

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



# Fun with analyzing @BillGates tweets Twitter API's-Step by Step analysis from Extraction, Data Visualization, and Sentiment Analysis



Senthil E · Following

Published in Towards Data Science

11 min read · May 10, 2019

[Listen](#)[Share](#)[More](#)

This is the 2nd post of the web scraping and API's series. The first post is here.

Please check it out. In this post, we can see how to extract the twitter data using Twitter API's and then do some basic visualization using word cloud, pie charts and then sentiment analysis using Textblob and Vader.

If you don't have Jupyter then please go ahead and install anaconda.

“

Data is the new oil”

Clive Humby



Let's do the following

1. Create a twitter account or use your existing Twitter account.
2. Request for twitter developer access key
3. Extract real-time tweets using Twitter Streaming API
4. Extract history tweets using Twitter Search/Rest API using Tweepy
5. Load the data to a pandas data frame
6. Wordcloud
7. Scatter Text

8. Do some statistical analysis and visualization.

9. Sentiment Analysis using **Textblob**.

10. Sentiment Analysis using **VADER**

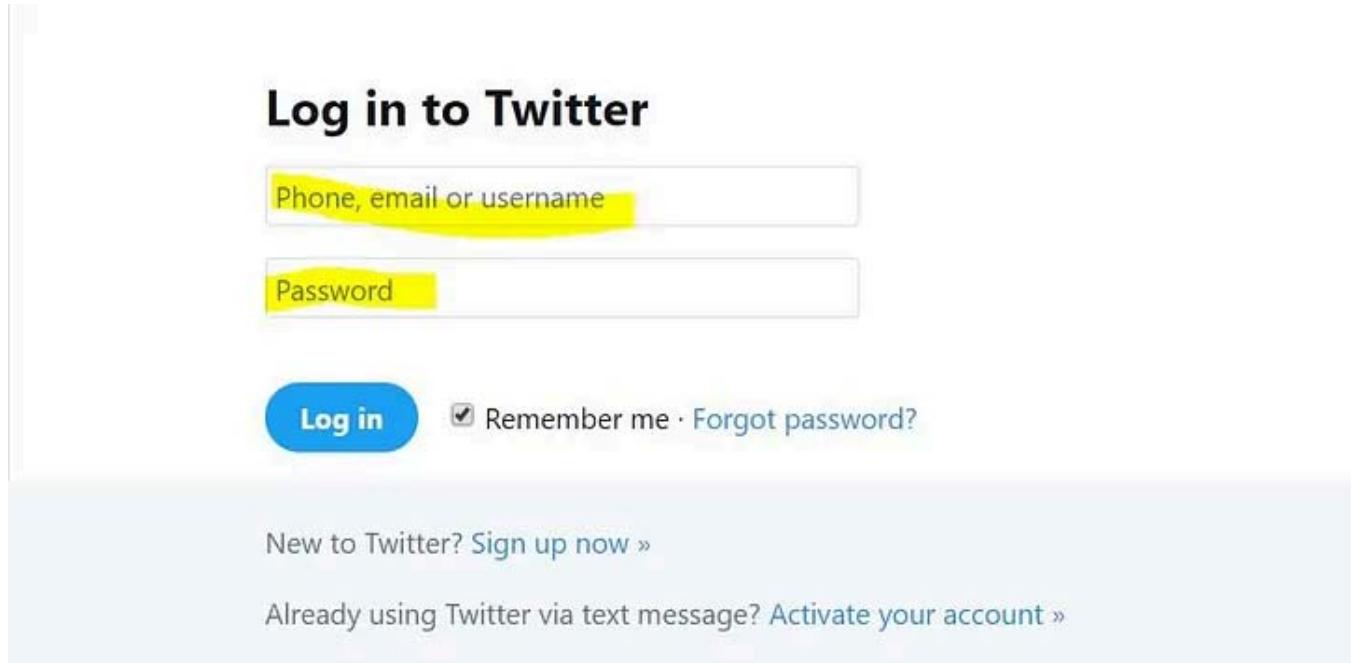
11. Sentiment Analysis using **Syuzhet(R)**

Before using the Twitter API we need a twitter account and then request for a developer key. First, create a twitter account if you don't have one. The steps to get the necessary developer keys are below

**Watch this youtube video for creating a twitter account**

1. Goto to <https://developer.twitter.com/en/apps>.

2. Log in using your twitter account. If you don't have one then create one.



3. After logged in then create an app.

[Create an app](#)[Details](#)

4. Fill out all the required fields.

### App details

The following app details will be visible to app users and are required to generate the API keys needed to authenticate Twitter developer products.

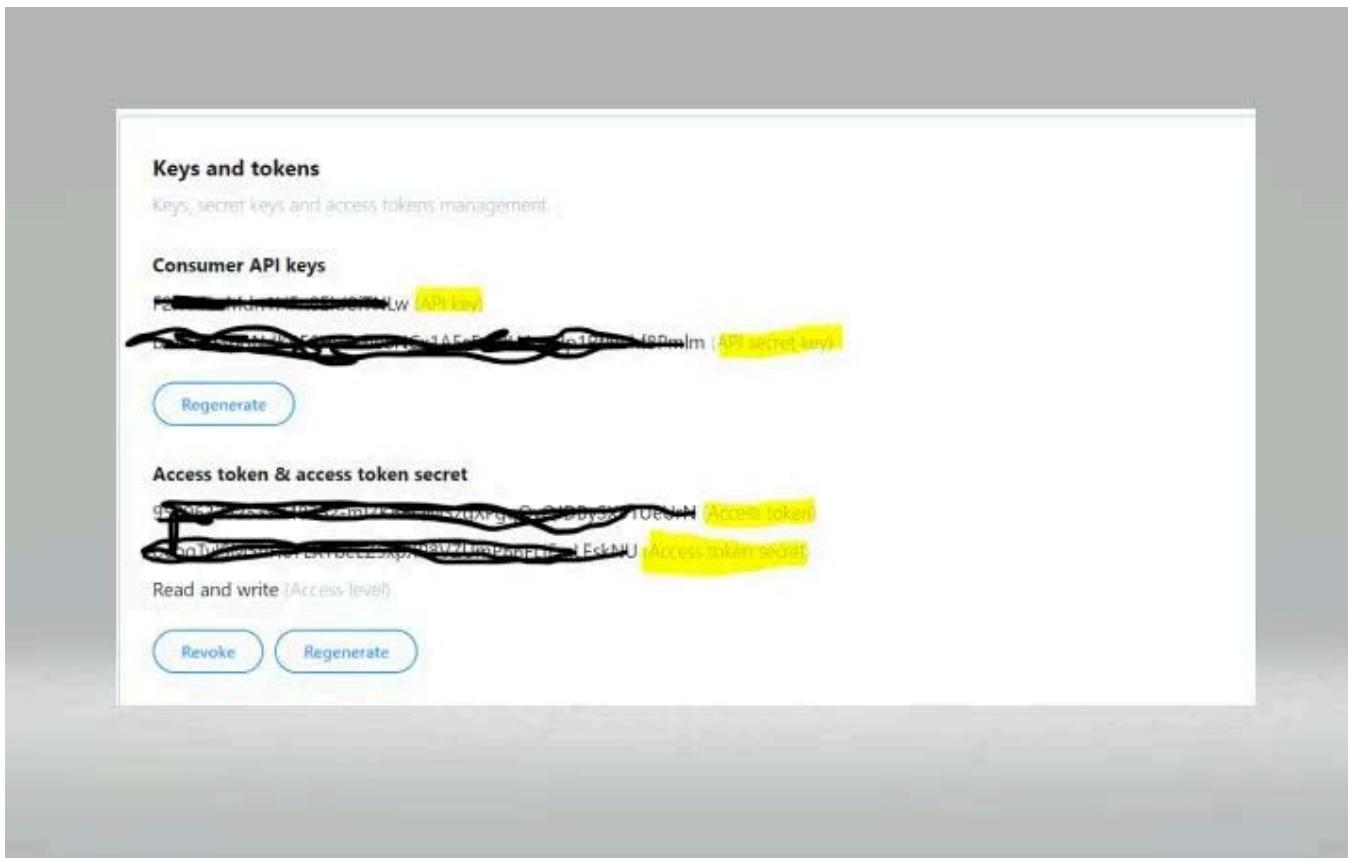
**App name** (required) [?](#)

Maximum characters: **32**

**Application description** (required)

Share a description of your app. This description will be visible to users so this is a good place to tell them what your app does.

5. You will get the following credentials



## 1. API Key

## 2. API secret key

## 3. Access token

## 4. Access token secret

[Check out this youtube video on creating a twitter](#)

[Check out his documentation to get the Twitter Dev Keys](#)

**Twitter provides 2 API's**

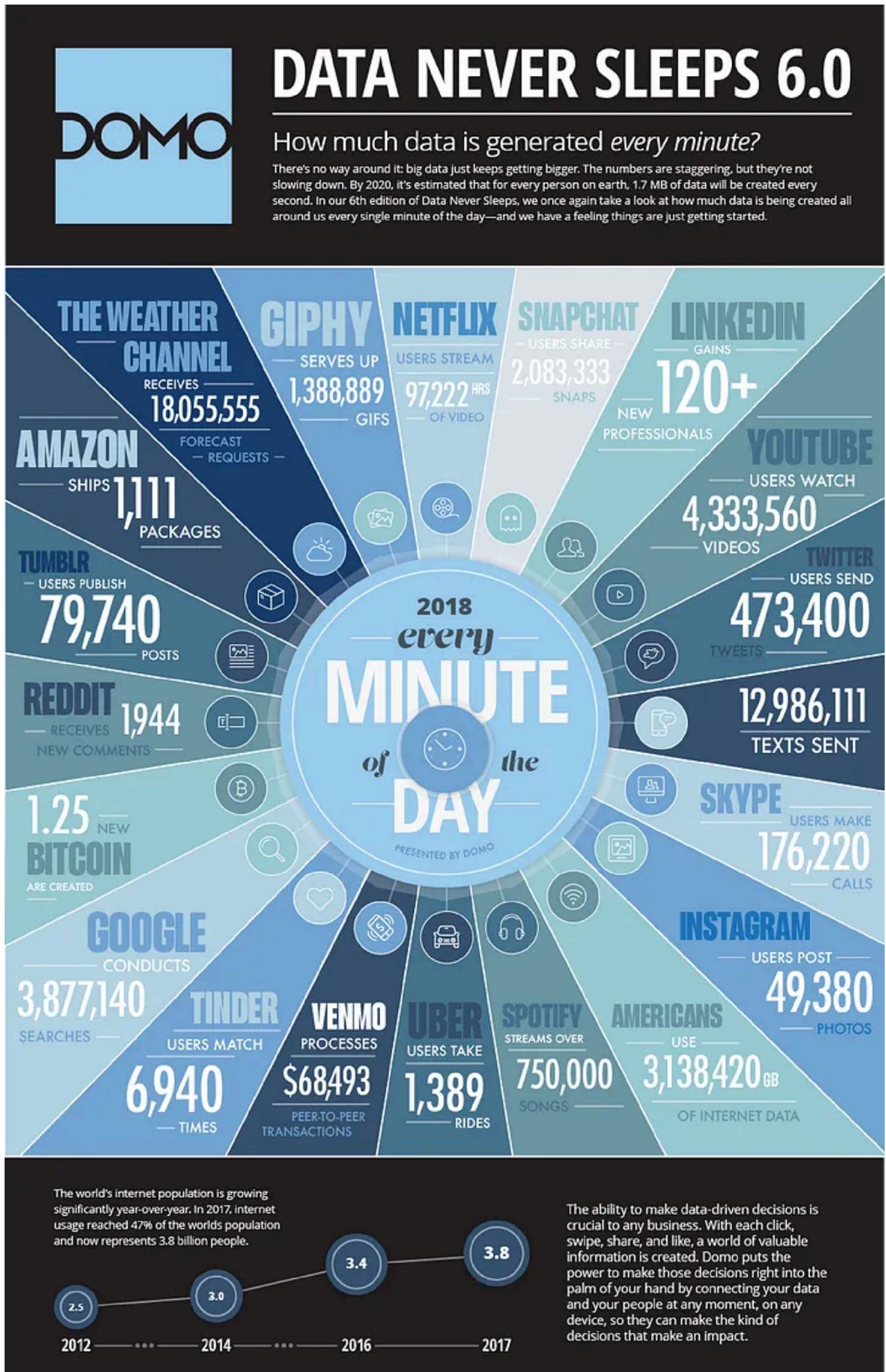
- *Streaming API*
- *Search API or Rest API*



From the twitter developer website, the available python related wrappers are

## Python

- [python-twitter](#) maintained by @bear — this library provides a pure Python interface for the Twitter API ([documentation](#))
- [tweepy](#) maintained by @applepie & more — a Python wrapper for the Twitter API ([documentation](#)) ([examples](#))
- [TweetPony](#) by @Mezgrman — A Python library aimed at simplicity and flexibility.
- [Python Twitter Tools](#) by @sixohsix — An extensive Python library for interfacing to the Twitter REST and streaming APIs (v1.0 and v1.1). Also features a command line Twitter client. Supports Python 2.6, 2.7, and 3.3+. ([documentation](#))
- [twitter-gobject](#) by @tchx84 — Allows you to access Twitter's 1.1 REST API via a set of GObject based objects for easy integration with your GLib2 based code. ([examples](#))
- [TwitterSearch](#) by @crw\_koepf — Python-based interface to the 1.1 Search API.
- [twython](#) by @ryanmcgrath — Actively maintained, pure Python wrapper for the Twitter API. Supports both normal and streaming Twitter APIs. Supports all v1.1 endpoints, including dynamic functions so users can make use of endpoints not yet in the library. ([docs](#))
- [TwitterAPI](#) by @boxnumber03 — A REST and Streaming API wrapper that supports python 2.x and python 3.x, TwitterAPI also includes iterators for both API's that are useful for processing streaming results as well as paged results.
- [Birdy](#) by @sect2k — “a super awesome Twitter API client for Python”



**GLOBAL INTERNET POPULATION GROWTH 2012-2017  
(IN BILLIONS)**[Learn more at domo.com](#)

SOURCES: STATISTA, LINKEDIN, INTERNET LIVE STATS, EXPANDED RAMBLINGS, SLASH FILM, RIAA, BUSINESS OF APPS, INTERNATIONAL TELECOMMUNICATIONS UNION, INTERNATIONAL DATA CORPORATION

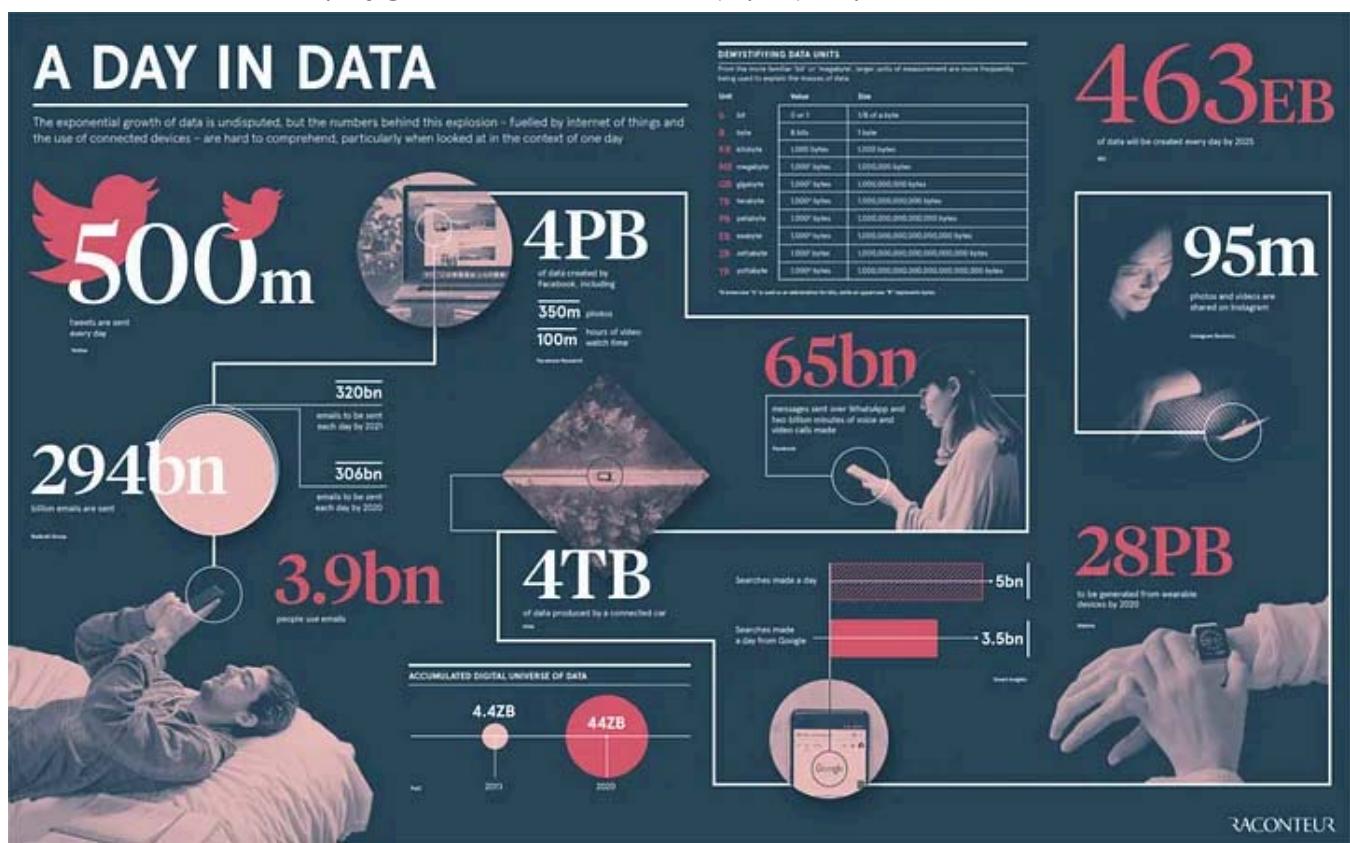


- **6000 tweets every second**
- **473,500 tweets per minute.**
- **500 million tweets are sent every day**
- **200 billion tweets per year.**
- **326 million people use Twitter every month**
- **2.5 quintillion bytes** of data generated every day.
- **50%** of the earth's population is on social media, with a total of **2,789 billion** people.
- By **2025**, it's estimated that **463 exabytes** of data will be created each day globally — that's the equivalent of **212,765,957 DVDs** per day!
- **Social media accounts for 33%** of the total time spent online.

