 'JJR'):
            att.append(pair[0])
print("Most common attributes")
Counter(att).most_common(25)
```

```
Most common attributes
```

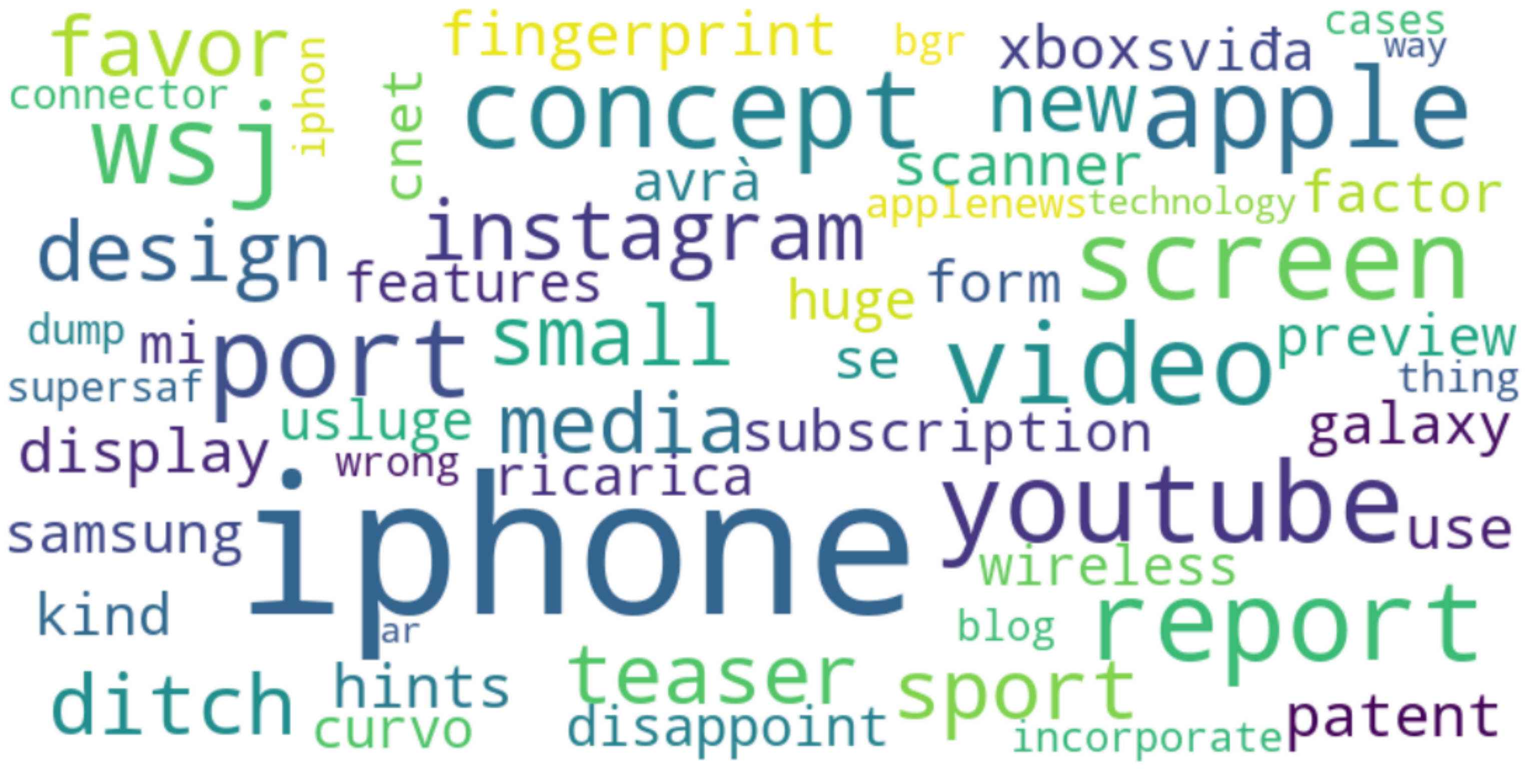
Out[38]:

```
[('iphone', 301474),
 ('apple', 148573),
 ('new', 78615),
 ('x', 46130),
 ('galaxy', 33910),
 ('samsung', 29306),
 ('phone', 28403),
 ('display', 23993),
 ('design', 21710),
 ('year', 20734),
 ('camera', 20017),
 ('screen', 19772),
 ('wireless', 18187),
 ('device', 15321),
 ('news', 15015),
 ('features', 14073),
 ('id', 13154),
 ('technology', 13132),
 ('touch', 13108),
 ('time', 12970),
 ('company', 12886),
 ('rumors', 12837),
 ('launch', 12752),
 ('smartphone', 12752),
 ('price', 12559)]
```

In [39]:

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

print("Top Attributes before Release")

NN_jt_text =(" ").join(att[:80])
type(NN_jt_text)
# top 200 active listing noun word cloud
# lower max_font_size, change the maximum number of word and lighten the backgro
und:
wc_jt = WordCloud(width=800, height=400, stopwords=stop, max_font_size=100, max_
words=200, background_color="white").generate(NN_jt_text)
#stopwords:set of strings or None
#The words that will be eliminated. If None, the build-in STOPWORDS list will be
used. Ignored if using generate_from_frequencies.

plt.figure( figsize=(20,10) )
plt.imshow(wc_jt, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
Top Attributes before Release
```



In [40]:

```
df = df_iphone_8_after
stop = stopwords.words('english')
df['Sound Bite Text'].apply(lambda x: x.lower())
df['tokenize'] = df['Sound Bite Text'].apply(word_tokenize)
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [word for word in x if word.isalpha()])
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [lemmatizer.lemmatize(y) for y in x])
#stemmer = PorterStemmer()
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [stemmer.stem(y) for y in x])
df['tokenize'] = df['tokenize'].apply(lambda x: [item.lower() for item in x])
df['sentence'] = df['tokenize'].apply(' '.join)
df['attr'] = df['tokenize'].apply(lambda x: nltk.pos_tag(x))
att = []
for values in df['attr']:
    for pair in values :
        if(pair[1] == 'NN' or pair[1] == 'NNS' or pair[1] == 'NNP' or pair[1]=='NNPS'):
            att.append(pair[0])
        if(pair[1] == 'JJ' or pair[1] == 'JJS' or pair[1] == 'JJR'):
            att.append(pair[0])
print("Most common attributes")
Counter(att).most_common(25)
```

```
Most common attributes
```

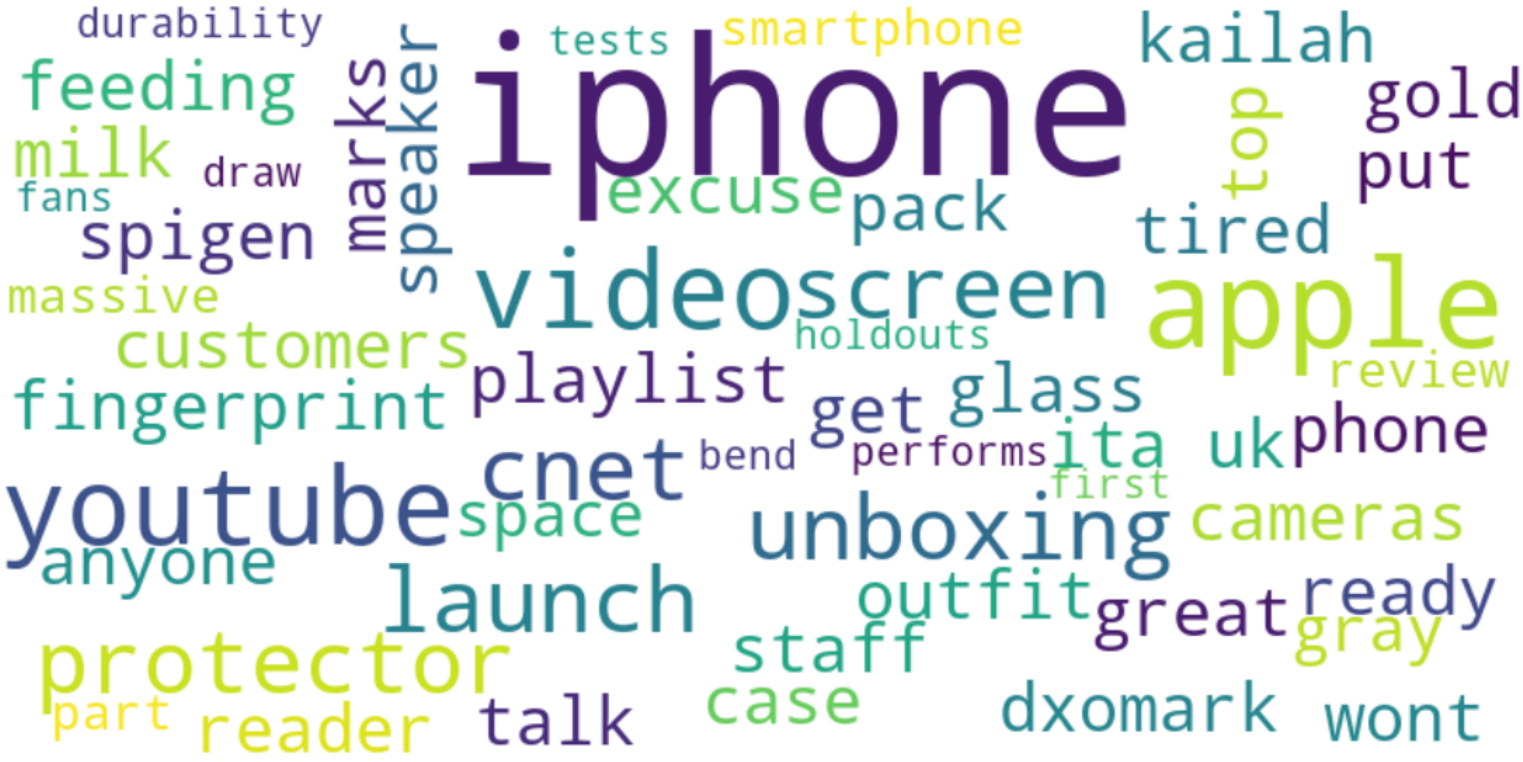
Out[40]:

```
[('iphone', 260491),
 ('apple', 93540),
 ('new', 51914),
 ('x', 45038),
 ('phone', 30134),
 ('camera', 22766),
 ('pixel', 17934),
 ('galaxy', 17523),
 ('samsung', 13719),
 ('wireless', 13418),
 ('ios', 13234),
 ('case', 13076),
 ('screen', 12780),
 ('note', 12039),
 ('google', 11847),
 ('phones', 11677),
 ('smartphone', 11494),
 ('battery', 11388),
 ('http', 11000),
 ('year', 10159),
 ('time', 10114),
 ('device', 9623),
 ('best', 9194),
 ('video', 8750),
 ('tags', 8497)]
```

In [41]:

```
NN_jt_text =(" ").join(att[:80])
type(NN_jt_text)

print("Top Attributes after Release")
# top 200 active listing noun word cloud
# lower max_font_size, change the maximum number of word and lighten the backgro
und:
wc_jt = WordCloud(width=800, height=400, stopwords=stop, max_font_size=100, max_
words=200, background_color="white").\
generate(NN_jt_text)
#stopwords:set of strings or None
#The words that will be eliminated. If None, the build-in STOPWORDS list will be
used. Ignored if using generate_from_frequencies.

plt.figure( figsize=(20,10) )
plt.imshow(wc_jt, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
Top Attributes after Release
```



In [42]:

```
#Splitting the Dataframe into before and after for iPhone X
df_iphone_X_before = pd.DataFrame(df_iphone_X[df_iphone_X['Before/After'] == 'Before'])
df_iphone_X_after = pd.DataFrame(df_iphone_X[df_iphone_X['Before/After'] == 'After'])


df = df_iphone_X_before
stop = stopwords.words('english')
df['Sound Bite Text'].apply(lambda x: x.lower())
df['tokenize'] = df['Sound Bite Text'].apply(word_tokenize)
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [word for word in x if word.isalpha()])
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [lemmatizer.lemmatize(y) for y in x])
#stemmer = PorterStemmer()
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [stemmer.stem(y) for y in x])
df['tokenize'] = df['tokenize'].apply(lambda x: [item.lower() for item in x])
df['sentence'] = df['tokenize'].apply(' '.join)
df['attr'] = df['tokenize'].apply(lambda x: nltk.pos_tag(x))
att = []
for values in df['attr']:
    for pair in values :
        if(pair[1] == 'NN' or pair[1] == 'NNS' or pair[1] == 'NNP' or pair[1]=='NNPS'):
            att.append(pair[0])
        if(pair[1] == 'JJ' or pair[1] == 'JJS' or pair[1] == 'JJR'):
            att.append(pair[0])
print("Most common attributes")
Counter(att).most_common(25)
```

```
Most common attributes
```

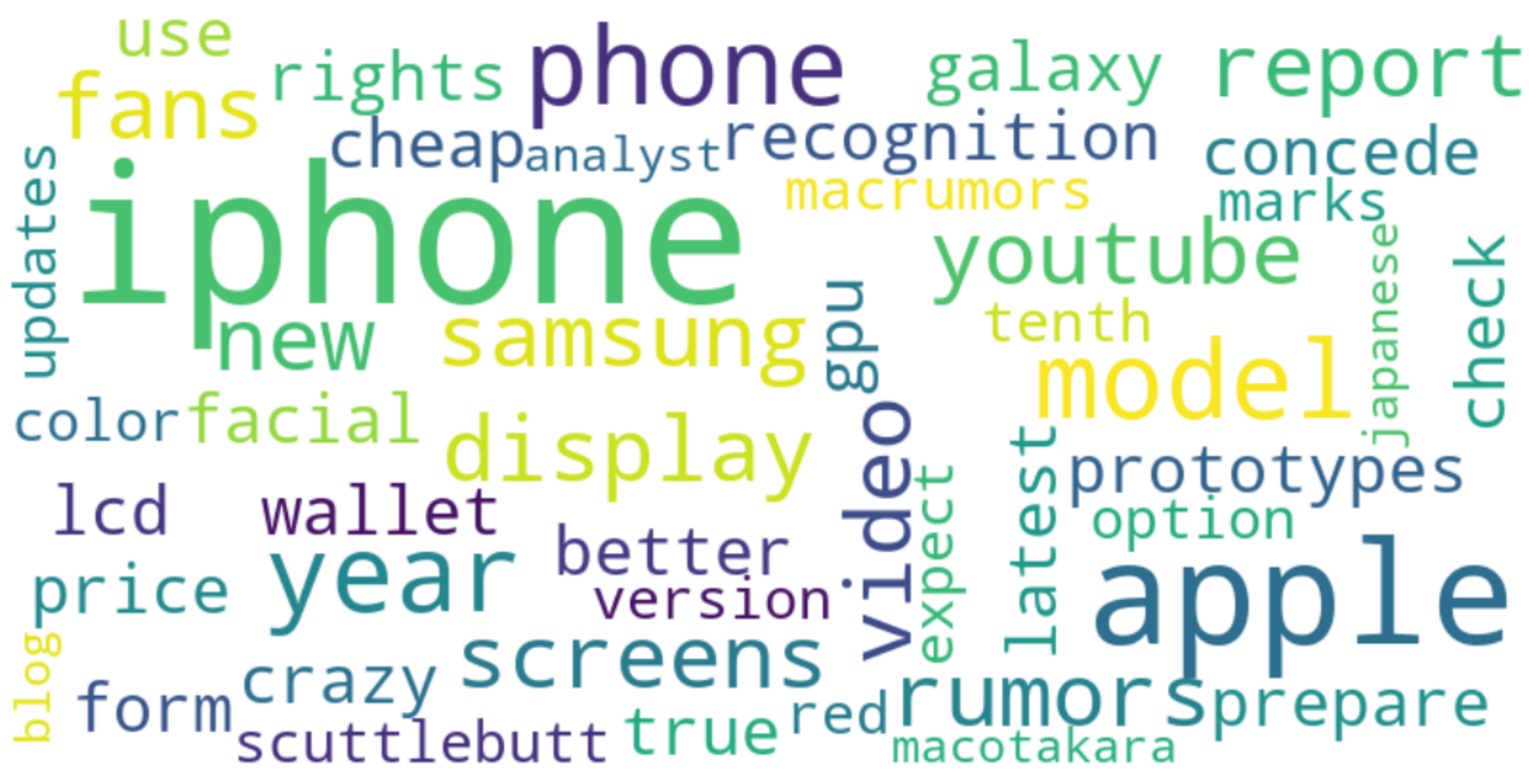
Out[42]:

```
[('iphone', 141619),
 ('apple', 59557),
 ('x', 44169),
 ('new', 33551),
 ('phone', 10510),
 ('wireless', 9924),
 ('galaxy', 8568),
 ('camera', 7957),
 ('samsung', 7751),
 ('display', 7392),
 ('screen', 7265),
 ('year', 6403),
 ('watch', 6209),
 ('phones', 5974),
 ('design', 5759),
 ('iphones', 5741),
 ('features', 5725),
 ('available', 5608),
 ('price', 5605),
 ('device', 4855),
 ('time', 4759),
 ('september', 4749),
 ('id', 4631),
 ('event', 4620),
 ('glass', 4501)]
```

```
In [43]:
```

```
NN_jt_text =(" ").join(att[:80])
type(NN_jt_text)

print("Top Attributes before Release")
# top 200 active listing noun word cloud
# lower max_font_size, change the maximum number of word and lighten the backgro
und:
wc_jt = WordCloud(width=800, height=400, stopwords=stop, max_font_size=100, max_
words=200, background_color="white").\
generate(NN_jt_text)
#stopwords:set of strings or None
#The words that will be eliminated. If None, the build-in STOPWORDS list will be
used. Ignored if using generate_from_frequencies.

plt.figure( figsize=(20,10) )
plt.imshow(wc_jt, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
Top Attributes before Release
```



In [44]:

```
df = df_iphone_X_after
stop = stopwords.words('english')
df['Sound Bite Text'].apply(lambda x: x.lower())
df['tokenize'] = df['Sound Bite Text'].apply(word_tokenize)
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [word for word in x if word.isalpha()])
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [lemmatizer.lemmatize(y) for y in x])
#stemmer = PorterStemmer()
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [stemmer.stem(y) for y in x])
df['tokenize'] = df['tokenize'].apply(lambda x: [item.lower() for item in x])
df['sentence'] = df['tokenize'].apply(' '.join)
df['attr'] = df['tokenize'].apply(lambda x: nltk.pos_tag(x))
att = []
for values in df['attr']:
    for pair in values :
        if(pair[1] == 'NN' or pair[1] == 'NNS' or pair[1] == 'NNP' or pair[1]=='NNPS'):
            att.append(pair[0])
        if(pair[1] == 'JJ' or pair[1] == 'JJS' or pair[1] == 'JJR'):
            att.append(pair[0])
print("Most common attributes")
Counter(att).most_common(25)
```

```
Most common attributes
```

