Interactive Twitter Data Visualization Word Cloud

Mehmet Duman[1]

[1]Data Science Graduate Student, University of Massachusetts Dartmouth, Dartmouth, MA 02747, USA

December 28, 2016

Abstract

The main contribution of this paper is analyzing the data available for Twitter in 2016 USA presidential election and trying to express the correlation between social media twitter top subjects before election and the actual result. I used Python multiprocessing system for manipulate tweet dataset, clean the data and create popular use word file. Moreover, this paper is also making assumptions about whether we can guess the results of a future election by observing the past tweets.

1 Introduction

Data visualization plays an important role in data analysis workflows. It enables data analysts to effectively discover patterns in large datasets through graphical means, and to represent these findings in a meaningful and effective way.

In this paper by using Python multiprocessing, I aim to analyze active Twitter data and try to achieve following goals:

1. Get data from twitter network and generate Most Used Word visualization
2. Find historical twitter data for 2016 Presidential Election day
3. Create Word Cloud to analyze twitter social media for selected filter

I picked Twitter for my paper because it is a very popular social network with many users from different ages. Twitter is an online news and social networking service where users post and read short 140-character messages called "tweets". Registered users can post and read tweets, but those who are unregistered can only read them [1]. By 2016, every second, on average, around 6,000 tweets are tweeted on Twitter, which corresponds to over 350,000 tweets sent per minute, 500 million tweets per day and around 200 billion tweets per year [2]. Users share thoughts, links and pictures on Twitter, journalists comment on live events, companies promote products and engage with customers.

For example, on the day of United States Presidential Election 2016, by 10 p.m., 40 million posts had been sent about the election, exceeding the 31 million sent on Election Day 2012 [3]. Although Twitter allows users to post as many tweets as they want, it only allows to get 3,200 of a Users most recent Tweets (by screen name or user id) [4].

2 Background

There are some studies that used Twitter,

1. Mining Twitter User Profile on Python [5] : This is an informative slideshare tutorial prepared by PHD student Weiai Xu, and explains how to download Twitter data. It keeps the data in SQLite, and does not do any wordclouds.
2. Analyzing a NHL Playoff Game With Twitter [6] : This NHL Playoff Game With Twitter analysis is written by Python and used Pandas, Python libraries Tweepy, pymongo, vincent and nltk (natural language processing library, for plot text with frequencies). There are some similarities like calculating word frequencies, using stopwords for data cleaning. They generate word versus frequencies plot, but I believe using wordcloud should be more effective.
3. Using Python to Grab Twitter User Data [7] : This work is about using Python Twython library to grab twitter user data. I used Python Tweepy library which is much easier and reliable to get tweets from Twitter Streaming API. This paper does not do any analysis on wordcloud.

3 Results

In this first part, I am going to explain how I collect the data, by creating different kind of word cloud visualization to analyze dataset. There are different options to collect data from Twitter. I preferred to get tweets by connecting Twitter API network and get real time tweets for my dataset.

In my project I used following programs;

- Python : Tweepy library for accessing Twitter API , JSON and Wordcloud library
- JavaScript
- D3.js Visualization

3.1 How to Get Tweets?

When I was using python, one of the first libraries I came across was Tweepy, an open source, easy to use python library for accessing the Twitter API. There are three different ways to get access to Twitters enormous amounts of data [6].

1. First is Twitters Search API which allows you to gather tweets which were made in the past week. You can collect tweets based on user, keywords, locations, its basically the same as when you are searching for tweets on Twitters website except you are limited to how much you can collect and you get raw json type Tweet files.
2. Second one is Twitter Streaming API. This allows you to compile tweets as they are happening in real time. In order to have access to Twitter data programmatically, we need to create an app that interacts with the Twitter Streaming API. You won't actually use the app for anything, you just need the password and authentication code.The disadvantage of the streaming API is you are not collecting all of the Tweets that are being sent in real time only a sample of 1% of the current traffic.
3. The only way to get the full stream of all tweets being sent that match your criteria is the Twitter Firehose. The only way to get access to the Firehose is to go through a third party such as Datasift or the recently acquired by twitter, GNIP.

3.1.1 Twitter API Access Information

In order to access Twitter Streaming API, we need to get 4 pieces of information from Twitter: API key, API secret, Access token and Access token secret. Then we follow the steps below to get all 4 elements:

- Create a twitter account if you do not already have one.
- Go to `https://apps.twitter.com/` and log in with your twitter credentials.
- Click "Create New App"
- Fill out the form, agree to the terms, and click "Create your Twitter application"
- In the next page, click on "API keys" tab, and copy your "API key" and "API secret".
- Scroll down and click "Create my access token", and copy your "Access token" and "Access token secret".

3.1.2 Connect Twitter Streaming API

I am using a Python library called Tweepy to connect to Twitter Streaming API and downloading the data. In order to authorize our app to access Twitter on our behalf, we need to use the OAuth interface. The api variable is now our entry point for most of the operations we can perform with Twitter.

```
import tweepy

#Variables that contains the user credentials to access Twitter API
access_token = ""
access_token_secret = ""
consumer_key = ""
consumer_secret = ""

#OAuth process, using the keys and tokens
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

#Creating of the actual interface, using authentication
api = tweepy.API(auth)
```

Figure 1

3.1.3 Streaming

In order to "keep the connection open", and gather all the upcoming tweets about a particular event, the Streaming API is what we need. We need to extend the StreamListener() to customize the way we process the incoming data. The Streaming API monitors for tweets and actions in real time and catches them when some event happens.