Source: Domo.com

# 2018 This Is What Happens In An Internet Minute





Source:visualcapitalist

Abbreviation	Unit	Value	Size (in bytes)
b	bit	0 or 1	1/8 of a byte
B	bytes	8 bits	1 byte
KB	kilobytes	1,000 bytes	1,000 bytes
MB	megabyte	1,000 <sup>2</sup> bytes	1,000,000 bytes
GB	gigabyte	1,000 <sup>3</sup> bytes	1,000,000,000 bytes
TB	terabyte	1,000 <sup>4</sup> bytes	1,000,000,000,000 bytes
PB	petabyte	1,000 <sup>5</sup> bytes	1,000,000,000,000,000 bytes
EB	exabyte	1,000 <sup>6</sup> bytes	1,000,000,000,000,000,000 bytes
ZB	zettabyte	1,000 <sup>7</sup> bytes	1,000,000,000,000,000,000 bytes
YB	yottabyte	1,000 <sup>8</sup> bytes	1,000,000,000,000,000,000 bytes

Source:visualcapitalist

## 1.Streaming API

The Twitter Search API and Twitter Streaming API work well for individuals that just want to access Twitter data for light analytics or statistical analysis.

Twitter's Streaming API is a push of data as tweets happens in near real-time. The major drawback of the Streaming API is that Twitter's Streaming API provides only a

**sample of tweets** that are occurring. The actual percentage of total tweets users receive with Twitter's Streaming API varies heavily based on the criteria users request and the current traffic. Studies have estimated that using Twitter's Streaming API users can expect to receive 1% of the tweets in near real-time. ([reference website](#))

Before starting install all the required libraries – command prompt using pip

**Please check out more on PIP**

```
pip install vaderSentiment
pip install nltk
pip install Textblob
pip install numpy
pip install pandas
```

After that import all the required libraries

```
1 #Declare the libraries needed
2 import tweepy
3 import pandas as pd
4 import sys
5 import csv
6 from wordcloud import WordCloud, STOPWORDS
7 import matplotlib.pyplot as plt
8 import re
9 from PIL import Image
10 import pandas_profiling
```

tw19 hosted with ❤ by GitHub

[view raw](#)

```
1 # The keys have been displayed here because they are not to be shared.
2 # The value inside quotes must be replaced by your keys if you are using this.
3 import twitter
4 #api = twitter.Api(consumer_key='consumer_key', consumer_secret='consumer_secret',
5 #access_token_key='access_token', access_token_secret='access_token_secret')
6 api = twitter.Api(consumer_key='enter your consumer key',
7                   consumer_secret='enter your consumer secret',
8                   access_token_key='enter your access token',
9                   access_token_secret='enter your secret token')
```

tw1 hosted with ❤ by GitHub

[view raw](#)

First, enter all your credentials. Login into twitter account and go to the app and then fill all the above information.

```
1 #Print your credentials and check
2 print (api.VerifyCredentials())
```

tw2 hosted with ❤ by GitHub

[view raw](#)

Just do print and check your credentials

You can filter by

1. User ID
2. User Name
3. Query by keywords
4. Geo Location

Filter by location and select the tweets I am filtering it by location = San Francisco

```
1 for tweet in api.GetStreamFilter(locations = ['-122.75,36.8,-121.75,37.8']):
2     print (tweet)
3     break
```

tw3 hosted with ❤ by GitHub

[view raw](#)

## locations

A comma-separated list of longitude,latitude pairs specifying a set of bounding boxes to filter Tweets by. Only geolocated Tweets falling within the requested bounding boxes will be included—unlike the Search API, the user's location field is not used to filter Tweets.

Each bounding box should be specified as a pair of longitude and latitude pairs, with the southwest corner of the bounding box coming first. For example:

Parameter value	Tracks Tweets from...
-122.75,36.8,-121.75,37.8	San Francisco
-74,40,-73,41	New York City
-122.75,36.8,-121.75,37.8,-74,40,-73,41	San Francisco OR New York City

Just one tweet contains so much information

```
{
  'created_at': 'Sat May 04 21:14:50 +0000 2019', 'id': 1124784475001384960, 'id_str': '1124784475001384960', 'text': '@AlaskaAir it doesn't seem Alaska airlines is up to meeting the norm for int'l flights-just nuts and chips on your_ https://t.co/WNoEDdoLZU', 'source': '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>', 'truncated': True, 'in_reply_to_status_id': None, 'in_reply_to_status_id_str': None, 'in_reply_to_user_id': 13192972, 'in_reply_to_user_id_str': '13192972', 'in_reply_to_screen_name': 'AlaskaAir', 'user': {'id': 770271189264543744, 'id_str': '770271189264543744', 'name': 'Vietnam-era Vet-PDX', 'screen_name': 'PdxVeteran', 'location': 'Portland, OR', 'url': None, 'description': 'Retired Air Force dude. Bike & walk a ton every day. My church is the outdoors, and I spend lots of time in church 😊 I follow the news & vote 4 progressives!', 'translator_type': 'none', 'protected': False, 'verified': False, 'followers_count': 600, 'friends_count': 1748, 'listed_count': 0, 'favourites_count': 5209, 'statuses_count': 4257, 'created_at': 'Mon Aug 29 14:45:37 +0000 2016', 'utc_offset': None, 'time_zone': None, 'geo_enabled': True, 'lang': 'en', 'contributors_enabled': False, 'is_translator': False, 'profile_background_color': '000000', 'profile_background_image_url': 'http://abs.twimg.com/images/themes/theme1/bg.png', 'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme1/bg.png', 'profile_background_tile': False, 'profile_link_color': '981CEB', 'profile_sidebar_border_color': '000000', 'profile_sidebar_fill_color': '000000', 'profile_text_color': '000000', 'profile_use_background_image': False, 'profile_image_url': 'http://pbs.twimg.com/profile_images/770276640706301952/QG8qVKem_normal.jpg', 'profile_image_url_https': 'https://pbs.twimg.com/profile_images/770276640706301952/QG8qVKem_normal.jpg', 'default_profile': False, 'default_profile_image': False, 'following': None, 'follow_request_sent': None, 'notifications': None}, 'geo': None, 'coordinates': None, 'place': {'id': 'fb6d2f5a4e4a15e', 'url': 'https://api.twitter.com/1.1/geo/id/fb6d2f5a4e4a15e.json', 'place_type': 'admin', 'name': 'California', 'full_name': 'California, USA', 'country_code': 'US', 'country': 'United States', 'bounding_box': {'type': 'Polygon', 'coordinates': [[[[-124.482003, 32.528832], [-124.482003, 42.009519], [-114.131212, 42.009519], [-114.131212, 32.528832]]]}}, 'attributes': {}, 'contributors': None, 'is_quote_status': False, 'extended_tweet': {'full_text': "@AlaskaAir it doesn't seem Alaska airlines is up to meeting the norm for int'l flights-just nuts and chips on your flights from Costa Rica! That's indefensible!!!!", 'display_text_range': [0, 163], 'entities': {'hashtags': [], 'urls': [], 'user_mentions': [{"screen_name": "AlaskaAir", "name": "Alaska Airlines", "id": 13192972, "id_str": "13192972", "indices": [0, 163]}, {"symbols": []}], 'quote_count': 0, 'reply_count': 0, 'retweet_count': 0, 'favorite_count': 0, 'entities': {'hashtags': [], 'urls': [{"url": "https://t.co/WNoEDdoLZU", "expanded_url": "https://twitter.com/i/web/status/1124784475001384960", "display_url": "twitter.com/i/web/status/1124784475001384960", "indices": [116, 139]}]}, 'user_mentions': [{"screen_name": "AlaskaAir", "name": "Alaska Airlines", "id": 13192972, "id_str": "13192972", "indices": [0, 163]}, {"symbols": []}], 'favorited': False, 'retweeted': False, 'filter_level': 'low', 'lang': 'en', 'timestamp_ms': '1557004490982'}
```

## Check out all the tweet objects available

The JSON object looks like

```
1
2     "created_at": "Thu Apr 06 15:24:15 +0000 2017",
3     "id_str": "850006245121695744",
4     "text": "1/\u2019re sharing our vision for the future of the Twitter API platform!\u2019",
5     "user": {
6         "id": 2244994945,
7         "name": "Twitter Dev",
8         "screen_name": "TwitterDev",
9         "location": "Internet",
10        "url": "https://dev.twitter.com/",
11        "description": "Your official source for Twitter Platform news, updates & events. Need tech",
12    },
13    "place": {
14    },
15    "entities": {
16        "hashtags": [
17        ],
18        "urls": [
19            {
20                "url": "https://t.co/XweGngmxlP",
21                "unwound": {
22                    "url": "https://cards.twitter.com/cards/18ce53wgo4h/3xo1c",
23                    "title": "Building the Future of the Twitter API Platform"
24                }
25            }
26        ],
27        "user_mentions": [
28        ]
29    }
30 }
```

tw18 hosted with ❤ by GitHub

[view raw](#)

You can also save it to the CSV file

```

1 import csv
2 geo_stream = api.GetStreamFilter(locations = ['-122.75,36.8,-121.75,37.8'])
3 #     print (tweet)
4 with open('tweets_sv1.csv', 'w+') as csv_file:
5     csv_writer = csv.writer(csv_file)
6     for line in geo_stream:
7         #Signal that the line represents a tweet
8         tweet = twitter.Status.NewFromJsonDict(line)
9         row = [tweet.id, tweet.user.screen_name, tweet.text]
10        csv_writer.writerow(row)
11    break

```

tw6 hosted with ❤ by GitHub

[view raw](#)

The output of the 1 tweet with selected info

```

tweets_sv1.csv - Notepad
File Edit Format View Help
1124796546149601283,irapolis,@woolie More so than anywhere else, if I'm not mistaken.

```

### For more info on the streaming API

The output from the Twitter API is in JSON format. So what is JSON

Please check this youtube video.

Please check the data camp article for more info on JSON file and python

Now we can download the file in JSON format

```

1 # lets save the json data to a file: stream.json
2 # "\n" is new line. We save the data to a file,
3 # with each new line we save all the information related to one tweet
4 import json
5 f = open('./stream1.json', 'w')
6 for tweet in api.GetStreamFilter(locations = ['-122.75,36.8,-121.75,37.8']):
7     f.write(json.dumps(tweet))
8     f.write('\n')
9     print (tweet)

```

tw7 hosted with ❤ by GitHub

[view raw](#)

Upload the JSON file to a list.

```

1 #Upload the json file and append it to a list
2 import json
3 data=[]
4 with open('./streamingData1.json', 'r') as jsonFile:
5     for line in jsonFile:
6         data.append(json.loads(line))
7 print ("Total number of tweets loaded: ", len(data))
8 print(data)

```

tw8 hosted with ❤ by GitHub

[view raw](#)

Check the keys and the tweet itself

```

1 print (data[0].keys())    # properties
2 print (data[0]["text"])   # how to access the tweet message itself

```

tw9 hosted with ❤ by GitHub

[view raw](#)

The output looks like

```

dict_keys(['created_at', 'id', 'id_str', 'text', 'source', 'truncated', 'in_reply_to_status_id', 'in_reply_to_status_id_str',
'in_reply_to_user_id', 'in_reply_to_user_id_str', 'in_reply_to_screen_name', 'user', 'geo', 'coordinates', 'place', 'contributors',
'quoted_status_id', 'quoted_status_id_str', 'quoted_status', 'quoted_status_permalink', 'is_quote_status', 'quote_count',
'reply_count', 'retweet_count', 'favorite_count', 'entities', 'favorited', 'retweeted', 'filter_level', 'lang', 'timestamp_millis'])
CUTTING HUMOR
"胸怀大志"

```

The dictionary consists of a lot of information like created, user ID, retweeted, timestamp, etc. Just read from the dictionary what information you need and then proceed with the text mining and visualization.

## 2. Search API or REST API

Twitter's Search API gives you access to a data set that already exists from tweets that have occurred. For an individual user, the maximum number of tweets you can receive is the last 3,200 tweets, regardless of the query criteria. With a specific keyword, you can typically only poll the last 5,000 tweets per keyword. You are further limited by the number of requests you can make in a certain time period. The Twitter request limits have changed over the years but are currently limited to 180 requests in a 15 minute period.

Let's use tweepy for search API.

1. pip install tweepy at the command prompt

```
(base) C:\>pip install tweepy
Requirement already satisfied: tweepy in c:\users\sesakkiappa
0)
Requirement already satisfied: requests>=2.11.1 in c:\users\s
ages (from tweepy) (2.18.4)
Requirement already satisfied: six>=1.10.0 in c:\users\sesakk
(from tweepy) (1.11.0)
Requirement already satisfied: PySocks>=1.5.7 in c:\users\ses
es (from tweepy) (1.6.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:
ite-packages (from tweepy) (1.2.0)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\us
-packages (from requests>=2.11.1->tweepy) (3.0.4)
Requirement already satisfied: idna<2.7,>=2.5 in c:\users\ses
es (from requests>=2.11.1->tweepy) (2.6)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in c:\us
-packages (from requests>=2.11.1->tweepy) (1.22)
Requirement already satisfied: certifi>=2017.4.17 in c:\users
ckages (from requests>=2.11.1->tweepy) (2018.4.16)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\se
ges (from requests-oauthlib>=0.7.0->tweepy) (3.0.1)
You are using pip version 19.0.3, however version 19.1 is ava
You should consider upgrading via the 'python -m pip install
```

Since I already installed it says that Requirement already satisfied.