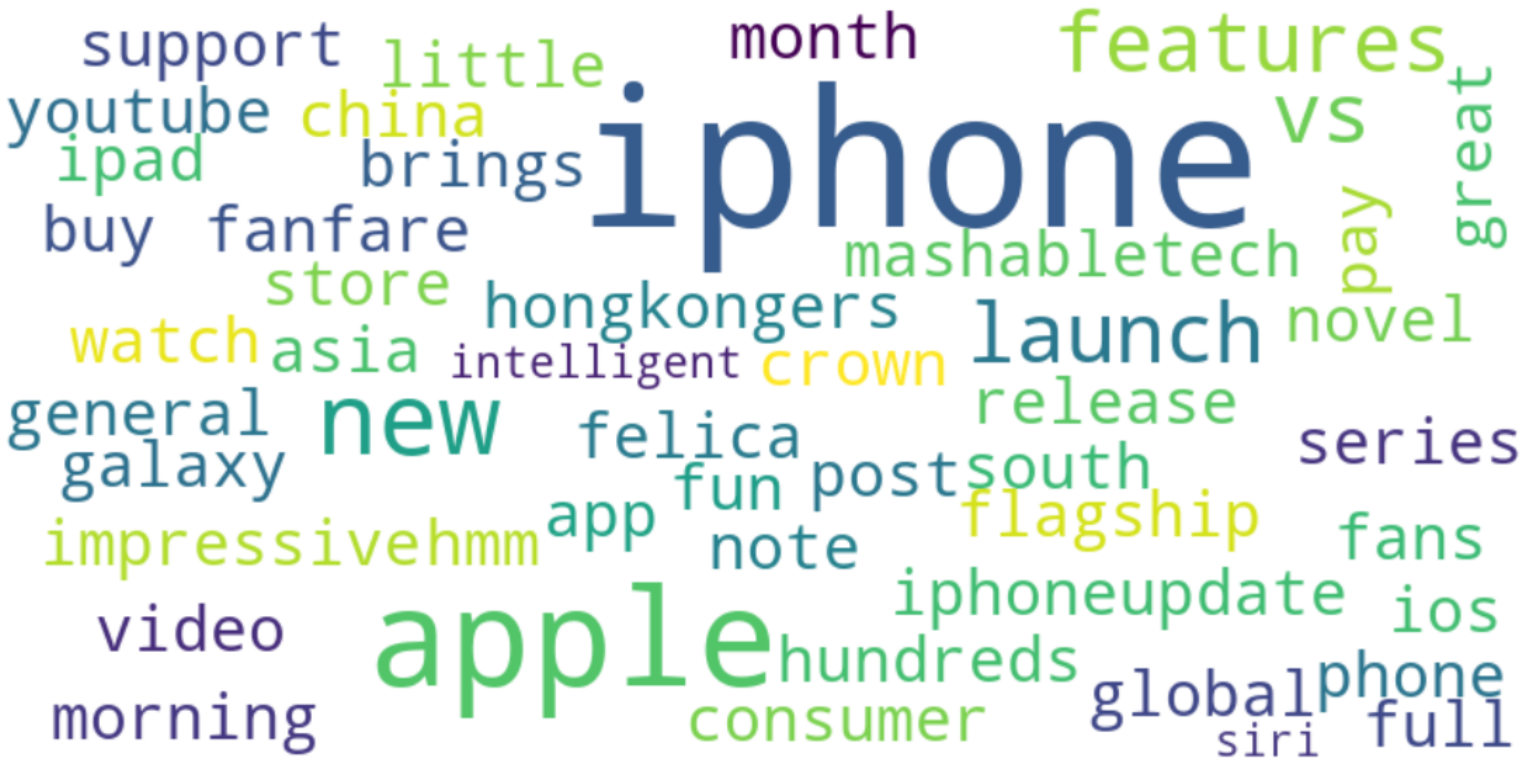
```
Out[44]:
```

```
[('iphone', 127384),
 ('apple', 47343),
 ('x', 43058),
 ('new', 23821),
 ('phone', 12148),
 ('galaxy', 9516),
 ('camera', 8764),
 ('wireless', 8473),
 ('samsung', 7701),
 ('pixel', 7537),
 ('screen', 6854),
 ('phones', 6506),
 ('year', 6328),
 ('display', 5391),
 ('google', 5296),
 ('battery', 5024),
 ('smartphone', 4926),
 ('time', 4917),
 ('price', 4751),
 ('device', 4720),
 ('design', 4588),
 ('note', 4535),
 ('iphones', 4406),
 ('devices', 4114),
 ('launch', 3994)]
```

In [45]:

```
NN_jt_text =(" ").join(att[:80])
type(NN_jt_text)

print("Top Attributes after Release")
# top 200 active listing noun word cloud
# lower max_font_size, change the maximum number of word and lighten the background:
wc_jt = WordCloud(width=800, height=400, stopwords=stop, max_font_size=100, max_words=200, background_color="white").\
generate(NN_jt_text)
#stopwords:set of strings or None
#The words that will be eliminated. If None, the build-in STOPWORDS list will be used. Ignored if using generate_from_frequencies.

plt.figure( figsize=(20,10) )
plt.imshow(wc_jt, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
Top Attributes after Release
```



In [46]:

```
#Splitting the Dataframe into before and after for Samsung Galaxy 8
df_galaxy_before = pd.DataFrame(df_galaxy[df_galaxy['Before/After'] == 'Before']
)
df_galaxy_after = pd.DataFrame(df_galaxy[df_galaxy['Before/After'] == 'After'])


df = df_galaxy_before
stop = stopwords.words('english')
df['Sound Bite Text'].apply(lambda x: x.lower())
df['tokenize'] = df['Sound Bite Text'].apply(word_tokenize)
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not
in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not
in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [word for word in x if word.isal
pha()])
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [lemmatizer.lemmati
ze(y) for y in x])
#stemmer = PorterStemmer()
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [stemmer.stem(y) fo
r y in x])
df['tokenize'] = df['tokenize'].apply(lambda x: [item.lower() for item in x])
df['sentence'] = df['tokenize'].apply(' '.join)
df['attr'] = df['tokenize'].apply(lambda x: nltk.pos_tag(x))
att = []
for values in df['attr']:
    for pair in values :
        if(pair[1] == 'NN' or pair[1] == 'NNS' or pair[1] == 'NNP' or pair[1]=='
NNPS'):
            att.append(pair[0])
        if(pair[1] == 'JJ' or pair[1] == 'JJS' or pair[1] == 'JJR'):
            att.append(pair[0])
print("Most common attributes")
Counter(att).most_common(25)
```

```
Most common attributes
```

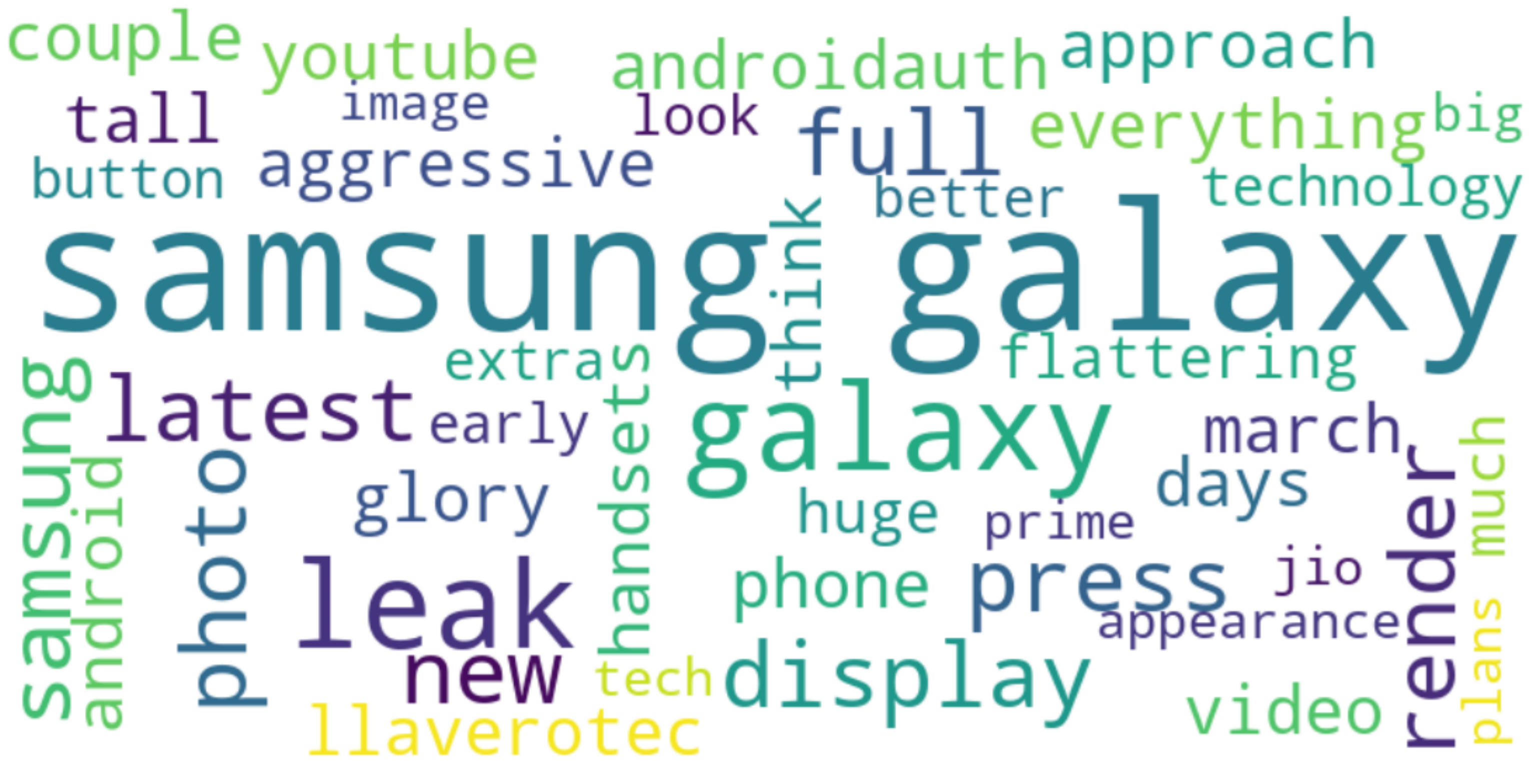
Out[46]:

```
[('galaxy', 35391),
 ('samsung', 26522),
 ('new', 10334),
 ('phone', 5301),
 ('bixby', 4251),
 ('launch', 3591),
 ('news', 3520),
 ('http', 3457),
 ('march', 3100),
 ('assistant', 2975),
 ('leaks', 2950),
 ('flagship', 2878),
 ('smartphone', 2871),
 ('company', 2643),
 ('device', 2635),
 ('camera', 2598),
 ('phones', 2510),
 ('black', 2486),
 ('android', 2418),
 ('note', 2387),
 ('time', 2296),
 ('first', 2210),
 ('devices', 2206),
 ('leak', 2173),
 ('screen', 2116)]
```

In [47]:

```
NN_jt_text =(" ").join(att[:80])
type(NN_jt_text)

print("Top Attributes before Release")
# top 200 active listing noun word cloud
# lower max_font_size, change the maximum number of word and lighten the backgro
und:
wc_jt = WordCloud(width=800, height=400, stopwords=stop, max_font_size=100, max_
words=200, background_color="white").\
generate(NN_jt_text)
#stopwords:set of strings or None
#The words that will be eliminated. If None, the build-in STOPWORDS list will be
used. Ignored if using generate_from_frequencies.

plt.figure( figsize=(20,10) )
plt.imshow(wc_jt, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
Top Attributes before Release
```



In [48]:

```
df = df_galaxy_after
stop = stopwords.words('english')
df['Sound Bite Text'].apply(lambda x: x.lower())
df['tokenize'] = df['Sound Bite Text'].apply(word_tokenize)
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [word for word in x if word.isalpha()])
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [lemmatizer.lemmatize(y) for y in x])
#stemmer = PorterStemmer()
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [stemmer.stem(y) for y in x])
df['tokenize'] = df['tokenize'].apply(lambda x: [item.lower() for item in x])
df['sentence'] = df['tokenize'].apply(' '.join)
df['attr'] = df['tokenize'].apply(lambda x: nltk.pos_tag(x))
att = []
for values in df['attr']:
    for pair in values :
        if(pair[1] == 'NN' or pair[1] == 'NNS' or pair[1] == 'NNP' or pair[1]=='NNPS'):
            att.append(pair[0])
        if(pair[1] == 'JJ' or pair[1] == 'JJS' or pair[1] == 'JJR'):
            att.append(pair[0])
print("Most common attributes")
Counter(att).most_common(25)
```

```
Most common attributes
```

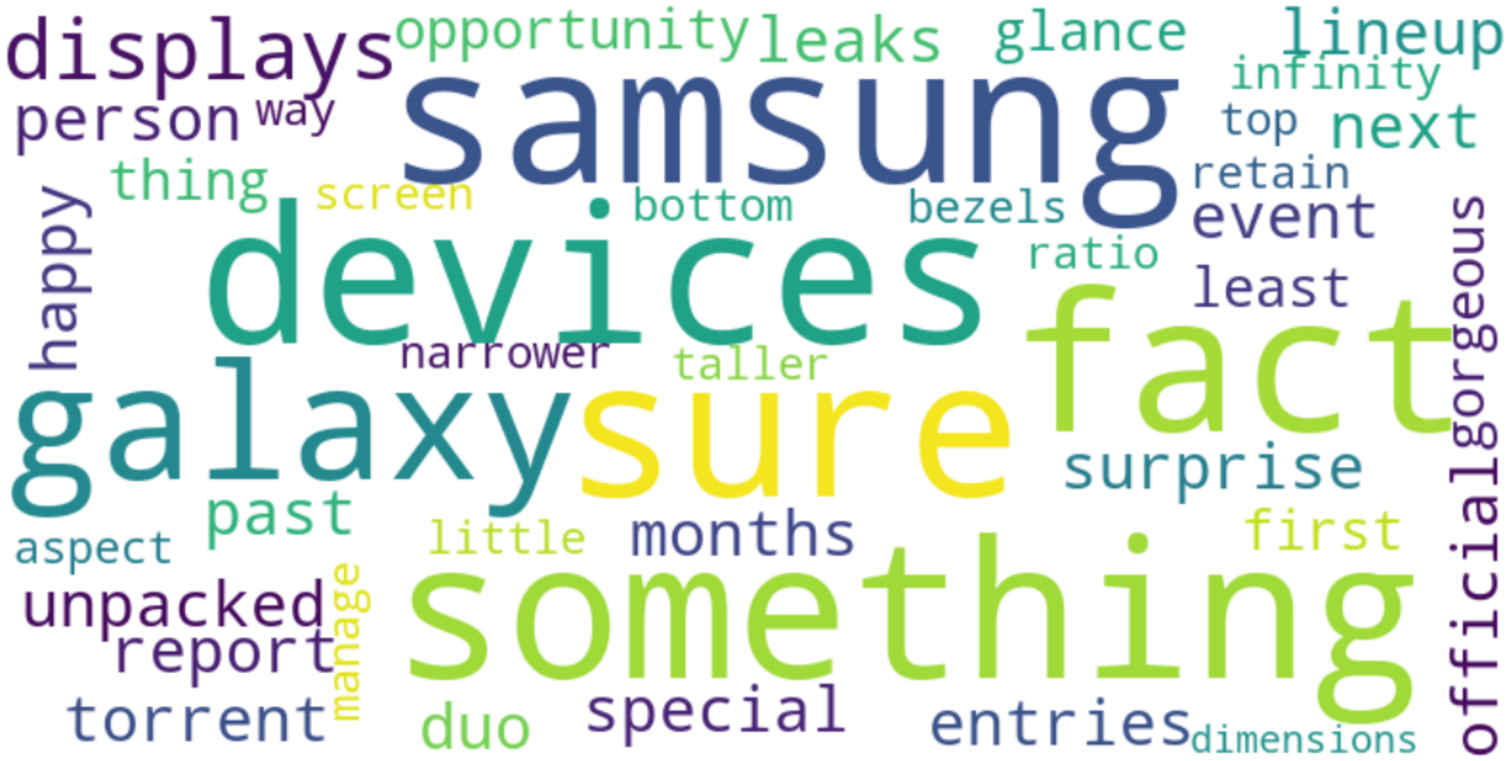
Out[48]:

```
[('galaxy', 333859),
 ('samsung', 238712),
 ('phone', 73763),
 ('new', 70786),
 ('screen', 35845),
 ('android', 35216),
 ('smartphone', 29416),
 ('camera', 27144),
 ('display', 26685),
 ('phones', 26242),
 ('note', 25275),
 ('device', 24085),
 ('bixby', 24058),
 ('google', 22648),
 ('case', 22611),
 ('http', 22551),
 ('devices', 21047),
 ('flagship', 18838),
 ('available', 18402),
 ('smartphones', 17217),
 ('company', 17020),
 ('time', 16827),
 ('features', 16523),
 ('design', 16523),
 ('launch', 16497)]
```

In [49]:

```
NN_jt_text =(" ").join(att[:50])
type(NN_jt_text)

print("Top Attributes after Release")
# top 200 active listing noun word cloud
# lower max_font_size, change the maximum number of word and lighten the backgro
und:
wc_jt = WordCloud(width=800, height=400, stopwords=stop, max_font_size=100, max_
words=200, background_color="white").\
generate(NN_jt_text)
#stopwords:set of strings or None
#The words that will be eliminated. If None, the build-in STOPWORDS list will be
used. Ignored if using generate_from_frequencies.

plt.figure( figsize=(20,10) )
plt.imshow(wc_jt, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
Top Attributes after Release
```



Calculating n-grams to check with the top attributes

In [50]:

```
#Put n-grams as a part of cleaning and important attributes
soundbite_list = normal_df['Sound Bite Text'].tolist()
```

In [51]:

```
#lowercase words as they are tokenized
review_lower = [tok.lower() for i in soundbite_list for tok in nltk.word_tokeniz
e(i)]
#print(len(review_lower))
#print(review_lower[:10])

#stopwords removal
nltk_stopwords = set(stopwords.words('english'))

review_lower_stop = [x for x in review_lower if not x in nltk_stopwords]
#print(len(review_lower_stop))
#print(review_lower_stop[:30])
```

