Twitter data analysis

|  |  |
| --- | --- |
| 1. [1. Visit the Twitter Developers Site](https://docs.inboundnow.com/guide/create-twitter-application/#toc-0) 2. [2. Sign in with your Twitter Account](https://docs.inboundnow.com/guide/create-twitter-application/#toc-1) 3. [3. Go to “My Applications”](https://docs.inboundnow.com/guide/create-twitter-application/#toc-2) 4. [4. Create a New Application](https://docs.inboundnow.com/guide/create-twitter-application/#toc-3) 5. [5. Fill in your Application Details](https://docs.inboundnow.com/guide/create-twitter-application/#toc-4) 6. [6. Create Your Access Token](https://docs.inboundnow.com/guide/create-twitter-application/#toc-5) 7. [7. Choose what Access Type You Need](https://docs.inboundnow.com/guide/create-twitter-application/#toc-6) 8. [8. Make a note of your OAuth Settings](https://docs.inboundnow.com/guide/create-twitter-application/#toc-7) |  |

Once you’ve done this, make a note of your OAuth settings.

* Consumer Key
* Consumer Secret
* OAuth Access Token
* OAuth Access Token Secret

Twitter\_streaming.py

----------------------------

#Import the necessary methods from tweepy library

from tweepy.streaming import StreamListener

from tweepy import OAuthHandler

from tweepy import Stream

#Variables that contains the user credentials to access Twitter API

access\_token = ""

access\_token\_secret = ""

consumer\_key = ""

consumer\_secret = ""

#This is a basic listener that just prints received tweets to stdout.

class StdOutListener(StreamListener):

def on\_data(self, data):

print data

return True

def on\_error(self, status):

print status

if \_\_name\_\_ == '\_\_main\_\_':

#This handles Twitter authetification and the connection to Twitter Streaming API

l = StdOutListener()

auth = OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

stream = Stream(auth, l)

#This line filter Twitter Streams to capture data by the keywords: 'python', 'javascript', 'ruby'

stream.filter(track=['python', 'javascript', 'ruby'])

pip install tweepy

python twiter\_streaming.py>twitter\_Data.txt

ctrl+c

twitter.py

import re

import json

import pandas as pd

import matplotlib.pyplot as plt

tweets\_data\_path = 'twitter\_Data.txt'

tweets\_data = []

tweets\_file = open(tweets\_data\_path, "r")

for line in tweets\_file:

try:

tweet = json.loads(line)

tweets\_data.append(tweet)

except:

continue

print len(tweets\_data)

#print(tweets\_data[0])

tweets = pd.DataFrame()

tweets['text'] = map(lambda tweet: tweet['text'], tweets\_data)

tweets['lang'] = map(lambda tweet: tweet['lang'], tweets\_data)

tweets['country'] = map(lambda tweet: tweet['place']['country'] if tweet['place'] != None else None, tweets\_data)

print(tweets)

tweets\_by\_lang = tweets['lang'].value\_counts()

print(tweets\_by\_lang)

fig, ax = plt.subplots()

ax.tick\_params(axis='x', labelsize=15)

ax.tick\_params(axis='y', labelsize=10)

ax.set\_xlabel('Languages', fontsize=15)

ax.set\_ylabel('Number of tweets' , fontsize=15)

ax.set\_title('Top 5 languages', fontsize=15, fontweight='bold')

tweets\_by\_lang[:5].plot(ax=ax, kind='bar', color='red')

tweets\_by\_country = tweets['country'].value\_counts()

print tweets\_by\_country

fig, ax = plt.subplots()

ax.tick\_params(axis='x', labelsize=15)

ax.tick\_params(axis='y', labelsize=10)

ax.set\_xlabel('Countries', fontsize=15)

ax.set\_ylabel('Number of tweets' , fontsize=15)

ax.set\_title('Top 5 countries', fontsize=15, fontweight='bold')

tweets\_by\_country[:5].plot(ax=ax, kind='bar', color='blue')

def word\_in\_text(word, text):

word = word.lower()

text = text.lower()

match = re.search(word, text)

if match:

return True

return False

tweets['python'] = tweets['text'].apply(lambda tweet: word\_in\_text('python', tweet))

tweets['javascript'] = tweets['text'].apply(lambda tweet: word\_in\_text('javascript', tweet))

tweets['ruby'] = tweets['text'].apply(lambda tweet: word\_in\_text('ruby', tweet))

print(tweets)

print tweets['python'].value\_counts()[True]

print tweets['javascript'].value\_counts()[True]

print tweets['ruby'].value\_counts()[True]

prg\_langs = ['python', 'javascript', 'ruby']

tweets\_by\_prg\_lang = [tweets['python'].value\_counts()[True], tweets['javascript'].value\_counts()[True], tweets['ruby'].value\_counts()[True]]

x\_pos = list(range(len(prg\_langs)))

width = 0.8

fig, ax = plt.subplots()

plt.bar(x\_pos, tweets\_by\_prg\_lang, width, alpha=1, color='g')

# Setting axis labels and ticks

ax.set\_ylabel('Number of tweets', fontsize=15)

ax.set\_title('Ranking: python vs. javascript vs. ruby (Raw data)', fontsize=10, fontweight='bold')

ax.set\_xticks([p + 0.4 \* width for p in x\_pos])

ax.set\_xticklabels(prg\_langs)

plt.grid()

tweets['programming'] = tweets['text'].apply(lambda tweet: word\_in\_text('programming', tweet))

tweets['tutorial'] = tweets['text'].apply(lambda tweet: word\_in\_text('tutorial', tweet))

tweets['relevant'] = tweets['text'].apply(lambda tweet: word\_in\_text('programming', tweet) or word\_in\_text('tutorial', tweet))

print tweets['programming'].value\_counts()[True]

print tweets['tutorial'].value\_counts()[True]

print tweets['relevant'].value\_counts()[True]

print tweets[tweets['relevant'] == True]['python'].value\_counts()[True]

print tweets[tweets['relevant'] == True]['javascript'].value\_counts()[True]

print tweets[tweets['relevant'] == True]['ruby'].value\_counts()[True]

tweets\_by\_prg\_lang = [tweets[tweets['relevant'] == True]['python'].value\_counts()[True],

tweets[tweets['relevant'] == True]['javascript'].value\_counts()[True],

tweets[tweets['relevant'] == True]['ruby'].value\_counts()[True]]

x\_pos = list(range(len(prg\_langs)))

width = 0.8

fig, ax = plt.subplots()

plt.bar(x\_pos, tweets\_by\_prg\_lang, width,alpha=1,color='g')

ax.set\_ylabel('Number of tweets', fontsize=15)

ax.set\_title('Ranking: python vs. javascript vs. ruby (Relevant data)', fontsize=10, fontweight='bold')

ax.set\_xticks([p + 0.4 \* width for p in x\_pos])

ax.set\_xticklabels(prg\_langs)

plt.grid()

def extract\_link(text):

regex = r'https?://[^\s<>"]+|www\.[^\s<>"]+'

match = re.search(regex, text)

if match:

return match.group()

return ''

tweets['link'] = tweets['text'].apply(lambda tweet: extract\_link(tweet))

tweets\_relevant = tweets[tweets['relevant'] == True]

tweets\_relevant\_with\_link = tweets\_relevant[tweets\_relevant['link'] != '']

print tweets\_relevant\_with\_link[tweets\_relevant\_with\_link['python'] == True]['link']

print tweets\_relevant\_with\_link[tweets\_relevant\_with\_link['javascript'] == True]['link']

print tweets\_relevant\_with\_link[tweets\_relevant\_with\_link['ruby'] == True]['link']

plt.show()