2. Import the libraries needed.

```
1 #Declare the libraries needed
2 import tweepy
3 import pandas as pd
4 import sys
5 import csv
6 from wordcloud import WordCloud, STOPWORDS
7 import matplotlib.pyplot as plt
8 import re
9 from PIL import Image
10 import pandas_profiling
```

tw19 hosted with ❤️ by GitHub

[view raw](#)

### 3. Provide all the credentials you got before

```
1 # Fill the credentials
2 # following the above mentioned procedure.
3 api = twitter.Api(consumer_key='Enter your consumer key',
4                     consumer_secret='Enter your consumer secret',
5                     access_token_key='Enter your access token',
6                     access_token_secret='enter your access token')
7
```

tw10 hosted with ❤️ by GitHub

[view raw](#)

### 4. Now we are going to extract the tweets The main function is below. I am extracting the tweets of Bill Gates. *The maximum tweets allowed is 3200*. I can't extract the whole tweet history of the user if they have tweeted more than 3200 tweets.

```
1 import tweepy
2 import pandas as pd
3 import sys
4 import csv
5
6 # Fill the X's with the credentials obtained by
7 # following the above mentioned procedure.
8
9 api = twitter.Api(consumer_key='Please enter your credentials',
10                     consumer_secret='Please enter your credentials',
11                     access_token_key='Please enter your credentials',
12                     access_token_secret='Please enter your credentials')
13
14 # Function to extract tweets
15 def get_tweets(username):
16
17     # Authorization to consumer key and consumer secret
18     auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
19
20     # Access to user's access key and access secret
21     auth.set_access_token(access_token_key, access_token_secret)
22
23     # Calling api
24     api = tweepy.API(auth)
25     #set count to however many tweets you want - max count is 3200 and this doesn't have an
26     # number_of_tweets = 5000
27
28     tfile = []
29     for tweet in tweepy.Cursor(api.user_timeline, screen_name = username).items():
30
31         #username, tweet id, date/time, text
32         tfile.append([username, tweet.id_str,tweet.source, tweet.created_at,tweet.retweet_c
33
34         #write to a new csv file from the array of tweets
35         outfile = username + "_tweets_V1.csv"
36         print ("writing to " + outfile)
37         with open(outfile, 'w+') as file:
38             writer = csv.writer(file, delimiter=',')
39             writer.writerow(['User_Name', 'Tweet_ID', 'Source', 'Created_date','Retweet_count',
40             writer.writerows(tfile)
41     # user name
42     get_tweets("@BillGates")
```



20. Bill Gates  
@BillGates

**Bio:** Sharing things I'm learning through my foundation work and other interests.  
**Location:** Seattle, WA

followers 47,134,550	following 196	tweets 3,057
-------------------------	------------------	-----------------

The downloaded file looks like below

---

User\_Name,Tweet\_ID,Source,created\_date,retweet\_count,favorite\_count,tweet

@BillGates,1124761460964458497,Sprinklr,2019-05-04 19:43:24,492,4536,Melinda and I love meeting with people who are trying to change the world. Here are four of the people who inspired... <https://t.co/MrmYszRNMG>

@BillGates,1124310352848674816,Sprinklr,2019-05-03 13:50:51,905,4574,What does a "dirt detective" do for a living? I'm eager to watch this talk from soil scientist Asmeret Asefaw Berhe... <https://t.co/Rw3GrZ1V3q>

@BillGates,1124070620722094087,Twitter Web Client,2019-05-02 21:58:14,301,2241,"A few months ago, Fred contributed a piece to my blog which outlined why educating the next generation of African l... <https://t.co/jPqGy50hR3>"

@BillGates,1124070618553618432,Twitter Web Client,2019-05-02 21:58:14,711,3570,"By the end of the century, almost half of the young people in the world will

5. Now upload the data into a panda data frame for data visualization and sentiment analysis

```

1 #Read the file and also assign it to dataframe
2 import pandas as pd
3 bg= pd.read_csv("@BillGates_tweets_V4.csv",encoding='utf-8')
4 bg.head(10)

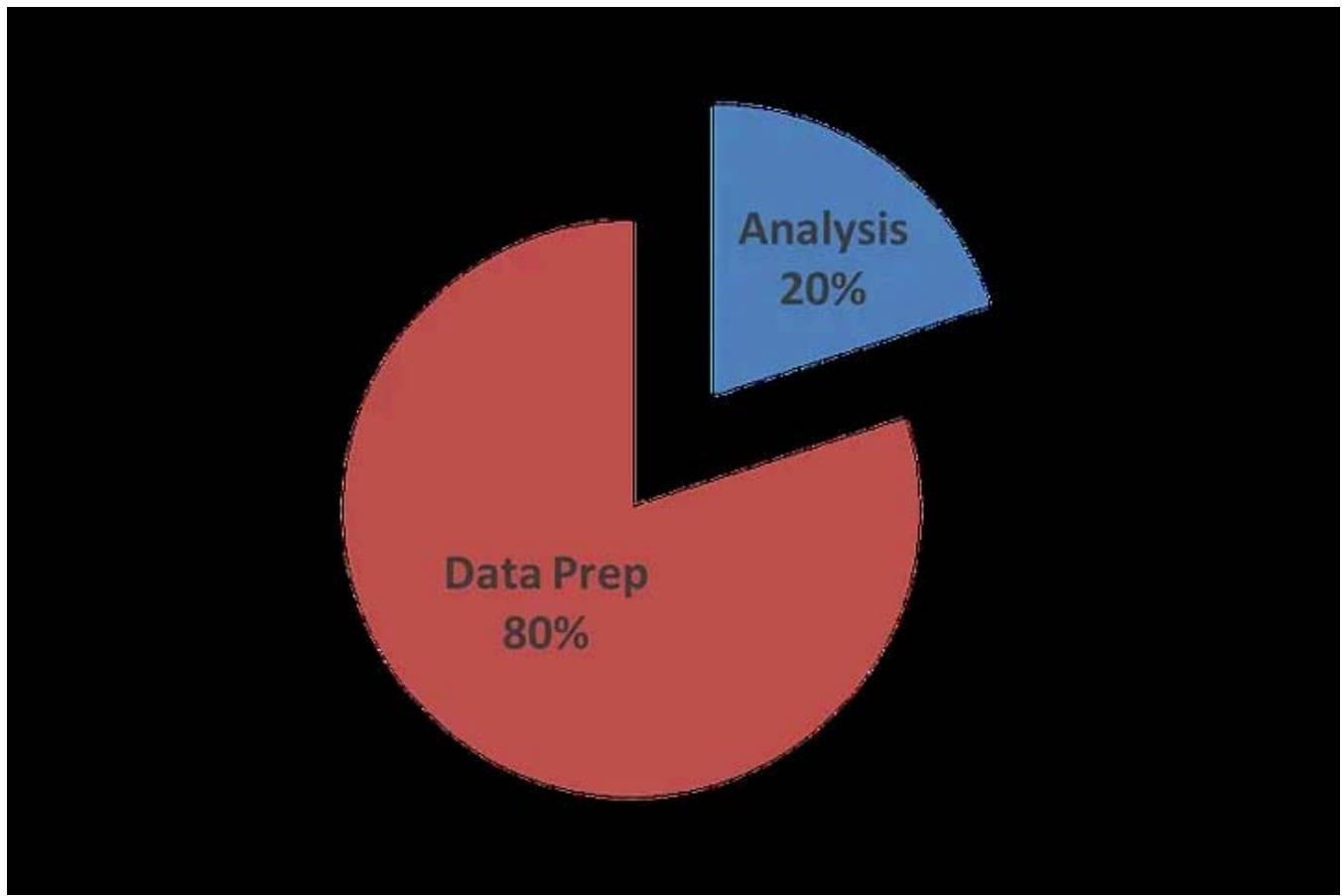
```

tw12 hosted with ❤ by GitHub

[view raw](#)

The data looks like below in the dataframe

	User_Name	Tweet_ID	Source	created_date	retweet_count	favorite_count	tweet
0	@BillGates	1124761460964458497	Sprinklr	2019-05-04 19:43:24	492	4536	Melinda and I love meeting with people who are...
1	@BillGates	1124310352848674816	Sprinklr	2019-05-03 13:50:51	905	4574	What does a "dirt detective" do for a living? ...
2	@BillGates	1124070620722094087	Twitter Web Client	2019-05-02 21:58:14	301	2241	A few months ago, Fred contributed a piece to ...
3	@BillGates	1124070618553618432	Twitter Web Client	2019-05-02 21:58:14	711	3570	By the end of the century, almost half of the ...
4	@BillGates	1124070613344292871	Twitter Web Client	2019-05-02 21:58:13	576	3869	...@FredSwaniker has an ambitious plan to unlock...
5	@BillGates	1123977200708018176	Sprinklr	2019-05-02 15:47:01	521	4229	It's hard for me to overstate how brave people...
6	@BillGates	1123726002654515202	Twitter Web Client	2019-05-01 23:08:51	228	2815	Congratulations @SueDHeilmann on your 5th anni...
7	@BillGates	1123625634943635457	Sprinklr	2019-05-01 16:30:01	608	2625	The world's historic progress to #endpolio wou...
8	@BillGates	1123263708258377730	Twitter Media Studio	2019-04-30 16:31:51	1122	5769	If you enjoyed the @Avengers this weekend, I t...
9	@BillGates	1123036364130406400	Sprinklr	2019-04-30 01:28:28	1592	7730	It's amazing to think about the creative ways ...



About 80% of the time is spent in preparing the data and only 20% in the analysis.

## 6. Do some basic cleaning of the tweets

```
1  bg2 = []
2  import re
3  pattern1 = re.compile(" ' # $ % & ' ( ) * + , - . / : ; < = > @ [ / ] ^ _ { | } ~")
4  pattern2 = re.compile("@[A-Za-z0-9]+")
5  pattern3 = re.compile("https?:///[A-Za-z0-9./]+")
6
7  for item in bg1:
8      tweet = re.sub(pattern1, "", item)    # version 1 of the tweet
9      tweet = re.sub(pattern2, "", tweet)
10     tweet = re.sub(pattern3, "", tweet)
11     bg2.append(tweet)
12
13 bg3 = pd.DataFrame(bg2,columns = ['tweet'])
```

tw13 hosted with ❤ by GitHub

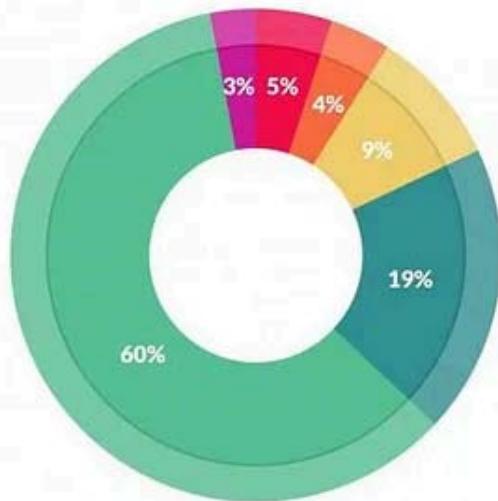
[view raw](#)

[Open in app ↗](#)



Search





#### What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

Source: Forbes

## 7. Now we can do basic data analysis and visualization

Let's do a word cloud. [Check the documentation for more info.](#)

The only required parameter is the text and all others are optional.