In [52]:

```
## remove non-alphabetic characters ( ex) punctations)
# function that takes a word and returns true if it consists only
#   of non-alphabetic characters
def alpha_filter(w):
    pattern = re.compile('^[^a-z]+$')
    if (pattern.match(w)):
        return True
    else:
        return False

# remove punctuations
review_lower_stop_pun = [y for y in review_lower_stop if not alpha_filter(y)]
print(len(review_lower_stop_pun))
print(review_lower_stop_pun[:30])
```

```
21422541
['samsung', 'galaxy', 's8', 'probably', 'looks', 'exactly', 'like',
'ift.tt/2lyvu1g', 'latest', 'leak', "'confirms", 'galaxy', 's8', "'s
", 'aggressive', 'approach', 'goo.gl/fb/6awphu', 'another', 'small',
'teaser', 'iphone', 'iphone8', 'iphone8edge', 'iphone8curve', 'pic.t
witter.com/tra2sjijom', 'galaxy', 's8', 'leaks', 'mound', 'ahead']
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```

```

Computing NMF for Topic Modelling

In [57]:

```
#Computing NMF
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation, NMF
```

In [58]:

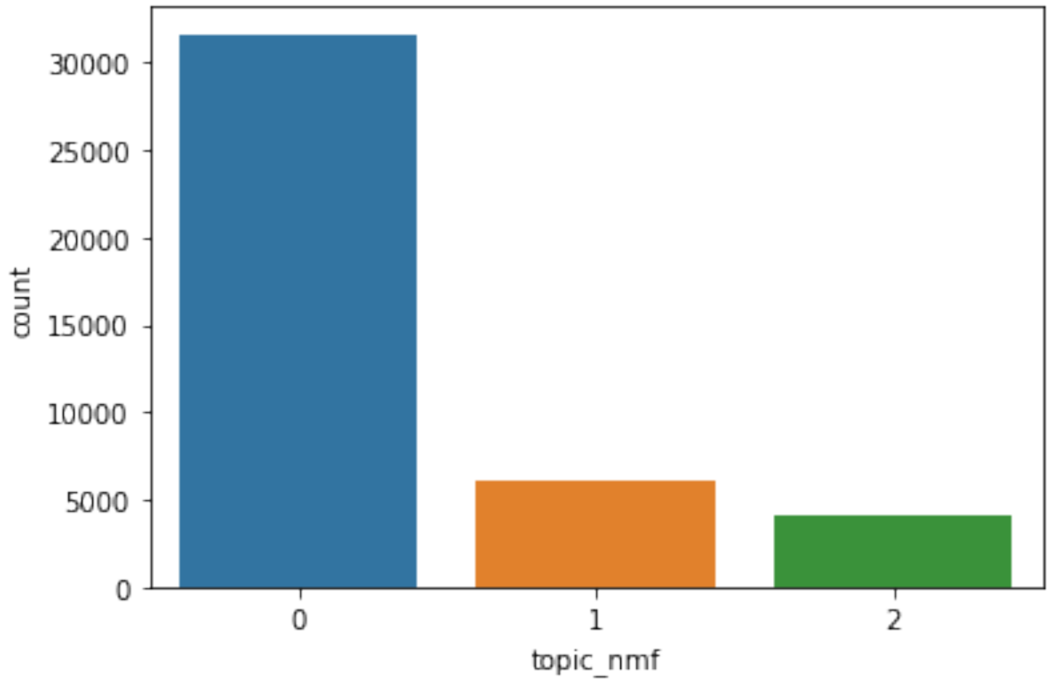
```
### NMF Topic Modeling

# The normalized corpus is then fed into a Term Frequency Vectorizer or Tf-idf v
ectorizer depending on the algorithm.
# Topic modeling is performed using NMF

# Non-Negative Matrix Factorization (NMF): The goal of NMF is to find two non-ne
gative matrices (W, H)
# whose product approximates the non- negative matrix X.
# This factorization can be used for example for dimensionality reduction, sourc
e separation or topic extraction.
#We will be using sklearn’s implementation of NMF.


def tfidf_nmf_function (dataframe, no_top_words, number) :
# Store only text contents
    data_text = dataframe[['Sound Bite Text']]
    data_text['index'] = dataframe.index
    # Assign to 'documents' which has texts and index of each
    documents = data_text
    ## Vectorization
    # NMF is able to use tf-idf - vectorize the corpus
    tfidf_vectorizer = TfidfVectorizer(max_df=0.9, min_df=10, max_features=10000
00
                                   , stop_words='english', token_pattern = r'\b[
^\d\W]+\b')
    # calculate the feature matrix
    tfidf = tfidf_vectorizer.fit_transform(dataframe['Sound Bite Text'])
    tfidf_feature_names = tfidf_vectorizer.get_feature_names()


    print ( "in the corpus of N documents, total of N unique features :")
    display(tfidf.shape)
    tfidf_feature_names = tfidf_vectorizer.get_feature_names()
    print("Length of unique features are : ", len(tfidf_vectorizer.get_feature_n
ames()))
```

```
    # Run NMF

    nmf = NMF(n_components=number, random_state=1, alpha=.1, l1_ratio=.5, init='
nndsvd')
    nmf_z = nmf.fit_transform(tfidf)

    for topic_idx, topic in enumerate(nmf.components_):
            print("Topic %d:" % (topic_idx))
            print(", ".join([tfidf_feature_names[i]
                        for i in topic.argsort()[:-no_top_words - 1:-1]]))

    TopicNumber=[]
    for i in range(len(nmf_z)):
        h=nmf_z[i].tolist().index(nmf_z[i].max())
        TopicNumber.append(h)
    documents['topic_nmf']=TopicNumber
    sns.countplot(x='topic_nmf', data=documents)



def getTermsAndSizes(topic_display_list_item):
    terms = []
    sizes = []
    for term, size in topic_display_list_item:
        terms.append(term)
        sizes.append(size)
    return terms, sizes
```

In [59]:

```
#iphone X _ output
#def nmf_function (tfidf, model, feature_names, no_top_words) :
#nmf_function (tfidf,nmf, tfidf_feature_names, 20)
#visualization topic distribution
tfidf_nmf_function(df_iphone_X,10 ,3)
```

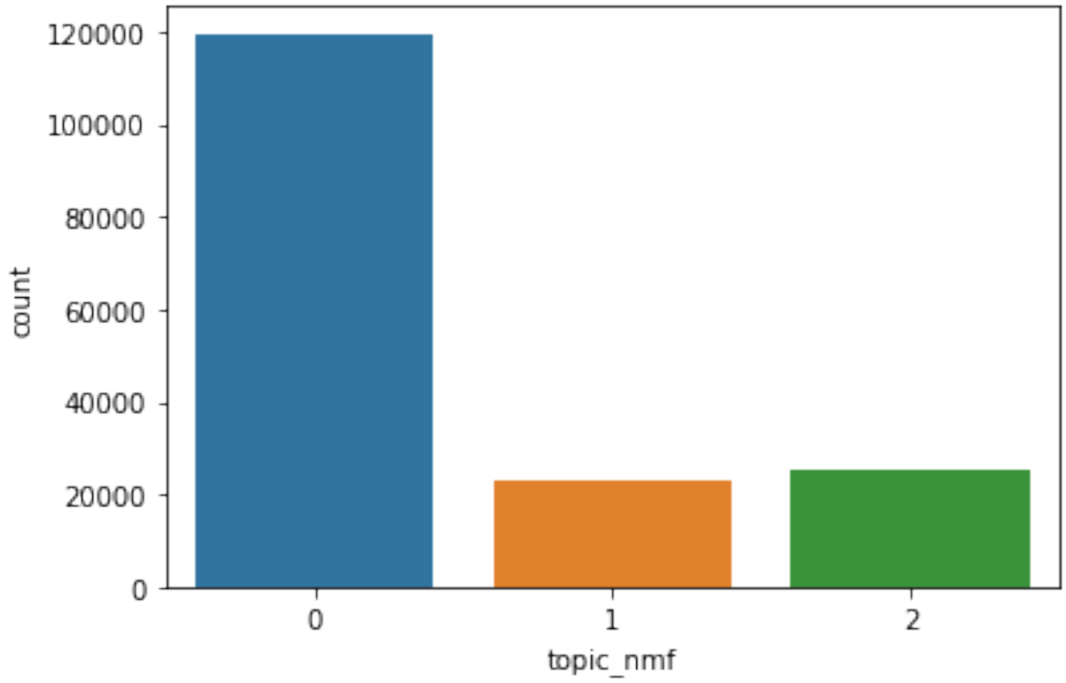
```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:15:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  from ipykernel import kernelapp as app

in the corpus of N documents, total of N unique features :

(41856, 12424)

Length of unique features are :  12424
Topic 0:
apple, s, new, plus, t, watch, phone, camera, screen, features
Topic 1:
galaxy, samsung, pixel, google, note, s, xl, vs, android, phones
Topic 2:
charging, wireless, qi, charge, charger, support, fast, pad, standar
d, pads


/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:45:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```



In [60]:

```
#iphone_8 output
tfidf_nmf_function(df_iphone_8,14,3)
```

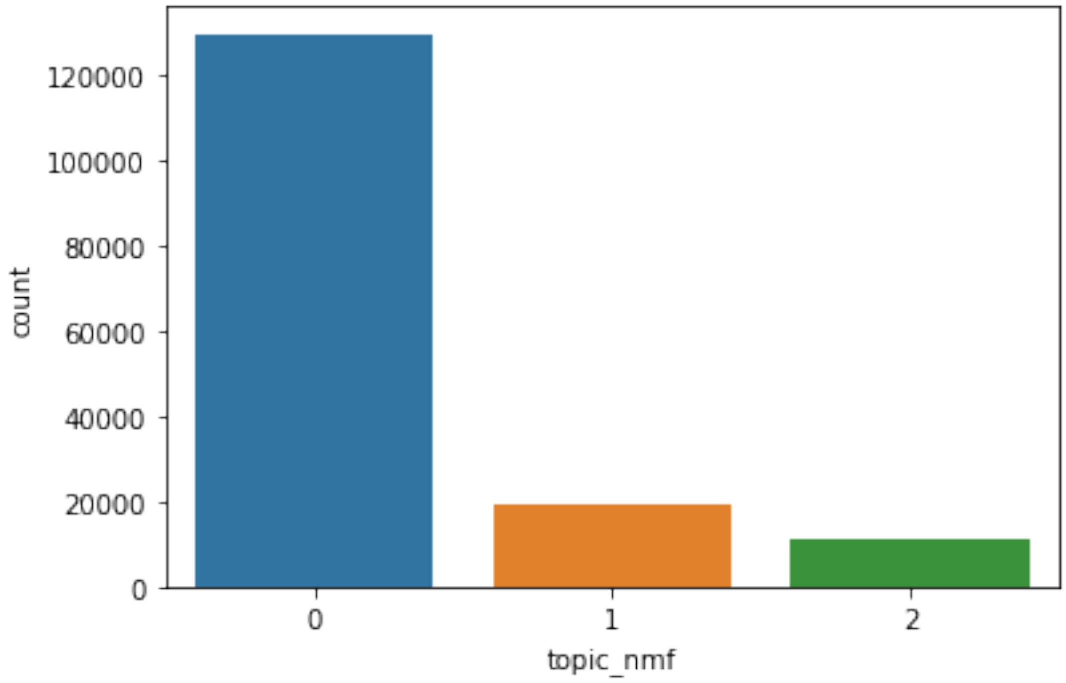
```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:15:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  from ipykernel import kernelapp as app

in the corpus of N documents, total of N unique features :

(167632, 21789)

Length of unique features are :  21789
Topic 0:
apple, plus, x, s, new, charging, wireless, t, phone, watch, ios, ye
ar, screen, just
Topic 1:
https, twitter, com, http, tags, t, ifttt, ift, tt, pic, ly, www, ne
ws, cnet
Topic 2:
galaxy, samsung, pixel, note, s, google, camera, phone, vs, xl, disp
lay, android, smartphone, phones

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:45:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```



In [61]:

```
# galaxy output
tfidf_nmf_function(df_galaxy,12,3)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:15:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  from ipykernel import kernelapp as app

in the corpus of N documents, total of N unique features :

(160320, 19744)

Length of unique features are :  19744
Topic 0:
samsung, s, phone, new, plus, smartphone, screen, android, t, displa
y, note, camera
Topic 1:
twitter, https, com, t, tags, http, ifttt, android, pic, rt, tech, m
artinguayott
Topic 2:
bixby, assistant, voice, button, s, samsung, launch, google, ai, vir
tual, siri, home

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:45:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```



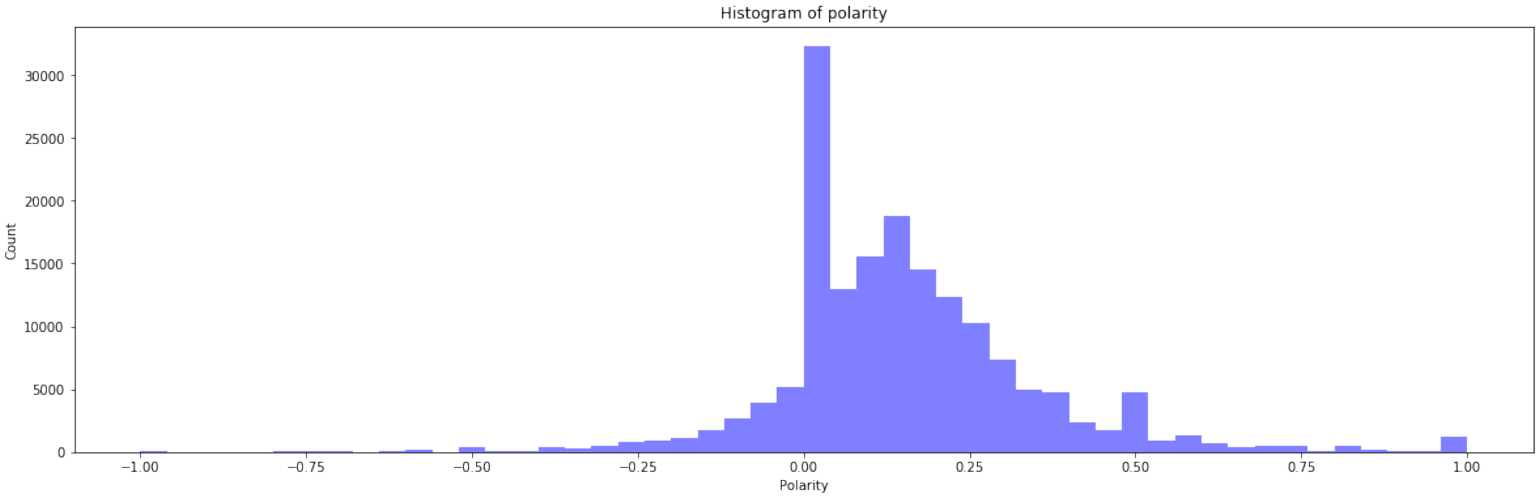
Starting Sentiment to get Top Liked and Dis-liked Attributes

In [62]:

```
from textblob import TextBlob
normal_blob_8 = pd.DataFrame(df_iphone_8)

def detect_polarity(text):
    return TextBlob(text).sentiment.polarity

normal_blob_8['polarity'] = df_iphone_8['Sound Bite Text'].apply(detect_polarity
)
```

In [63]:

```
num_bins = 50
plt.figure(figsize=(20,6))
n, bins, patches = plt.hist(normal_blob_8.polarity, num_bins, facecolor='blue',
alpha=0.5)
plt.xlabel('Polarity')
plt.ylabel('Count')
plt.title('Histogram of polarity')
plt.show()
```



In [64]:

```
normal_negative_8 = normal_blob_8[normal_blob_8['polarity']<0]
normal_negative_8['Group'] = "Negative"

normal_positive_8 = normal_blob_8[normal_blob_8['polarity']>0.1]
normal_positive_8['Group'] = "Positive"

normal_neutral_8 = normal_blob_8[(normal_blob_8['polarity'] >=0) & (normal_blob_
8['polarity']<0.1)]
normal_neutral_8['Group'] = "Netural"

grouping_8 = [normal_negative_8,normal_positive_8,normal_neutral_8]
normal_groups_8 = pd.concat(grouping_8)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  """
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```

In [65]:

```
normal_blob_X = pd.DataFrame(df_iphone_X)
normal_blob_X['polarity'] = df_iphone_X['Sound Bite Text'].apply(detect_polarity
)

normal_negative_X = normal_blob_X[normal_blob_X['polarity']<0]
normal_negative_X['Group'] = "Negative"

normal_positive_X = normal_blob_X[normal_blob_X['polarity']>0.1]
normal_positive_X['Group'] = "Positive"

normal_neutral_X = normal_blob_X[(normal_blob_X['polarity'] >=0) & (normal_blob_
X['polarity']<0.1)]
normal_neutral_X['Group'] = "Netural"

grouping_X = [normal_negative_X,normal_positive_X,normal_neutral_X]
normal_groups_X = pd.concat(grouping_X)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  """
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:11:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  # This is added back by InteractiveShellApp.init_path()
```

In [66]:

```
normal_blob_G = pd.DataFrame(df_galaxy)
normal_blob_G['polarity'] = df_galaxy['Sound Bite Text'].apply(detect_polarity)

normal_negative_G = normal_blob_G[normal_blob_G['polarity']<0]
normal_negative_G['Group'] = "Negative"

normal_positive_G = normal_blob_G[normal_blob_G['polarity']>0.1]
normal_positive_G['Group'] = "Positive"

normal_neutral_G = normal_blob_G[(normal_blob_G['polarity'] >=0) & (normal_blob_
G['polarity']<0.1)]
normal_neutral_G['Group'] = "Netural"

grouping_G = [normal_negative_G,normal_positive_G,normal_neutral_G]
normal_groups_G = pd.concat(grouping_G)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  """
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:11:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
  # This is added back by InteractiveShellApp.init_path()
```

In [67]:

```
#Distributing the dataframes into Positive, Negative and Neutral Depening on the
polarity
```

In [69]:

```
df_iphone_8_negative = pd.DataFrame(normal_groups_8[normal_groups_8['Group'] == 'Negative'])
df_iphone_8_positive = pd.DataFrame(normal_groups_8[normal_groups_8['Group'] == 'Positive'])
df_iphone_8_netural = pd.DataFrame(normal_groups_8[normal_groups_8['Group'] == 'Netural'])


df = df_iphone_8_negative
stop = stopwords.words('english')
df['Sound Bite Text'].apply(lambda x: x.lower())
df['tokenize'] = df['Sound Bite Text'].apply(word_tokenize)
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [word for word in x if word.isalpha()])
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [lemmatizer.lemmatize(y) for y in x])
#stemmer = PorterStemmer()
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [stemmer.stem(y) for y in x])
df['tokenize'] = df['tokenize'].apply(lambda x: [item.lower() for item in x])
df['sentence'] = df['tokenize'].apply(' '.join)
df['attr'] = df['tokenize'].apply(lambda x: nltk.pos_tag(x))
att = []
for values in df['attr']:
    for pair in values :
        if(pair[1] == 'NN' or pair[1] == 'NNS' or pair[1] == 'NNP' or pair[1]=='NNPS'):
            att.append(pair[0])
        if(pair[1] == 'JJ' or pair[1] == 'JJS' or pair[1] == 'JJR'):
            att.append(pair[0])
print("Most common negative attributes")
Counter(att).most_common(25)
```

```
Most common negative attributes
```