The below script will retrieve each tweet in json file format. I got these tweets text future only and append it as a new line in text file format. Depending on the search term, we can gather tons of tweets within a few minutes. This is especially true for election or live events with a world-wide coverage (World Cups, Super Bowls, Academy Awards, etc) [8].

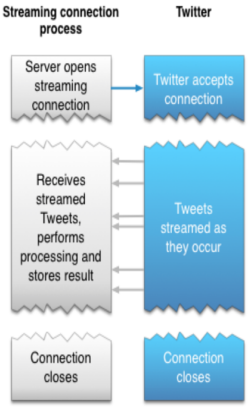

Figure 2

```
class StdOutListener(StreamListener):
    def __init__(self, api=None):
        super(StdOutListener, self).__init__()
        self.num_alltweets = 0
        self.num_print = 2
        self.count = 0
        self.tstart = time.time()
        print( tf_filter, Sf_max, Sf_save_num, Sf_save_sec, tf_text)

    def on_status(self, status):
        self.num_alltweets += 1
        self.count += 1

        if self.num_alltweets <= Sf_max :
            ft_save.write(status.text+'\n')

            time_past = (time.time() - self.tstart)

            if self.count >= Sf_save_num or time_past > Sf_save_sec :
                print('self.count',self.count,'Sf_save_num',Sf_save_num,
                      'time_past',time_past,'Sf_save_sec',Sf_save_sec)
                self.count = 0
                self.tstart = time.time()

                tf_filter_str = '_'.join(map(str, tf_filter))
                createWordFile(self.num_alltweets, 'F_'+ tf_filter_str )
            return True
        else:
            print("Time limit or Max Tweet Recieved")
            return False

    def on_error(self, status):
        print('on_error(self, status):', status,self )
        return True

def getTweetsByTrack():
    #This handles Twitter authetification and the connection to Twitter Streaming API
    try:
        auth = OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_token, access_token_secret)
        if tfscr_name != '' :
            api = tweepy.API(auth)                    ## Get all tweeets for give tfscr_name on the se
            tw_tot = getTweetsByScreenName(api)
        else:
            l = StdOutListener()                  ## Get new tweets
            stream = Stream(auth, l)
            if tf_filter == ['']:
                stream.sample()
            else:
                stream.filter(track = tf_filter )
            ft_save.close()
```

Figure 3

3.1.4 Get Tweets for a Specified User

I am using Tweepy library and python to return the tweets of a specified user. According to the Python Twitter Api, they'll allow you to extract a maximum of 3,200 tweets, 200 at a time. To get this 3200 tweets I need to create a loop and each time I could retrieve 200 tweets from user timeline method. This method is different than Twitter API Streaming method. You are getting tweets from specified users timeline and there is no time limits for this last 3200 tweets.

3.2 Tweet Sample

Tweet text is limited by 140 character, but single tweet is around 2000 character.

{"created_at":"Fri Nov 25 19:31:34 +0000 2016","id":802233284218368001,"id_str":"802233284218368001","text":"OH MY GOD: LOOK WHAT HAPPENED IMMEDIATELY AFTER TRUMP LANDED IN FLORIDA FOR THANKSGIVING! https:\/\/t.co\/WzB6edU0RI https:\/\/t.co\/kTWKZxPCdF","display_text_range":[0,113],"source":"\u003ca href=\"http:\/\/dlvr.it\" rel=\"nofollow\"\u003edlvr.it\u003c\/a\u003e","truncated":false,"in_reply_to_status_id":null,"in_reply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":{"id":772922637953667072,"id_str":"772922637953667072","name":"Students4Trump8","screen_name":"AnnaAlvarez1992","location":"South Carolina, USA","url":null,"description":"We believe students who think in honest objective terms make better citizens.\u201c.@students4trump8","protected":false,"verified":false,"followers_count":2254,"friends_count":2307,"listed_count":16,"favourites_count":7,"statuses_count":47840,"created_at":"Mon Sep 05 22:21:31 +0000 2016","utc_offset":-25200,"time_zone":"Arizona","geo_enabled":false,"lang":"es","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":"F58EA8","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\/772925759614750720\/D2dkyku0_normal.jpg","profile_image_url_https":"https:\/\/pbs.twimg.com\/profile_images\/772925759614750720\/D2dkyku0_normal.jpg","profile_banner_url":"https:\/\/pbs.twimg.com\/profile_banners\/772922637953667072\/1473123472","default_profile":false,"default_profile_image":false,"following":null,"follow_request_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors":null,"is_quote_status":false,"retweet_count":0,"favorite_count":0,"entities":{"hashtags":[],"urls":[{"url":"https:\/\/t.co\/WzB6edU0RI","expanded_url":"http:\/\/viid.me\/qq2WHi","display_url":"viid.me\/qq2WHi","indices":[90,113]}],"user_mentions":[],"symb

Figure 4

3.3 Using Python Multiprocessing to Read and Process Tweets

Python multiprocessing pool: It is a process pool object which controls a pool of worker processes to which jobs can be submitted.