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib as mpl
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 from subprocess import check_output
7 from wordcloud import WordCloud, STOPWORDS
8
9 mpl.rcParams['figure.figsize']=(16.0,10.0)
10 mpl.rcParams['font.size']=12
11 mpl.rcParams['savefig.dpi']=1400
12 mpl.rcParams['figure.subplot.bottom']=.1
13
14 stopwords = set(STOPWORDS)
15 text = " ".join(tweet for tweet in bg3.tweet)
16 print ("There are {} words in the combination of all tweets.".format(len(text)))
17
18 wordcloud = WordCloud(
19                     background_color='white',
20                     stopwords=stopwords,
21                     max_words=200,
22                     max_font_size=40,
23                     random_state=42
24                 ).generate(str(text))
25
26 print(wordcloud)
27 fig = plt.figure(1)
28 plt.imshow(wordcloud)
29 plt.axis('off')
30 plt.show()
31 fig.savefig("word1.png", dpi=1400)
```

tw14 hosted with ❤ by GitHub

[view raw](#)

The output is



You can also do a mask

The original picture is



and the wordcloud



and the code is



**There are 292731 words in the combination of all tweets.**

### **Scatter Text**

A tool for finding distinguishing terms in small-to-medium-sized corpora, and presenting them in a sexy, interactive scatter plot with non-overlapping term labels. Exploratory data analysis just got more fun.

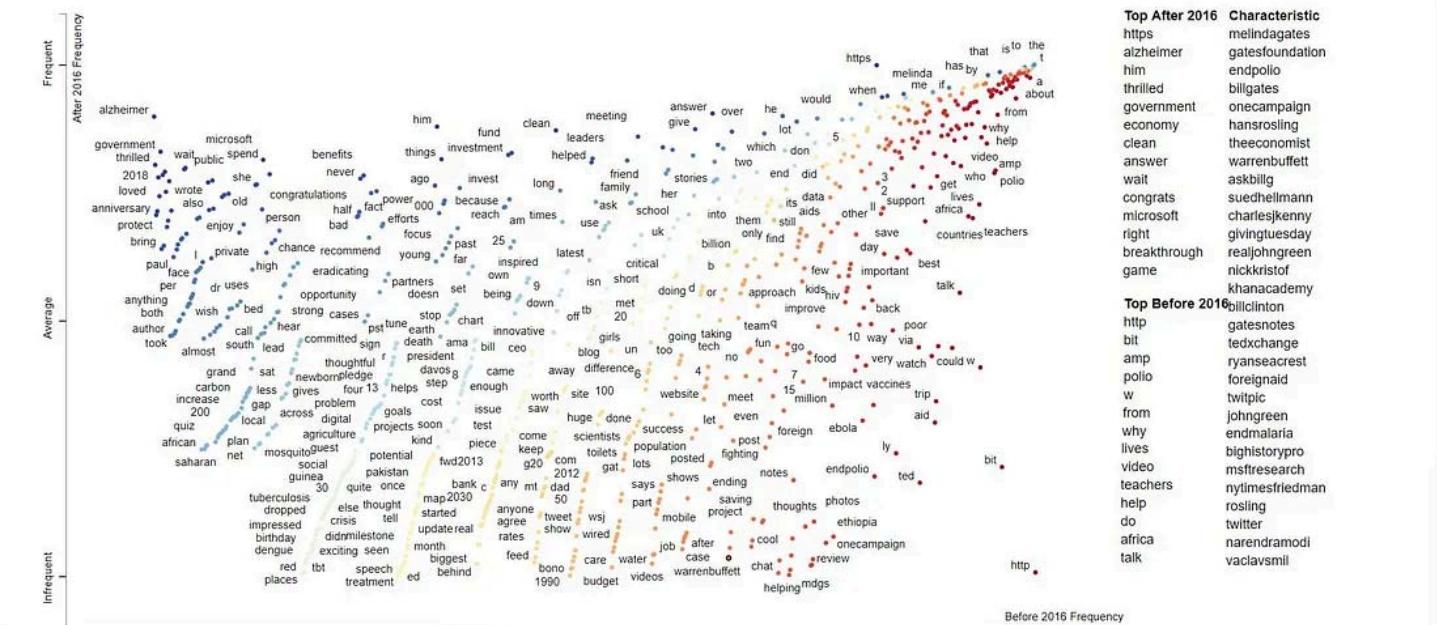
### **Check out the GitHub link**

The visual is stunning. I tried to just use the tweets to do a scatter text. Please refer the GitHub for sample code and scenarios. I think definitely my scatter text can be

improved.

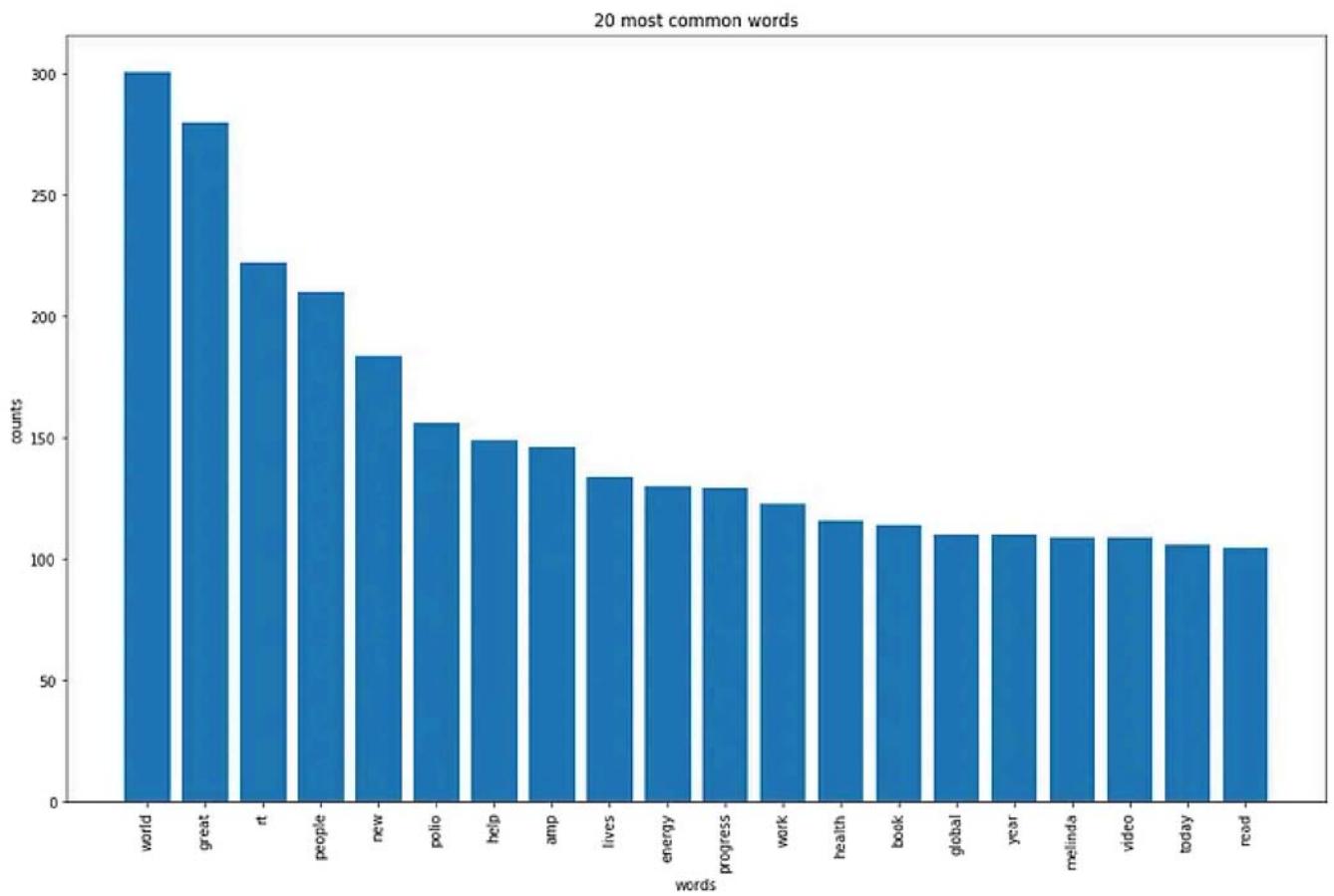
The code is

I love the graph in the HTML display. I have attached a screenshot.



## Top 20 common words in the tweets





Now do some basic analysis. Find the most liked tweets and most retweeted tweets.

The code is

## The output

The tweet with more likes is:

Congratulations to the Indian government on the first 100 days of @AyushmanNHA. It's great to see how many people h...

<https://t.co/mGXaz16H7W>

**Number of likes: 56774**

The tweet with more retweets is:

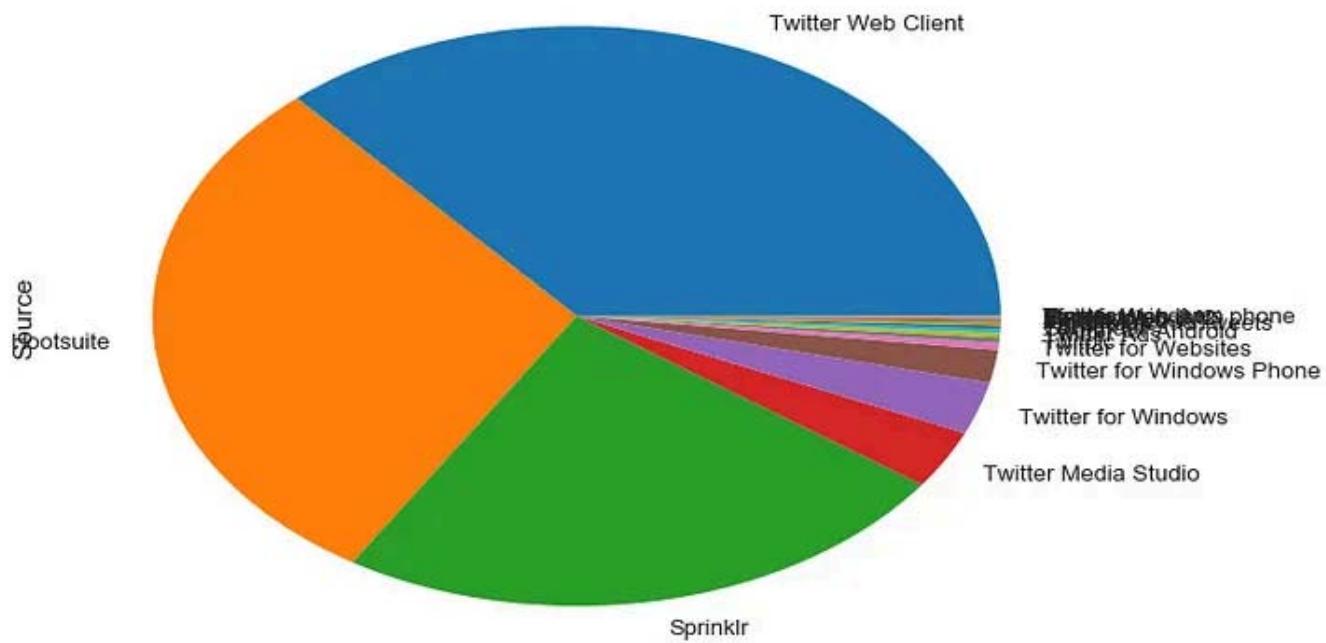
RT @WarrenBuffett: Warren is in the house.

**Number of retweets: 39904**

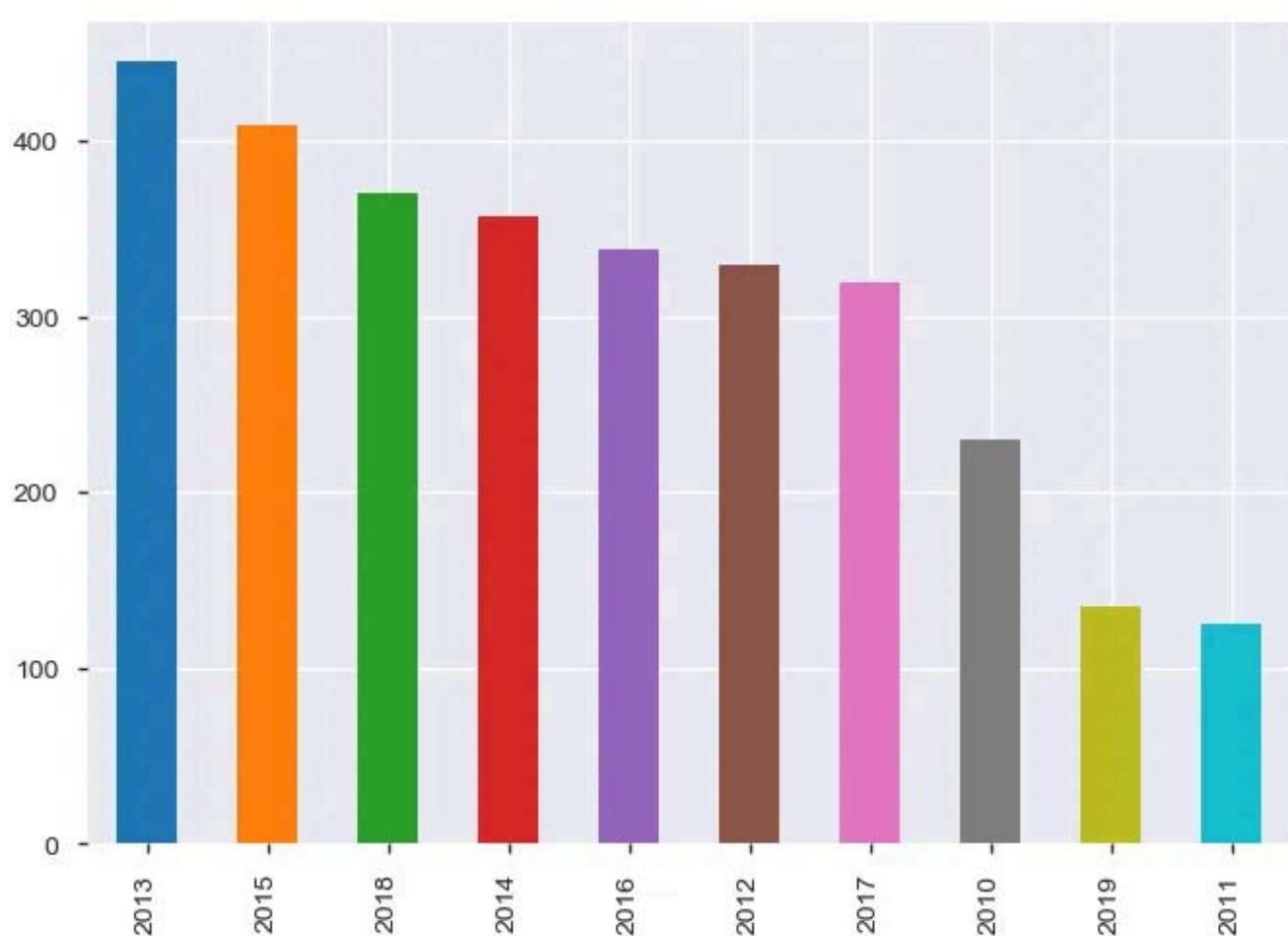
The most liked tweet is

Source of the tweet — Obviously no *iPhone and Android* 😊

<b>Twitter Web Client</b>	<b>1115</b>
<b>Hootsuite</b>	<b>907</b>
<b>Sprinklr</b>	<b>733</b>
<b>Twitter Media Studio</b>	<b>100</b>
<b>Twitter for Windows</b>	<b>89</b>
<b>Twitter for Windows Phone</b>	<b>56</b>
<b>Twitter for Websites</b>	<b>11</b>
<b>Twitpic</b>	<b>10</b>
<b>Twitter Ads</b>	<b>8</b>
<b>Spredfast</b>	<b>6</b>
<b>Twitter for Android</b>	<b>6</b>
<b>Panoramic moTweets</b>	<b>5</b>
<b>Mobile Web</b>	<b>3</b>
<b>Seesmic Look</b>	<b>3</b>
<b>Yfrog</b>	<b>1</b>
<b>Vine for windows phone</b>	<b>1</b>
<b>Facebook</b>	<b>1</b>
<b>Twitter Web App</b>	<b>1</b>
<b>Mobile Web (M2)</b>	<b>1</b>
<b>Name: Source, dtype: int64</b>	



## Tweets by Year

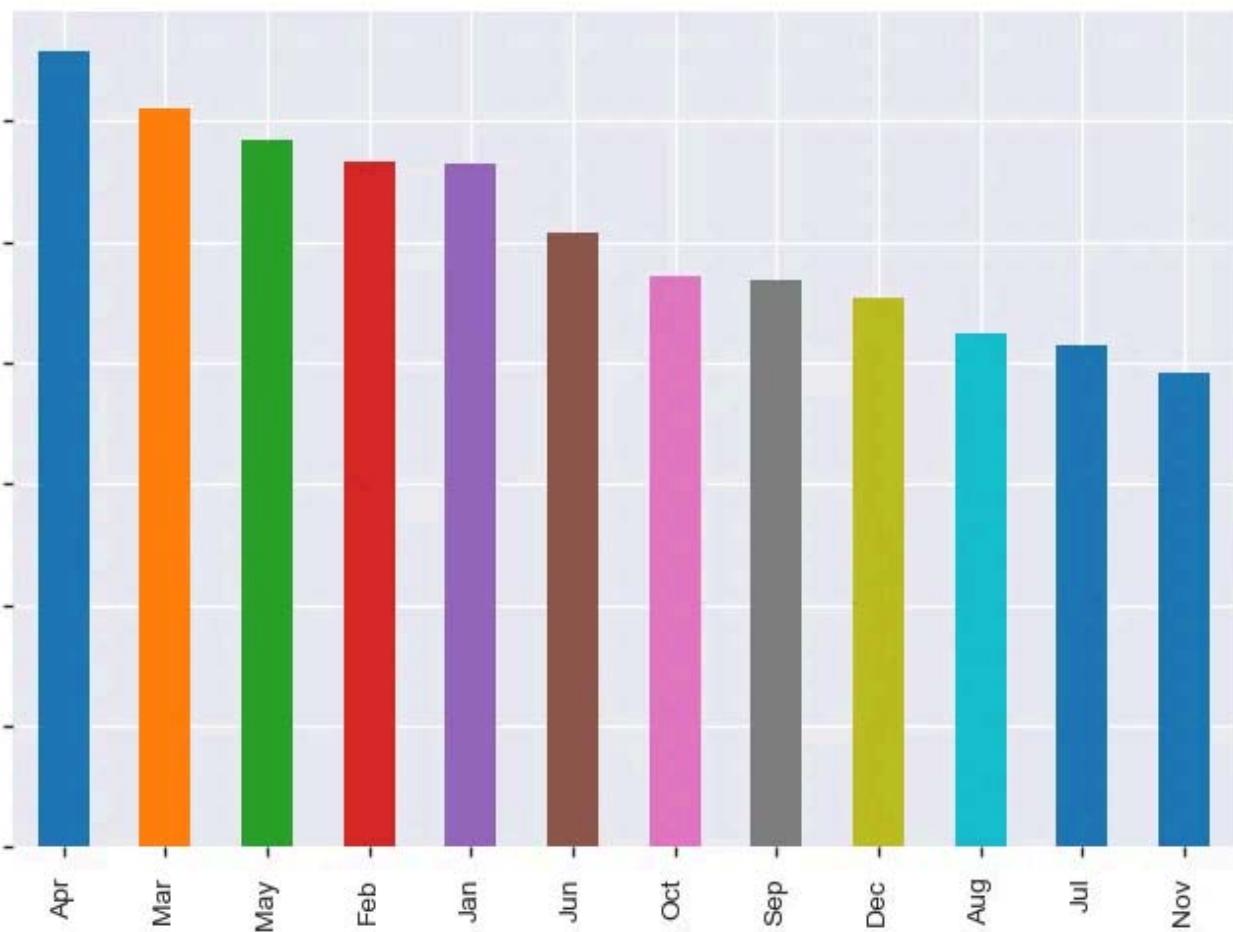


2013 is the year in which Gates tweeted 445 tweets and 409 in 2015. The lowest is 2011 with 125 tweets.

<b>2013</b>	<b>445</b>
<b>2015</b>	<b>409</b>
<b>2018</b>	<b>370</b>
<b>2014</b>	<b>357</b>
<b>2016</b>	<b>338</b>
<b>2012</b>	<b>329</b>
<b>2017</b>	<b>319</b>
<b>2010</b>	<b>230</b>
<b>2019</b>	<b>135</b>
<b>2011</b>	<b>125</b>

## Tweets by Month

April is the month he tweeted the most with 329 and the lowest is November with 196

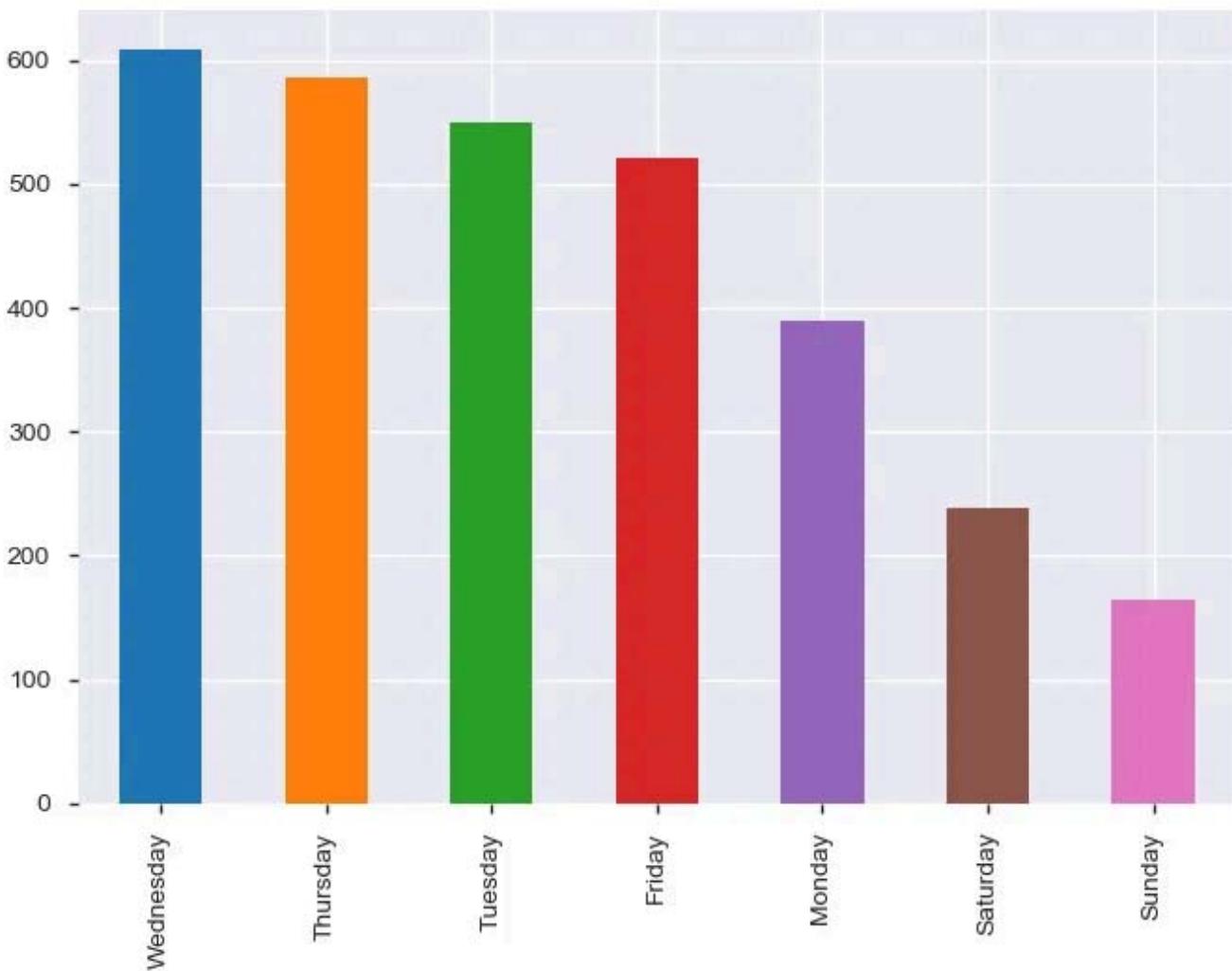


<b>Apr</b>	<b>329</b>
<b>Mar</b>	<b>305</b>
<b>May</b>	<b>292</b>
<b>Feb</b>	<b>283</b>
<b>Jan</b>	<b>282</b>
<b>Jun</b>	<b>254</b>
<b>Oct</b>	<b>236</b>

<b>Sep</b>	<b>234</b>
<b>Dec</b>	<b>227</b>
<b>Aug</b>	<b>212</b>
<b>Jul</b>	<b>207</b>
<b>Nov</b>	<b>196</b>

## Tweets by day

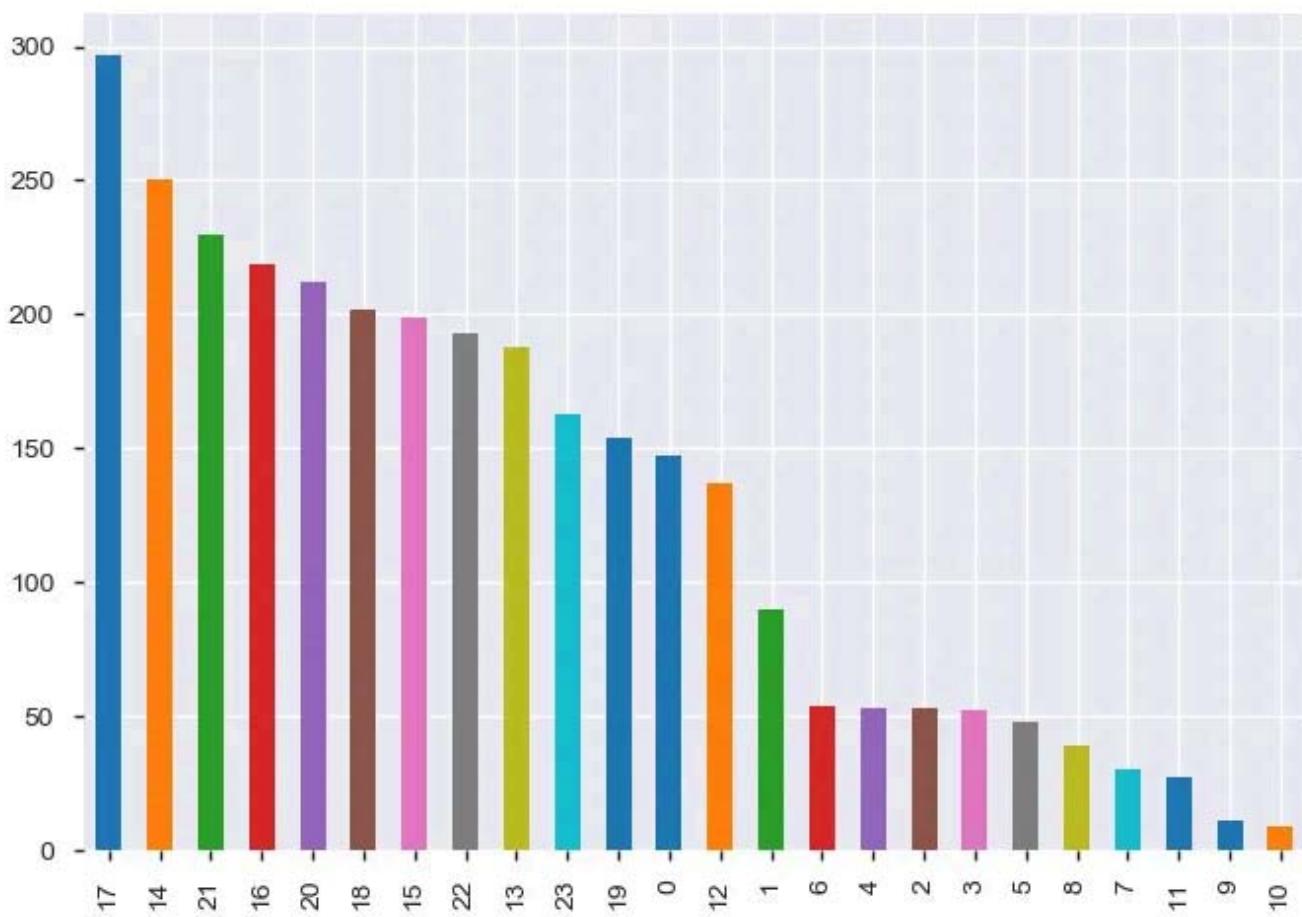
Wednesday is the day in which he tweeted almost 609 tweets and next is Thursday with 586. The weekends are the lowest 😊



<b>Wednesday</b>	<b>609</b>
<b>Thursday</b>	<b>586</b>
<b>Tuesday</b>	<b>549</b>
<b>Friday</b>	<b>521</b>
<b>Monday</b>	<b>389</b>
<b>Saturday</b>	<b>239</b>
<b>Sunday</b>	<b>164</b>

## Tweets by the hour:

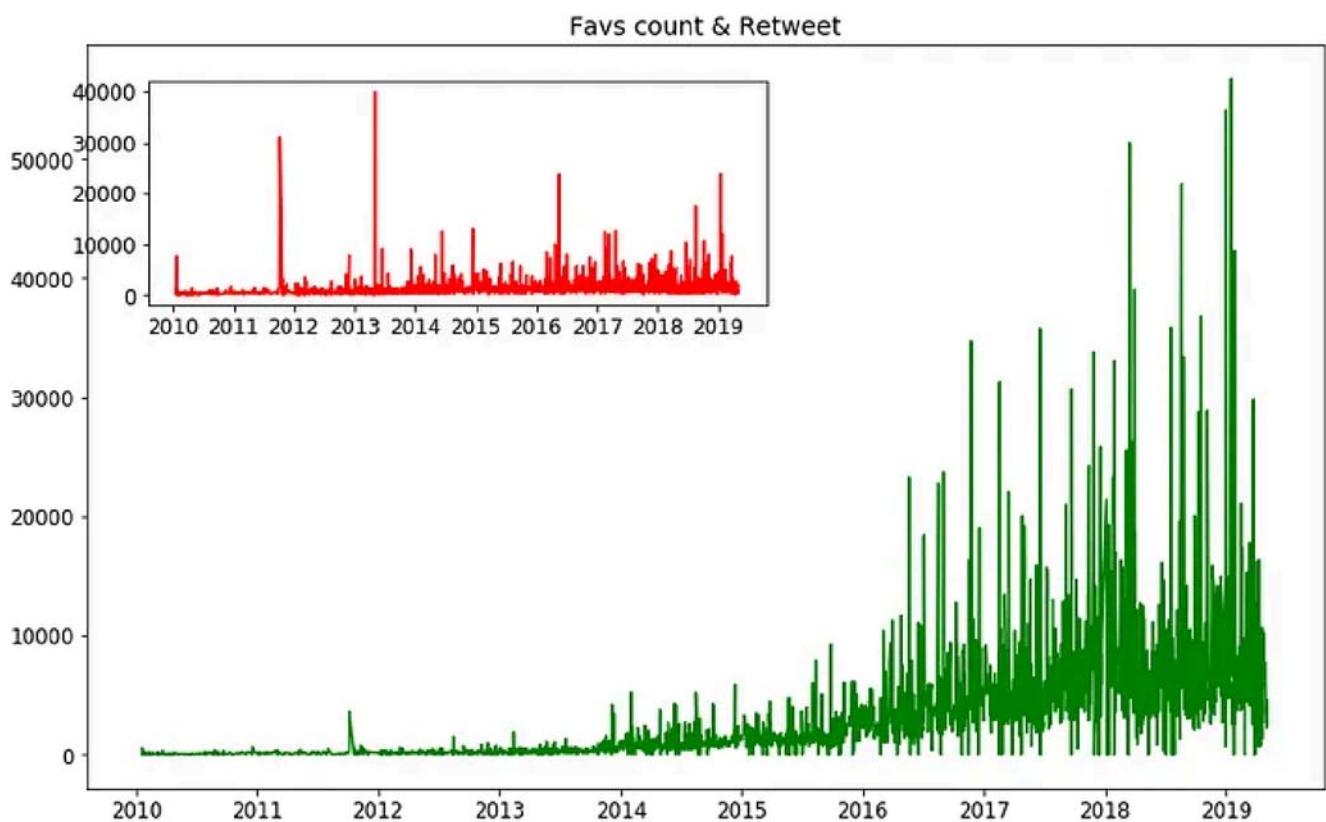
Most of the tweets are in the PM. The maximum is around 5.00 PM with 297. Even he tweeted at midnight or 1.00 AM 😊 .AM tweets are very less.



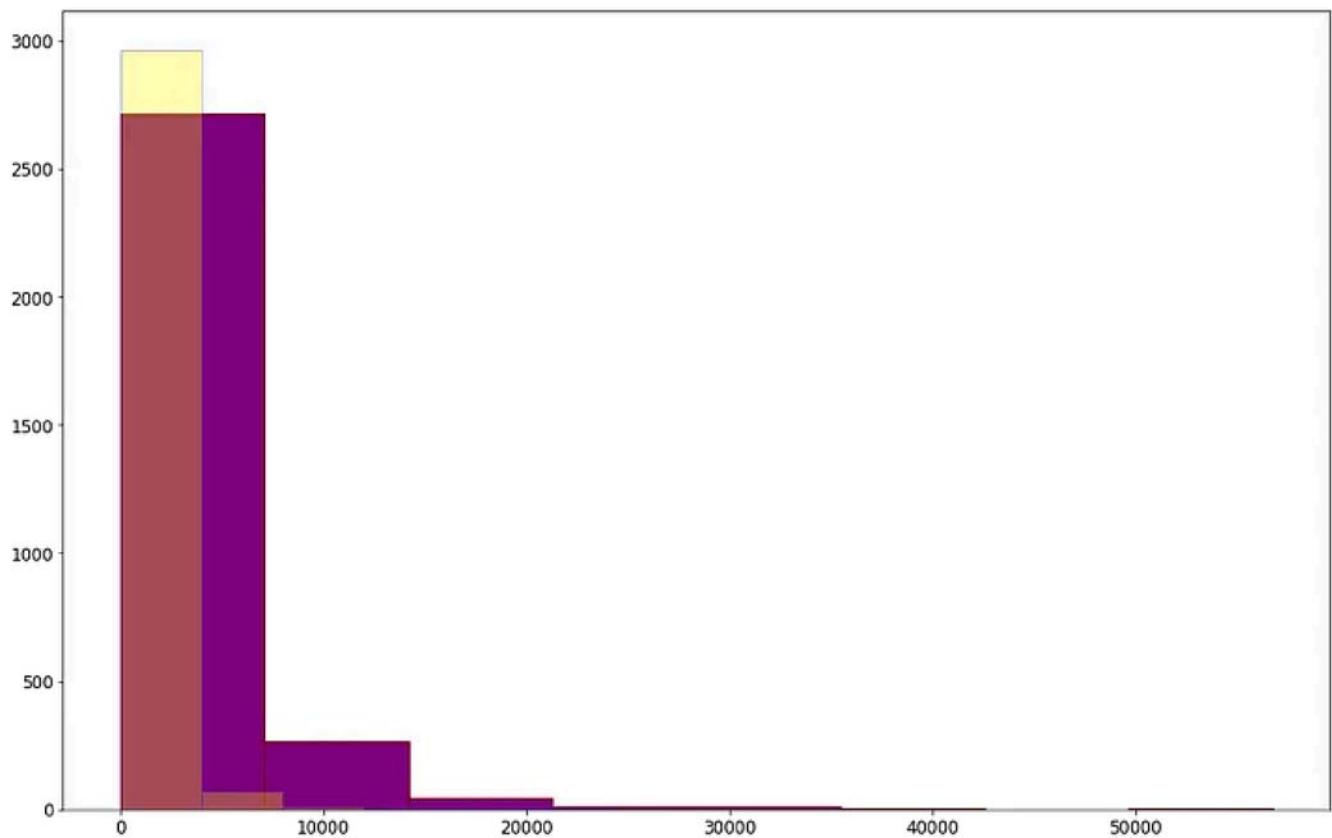
<b>17</b>	<b>297</b>
<b>14</b>	<b>250</b>
<b>21</b>	<b>230</b>
<b>16</b>	<b>219</b>
<b>20</b>	<b>212</b>
<b>18</b>	<b>202</b>
<b>15</b>	<b>199</b>
<b>22</b>	<b>193</b>
<b>13</b>	<b>188</b>
<b>23</b>	<b>163</b>
<b>19</b>	<b>154</b>
<b>0</b>	<b>147</b>
<b>12</b>	<b>137</b>
<b>1</b>	<b>90</b>
<b>6</b>	<b>54</b>
<b>4</b>	<b>53</b>
<b>2</b>	<b>53</b>
<b>3</b>	<b>52</b>