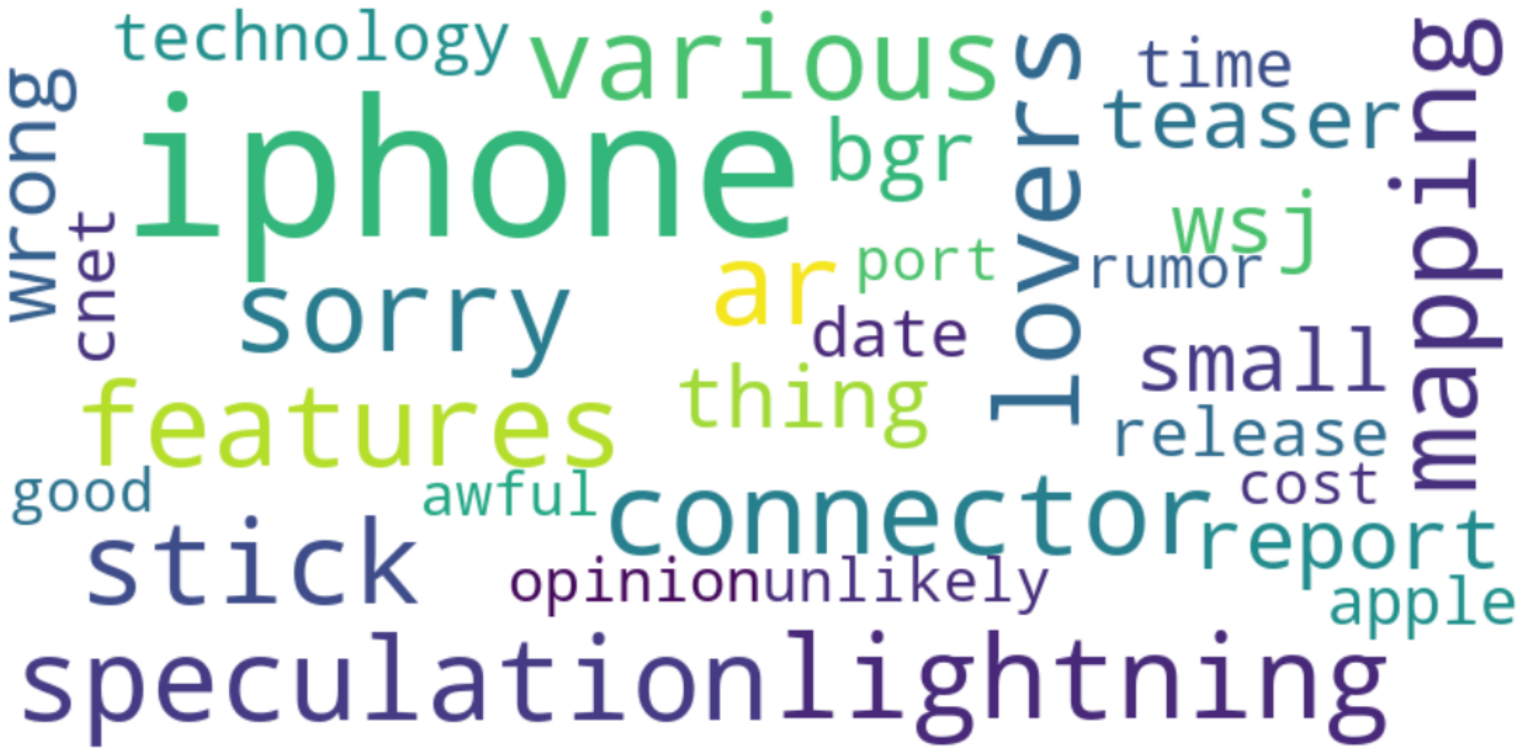
```
Out[69]:
```

```
[('iphone', 42364),
 ('apple', 21523),
 ('new', 6926),
 ('x', 6109),
 ('phone', 4535),
 ('irrelevant', 4305),
 ('flag', 3796),
 ('ios', 2580),
 ('year', 2445),
 ('samsung', 2413),
 ('news', 2385),
 ('http', 2342),
 ('galaxy', 2269),
 ('battery', 2149),
 ('screen', 2025),
 ('camera', 2017),
 ('https', 1927),
 ('tags', 1907),
 ('launch', 1854),
 ('id', 1773),
 ('time', 1724),
 ('case', 1697),
 ('display', 1675),
 ('device', 1641),
 ('design', 1612)]
```

In [70]:

```
NN_jt_text =(" ").join(att[:50])
type(NN_jt_text)

print("Top Dis-liked attributes")
# top 200 active listing noun word cloud
# lower max_font_size, change the maximum number of word and lighten the background:
wc_jt = WordCloud(width=800, height=400, stopwords=stop, max_font_size=100, max_words=200, background_color="white").\
generate(NN_jt_text)
#stopwords:set of strings or None
#The words that will be eliminated. If None, the build-in STOPWORDS list will be used. Ignored if using generate_from_frequencies.

plt.figure( figsize=(20,10) )
plt.imshow(wc_jt, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
Top Dis-liked attributes
```



In [73]:

```
df = df_iphone_8_positive
stop = stopwords.words('english')
df['Sound Bite Text'].apply(lambda x: x.lower())
df['tokenize'] = df['Sound Bite Text'].apply(word_tokenize)
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [word for word in x if word.isalpha()])
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [lemmatizer.lemmatize(y) for y in x])
#stemmer = PorterStemmer()
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [stemmer.stem(y) for y in x])
df['tokenize'] = df['tokenize'].apply(lambda x: [item.lower() for item in x])
df['sentence'] = df['tokenize'].apply(' '.join)
df['attr'] = df['tokenize'].apply(lambda x: nltk.pos_tag(x))
att = []
for values in df['attr']:
    for pair in values :
        if(pair[1] == 'NN' or pair[1] == 'NNS' or pair[1] == 'NNP' or pair[1]=='NNPS'):
            att.append(pair[0])
        if(pair[1] == 'JJ' or pair[1] == 'JJS' or pair[1] == 'JJR'):
            att.append(pair[0])
print("Most common liked attributes")
Counter(att).most_common(25)
```

```
Most common liked attributes
```

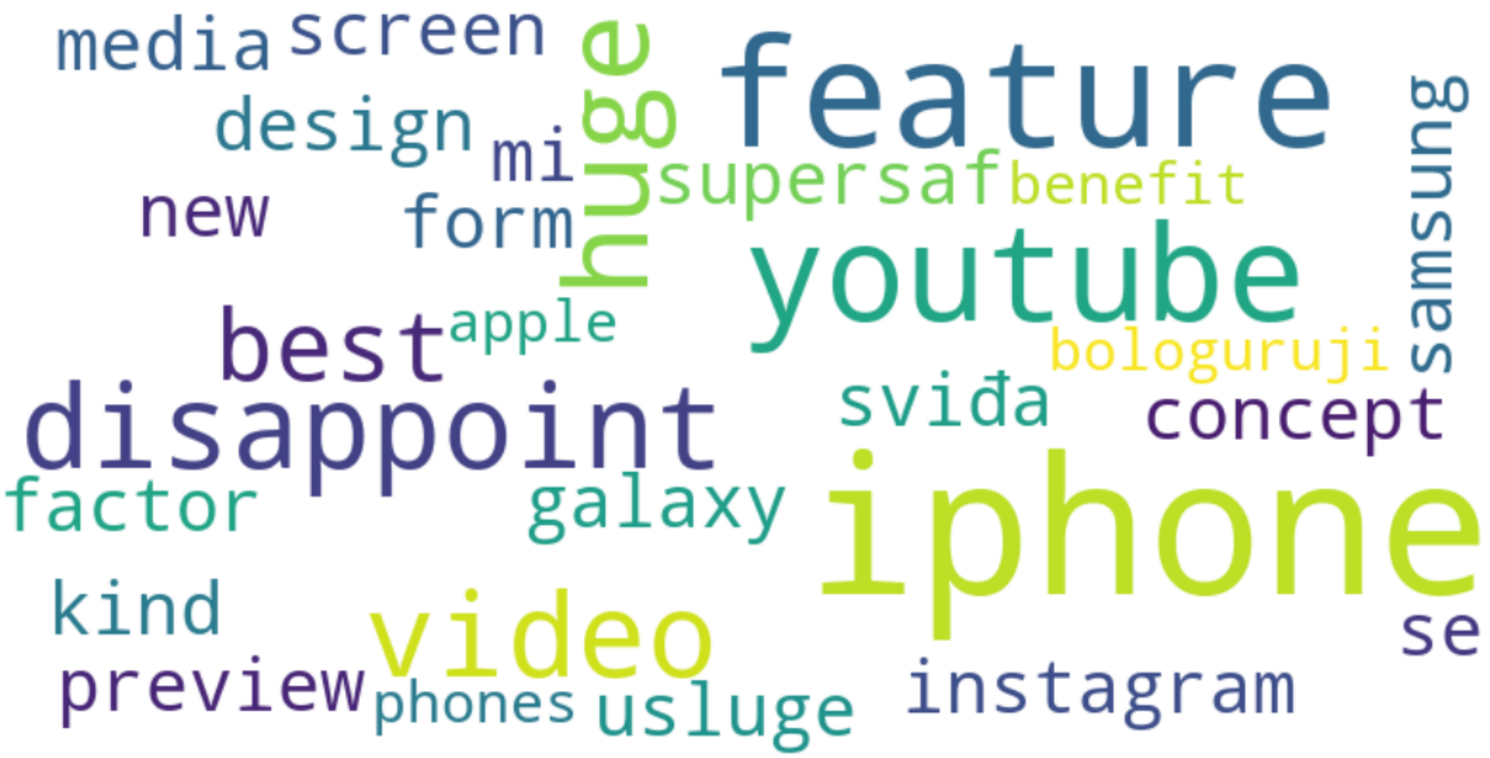
Out[73]:

```
[('iphone', 362634),
 ('apple', 147387),
 ('new', 92016),
 ('x', 62771),
 ('phone', 38827),
 ('galaxy', 37739),
 ('camera', 31567),
 ('samsung', 29270),
 ('wireless', 22004),
 ('screen', 20993),
 ('display', 20771),
 ('design', 19764),
 ('year', 18647),
 ('phones', 16960),
 ('pixel', 16827),
 ('smartphone', 16403),
 ('note', 15997),
 ('device', 15586),
 ('features', 15103),
 ('ios', 15062),
 ('best', 14936),
 ('time', 14894),
 ('case', 14520),
 ('latest', 13846),
 ('news', 13694)]
```

In [74]:

```
NN_jt_text =(" ").join(att[:50])
type(NN_jt_text)

print("Top Liked attributes")
# top 200 active listing noun word cloud
# lower max_font_size, change the maximum number of word and lighten the backgro
und:
wc_jt = WordCloud(width=800, height=400, stopwords=stop, max_font_size=100, max_
words=200, background_color="white").\
generate(NN_jt_text)
#stopwords:set of strings or None
#The words that will be eliminated. If None, the build-in STOPWORDS list will be
used. Ignored if using generate_from_frequencies.

plt.figure( figsize=(20,10) )
plt.imshow(wc_jt, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
Top Liked attributes
```



In [75]:

```
df = df_iphone_8_netural
stop = stopwords.words('english')
df['Sound Bite Text'].apply(lambda x: x.lower())
df['tokenize'] = df['Sound Bite Text'].apply(word_tokenize)
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [item for item in x if item not in stop])
df['tokenize'] = df['tokenize'].apply(lambda x: [word for word in x if word.isalpha()])
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [lemmatizer.lemmatize(y) for y in x])
#stemmer = PorterStemmer()
#df['tokenized_text'] = df['tokenized_text'].apply(lambda x: [stemmer.stem(y) for y in x])
df['tokenize'] = df['tokenize'].apply(lambda x: [item.lower() for item in x])
df['sentence'] = df['tokenize'].apply(' '.join)
df['attr'] = df['tokenize'].apply(lambda x: nltk.pos_tag(x))
att = []
for values in df['attr']:
    for pair in values :
        if(pair[1] == 'NN' or pair[1] == 'NNS' or pair[1] == 'NNP' or pair[1]=='NNPS'):
            att.append(pair[0])
        if(pair[1] == 'JJ' or pair[1] == 'JJS' or pair[1] == 'JJR'):
            att.append(pair[0])
print("Most common attributes")
Counter(att).most_common(25)
```

```
Most common attributes
```

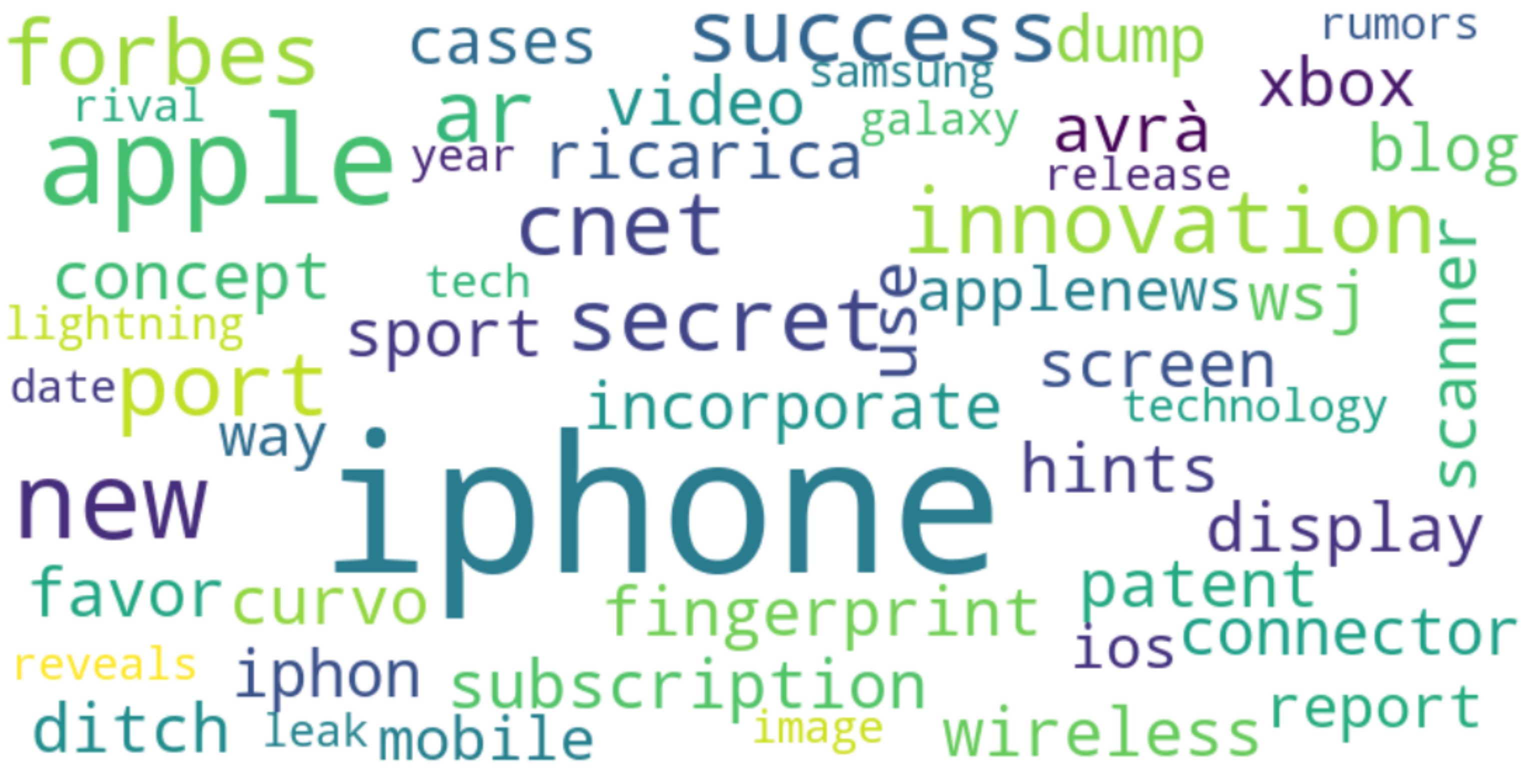
Out[75]:

```
[('iphone', 155181),
 ('apple', 72532),
 ('new', 31587),
 ('x', 22051),
 ('phone', 15006),
 ('galaxy', 11341),
 ('samsung', 11288),
 ('year', 9736),
 ('screen', 9469),
 ('display', 9345),
 ('camera', 9106),
 ('wireless', 7972),
 ('design', 7935),
 ('device', 7675),
 ('http', 7289),
 ('news', 6745),
 ('launch', 6642),
 ('ios', 6584),
 ('company', 6567),
 ('smartphone', 6454),
 ('https', 6445),
 ('time', 6400),
 ('technology', 6363),
 ('tags', 6131),
 ('touch', 6044)]
```

In [76]:

```
NN_jt_text =(" ").join(att[:80])
type(NN_jt_text)

print("Top Netural attributes")
# top 200 active listing noun word cloud
# lower max_font_size, change the maximum number of word and lighten the backgro
und:
wc_jt = WordCloud(width=800, height=400, stopwords=stop, max_font_size=100, max_
words=200, background_color="white").\
generate(NN_jt_text)
#stopwords:set of strings or None
#The words that will be eliminated. If None, the build-in STOPWORDS list will be
used. Ignored if using generate_from_frequencies.

plt.figure( figsize=(20,10) )
plt.imshow(wc_jt, interpolation="bilinear")
plt.axis("off")
plt.show()
```

```
Top Netural attributes
```



In [ ]:

In [ ]:

In [ ]:

How did Customers feel about Quality, Price and Value

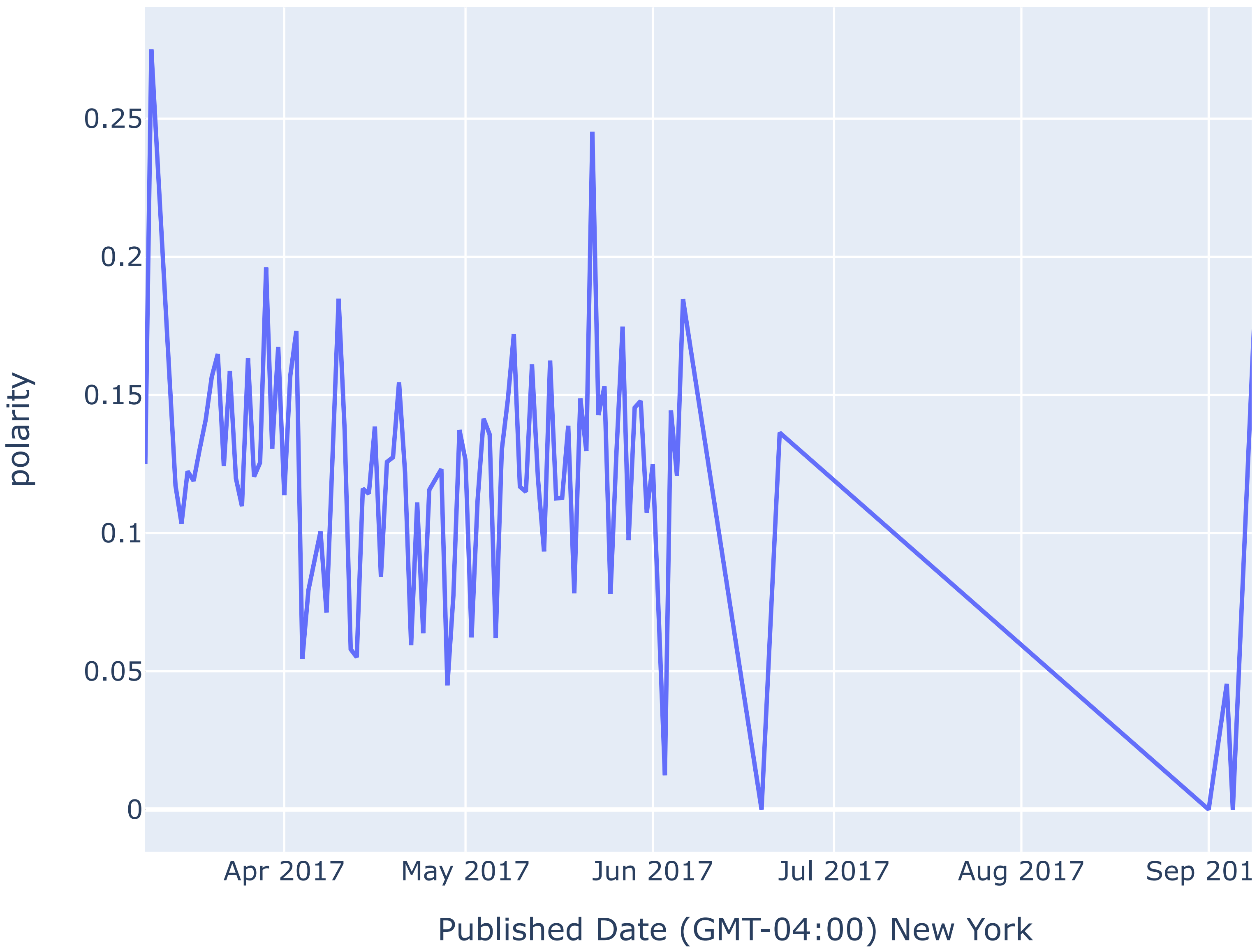
In [1]:

```
#On the PQV File
```

Time Series to show user's Sentiment before and after the release of the products

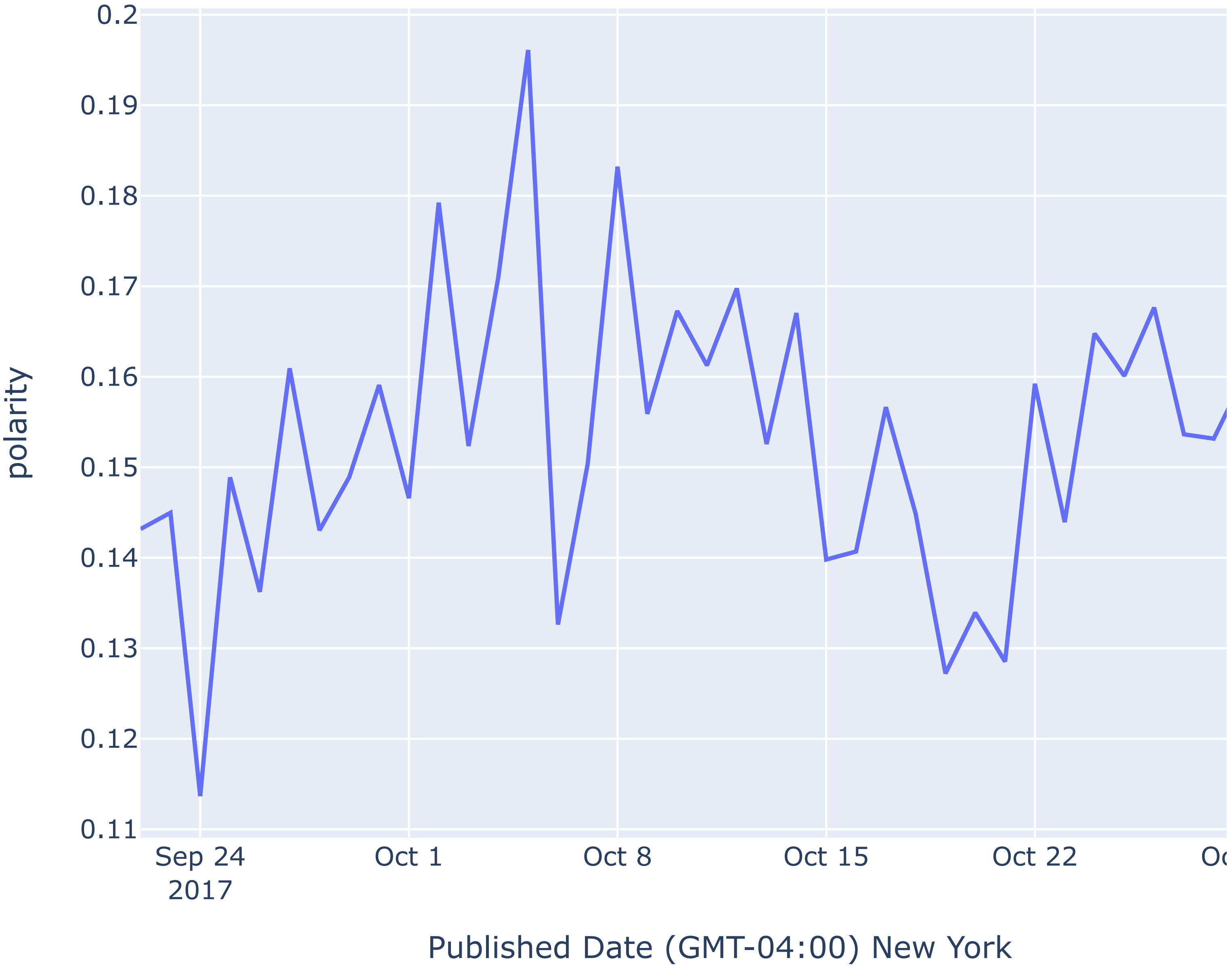
In [78]:

```
#iPhone X
import plotly.express as px
normal_before = normal_groups_X[normal_groups_X['Before/After']=='Before']
#normal_negative_before['Before/After'].str.contains("After")

plotting_before = normal_before.groupby(['Published Date (GMT-04:00) New York'])
['polarity'].mean()
#positive_plotting.head()

before=pd.DataFrame({'Published Date (GMT-04:00) New York':plotting_before.index
, 'polarity':plotting_before.values})

fig = px.line(before, x='Published Date (GMT-04:00) New York', y='polarity')
fig.show()
```


polarity
0.25
0.2
0.15
0.1
0.05
0
Apr 2017
May 2017
Jun 2017
Jul 2017
Aug 2017
Sep 201
Published Date (GMT-04:00) New York

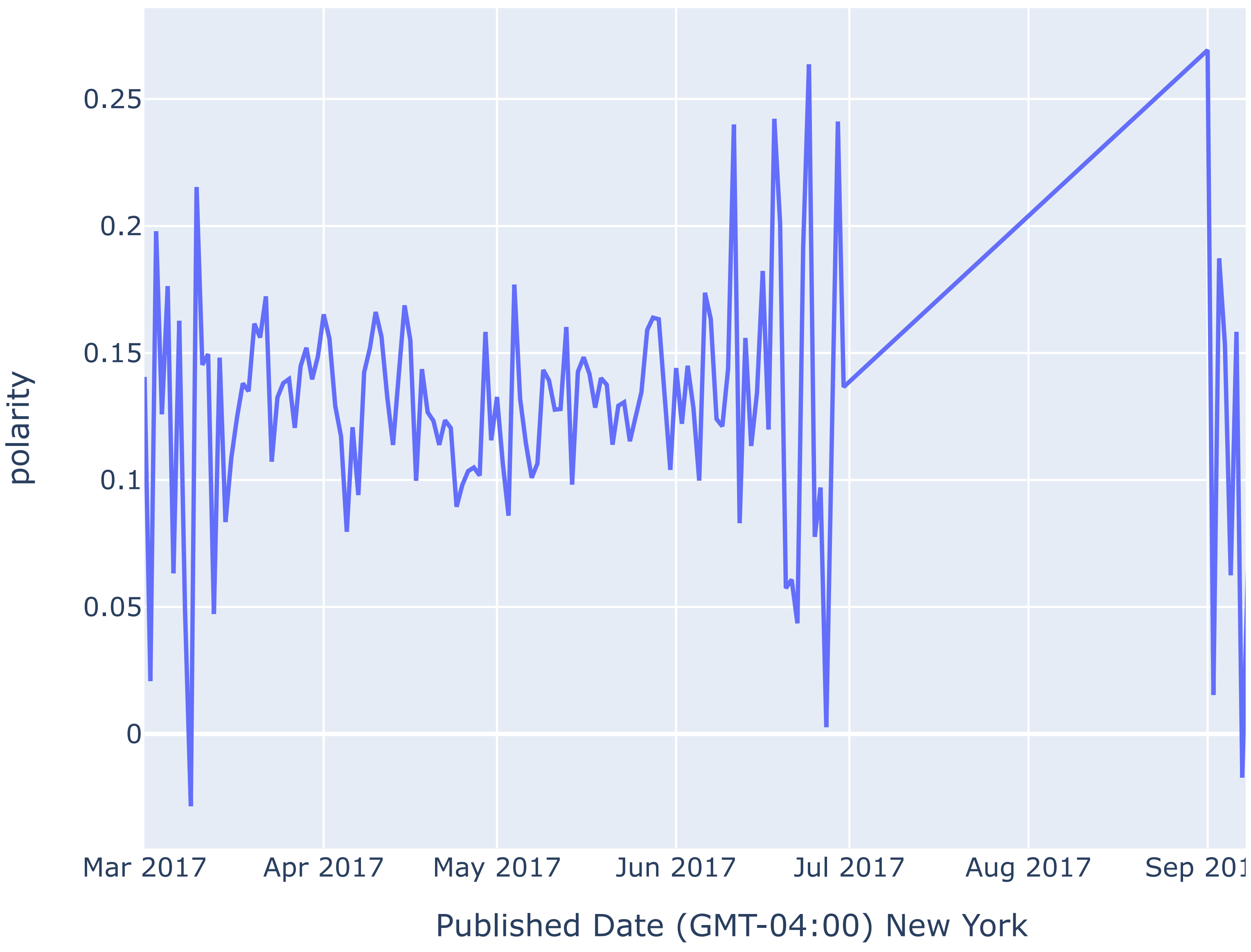
In [79]:

```
normal_after = normal_groups_X[normal_groups_X['Before/After']=='After']
#normal_negative_before['Before/After'].str.contains("After")

plotting_after = normal_after.groupby(['Published Date (GMT-04:00) New York'])['
polarity'].mean()
#positive_plotting.head()

after=pd.DataFrame({'Published Date (GMT-04:00) New York':plotting_after.index,
'polarity':plotting_after.values})

fig = px.line(after, x='Published Date (GMT-04:00) New York', y='polarity')
fig.show()
```



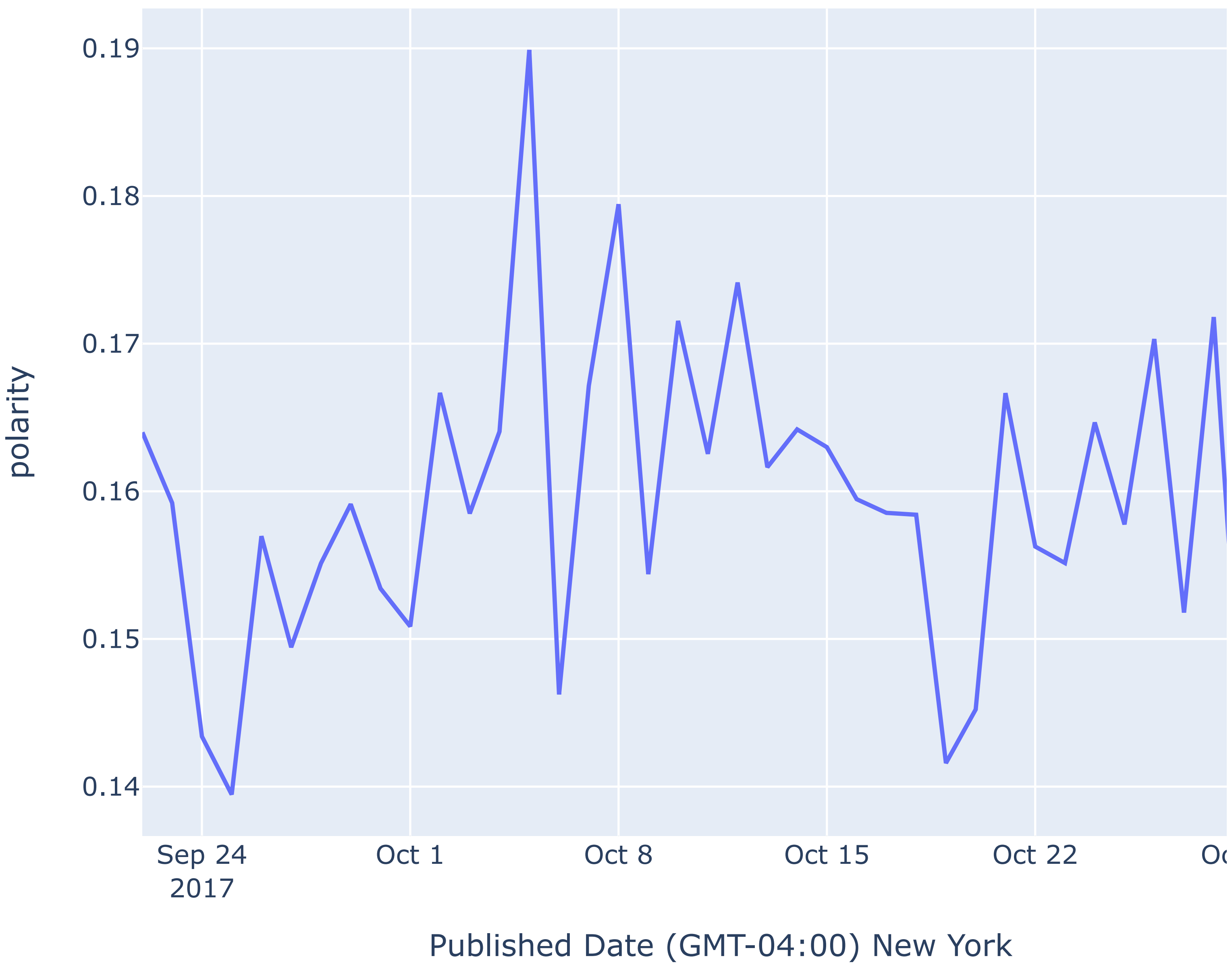
In [80]:

```
normal_before = normal_blob_8[normal_blob_8['Before/After']=='Before']
#normal_negative_before['Before/After'].str.contains("After")

plotting_before = normal_before.groupby(['Published Date (GMT-04:00) New York'])
['polarity'].mean()
#positive_plotting.head()

before=pd.DataFrame({'Published Date (GMT-04:00) New York':plotting_before.index
, 'polarity':plotting_before.values})

fig = px.line(before, x='Published Date (GMT-04:00) New York', y='polarity')
fig.show()
```



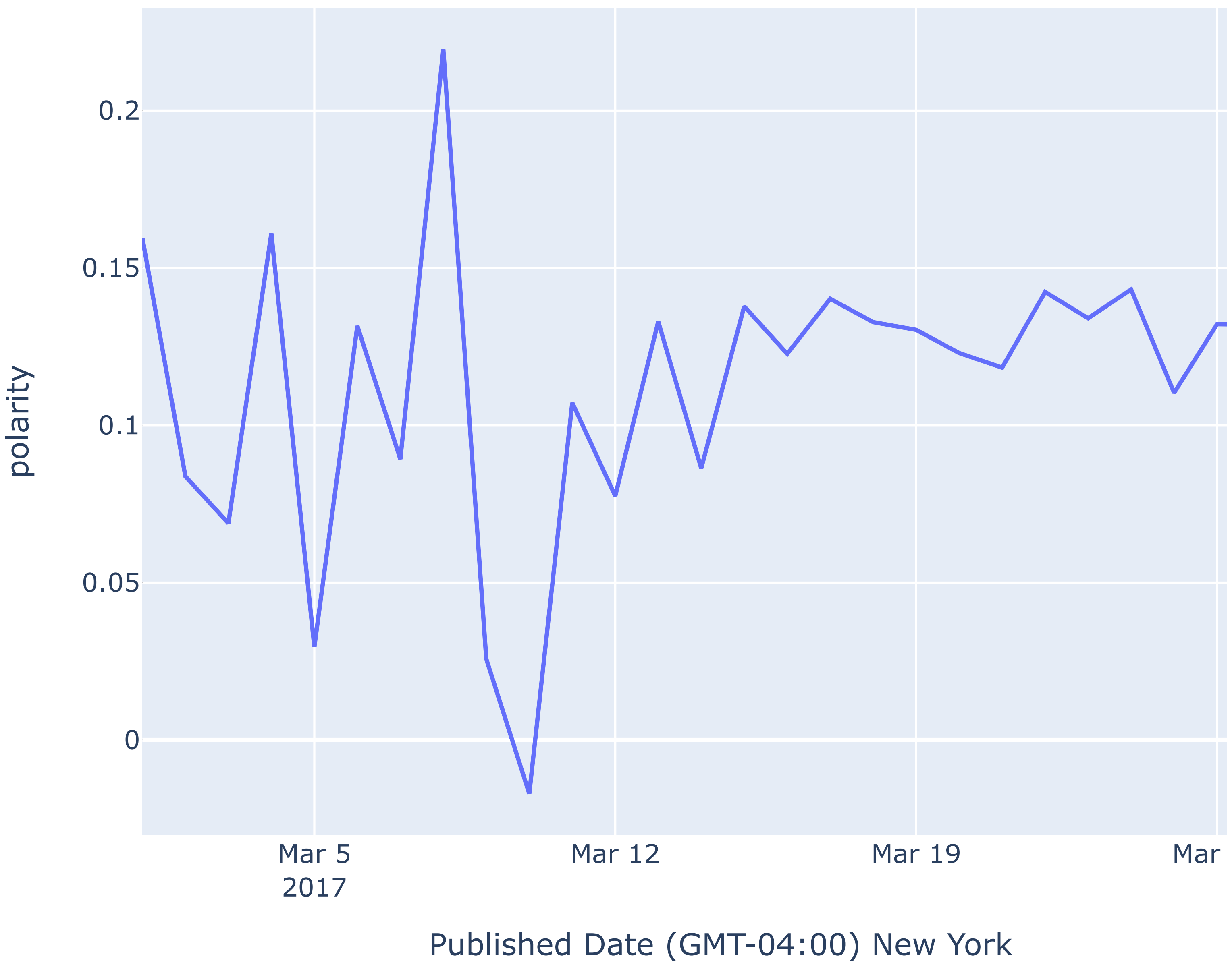
In [81]:

```
normal_after = normal_blob_8[normal_blob_8['Before/After']=='After']
#normal_negative_before['Before/After'].str.contains("After")

plotting_after = normal_after.groupby(['Published Date (GMT-04:00) New York'])['
polarity'].mean()
#positive_plotting.head()

after=pd.DataFrame({'Published Date (GMT-04:00) New York':plotting_after.index,
'polarity':plotting_after.values})

fig = px.line(after, x='Published Date (GMT-04:00) New York', y='polarity')
fig.show()
```