<b>5</b>	<b>48</b>
<b>8</b>	<b>39</b>
<b>7</b>	<b>30</b>
<b>11</b>	<b>27</b>

9	11
10	9

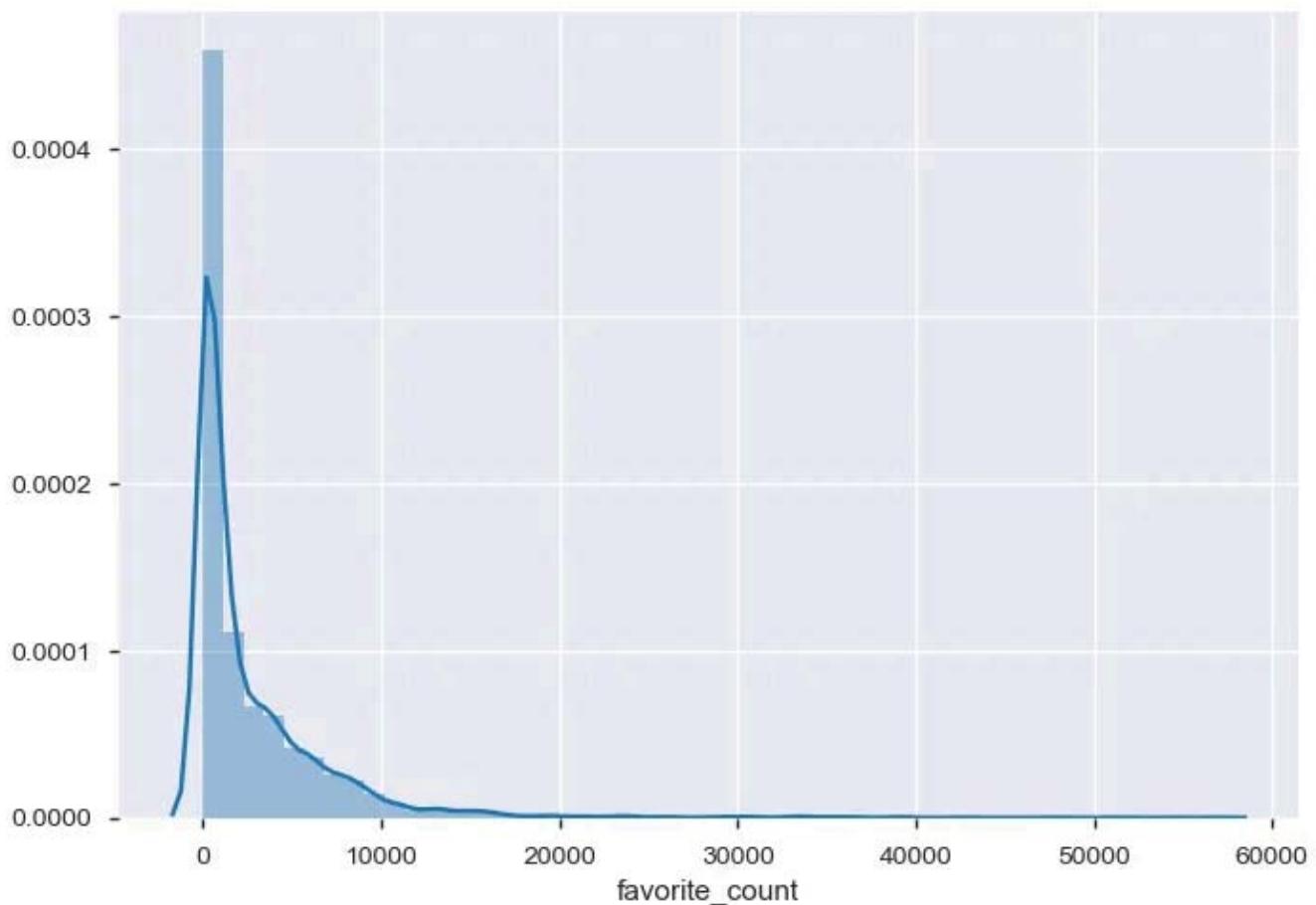
## Fav count & Retweets over the years

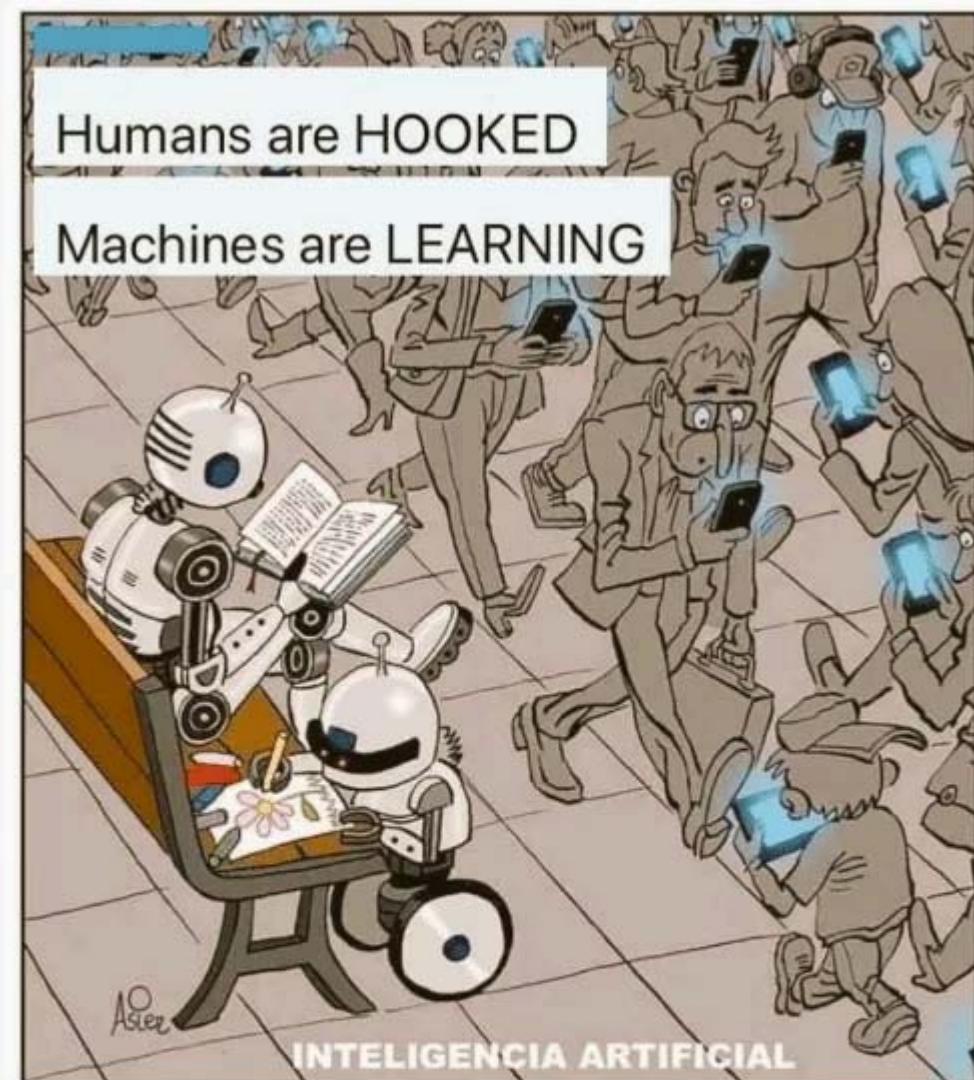


## Fav and Retweet Histogram



Fav count — Seaborn





Machines are LEARNING,  
Humans are HOOKED. Received  
as a WhatsApp forward from a  
friend.

## Sentiment Analysis



Image Source: Medium Article

## What is the Sentiment Analysis?

Sentiment Analysis also is known as *Opinion Mining* is a field within Natural Language Processing (NLP) that builds systems that try to identify and extract opinions within the text. Usually, besides identifying the opinion, these systems extract attributes of the expression e.g.:

- *Polarity*: if the speaker expresses a *positive* or *negative* opinion,
- *Subject*: the thing that is being talked about,
- *Opinion holder*: the person, or entity that expresses the opinion.

With the help of sentiment analysis systems, this unstructured information could be automatically transformed into structured data of public opinions about products, services, brands, politics, or any topic that people can express opinions about. This data can be very useful for commercial applications like marketing analysis, public relations, product reviews, net promoter scoring, product feedback, and customer service.

## Some of the Python Sentiment Analysis API's & Libraries

- Scikit-learn
- NLTK

- SpaCy
- TensorFlow
- Keras
- PyTorch
- Gensim
- Polglot
- TextBlob
- Pattern
- Vocabulary
- Stanford CoreNLP Python
- Montylingua

### **Sentiment Analysis by TextBlob:**

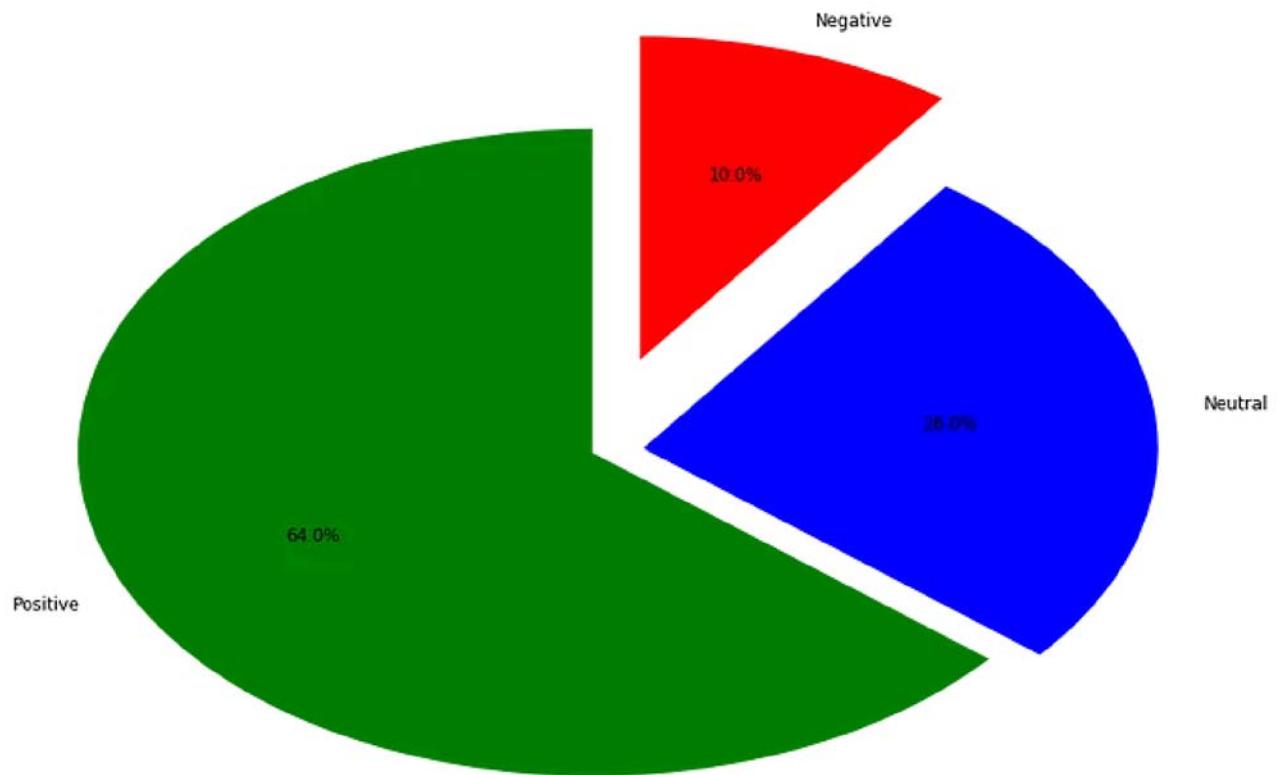
*TextBlob* is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

The code is

**Percentage of positive tweets: 63.23192672554792%**  
**Percentage of neutral tweets: 26.66012430487406%**  
**Percentage of negative tweets: 10.107948969578018%**

The output is above

The pie chart is



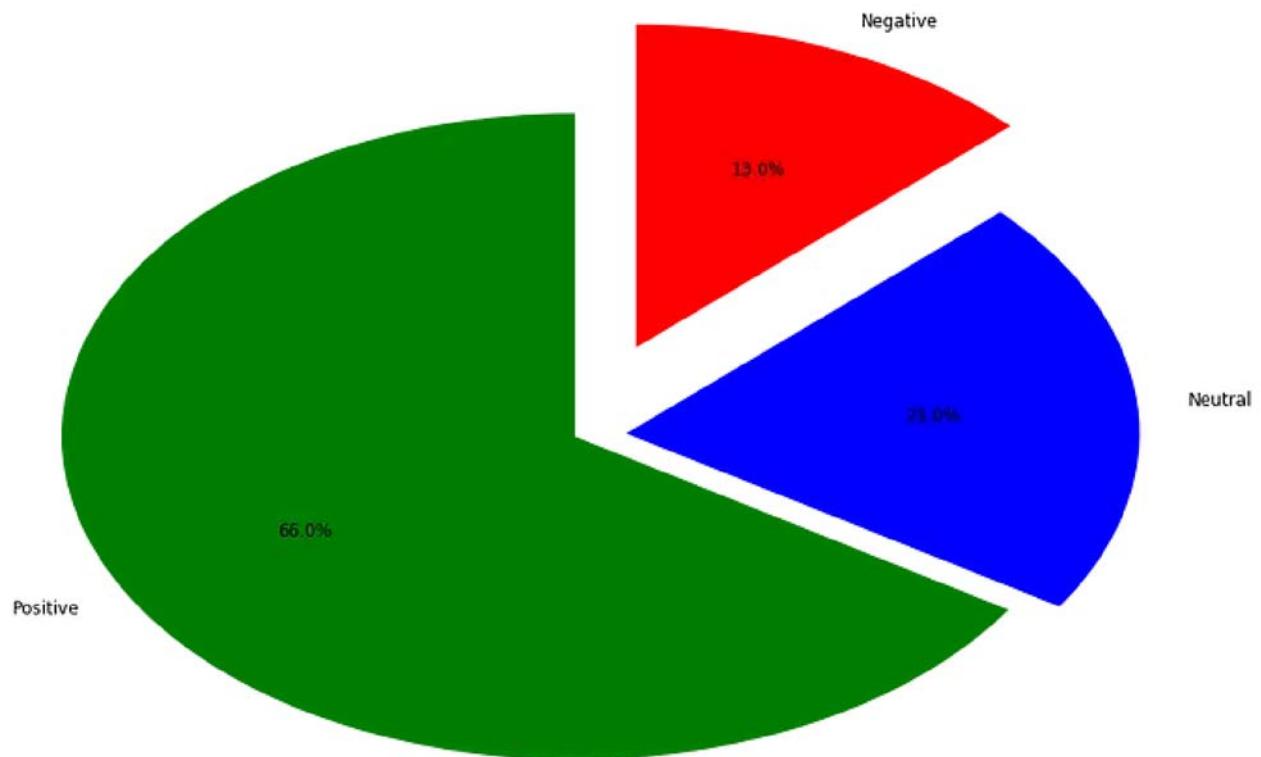
### **Sentiment Analysis by Vader:**

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is *specifically attuned to sentiments expressed in social media*. It is fully open-sourced under the [\[MIT License\]](#)

[For more information please check this GitHub link](#)

The code is similar to the above in calculating the sentiment score. Only the library is different. We are considering only the compound score.

**Percentage of positive tweets: 66.47039581288846%**  
**Percentage of neutral tweets: 20.870134118416747%**  
**Percentage of negative tweets: 12.659470068694798%**



	tweet	SA	VSA
0	Melinda and I love meeting with people who are trying to change the world. Here are four of the people who inspired...	1	1
1	What does a "dirt detective" do for a living? I'm eager to watch this talk from soil scientist Asmeret Asefaw Berhe...	0	1
2	A few months ago, Fred contributed a piece to my blog which outlined why educating the next generation of African I...	-1	0
3	By the end of the century, almost half of the young people in the world will be in sub-Saharan Africa. -1 0	-1	0
4	, has an ambitious plan to unlock the talents of young Africans. I was thrilled to see him featured in...	1	1
5	It's hard for me to overstate how brave people who fight outbreaks are. They face a near-impossible task with grace...	1	1
6	Congratulations on your 5th anniversary . Your leadership on some of the world's tou...	0	1
7	The world's historic progress to #endpolio would not be possible without the strong commitment of partners like...	1	-1
8	If you enjoyed the this weekend, I think you'll be inspired by these real-life heroes:...	1	1
9	It's amazing to think about the creative ways we've delivered vaccines over the years (even by dog sled, when neces...	1	1
10	The promise and perils of gene-editing might be the most important public debate we haven't been having widely enou...	1	1
11	One of my favorite authors, , co-wrote a piece about why nuclear power is so important in fighting climate...	1	1
12	Deaths from malaria have dropped 42% since 2000. I'm as optimistic as ever that we can get to zero deaths in our ge...	0	1
13	Stories like this one from Kenya remind me of why Melinda and I started our foundation. I'm optimistic we will cont...	0	1
14	Melinda's new book, The #MomentofLift, is out today: \n\n\nShe combines her mastery of data wit...	1	0
15	On #EarthDay, I'm inspired by the inventors who are tackling climate change and all the people who are supporting t...	1	1
16	Melinda's new book The #MomentofLift is an urgent call to action. I'll commit to the lift by working to ensure that...	1	1
17	Gene drive is a technique where you give a few mosquitoes an edited gene and it drives itself into all their offspr...	-1	0
18	Less than a century ago, families everywhere lived in fear of a mosquito bite. Now 75% of all malaria cases occur i...	-1	-1
19	3: Which of the following factors affect the spread of malaria? The answer is in my latest blog post: 1 0	1	0
20	2: What kind of pathogen causes malaria? Find the answer here: 1 0	1	0

### SA — Sentiment Analysis Score from Textblob

## VSA — *Sentiment Analysis Score from Vader*

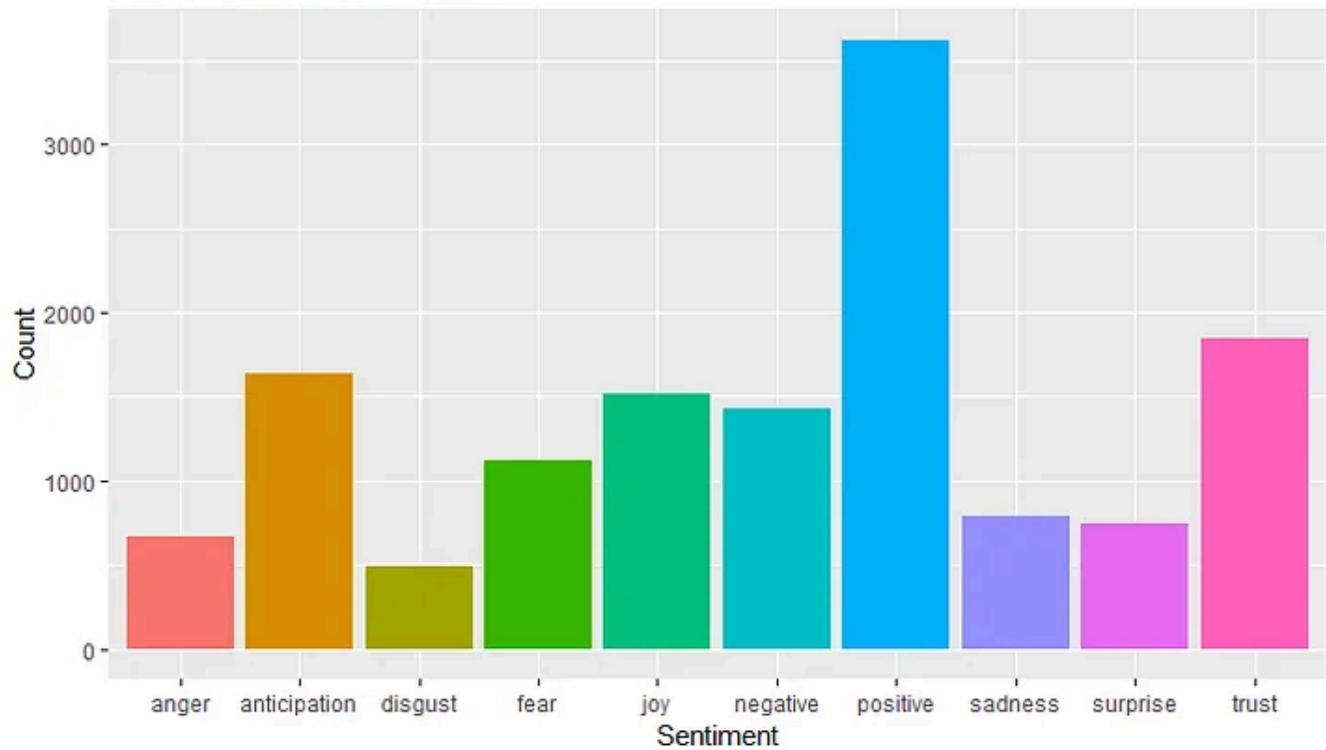
### **Sentiment Analysis by Syuzhet**

This package comes with four sentiment dictionaries and provides a method for accessing the robust, but computationally expensive, sentiment extraction tool developed in the NLP group at Stanford.

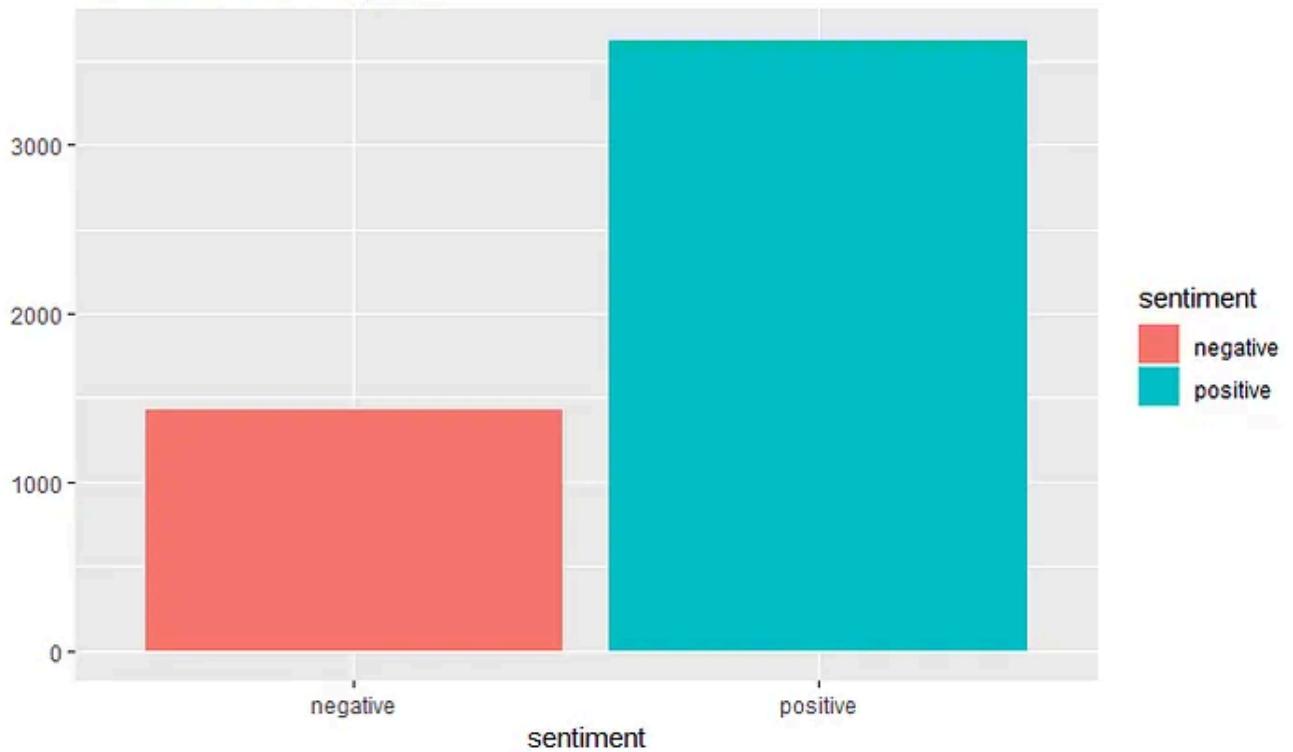
**For more information please check the link.**

This package is available in R and the code is below. I referred to the sample code and did it. If you want more details then please check the link I provided and also check Rcran projects.

### BG Total Sentiment Score



### BG Sentiment Analysis

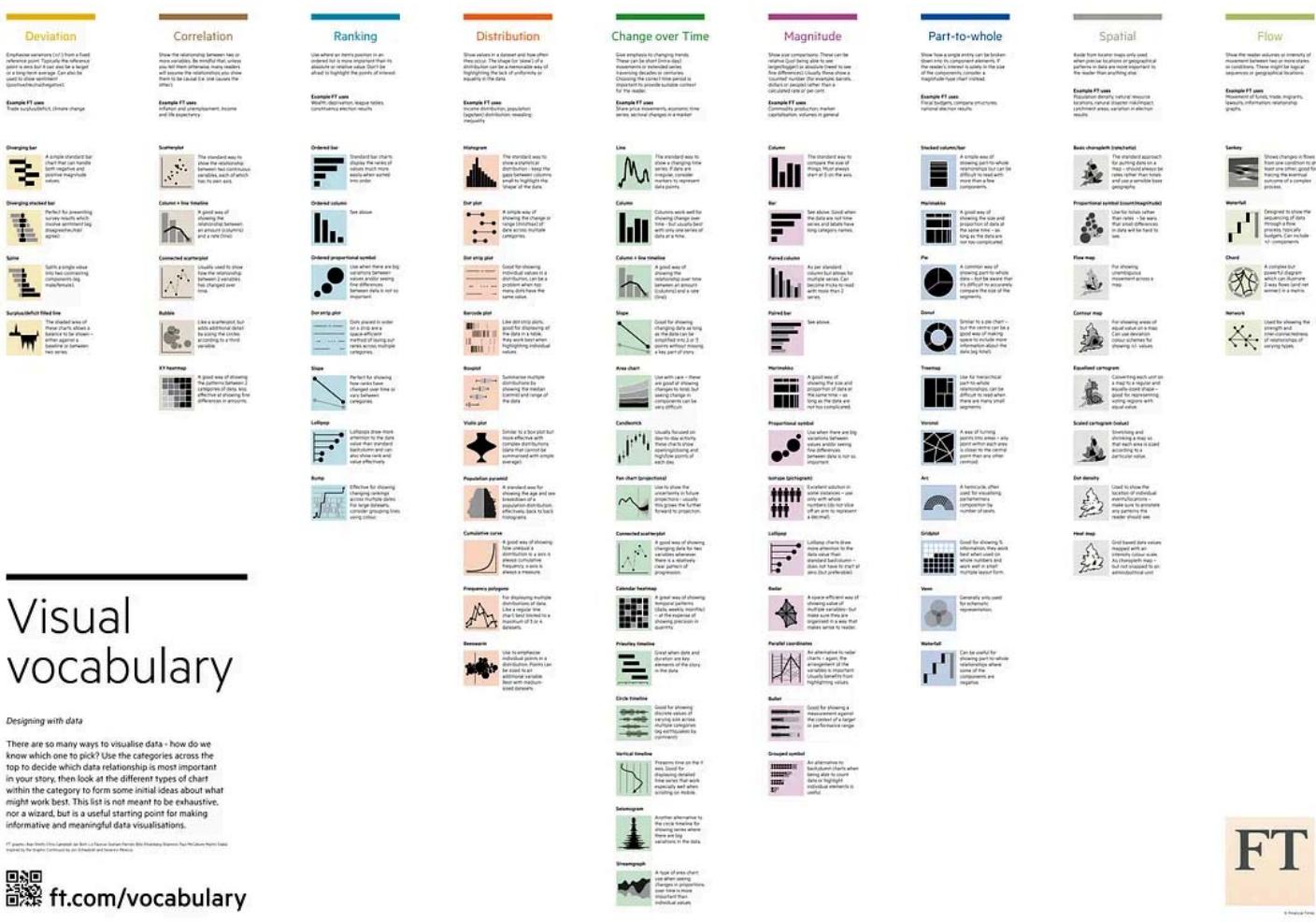


	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	2	0	1	1	0	2
3	0	3	2	1	1	0	1	2	2	3
4	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	1	0	1	0	0	1
6	0	2	0	0	1	0	1	0	0	1

## Conclusion

Hope you enjoyed reading this article. Now you know how easy to extract data by using Twitter API's -Streaming and Search. With only a few lines of code, you can get the data from Twitter and use the pandas for all the analyses. In the next post, I will write about scrapy and selenium. I will also update my GitHub sometime soon and provide the link. Meanwhile, if you want to provide feedback or suggestions or have any questions then please go ahead and send an email to epmanalytics100@gmail.com. Thanks again for reading my post 

I found this chart very helpful. (Source: FT)

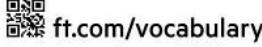


# Visual vocabulary

Designing with data

There are so many ways to visualise data – how do we know which one to pick? Use the categories across the top to decide which data relationship is most important in your story, then look at the different types of chart within the category to form some initial ideas about what might work best. This list is not meant to be exhaustive, nor a wizard, but is a useful starting point for making informative and meaningful data visualisations.

PT grants: Alan Smith, Chris Lamont, Ian Scott, Le Faouine, Graham Parsons, Billy Hwang, Shannon, Paul McCullagh, Hyun-Jeong Kim, and others. This work was partially funded by grants from the National Science Foundation (NSF) and the Defense Advanced Research Projects Agency (DARPA). The authors would like to thank the anonymous reviewers for their useful comments and suggestions.



FT

## Jupyter Cheat Sheet (Source: Datacamp)

**Python For Data Science Cheat Sheet**

**Jupyter Notebook**

Learn More Python for Data Science Interactively at [www.DataCamp.com](http://www.DataCamp.com)

**Saving/Loading Notebooks**

- Create new notebook
- Make a copy of the current notebook
- Save current notebook and record checkpoint
- Preview of the printed notebook
- Close notebook & stop running any scripts
- Open an existing notebook
- Rename notebook
- Revert notebook to a previous checkpoint
- Download notebook as:
  - IPython notebook
  - Python
  - HTML
  - Markdown
  - reST
  - LaTeX
  - PDF

**Working with Different Programming Languages**

Kernels provide computation and communication with front-end interfaces like the notebooks. There are three main kernels:

- IPython
- IRkernel
- Julia

Installing Jupyter Notebook will automatically install the IPython kernel.

**Widgets**

Notebook widgets provide the ability to visualize and control changes in your data, often as a control like a slider, textbox, etc.

You can use them to build interactive GUIs for your notebooks or to synchronize stateful and stateless information between Python and JavaScript.

**Command Mode:**

**Edit Mode:**

**Executing Cells**

**View Cells**

**Asking For Help**

1. Save and checkpoint  
2. Insert cell below  
3. Cut cell  
4. Copy cell(s)  
5. Paste cell(s) below  
6. Move cell up  
7. Move cell down  
8. Run current cell  
9. Interrupt kernel  
10. Restart kernel  
11. Display characteristics  
12. Open command palette  
13. Current kernel  
14. Kernel status  
15. Log out from notebook server

**Help Menu:**

- User Interface Tour
- Keyboard Shortcuts
- Edit Keyboard Shortcuts
- Notebook Help
- Markdown
- Jupyter-contrib notebooks
- Python
- IPython
- NumPy
- SciPy
- Matplotlib
- Sympy
- pandas
- About

**DataCamp**  
Learn Python for Data Science Interactively

## Pandas Cheat Sheet 1(Source: Datacamp)

## Python For Data Science Cheat Sheet

### Pandas Basics

Learn Python for Data Science Interactively at [www.DataCamp.com](http://www.DataCamp.com)



### Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.



Use the following import convention:

```
>>> import pandas as pd
```

### Pandas Data Structures

#### Series

A one-dimensional labeled array capable of holding any data type

a	3
b	-5
c	7
d	4

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

#### DataFrame

Columns	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasilia	207847528

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
   'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
   'Population': [11190846, 1303171035, 207847528]}
```

```
>>> df = pd.DataFrame(data,
   columns=['Country', 'Capital', 'Population'])
```

### I/O

#### Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

#### Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
Read multiple sheets from the same file
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

### Asking For Help

```
>>> help(pd.Series.loc)
```

### Selection

#### Also see NumPy Arrays

#### Getting

```
>>> s['b']
-5
>>> df[1]
   Country    Capital  Population
1  India      New Delhi     1303171035
2  Brazil     Brasilia     207847528
```

Get one element

Get subset of a DataFrame

### Selecting, Boolean Indexing & Setting

#### By Position

```
>>> df.iloc[[0], [0]]
   'Belgium'
>>> df.iat[[0], [0]]
   'Belgium'
```

Select single value by row & column

#### By Label

```
>>> df.loc[[0], ['Country']]
   'Belgium'
>>> df.at[[0], ['Country']]
   'Belgium'
```

Select single value by row & column labels

#### By Label/Position

```
>>> df.ix[2]
   Country    Brazil
   Capital   Brasilia
   Population 207847528
>>> df.ix[:, 'Capital']
   0    Brussels
   1    New Delhi
   2    Brasilia
```

Select single row of subset of rows

```
>>> df.ix[1, 'Capital']
   'New Delhi'
```

Select a single column of subset of columns

#### Setting

```
>>> s1 = s[s > 1]
>>> s1[s < -1] | (s > 2)
>>> df[df['Population']>1200000000]
```

Series s where value is not > 1  
s where value is <-1 or > 2  
Use filter to adjust DataFrame

```
>>> s1['a'] = 6
```

Set index a of Series s to 6

### Dropping

```
>>> s.drop(['a', 'c'])
Drop values from rows (axis=0)
```

```
>>> df.drop('Country', axis=1)
Drop values from columns(axis=1)
```

### Sort & Rank

```
>>> df.sort_index()
>>> df.sort_values(by='Country')
>>> df.rank()
```

Sort by labels along an axis  
Sort by the values along an axis  
Assign ranks to entries

### Retrieving Series/DataFrame Information

#### Basic Information

```
>>> df.shape
(rows,columns)
>>> df.index
Describe index
>>> df.columns
Describe DataFrame columns
>>> df.info()
Info on DataFrame
>>> df.count()
Number of non-NA values
```

#### Summary

```
>>> df.sum()
>>> df.cumsum()
>>> df.min() / df.max()
>>> df.idxmin() / df.idxmax()
>>> df.describe()
>>> df.mean()
>>> df.median()
```

Sum of values  
Cumulative sum of values  
Minimum/maximum values  
Minimum/Maximum index value  
Summary statistics  
Mean of values  
Median of values

### Applying Functions

```
>>> f = lambda x: x**2
>>> df.apply(f)
>>> df.applymap(f)
```

Apply function  
Apply function element-wise

### Data Alignment

#### Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s = s3
   a    10.0
   b    NaN
   c    5.0
   d    7.0
```

### Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s3.add(s3, fill_value=0)
   a    10.0
   b    -5.0
   c    5.0
   d    7.0
```

```
>>> s3.sub(s3, fill_value=2)
   a    10.0
   b    -5.0
   c    5.0
   d    7.0
```

```
>>> s3.div(s3, fill_value=4)
   a    10.0
   b    -5.0
   c    5.0
   d    7.0
```

```
>>> s3.mul(s3, fill_value=3)
   a    10.0
   b    -5.0
   c    5.0
   d    7.0
```

DataCamp

Learn Python for Data Science Interactively



## Python For Data Science Cheat Sheet

### Pandas

Learn Python for Data Science Interactively at [www.DataCamp.com](http://www.DataCamp.com)



### Reshaping Data

#### Pivot

>>> df3 = df2.pivot(index='Date', columns='Type', values='Value')

Spread rows into columns

Date	Type	Value
0 2016-03-01	a	11.432
1 2016-03-02	b	13.031
2 2016-03-01	c	20.784
3 2016-03-03	a	99.906
4 2016-03-02	a	1.303
5 2016-03-03	c	20.784

2016-03-01 11.432  
2016-03-02 13.031  
2016-03-01 20.784  
2016-03-03 99.906  
2016-03-02 1.303  
2016-03-03 20.784

#### Pivot Table

>>> df4 = pd.pivot\_table(df2, values='Value', index='Date', columns='Type')

Spread rows into columns

#### Stack / Unstack

>>> stacked = df5.stack()  
>>> stacked.unstack()

Pivot a level of column labels  
Pivot a level of index labels

	0	1
0	0.233482	0.390959
1	0.184713	0.237102
2	0.433522	0.429401
3		

Unstacked

Stacked

#### Melt

>>> pd.melt(df2, id\_vars=['Date'], value\_vars=['Type', 'Value'], value\_name='Observations')

Gather columns into rows

Date	Type	Value	Observations
0 2016-03-01	a	11.432	
1 2016-03-02	b	13.031	
2 2016-03-01	c	20.784	
3 2016-03-03	a	99.906	
4 2016-03-02	a	1.303	
5 2016-03-03	c	20.784	

Unstacked

Stacked

#### Iteration

>>> df.iteritems()  
>>> df.iterrows()

(Column-index, Series) pairs  
(Row-index, Series) pairs

## Advanced Indexing

### Selecting

```
>>> df3.loc[:, (df3>1).any()]
>>> df3.loc[:, (df3>1).all()]
>>> df3.loc[:, df3.isnull().any()]
>>> df3.loc[:, df3.notnull().all()]
```

### Indexing With isin

```
>>> df[(df.Country.isin(df2.Type))]
>>> df.filter(items="a","b")
>>> df.select(lambda x: not x>5)
```

### Where

```
>>> s.where(s > 0)
```

### Query

```
>>> df6.query('second > first')
```

## Also see NumPy Arrays

Select cols with any vals >  
Select cols with vals > 1  
Select cols with NaN  
Select cols without NaN  
Find same elements  
Filter on values  
Select specific elements

Subset the data

Query DataFrame

## Combining Data

data1	data2
X1	X2
a	11.432
b	1.303
c	99.906

data1	data2
a	11.432
b	NaN
c	99.906

### Merge

```
>>> pd.merge(data1, data2, how='left', on='X1')
```

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN

### Setting/Resetting Index

```
>>> df.set_index('Country')
>>> df4 = df.reset_index()
>>> df = df.rename(index=reset,
                   columns=[{"Country": "entity",
                             "Capital": "capitl",
                             "Population": "pplmn"}])
```

Set the index  
Reset the index  
Rename DataFrame

### Reindexing

```
>>> s2 = s.reindex(['a','c','d','e','b'])
```

### Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

Forward Filling

```
>>> df.reindex(range(4), method='bfill')
```

Backward Filling

```
>>> df.reindex(range(5), method='bfill')
```

Forward Filling

```
>>> df.reindex(range(4), method='ffill')
```

Backward Filling

```
>>> df.reindex(range(5), method='ffill')
```

## Python For Data Science Cheat Sheet

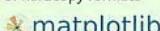
### Matplotlib

Learn Python Interactively at [www.DataCamp.com](http://www.DataCamp.com)



### Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.