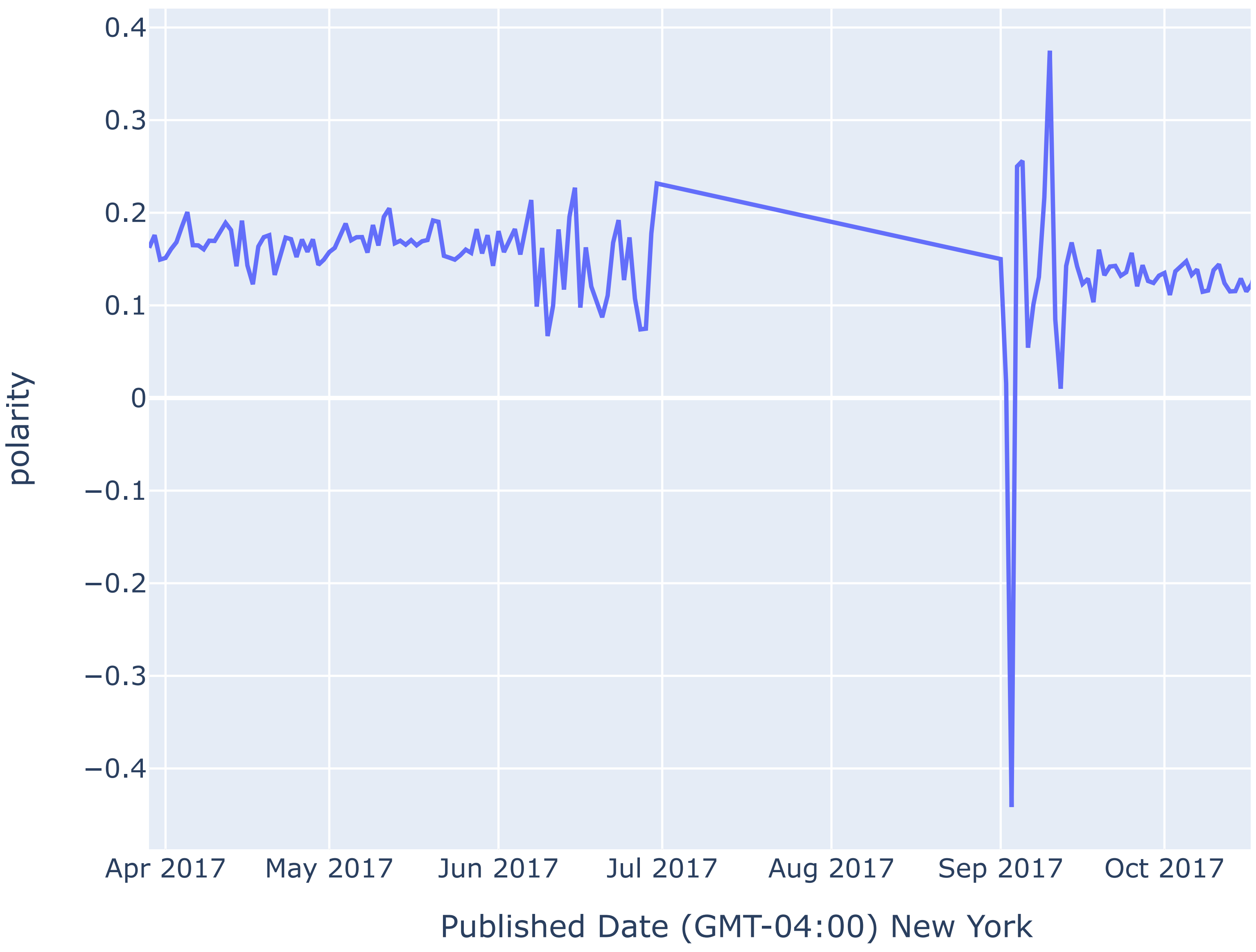
In [82]:

```
normal_before = normal_groups_G[normal_groups_G['Before/After']=='Before']
#normal_negative_before['Before/After'].str.contains("After")

plotting_before = normal_before.groupby(['Published Date (GMT-04:00) New York'])
['polarity'].mean()
#positive_plotting.head()

before=pd.DataFrame({'Published Date (GMT-04:00) New York':plotting_before.index
, 'polarity':plotting_before.values})

fig = px.line(before, x='Published Date (GMT-04:00) New York', y='polarity')
fig.show()
```



In [83]:

```
normal_after = normal_groups_G[normal_groups_G['Before/After']=='After']
#normal_negative_before['Before/After'].str.contains("After")

plotting_after = normal_after.groupby(['Published Date (GMT-04:00) New York'])['
polarity'].mean()
#positive_plotting.head()

after=pd.DataFrame({'Published Date (GMT-04:00) New York':plotting_after.index,
'polarity':plotting_after.values})

fig = px.line(after, x='Published Date (GMT-04:00) New York', y='polarity')
fig.show()
```



Predicting the uptake or adoption of the products

In [84]:

```
## Time Series - Moving average
```

In [85]:

```
#from textblob import TextBlob
#df_galaxy.head()
```

In [86]:

```
normal_blob = pd.DataFrame(df_galaxy)

#def detect_polarity(text):
#    return TextBlob(text).sentiment.polarity

#def translate_english(text):
   # return TextBlob(text).translate(to='en')

#normal_blob['Sound Bite Translated'] = normal_df['Sound Bite Text'].apply(translate_english)

#normal_blob['polarity'] = df_galaxy['Sound Bite Text'].apply(detect_polarity)


normal_positive = normal_blob[normal_blob['polarity']>0.1]
normal_positive['Group'] = "Positive"


plotting_before = normal_positive.groupby(['Published Date (GMT-04:00) New York'])['polarity'].mean()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:15:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```

In [87]:

```
galaxy_timeseries = pd.DataFrame(plotting_before).reset_index()
galaxy_timeseries.head(3)
```

Out[87]:

	Published Date (GMT-04:00) New York	polarity
0	2017-03-01	0.310536
1	2017-03-02	0.330682
2	2017-03-03	0.287879

In [88]:

```
galaxy_timeseries['MA'] = galaxy_timeseries['polarity'].rolling(window=5,center=
False).mean()
```

In [89]:

```
galaxy_timeseries
```

Out[89]:

	Published Date (GMT-04:00) New York	polarity	MA
0	2017-03-01	0.310536	NaN
1	2017-03-02	0.330682	NaN
2	2017-03-03	0.287879	NaN
3	2017-03-04	0.392172	NaN
4	2017-03-05	0.218182	0.307890
5	2017-03-06	0.293944	0.304572
6	2017-03-07	0.330556	0.304546
7	2017-03-08	0.414744	0.329919
8	2017-03-09	0.187205	0.288926
9	2017-03-10	0.201515	0.285593
10	2017-03-11	0.241035	0.275011
11	2017-03-12	0.315584	0.272017
12	2017-03-13	0.307102	0.250489
13	2017-03-14	0.221039	0.257255
14	2017-03-15	0.234030	0.263758
15	2017-03-16	0.231675	0.261886
16	2017-03-17	0.240229	0.246815

17	2017-03-18	0.251468	0.235688
18	2017-03-19	0.224011	0.236282
19	2017-03-20	0.216300	0.232737
20	2017-03-21	0.211343	0.228670
21	2017-03-22	0.236616	0.227947
22	2017-03-23	0.227806	0.223215
23	2017-03-24	0.277342	0.233881
24	2017-03-25	0.212585	0.233138
25	2017-03-26	0.256653	0.242200
26	2017-03-27	0.244301	0.243737
27	2017-03-28	0.246786	0.247533
28	2017-03-29	0.252248	0.242515
29	2017-03-30	0.266959	0.253390
30	2017-03-31	0.266753	0.255410
31	2017-04-01	0.260166	0.258583
32	2017-04-02	0.275248	0.264275
33	2017-04-03	0.265187	0.266863
34	2017-04-04	0.280572	0.269585
35	2017-04-05	0.292805	0.274796
36	2017-04-06	0.258510	0.274464
37	2017-04-07	0.254165	0.270248
38	2017-04-08	0.261247	0.269460
39	2017-04-09	0.266961	0.266738
40	2017-04-10	0.269168	0.262010
41	2017-04-11	0.273920	0.265092
42	2017-04-12	0.283524	0.270964
43	2017-04-13	0.280800	0.274875
44	2017-04-14	0.255527	0.272588
45	2017-04-15	0.305753	0.279905
46	2017-04-16	0.278183	0.280757
47	2017-04-17	0.256467	0.275346
48	2017-04-18	0.285575	0.276301
49	2017-04-19	0.286453	0.282486
50	2017-04-20	0.295996	0.280535
51	2017-04-21	0.296326	0.284163

52	2017-04-22	0.288633	0.290596
53	2017-04-23	0.306609	0.294803
54	2017-04-24	0.297099	0.296933
55	2017-04-25	0.273895	0.292512
56	2017-04-26	0.303774	0.294002
57	2017-04-27	0.276170	0.291509
58	2017-04-28	0.299734	0.290134
59	2017-04-29	0.304475	0.291609
60	2017-04-30	0.297218	0.296274
61	2017-05-01	0.280308	0.291581
62	2017-05-02	0.303593	0.297066
63	2017-05-03	0.298991	0.296917
64	2017-05-04	0.289376	0.293897
65	2017-05-05	0.302817	0.295017
66	2017-05-06	0.308979	0.300751
67	2017-05-07	0.321553	0.304343
68	2017-05-08	0.290013	0.302548
69	2017-05-09	0.316481	0.307969
70	2017-05-10	0.300882	0.307582
71	2017-05-11	0.305872	0.306960
72	2017-05-12	0.313543	0.305358
73	2017-05-13	0.311202	0.309596
74	2017-05-14	0.310671	0.308434
75	2017-05-15	0.308755	0.310009
76	2017-05-16	0.288788	0.306592
77	2017-05-17	0.281128	0.300109
78	2017-05-18	0.291714	0.296211
79	2017-05-19	0.305932	0.295264
80	2017-05-20	0.330519	0.299616
81	2017-05-21	0.329594	0.307777
82	2017-05-22	0.300593	0.311670
83	2017-05-23	0.295018	0.312331
84	2017-05-24	0.286150	0.308375
85	2017-05-25	0.295443	0.301359
86	2017-05-26	0.295295	0.294500
87	2017-05-27	0.291964	0.292774

```
87                          2017-05-27  0.291964  0.292774
88                          2017-05-28  0.341749  0.302120
89                          2017-05-29  0.284936  0.301877
90                          2017-05-30  0.296157  0.302020
91                          2017-05-31  0.276479  0.298257
92                          2017-06-01  0.293778  0.298620
93                          2017-06-02  0.285941  0.287458
94                          2017-06-03  0.305853  0.291642
95                          2017-06-04  0.323952  0.297201
96                          2017-06-05  0.297537  0.301412
97                          2017-06-06  0.315481  0.305753
98                          2017-06-07  0.506657  0.349896
99                          2017-06-08  0.282474  0.345220
100                         2017-06-09  0.351566  0.350743
101                         2017-06-10  0.578788  0.406993
102                         2017-06-11  0.274405  0.398778
103                         2017-06-12  0.540625  0.405572
104                         2017-06-13  0.515341  0.452145
105                         2017-06-14  0.472338  0.476299
106                         2017-06-15  0.600000  0.480542
107                         2017-06-16  0.373686  0.500398
108                         2017-06-17  0.385065  0.469286
109                         2017-06-18  0.555000  0.477218
110                         2017-06-19  0.259758  0.434702
111                         2017-06-20  0.493636  0.413429
112                         2017-06-21  0.294413  0.397574
113                         2017-06-22  0.503030  0.421168
114                         2017-06-23  0.501042  0.410376
115                         2017-06-24  0.320833  0.422591
116                         2017-06-25  0.444129  0.412689
117                         2017-06-26  0.373154  0.428438
118                         2017-06-27  0.305177  0.388867
119                         2017-06-28  0.368123  0.362283
120                         2017-06-29  0.519110  0.401938
121                         2017-06-30  0.376406  0.388394
122                         2017-09-01  0.600000  0.403763
```

122	2017-09-01	0.600000	0.433763
123	2017-09-02	0.252399	0.423208
124	2017-09-04	1.000000	0.549583
125	2017-09-05	0.578788	0.561519
126	2017-09-06	0.223990	0.531035
127	2017-09-07	0.550000	0.521035
128	2017-09-08	0.425000	0.555556
129	2017-09-09	0.488889	0.453333
130	2017-09-10	0.750000	0.487576
131	2017-09-11	0.875000	0.617778
132	2017-09-12	0.275000	0.562778
133	2017-09-13	0.316236	0.541025
134	2017-09-14	0.331482	0.509544
135	2017-09-15	0.287539	0.417051
136	2017-09-16	0.285435	0.299138
137	2017-09-17	0.306680	0.305474
138	2017-09-18	0.242732	0.290774
139	2017-09-19	0.309559	0.286389
140	2017-09-20	0.287591	0.286399
141	2017-09-21	0.297743	0.288861
142	2017-09-22	0.314732	0.290471
143	2017-09-23	0.299703	0.301866
144	2017-09-24	0.321277	0.304209
145	2017-09-25	0.317436	0.310178
146	2017-09-26	0.275926	0.305815
147	2017-09-27	0.289868	0.300842
148	2017-09-28	0.296005	0.300102
149	2017-09-29	0.273679	0.290583
150	2017-09-30	0.313593	0.289814
151	2017-10-01	0.322214	0.299072
152	2017-10-02	0.267314	0.294561
153	2017-10-03	0.325774	0.300515
154	2017-10-04	0.273175	0.300414
155	2017-10-05	0.314300	0.300556
156	2017-10-06	0.344983	0.305109
157	2017-10-07	0.287556	0.309153

157	2017-10-07	0.287556	0.309158
158	2017-10-08	0.308766	0.305756
159	2017-10-09	0.297675	0.310656
160	2017-10-10	0.285932	0.304982
161	2017-10-11	0.314557	0.298897
162	2017-10-12	0.278820	0.297150
163	2017-10-13	0.289143	0.293225
164	2017-10-14	0.304317	0.294554
165	2017-10-15	0.343765	0.306120
166	2017-10-16	0.279894	0.299188
167	2017-10-17	0.280703	0.299564
168	2017-10-18	0.265719	0.294879
169	2017-10-19	0.291979	0.292412
170	2017-10-20	0.286850	0.281029
171	2017-10-21	0.325950	0.290240
172	2017-10-22	0.323956	0.298891
173	2017-10-23	0.289151	0.303577
174	2017-10-24	0.313321	0.307845
175	2017-10-25	0.294462	0.309368
176	2017-10-26	0.293324	0.302843
177	2017-10-27	0.299291	0.297910
178	2017-10-28	0.311847	0.302449
179	2017-10-29	0.319121	0.303609
180	2017-10-30	0.280825	0.300881
181	2017-10-31	0.255679	0.293352

In [90]:

```
galaxy_timeseries['polarity']
```

Out[90]:

```
0       0.310536
1       0.330682
2       0.287879
3       0.392172
4       0.218182
5       0.293944
6       0.330556
7       0.414744
8       0.187205
9       0.201515
```

```
9       0.201515
10      0.241035
11      0.315584
12      0.307102
13      0.221039
14      0.234030
15      0.231675
16      0.240229
17      0.251468
18      0.224011
19      0.216300
20      0.211343
21      0.236616
22      0.227806
23      0.277342
24      0.212585
25      0.256653
26      0.244301
27      0.246786
28      0.252248
29      0.266959
30      0.266753
31      0.260166
32      0.275248
33      0.265187
34      0.280572
35      0.292805
36      0.258510
37      0.254165
38      0.261247
39      0.266961
40      0.269168
41      0.273920
42      0.283524
43      0.280800
44      0.255527
45      0.305753
46      0.278183
47      0.256467
48      0.285575
49      0.286453
50      0.295996
51      0.296326
52      0.288633
53      0.306609
54      0.297099
55      0.273895
56      0.303774
57      0.276170
58      0.299734
59      0.304475
60      0.297218
61      0.280308
62      0.303593
```