### 1) Prepare The Data

Also see [Lists & NumPy](#)

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

#### 2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> V, X = np.mgrid[-2:2:100j, -2:2:100j]
>>> U = -X**2 - V
>>> V = 1 - X**2 - V**2
>>> from matplotlib.cbook import get_sample_data
>>> img = np.loadtxt(get_sample_data('axes_grid/bivariate_normal.npy'))
```

### 2) Create Plot

```
>>> import matplotlib.pyplot as plt
```

#### Figure

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

#### Axes

All plotting is done with respect to an `Axes`. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) # row-col-num
>>> ax2 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(rows=2,ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

### 3) Plotting Routines

#### 1D Data

```
>>> fig, ax = plt.subplots()
>>> lines = ax.plot(X,y)
>>> ax.scatter(X,y)
>>> axes[0,0].bar([1,2,3],[3,4,5])
>>> axes[1,0].barh([0.5,1,2.5],[0,1,2])
>>> axes[1,1].axhline(0.45)
>>> axes[0,1].axvline(0.85)
>>> ax.hill(X,y,color='blue')
>>> ax.fill_between(X,y,color='yellow')
```

Draw points with lines or markers connecting them  
Draw unconnected points, scaled or colored  
Plot vertical rectangles (constant width)  
Plot horizontal rectangles (constant height)  
Draw a horizontal line across axes  
Draw a vertical line across axes  
Draw filled polygons  
Fill between y-values and o

#### 2D Data or Images

```
>>> fig, ax = plt.subplots()
>>> im = ax.imshow(img,
                  cmap='gist_earth',
                  interpolation='nearest',
                  vmin=-2,
                  vmax=2)
```

Colormapped or RGB arrays

#### Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)
>>> axes[1,1].quiver(y,z)
>>> axes[0,1].streamplot(X,Y,U,V)
```

Add an arrow to the axes  
Plot a 2D field of arrows  
Plot a 2D field of arrows

#### Data Distributions

```
>>> ax1.hist(y)
>>> ax1.boxplot(y)
>>> ax2.violinplot(z)
```

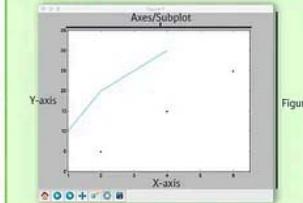
Plot a histogram  
Make a box and whisker plot  
Make a violin plot

```
>>> axes2[0].colorbar(data2)
>>> axes2[0].pcolormesh(data)
>>> CS = plt.contourf(X,X,U)
>>> axes2[2].contourf(data1)
>>> axes2[2] = ax.clabel(CS)
```

Pseudocolor plot of 2D array  
Pseudocolor plot of 2D array  
Plot contours  
Plot filled contours  
Label a contour plot

## Plot Anatomy & Workflow

### Plot Anatomy



### Workflow

The basic steps to creating plots with matplotlib are:

- 1 Prepare data
  - 2 Create plot
  - 3 Plot
  - 4 Customize plot
  - 5 Save plot
  - 6 Show plot
- ```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4] <Step 1>
>>> y = [10,20,25,30] <Step 2>
>>> fig = plt.figure() <Step 3>
>>> ax = fig.add_subplot(111) <Step 3>
>>> ax.plot(x, y, color='lightblue', linewidth=3) <Step 4>
>>> ax.scatter([2,4,6], [5,15,25], color='darkgreen', marker='') <Step 4>
>>> ax.set_xlim(1, 6.5) <Step 5>
>>> plt.savefig('foo.png') <Step 6>
>>> plt.show()
```

### 4) Customize Plot

#### Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha=0.4)
>>> ax.plot(X, Y, c='k')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img, cmap='seismic')
```

#### Mathtext

```
>>> plt.title(r'$\sigma_i=15$', fontsize=20)
```

#### Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker=".")
>>> ax.plot(x,y,marker="o")
```

#### LineStyles

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,ls='solid')
>>> plt.plot(x,y,ls='dashed')
>>> plt.plot(x,y,ls='dashdot')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

#### Text & Annotations

```
>>> ax.text(1, 1, 'Example Graph', style='italic')
>>> ax.annotate("Sine", xy=(8, 0), xycoords='data',
               xytext=(10.5, 0),
               textcoords='data',
               arrowprops=dict(arrowstyle="->",
                               connectionstyle="arc3"))
```

#### Legends

```
>>> ax.set(title='An Example Axes',
          xlabel='Y-axis',
          ylabel='X-axis')
>>> ax.legend(loc='best')
```

#### Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),
                ticklabels=[3,100,-12,"foo"])
>>> ax.tick_params(axis='y', direction='inout',
                  length=10)
```

#### Subplot Spacing

```
>>> fig.subplots_adjust(wspace=0.5,
                        hspace=0.3,
                        left=0.12,
                        right=0.9,
                        top=0.9,
                        bottom=0.1)
```

#### Axis Spines

```
>>> ax1.spines['top'].set_visible(False)
```

```
>>> ax1.spines['bottom'].set_position('outward',10)
```

Add padding to a plot  
Set the aspect ratio of the plot to 1  
Set limits for x- and y-axis  
Set limits for x-axis

Set a title and x- and y-axis labels

No overlapping plot elements

Manually set x-ticks  
Make y-ticks longer and go in and out

Adjust the spacing between subplots

Fit subplot(s) in to the figure area

Make the top axis line for a plot invisible  
Move the bottom axis line outward

### 5) Save Plot

#### Save figures

```
>>> plt.savefig('foo.png')
```

#### Save transparent figures

```
>>> plt.savefig('foo.png', transparent=True)
```

### 6) Show Plot

```
>>> plt.show()
```

### Close & Clear

```
>>> plt.clf()
```

```
>>> plt.cla()
```

```
>>> plt.close()
```

Clear an axis

Clear the entire figure

Close a window

DataCamp  
Learn Python for Data Science Interactively

Thanks again for reading my post 😊

NLP

Python

Data Visualization

Data Science

Twitter



Following

**Written by Senthil E**

2.7K Followers · Writer for Towards Data Science

ML/DS - Certified GCP Professional Machine Learning Engineer, Certified AWS Professional Machine learning Speciality,Certified GCP Professional Data Engineer .

---

## More from Senthil E and Towards Data Science



 Senthil E in Level Up Coding

## Navigating the World of LLMs: A Beginner’s Guide to Prompt Engineering-Part 1

From Basics To Advanced Techniques

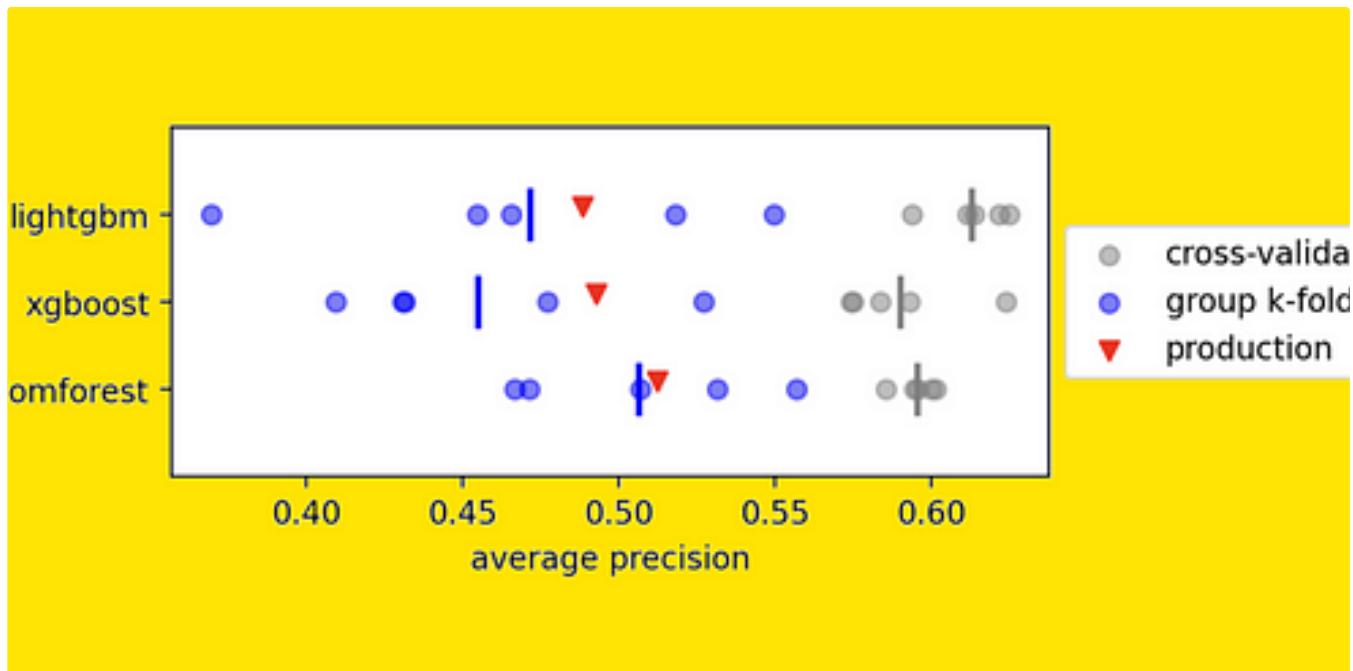
26 min read · Mar 17, 2024

 213

 1



...



 Samuele Mazzanti in Towards Data Science

## Why You Should Never Use Cross-Validation

In real-world applications, using randomized cross-validation is always a bad choice. Here is why.

◆ · 12 min read · Mar 27, 2024

 1.1K  29



...



 Dario Radečić  in Towards Data Science

## Pandas vs. Polars—Time to Switch?

Looking to speed up your data processing pipelines up to 10 times? Maybe it's time to say goodbye to Pandas.

◆ · 7 min read · Apr 7, 2024

👏 1K 🗣 17



...



👤 Senthil E in Level Up Coding

## Navigating the World of LLMs: A Beginner’s Guide to Prompt Engineering-Part 2

From Basics To Advanced Techniques

32 min read · Mar 17, 2024

👏 302 🗣

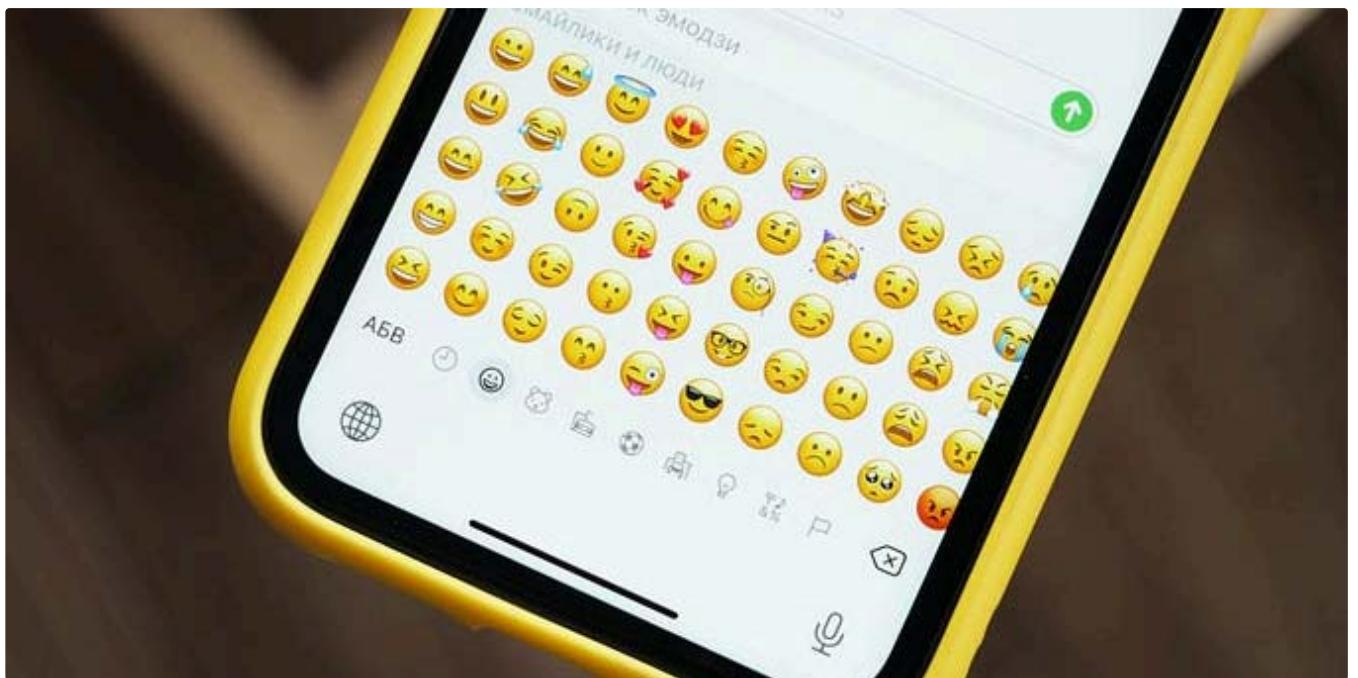


...

See all from Senthil E

See all from Towards Data Science

## Recommended from Medium



 Bale Chen in Towards Data Science

### Emojis Aid Social Media Sentiment Analysis: Stop Cleaning Them Out!

Leverage emojis in social media sentiment analysis to improve accuracy.

14 min read · Jan 31, 2023

 332

 2



...



 Eulene

## Sentiment Analysis of Amazon Reviews Using Natural Language Processing

11 min read · Feb 8, 2024

 12



...

---

### Lists



#### Coding & Development

11 stories · 567 saves



#### Predictive Modeling w/ Python

20 stories · 1111 saves



#### Practical Guides to Machine Learning

10 stories · 1323 saves



#### ChatGPT prompts

47 stories · 1439 saves



 Nuno Bispo in Django Unleashed

## How to Build an AI Chatbot for Q&A on Any Website

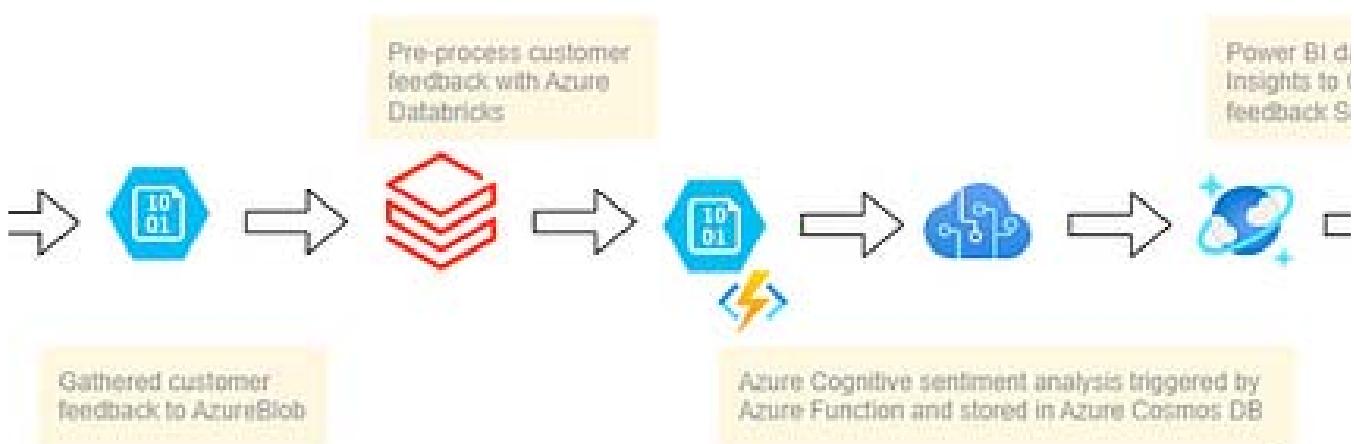
This article delves into the development of a chatbot, designed to read a website's content with web scraping and conversational AI...

◆ · 11 min read · Apr 5, 2024

 11 

 +

...



 Merwan Chinta in CodeNx

## Automated Customer Review Sentiment Analysis for Businesses

## 🌟 Objective

12 min read · Nov 13, 2023

👏 100



+

...



 Ubaid Shah

## Twitter Sentiment Analysis — A Step by Guide

In today's digital age, social media platforms like Twitter have become a goldmine for valuable data. Businesses, researchers, and...

6 min read · Nov 8, 2023

👏 30



+

...



 Onat Kaya

## How to Scrape Tweets From Twitter For Free (2023)

A Free and Easy Way To Scrape Tweets

7 min read · Nov 27, 2023

 39



...

[See more recommendations](#)