```
63      0.298991

64      0.289376
65      0.302817
66      0.308979
67      0.321553
68      0.290013
69      0.316481
70      0.300882
71      0.305872
72      0.313543
73      0.311202
74      0.310671
75      0.308755
76      0.288788
77      0.281128
78      0.291714
79      0.305932
80      0.330519
81      0.329594
82      0.300593
83      0.295018
84      0.286150
85      0.295443
86      0.295295
87      0.291964
88      0.341749
89      0.284936
90      0.296157
91      0.276479
92      0.293778
93      0.285941
94      0.305853
95      0.323952
96      0.297537
97      0.315481
98      0.506657
99      0.282474
100     0.351566
101     0.578788
102     0.274405
103     0.540625
104     0.515341
105     0.472338
106     0.600000
107     0.373686
108     0.385065
109     0.555000
110     0.259758
111     0.493636
112     0.294413
113     0.503030
114     0.501042
115     0.320833
```

116	0.444129
117	0.373154
118	0.305177
119	0.368123
120	0.519110
121	0.376406
122	0.600000
123	0.252399
124	1.000000
125	0.578788
126	0.223990
127	0.550000
128	0.425000
129	0.488889
130	0.750000
131	0.875000
132	0.275000
133	0.316236
134	0.331482
135	0.287539
136	0.285435
137	0.306680
138	0.242732
139	0.309559
140	0.287591
141	0.297743
142	0.314732
143	0.299703
144	0.321277
145	0.317436
146	0.275926
147	0.289868
148	0.296005
149	0.273679
150	0.313593
151	0.322214
152	0.267314
153	0.325774
154	0.273175
155	0.314300
156	0.344983
157	0.287556
158	0.308766
159	0.297675
160	0.285932
161	0.314557
162	0.278820
163	0.289143
164	0.304317
165	0.343765
166	0.279894
167	0.280703
168	0.265719

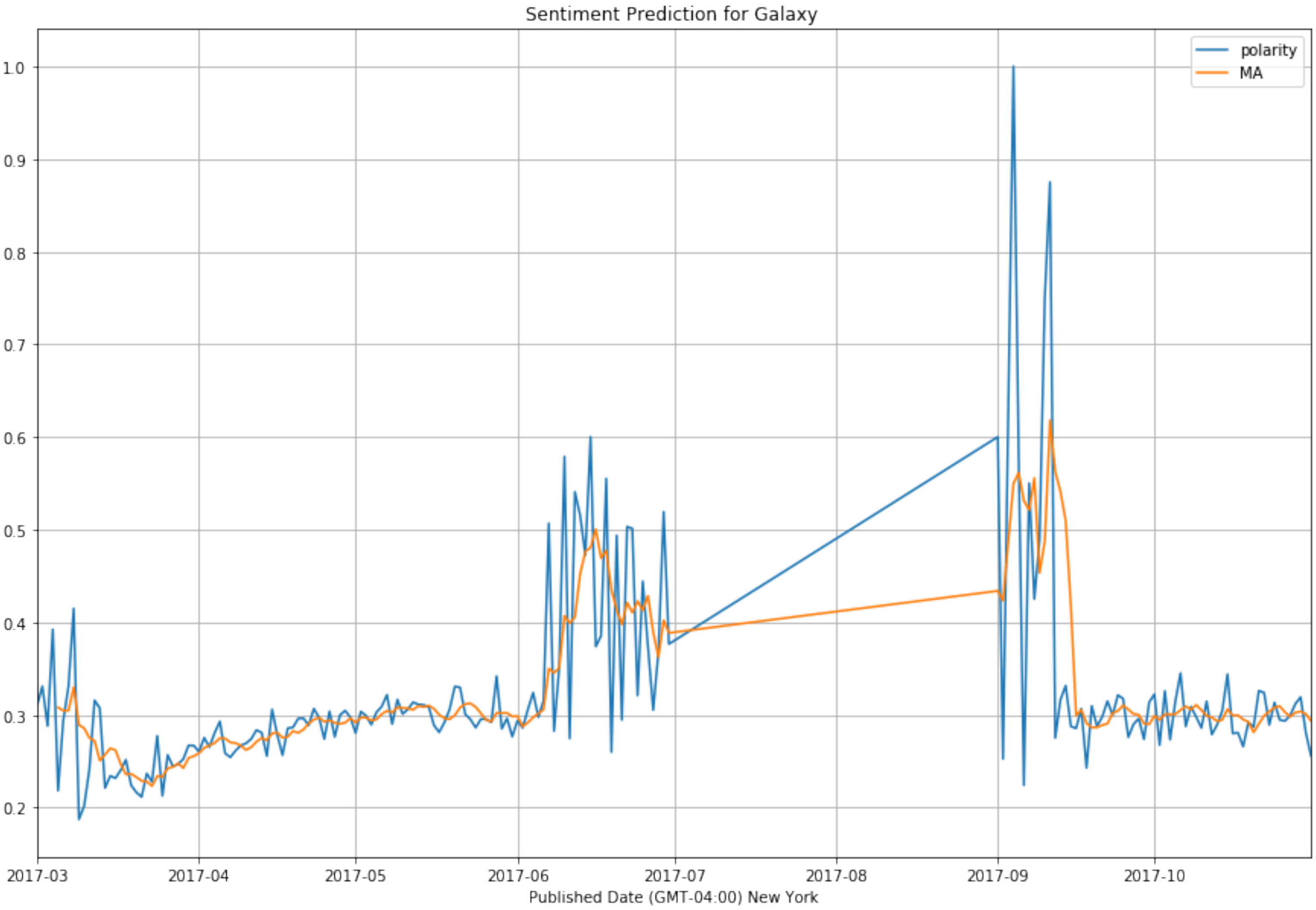
```
169     0.291979

170     0.286850
171     0.325950
172     0.323956
173     0.289151
174     0.313321
175     0.294462
176     0.293324
177     0.299291
178     0.311847
179     0.319121
180     0.280825
181     0.255679
Name: polarity, dtype: float64
```

In [91]:

```
galaxy_timeseries.plot(x='Published Date (GMT-04:00) New York', \
                             y=['polarity', 'MA'], title= "Sentiment Prediction
for Galaxy ", figsize=(15,10), grid=True)
```

Out[91]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd030ad50>
```



Plotting Accuracy for prediction of adoption of Samsung Galaxy S8

In [92]:

```
galaxy_timeseries['MA'] = galaxy_timeseries['polarity'].rolling(window=5,center=False).mean()
new_num = (1 - (abs(galaxy_timeseries['polarity']- galaxy_timeseries['MA']))/ galaxy_timeseries['polarity']) * 100
galaxy_timeseries['accuracy'] = new_num
galaxy_timeseries
```

Out[92]:

	Published Date (GMT-04:00) New York	polarity	MA	accuracy
0	2017-03-01	0.310536	NaN	NaN
1	2017-03-02	0.330682	NaN	NaN
2	2017-03-03	0.287879	NaN	NaN
3	2017-03-04	0.392172	NaN	NaN
4	2017-03-05	0.218182	0.307890	58.883775
5	2017-03-06	0.293944	0.304572	96.384429
6	2017-03-07	0.330556	0.304546	92.131668
7	2017-03-08	0.414744	0.329919	79.547777
8	2017-03-09	0.187205	0.288926	45.663608
9	2017-03-10	0.201515	0.285593	58.277303
10	2017-03-11	0.241035	0.275011	85.904286
11	2017-03-12	0.315584	0.272017	86.194617
12	2017-03-13	0.307102	0.250489	81.565178
13	2017-03-14	0.221039	0.257255	83.615534
14	2017-03-15	0.234030	0.263758	87.297081
15	2017-03-16	0.231675	0.261886	86.959808
16	2017-03-17	0.240229	0.246815	97.258365
17	2017-03-18	0.251468	0.235688	93.725030
18	2017-03-19	0.224011	0.236282	94.521760
19	2017-03-20	0.216300	0.232737	92.401273
20	2017-03-21	0.211343	0.228670	91.801362
21	2017-03-22	0.236616	0.227947	96.336419
22	2017-03-23	0.227806	0.223215	97.984904
23	2017-03-24	0.277342	0.233881	84.329677
24	2017-03-25	0.212585	0.233138	90.331716
25	2017-03-26	0.256653	0.242200	94.368675

26	2017-03-27	0.244301	0.243737	99.769165
27	2017-03-28	0.246786	0.247533	99.697243
28	2017-03-29	0.252248	0.242515	96.141284
29	2017-03-30	0.266959	0.253390	94.916958
30	2017-03-31	0.266753	0.255410	95.747607
31	2017-04-01	0.260166	0.258583	99.391422
32	2017-04-02	0.275248	0.264275	96.013488
33	2017-04-03	0.265187	0.266863	99.368286
34	2017-04-04	0.280572	0.269585	96.084243
35	2017-04-05	0.292805	0.274796	93.849414
36	2017-04-06	0.258510	0.274464	93.828397
37	2017-04-07	0.254165	0.270248	93.672344
38	2017-04-08	0.261247	0.269460	96.856373
39	2017-04-09	0.266961	0.266738	99.916290
40	2017-04-10	0.269168	0.262010	97.340908
41	2017-04-11	0.273920	0.265092	96.777369
42	2017-04-12	0.283524	0.270964	95.569937
43	2017-04-13	0.280800	0.274875	97.889724
44	2017-04-14	0.255527	0.272588	93.323259
45	2017-04-15	0.305753	0.279905	91.546099
46	2017-04-16	0.278183	0.280757	99.074614
47	2017-04-17	0.256467	0.275346	92.638838
48	2017-04-18	0.285575	0.276301	96.752610
49	2017-04-19	0.286453	0.282486	98.615274
50	2017-04-20	0.295996	0.280535	94.776405
51	2017-04-21	0.296326	0.284163	95.895635
52	2017-04-22	0.288633	0.290596	99.319769
53	2017-04-23	0.306609	0.294803	96.149681
54	2017-04-24	0.297099	0.296933	99.944002
55	2017-04-25	0.273895	0.292512	93.202684
56	2017-04-26	0.303774	0.294002	96.782995
57	2017-04-27	0.276170	0.291509	94.445664
58	2017-04-28	0.299734	0.290134	96.797342
59	2017-04-29	0.304475	0.291609	95.774588
60	2017-04-30	0.297218	0.296274	99.682496

61	2017-05-01	0.280308	0.291581	95.978499
62	2017-05-02	0.303593	0.297066	97.849873
63	2017-05-03	0.298991	0.296917	99.306368
64	2017-05-04	0.289376	0.293897	98.437683
65	2017-05-05	0.302817	0.295017	97.424154
66	2017-05-06	0.308979	0.300751	97.337084
67	2017-05-07	0.321553	0.304343	94.647933
68	2017-05-08	0.290013	0.302548	95.677936
69	2017-05-09	0.316481	0.307969	97.310284
70	2017-05-10	0.300882	0.307582	97.773230
71	2017-05-11	0.305872	0.306960	99.644128
72	2017-05-12	0.313543	0.305358	97.389596
73	2017-05-13	0.311202	0.309596	99.483931
74	2017-05-14	0.310671	0.308434	99.279884
75	2017-05-15	0.308755	0.310009	99.594023
76	2017-05-16	0.288788	0.306592	93.834940
77	2017-05-17	0.281128	0.300109	93.248465
78	2017-05-18	0.291714	0.296211	98.458401
79	2017-05-19	0.305932	0.295264	96.512697
80	2017-05-20	0.330519	0.299616	90.650368
81	2017-05-21	0.329594	0.307777	93.380874
82	2017-05-22	0.300593	0.311670	96.314779
83	2017-05-23	0.295018	0.312331	94.131636
84	2017-05-24	0.286150	0.308375	92.233049
85	2017-05-25	0.295443	0.301359	97.997342
86	2017-05-26	0.295295	0.294500	99.730671
87	2017-05-27	0.291964	0.292774	99.722674
88	2017-05-28	0.341749	0.302120	88.404185
89	2017-05-29	0.284936	0.301877	94.054360
90	2017-05-30	0.296157	0.302020	98.020283
91	2017-05-31	0.276479	0.298257	92.123104
92	2017-06-01	0.293778	0.298620	98.351811
93	2017-06-02	0.285941	0.287458	99.469394
94	2017-06-03	0.305853	0.291642	95.353453
95	2017-06-04	0.323952	0.297201	91.742106

96	2017-06-05	0.297537	0.301412	98.697588
97	2017-06-06	0.315481	0.305753	96.916430
98	2017-06-07	0.506657	0.349896	69.059722
99	2017-06-08	0.282474	0.345220	77.786950
100	2017-06-09	0.351566	0.350743	99.766036
101	2017-06-10	0.578788	0.406993	70.318209
102	2017-06-11	0.274405	0.398778	54.675257
103	2017-06-12	0.540625	0.405572	75.019009
104	2017-06-13	0.515341	0.452145	87.737036
105	2017-06-14	0.472338	0.476299	99.161282
106	2017-06-15	0.600000	0.480542	80.090278
107	2017-06-16	0.373686	0.500398	66.091291
108	2017-06-17	0.385065	0.469286	78.128122
109	2017-06-18	0.555000	0.477218	85.985169
110	2017-06-19	0.259758	0.434702	32.651423
111	2017-06-20	0.493636	0.413429	83.751749
112	2017-06-21	0.294413	0.397574	64.960223
113	2017-06-22	0.503030	0.421168	83.726083
114	2017-06-23	0.501042	0.410376	81.904546
115	2017-06-24	0.320833	0.422591	68.283353
116	2017-06-25	0.444129	0.412689	92.921109
117	2017-06-26	0.373154	0.428438	85.184812
118	2017-06-27	0.305177	0.388867	72.576492
119	2017-06-28	0.368123	0.362283	98.413709
120	2017-06-29	0.519110	0.401938	77.428398
121	2017-06-30	0.376406	0.388394	96.815228
122	2017-09-01	0.600000	0.433763	72.293850
123	2017-09-02	0.252399	0.423208	32.325975
124	2017-09-04	1.000000	0.549583	54.958302
125	2017-09-05	0.578788	0.561519	97.016307
126	2017-09-06	0.223990	0.531035	-37.080045
127	2017-09-07	0.550000	0.521035	94.733701
128	2017-09-08	0.425000	0.555556	69.281046
129	2017-09-09	0.488889	0.453333	92.727273
130	2017-09-10	0.750000	0.487576	65.010101

131	2017-09-11	0.875000	0.617778	70.603175
132	2017-09-12	0.275000	0.562778	-4.646465
133	2017-09-13	0.316236	0.541025	28.917256
134	2017-09-14	0.331482	0.509544	46.283255
135	2017-09-15	0.287539	0.417051	54.958106
136	2017-09-16	0.285435	0.299138	95.199157
137	2017-09-17	0.306680	0.305474	99.606830
138	2017-09-18	0.242732	0.290774	80.208051
139	2017-09-19	0.309559	0.286389	92.515196
140	2017-09-20	0.287591	0.286399	99.585751
141	2017-09-21	0.297743	0.288861	97.016878
142	2017-09-22	0.314732	0.290471	92.291549
143	2017-09-23	0.299703	0.301866	99.278389
144	2017-09-24	0.321277	0.304209	94.687444
145	2017-09-25	0.317436	0.310178	97.713567
146	2017-09-26	0.275926	0.305815	89.167859
147	2017-09-27	0.289868	0.300842	96.213968
148	2017-09-28	0.296005	0.300102	98.615678
149	2017-09-29	0.273679	0.290583	93.823611
150	2017-09-30	0.313593	0.289814	92.417341
151	2017-10-01	0.322214	0.299072	92.817643
152	2017-10-02	0.267314	0.294561	89.807129
153	2017-10-03	0.325774	0.300515	92.246474
154	2017-10-04	0.273175	0.300414	90.028844
155	2017-10-05	0.314300	0.300556	95.626982
156	2017-10-06	0.344983	0.305109	88.441802
157	2017-10-07	0.287556	0.309158	92.487759
158	2017-10-08	0.308766	0.305756	99.025210
159	2017-10-09	0.297675	0.310656	95.639110
160	2017-10-10	0.285932	0.304982	93.337458
161	2017-10-11	0.314557	0.298897	95.021625
162	2017-10-12	0.278820	0.297150	93.425994
163	2017-10-13	0.289143	0.293225	98.588177
164	2017-10-14	0.304317	0.294554	96.791867
165	2017-10-15	0.343765	0.306120	89.049362

166	2017-10-16	0.279894	0.299188	93.106618
167	2017-10-17	0.280703	0.299564	93.280760
168	2017-10-18	0.265719	0.294879	89.025820
169	2017-10-19	0.291979	0.292412	99.851767
170	2017-10-20	0.286850	0.281029	97.970649
171	2017-10-21	0.325950	0.290240	89.044425
172	2017-10-22	0.323956	0.298891	92.262876
173	2017-10-23	0.289151	0.303577	95.010777
174	2017-10-24	0.313321	0.307845	98.252419
175	2017-10-25	0.294462	0.309368	94.938043
176	2017-10-26	0.293324	0.302843	96.754853
177	2017-10-27	0.299291	0.297910	99.538572
178	2017-10-28	0.311847	0.302449	96.986397
179	2017-10-29	0.319121	0.303609	95.139190
180	2017-10-30	0.280825	0.300881	92.857925
181	2017-10-31	0.255679	0.293352	85.265537

In [93]:

```
galaxy_timeseries['accuracy'].mean()
```

Out[93]:

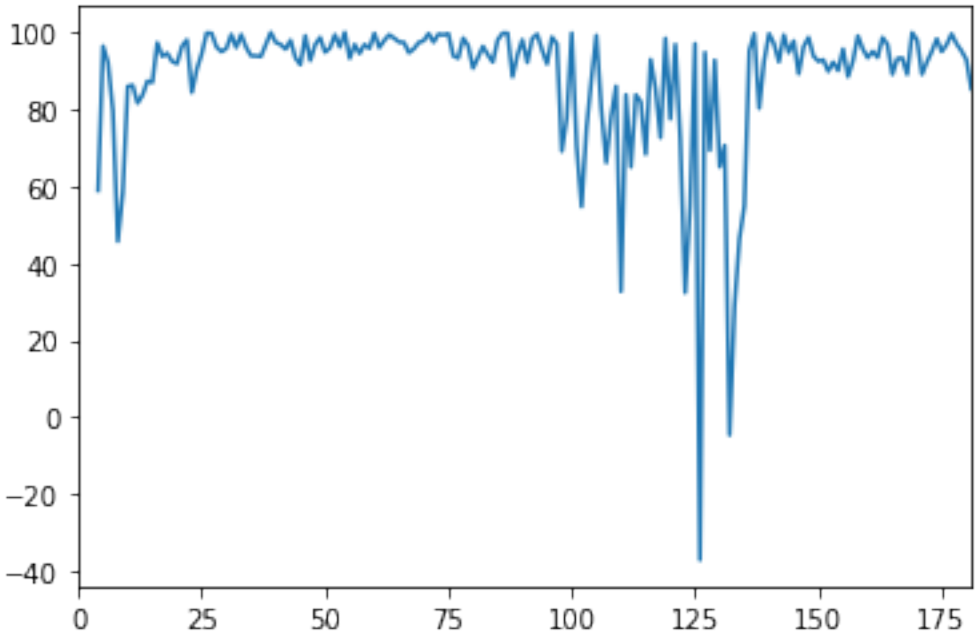
```
88.59157596806487
```

In [94]:

```
s = pd.Series(galaxy_timeseries['accuracy'])
s.plot.line()
```

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd0316550>
```



In [ ]:

In [95]:

```
normal_blob = pd.DataFrame(df_iphone_8)

#def detect_polarity(text):
#    return TextBlob(text).sentiment.polarity

#def translate_english(text):
   # return TextBlob(text).translate(to='en')

#normal_blob['Sound Bite Translated'] = normal_df['Sound Bite Text'].apply(trans
late_english)

#normal_blob['polarity'] = df_iphone_8['Sound Bite Text'].apply(detect_polarity)


normal_positive = normal_blob[normal_blob['polarity']>0.1]
normal_positive['Group'] = "Positive"


plotting_before = normal_positive.groupby(['Published Date (GMT-04:00) New York'
])['polarity'].mean()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:15:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```

In [96]:

```
iphone_timeseries = pd.DataFrame(plotting_before).reset_index()
iphone_timeseries.head(3)
```

Out[96]:

	Published Date (GMT-04:00) New York	polarity
0	2017-03-01	0.500710
1	2017-03-02	0.347078
2	2017-03-03	0.554261

In [97]:

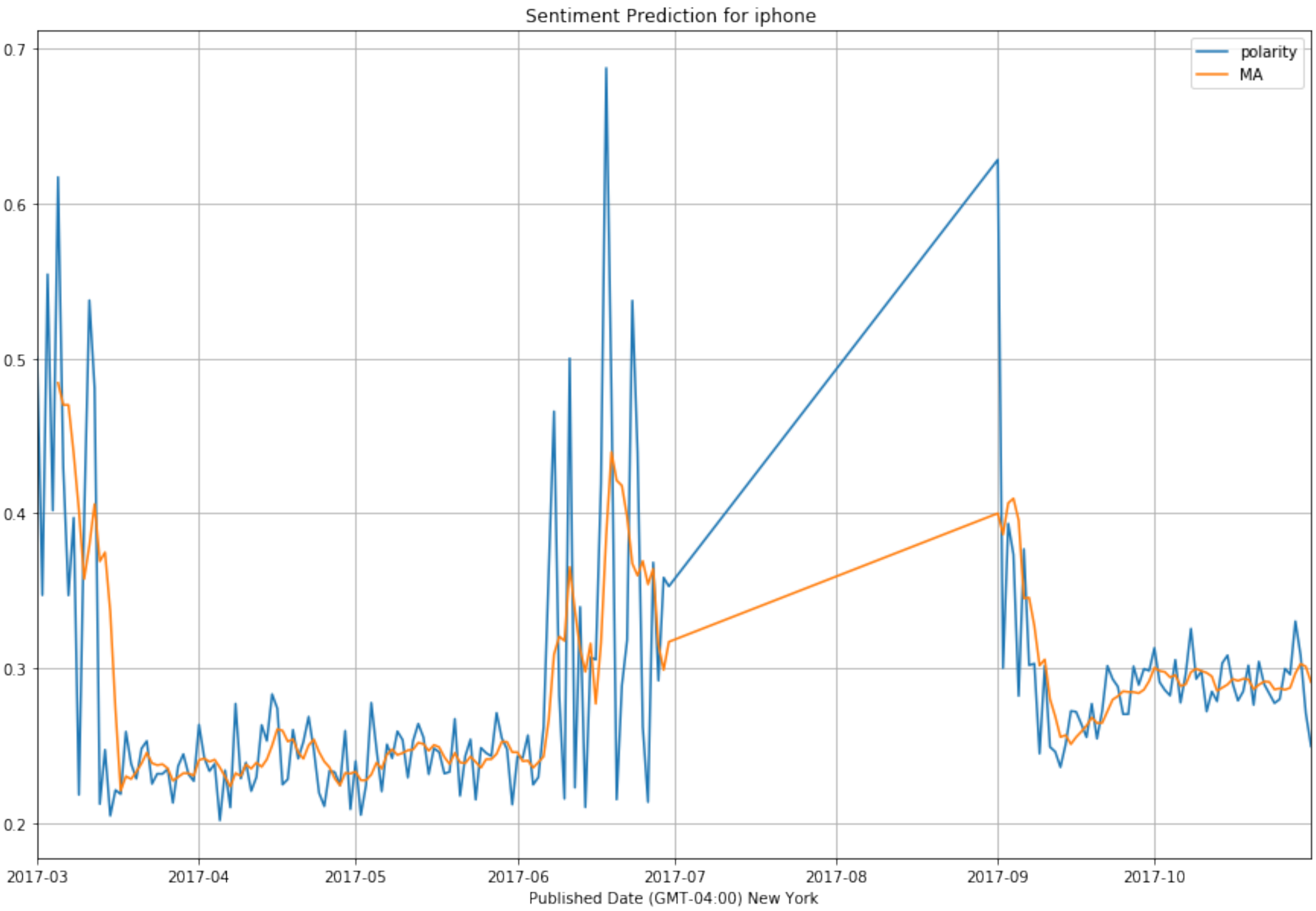
```
iphone_timeseries['MA'] = iphone_timeseries['polarity'].rolling(window=5,center=
False).mean()
```

In [98]:

```
iphone_timeseries.plot(x='Published Date (GMT-04:00) New York', \
                             y=['polarity', 'MA'], title= "Sentiment Prediction
for iphone ", figsize=(15,10), grid=True)
```

Out[98]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd02cb510>
```



Plotting Accuracy for prediction of adoption of Apple iPhone 8

In [99]:

```
iphone_timeseries['MA'] = iphone_timeseries['polarity'].rolling(window=5,center=
False).mean()
new_num = (1 - (abs(iphone_timeseries['polarity']- iphone_timeseries['MA']))/ ip
hone_timeseries['polarity']) * 100
iphone_timeseries['accuracy'] = new_num
iphone_timeseries
```

Out[99]:

	Published Date (GMT-04:00) New York	polarity	MA	accuracy
0	2017-03-01	0.500710	NaN	NaN
1	2017-03-02	0.347078	NaN	NaN
2	2017-03-03	0.554261	NaN	NaN
3	2017-03-04	0.401768	NaN	NaN
4	2017-03-05	0.617045	0.484173	78.466266
5	2017-03-06	0.429545	0.469940	90.596078
6	2017-03-07	0.346970	0.469918	64.565138
7	2017-03-08	0.397112	0.438488	89.580700
8	2017-03-09	0.218182	0.401771	15.855035
9	2017-03-10	0.395806	0.357523	90.327774
10	2017-03-11	0.537500	0.379114	70.532820
11	2017-03-12	0.480909	0.405902	84.403018
12	2017-03-13	0.212182	0.368916	26.132228
13	2017-03-14	0.247227	0.374725	48.428787
14	2017-03-15	0.204680	0.336500	35.597405
15	2017-03-16	0.221228	0.273245	76.487241
16	2017-03-17	0.218602	0.220784	99.001903
17	2017-03-18	0.258947	0.230137	88.874222
18	2017-03-19	0.238206	0.228333	95.855076
19	2017-03-20	0.228555	0.233108	98.007898
20	2017-03-21	0.248090	0.238480	96.126436
21	2017-03-22	0.252904	0.245340	97.009112
22	2017-03-23	0.225268	0.238605	94.079711
23	2017-03-24	0.231741	0.237312	97.596215
24	2017-03-25	0.231664	0.237933	97.293825
25	2017-03-26	0.234818	0.235279	99.803679
26	2017-03-27	0.213010	0.227300	93.291296
27	2017-03-28	0.236721	0.229591	96.988056
28	2017-03-29	0.244432	0.232129	94.966588
29	2017-03-30	0.231352	0.232067	99.691052
30	2017-03-31	0.227046	0.230512	98.473434
31	2017-04-01	0.263454	0.240601	91.325518
32	2017-04-02	0.242223	0.241702	99.784695
33	2017-04-03	0.233444	0.239504	97.404119

34	2017-04-04	0.238158	0.240865	98.863151
35	2017-04-05	0.201651	0.235786	83.072325
36	2017-04-06	0.233999	0.229895	98.246146
37	2017-04-07	0.209982	0.223447	93.587753
38	2017-04-08	0.277004	0.232159	83.810706
39	2017-04-09	0.228689	0.230265	99.310949
40	2017-04-10	0.238980	0.237731	99.477406
41	2017-04-11	0.220608	0.235053	93.452223
42	2017-04-12	0.229580	0.238972	95.909181
43	2017-04-13	0.263234	0.236218	89.737101
44	2017-04-14	0.253020	0.241084	95.282561
45	2017-04-15	0.283151	0.249919	88.263324
46	2017-04-16	0.273933	0.260584	95.126722
47	2017-04-17	0.224779	0.259623	84.498270
48	2017-04-18	0.228198	0.252616	89.299322
49	2017-04-19	0.260137	0.254040	97.655969
50	2017-04-20	0.241518	0.245713	98.263101
51	2017-04-21	0.252485	0.241423	95.618816
52	2017-04-22	0.268601	0.250188	93.144865
53	2017-04-23	0.246976	0.253943	97.178875
54	2017-04-24	0.219372	0.245790	87.957203
55	2017-04-25	0.210794	0.239646	86.313068
56	2017-04-26	0.233706	0.235890	99.065536
57	2017-04-27	0.232815	0.228733	98.246523
58	2017-04-28	0.224395	0.224216	99.920318
59	2017-04-29	0.259520	0.232246	89.490648
60	2017-04-30	0.208827	0.231853	88.973727
61	2017-05-01	0.239926	0.233097	97.153501
62	2017-05-02	0.205137	0.227561	89.068790
63	2017-05-03	0.224566	0.227595	98.650940
64	2017-05-04	0.277583	0.231208	83.293294
65	2017-05-05	0.247554	0.238953	96.525560
66	2017-05-06	0.220383	0.235044	93.347273
67	2017-05-07	0.250718	0.244161	97.384726
68	2017-05-08	0.241806	0.247609	97.600358
69	2017-05-09	0.259147	0.243222	94.184684

69	2017-05-09	0.259147	0.243922	94.124684
70	2017-05-10	0.253624	0.245136	96.653192
71	2017-05-11	0.229293	0.246918	92.313422
72	2017-05-12	0.252957	0.247366	97.789437
73	2017-05-13	0.264058	0.251816	95.363979
74	2017-05-14	0.255222	0.251031	98.357688
75	2017-05-15	0.231462	0.246598	93.460611
76	2017-05-16	0.248162	0.250372	99.109308
77	2017-05-17	0.245782	0.248937	98.716392
78	2017-05-18	0.231888	0.242503	95.422152
79	2017-05-19	0.232959	0.238051	97.814454
80	2017-05-20	0.267195	0.245197	91.767148
81	2017-05-21	0.217505	0.239066	90.087094
82	2017-05-22	0.243120	0.238533	98.113594
83	2017-05-23	0.253931	0.242942	95.672554
84	2017-05-24	0.215068	0.239364	88.703235
85	2017-05-25	0.248497	0.235624	94.819656
86	2017-05-26	0.245155	0.241154	98.368107
87	2017-05-27	0.243117	0.241153	99.192317
88	2017-05-28	0.271053	0.244578	90.232601
89	2017-05-29	0.254670	0.252498	99.147177
90	2017-05-30	0.247684	0.252336	98.122074
91	2017-05-31	0.211875	0.245680	84.044779
92	2017-06-01	0.243051	0.245667	98.923783
93	2017-06-02	0.242007	0.239857	99.111799
94	2017-06-03	0.256602	0.240244	93.625118
95	2017-06-04	0.224708	0.235648	95.131088
96	2017-06-05	0.229522	0.239178	95.793234
97	2017-06-06	0.261635	0.242895	92.837120
98	2017-06-07	0.362798	0.267053	73.609344
99	2017-06-08	0.465758	0.308884	66.318644
100	2017-06-09	0.282282	0.320399	86.496900
101	2017-06-10	0.215625	0.317620	52.698173
102	2017-06-11	0.500000	0.365292	73.058496
103	2017-06-12	0.222822	0.337297	48.624734
104	2017-06-13	[illegible]	[illegible]	[illegible]

104	2017-06-13	0.339510	0.312048	91.911131
105	2017-06-14	0.210101	0.297612	58.348280
106	2017-06-15	0.306960	0.315879	97.094571
107	2017-06-16	0.305424	0.276964	90.681572
108	2017-06-17	0.419531	0.316305	75.394972
109	2017-06-18	0.687500	0.385903	56.131399
110	2017-06-19	0.478273	0.439538	91.901026
111	2017-06-20	0.215152	0.421176	4.242132
112	2017-06-21	0.288636	0.417818	55.244012
113	2017-06-22	0.318182	0.397549	75.056176
114	2017-06-23	0.537273	0.367503	68.401588
115	2017-06-24	0.439394	0.359727	81.868966
116	2017-06-25	0.262121	0.369121	59.179191
117	2017-06-26	0.213485	0.354091	34.137686
118	2017-06-27	0.368182	0.364091	98.888889
119	2017-06-28	0.291919	0.315020	92.086505
120	2017-06-29	0.358434	0.298828	83.370438
121	2017-06-30	0.352794	0.316963	89.843690
122	2017-09-01	0.628283	0.399922	63.653236
123	2017-09-02	0.300000	0.386286	71.238005
124	2017-09-03	0.393308	0.406564	96.629695
125	2017-09-04	0.373252	0.409527	90.281226
126	2017-09-05	0.282008	0.395370	59.801622
127	2017-09-06	0.376813	0.345076	91.577585
128	2017-09-07	0.301871	0.345450	85.563690
129	2017-09-08	0.302879	0.327364	91.915700
130	2017-09-09	0.244766	0.301667	76.752721
131	2017-09-10	0.301515	0.305569	98.655590
132	2017-09-11	0.249083	0.280023	87.578494
133	2017-09-12	0.246005	0.268850	90.713938
134	2017-09-13	0.236024	0.255479	91.757329
135	2017-09-14	0.250876	0.256701	97.678243
136	2017-09-15	0.272290	0.250856	92.128008
137	2017-09-16	0.271789	0.255397	93.968894
138	2017-09-17	0.263827	0.258961	98.155826
139	2017-09-18	0.255275	0.262881	97.030330

139	2017-09-18	0.255375	0.262831	97.080339
140	2017-09-19	0.276978	0.268052	96.777241
141	2017-09-20	0.254427	0.264479	96.049165
142	2017-09-21	0.272453	0.264612	97.121970
143	2017-09-22	0.301471	0.272141	90.271096
144	2017-09-23	0.292910	0.279648	95.472239
145	2017-09-24	0.287979	0.281848	97.871061
146	2017-09-25	0.270275	0.285018	94.545158
147	2017-09-26	0.270261	0.284579	94.702162
148	2017-09-27	0.301133	0.284512	94.480355
149	2017-09-28	0.289147	0.283759	98.136482
150	2017-09-29	0.299535	0.286070	95.504906
151	2017-09-30	0.298253	0.291666	97.791350
152	2017-10-01	0.313004	0.300214	95.913892
153	2017-10-02	0.290889	0.298166	97.498429
154	2017-10-03	0.285822	0.297501	95.913975
155	2017-10-04	0.282265	0.294047	95.826023
156	2017-10-05	0.305326	0.295461	96.769175
157	2017-10-06	0.277662	0.288393	96.135424
158	2017-10-07	0.296423	0.289499	97.664317
159	2017-10-08	0.325442	0.297423	91.390735
160	2017-10-09	0.293002	0.299571	97.758148
161	2017-10-10	0.298008	0.298107	99.966760
162	2017-10-11	0.271989	0.296973	90.814507
163	2017-10-12	0.284832	0.294655	96.551402
164	2017-10-13	0.278561	0.285279	97.588537
165	2017-10-14	0.303254	0.287329	94.748685
166	2017-10-15	0.308221	0.289371	93.884434
167	2017-10-16	0.289938	0.292961	98.957308
168	2017-10-17	0.278984	0.291792	95.409082
169	2017-10-18	0.284867	0.293053	97.126448
170	2017-10-19	0.301736	0.292749	97.021566
171	2017-10-20	0.276218	0.286349	96.332526
172	2017-10-21	0.304148	0.289191	95.082129
173	2017-10-22	0.289737	0.291341	99.446173
174	2017-10-23	[illegible]	[illegible]	[illegible]

174	2017-10-23	0.283493	0.291066	97.328500
175	2017-10-24	0.277351	0.286189	96.813296
176	2017-10-25	0.279893	0.286924	97.487808
177	2017-10-26	0.299648	0.286024	95.453391
178	2017-10-27	0.295891	0.287255	97.081364
179	2017-10-28	0.330225	0.296602	89.818047
180	2017-10-29	0.308164	0.302764	98.247699
181	2017-10-30	0.270754	0.300937	88.852373
182	2017-10-31	0.249453	0.290897	83.385758

In [100]:

```
iphone_timeseries['accuracy'].mean()
```

Out[100]:

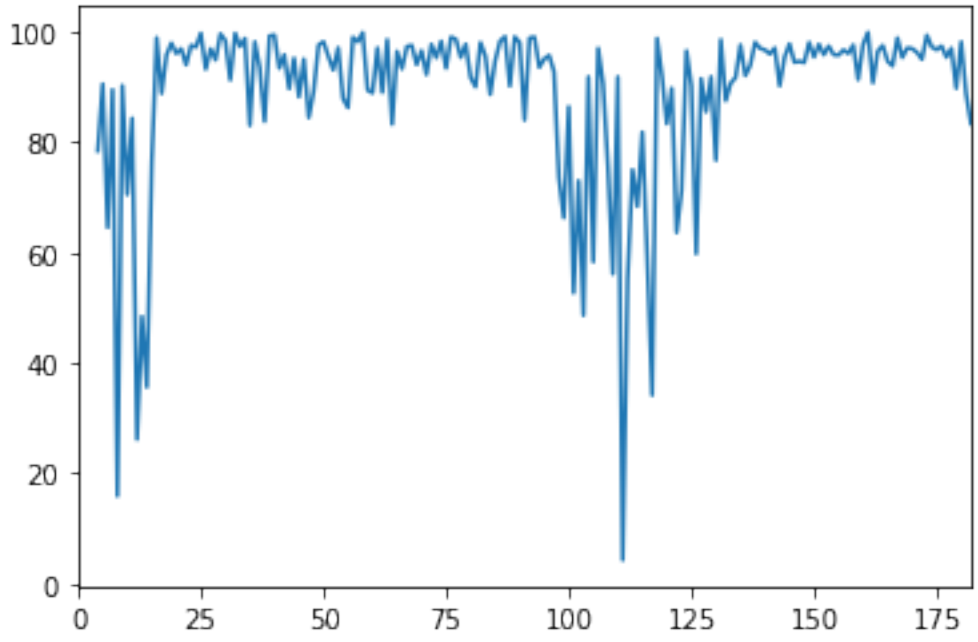
```
89.01040839143236
```

In [101]:

```
s = pd.Series(iphone_timeseries['accuracy'])
s.plot.line()
```

Out[101]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd02c11d0>
```



Prediction for iPhone 10. Since we do not have much data on iPhone 10, we cannot train our model and give an accurate prediction whether the customers adopt it or not.

In [ ]:

```
```