Python map (func, iterable, [chunksize]): This object will lock the main program until all a process is finished, which is quite useful if we want to obtain result in a particular order for certain applications.

```
def createWordFile(tw_tot, filterORscrname):
#   Print tw_tot total tweet number to setup file
    with open(tf_setup, 'r+') as fsetup:
        fsetup_line = json.load(fsetup)
        fsetup_line["t_number"] = tw_tot

        fsetup.seek(0)
        fsetup.write(json.dumps(fsetup_line))
        fsetup.truncate()

    stopWords = getStopWordList(tf_stopword)

    pool = multiprocessing.Pool(4)
    part_func = partial(processTweet, stopWords)
    with open(tf_text) as source_file:
        result = pool.map(part_func, source_file,4)

    pool.close()
    pool.join()
```

Figure 5

While I am getting tweets from the Twitter Streaming API, I am writing every tweets text info in a "tweet text file". If my tweet counter exceed the number from TwitterSetup.txt "tweet number", I am processing "tweet text file" to generate popular used word files (You can see the python code above Figure:3) and wordcloud image. During this process I am using python Multiprocessing to speed up. This generating file process is keep repeating whenever tweet counter exceeds the "tweet number" and generates wordcloud images.

```
def processTweet(stopWords,line ):
    try:
        tweet = line.strip()
    except:
        return
    WordList=''

    tweet = tweet.lower()                                          #Convert to lower case
    tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet) #Convert www.* or https?://* to UR
    tweet = re.sub('@[^\s]+','AT_USER',tweet)                      #Convert @username to AT_USER
    tweet = re.sub('[\s]+', ' ', tweet)                            #Remove additional white spaces
    tweet = re.sub(r'#([^\s]+)', r'\1', tweet)                     #Replace #word with word
    tweet = tweet.strip('\'"')                                     #trim
    words = tweet.split()                                          #split tweet into words

    for w in words:
        w = replaceTwoOrMore(w)                                    #replace two or more with two occu
        w = w.strip('\'"?,.')                                      #strip punctuation
        val = re.search(r"^[a-zA-Z][a-zA-Z0-9]*$", w)              #check if the word stats with an a
        if(w in stopWords or val is None):                         #ignore if it is a stop word
            continue
        else:
            WordList = WordList +  w.lower() + ','

        if WordList == None:
            WordList =''

        if WordList[:-1].endswith(','):
            WordList = WordList[:-1]

    return WordList
```

Figure 6

3.4 Creating Word Cloud by Python WordCloud Library

A worldcloud is a collage of words and those words that are bigger in size have a high frequency. Python wordcloud library is very special and easy to use. This library module will automatically clean a text such as split the sentence and delete some stop-words and generate the wordcloud.

```
    # Generate a word cloud image  the matplotlib way:
    WordListAll_text = " ".join(str(x) for x in result)

    wordcloud = WordCloud().generate( WordListAll_text )

    plt.imshow(wordcloud)
    plt.axis("off")
    plt.title('Number of tweets : ' + str(tw_tot) ,  fontsize=20)
```

Figure 7

After I generate "result" list variable (from Figure:5) by using Python Multiprocessing, I just use Python wordcloud library to generate the word cloud and save it to file. Creating wordcloud process is like, I am taking a picture of current tweets status and copying current tweets file to another file. After I generate my word cloud, I keep continue to retrieve tweets from Twitter and append to the existing file.

By generating wordcloud image over and over again in certain time or by certain tweet number (this

control number is defined in TwitterSetup.txt), I can follow the changes on the word clouds in time period and keep it in record.

3.5 Creating animated Word Cloud by D3.javascript

Python wordcloud library is a great tool, but I like to show and update the changes on the screen in real time. This is not possible by Python wordcloud. Therefore, I used animated d3 word cloud example [9] with prepared by D3.javascript and implemented into my project. By generating this d3.javascript application, I am able to generate animated word cloud, bring focus on the important changes by transition animation and get updates for new popular words interactively.

For this job, I need to calculate/define each popular words weight manually. I count the words frequencies and created popular top 30 words from text file. When I am creating this file, I used below mathematical logic to define words weight from popular words frequencies.

$$\text{LinearSize} = f_{min} + (\frac{(\text{WordFrequency} - w_{min})}{(w_{max} - w_{min})} * (f_{max} - f_{min})$$

$$\text{Weight} = (\frac{(\text{LinearSize})}{8})^3$$

w_{min} : minimum word frequency
w_{max} : maximum word frequency
f_{min} : minimum font size
f_{max} : maximum font size

Figure 8

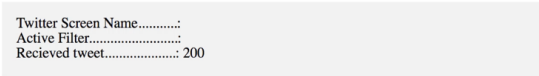

Figure 9: Sample for animated wordcloud

I created word clouds by Python wordcloud library and D3.js. Each visualization has different kind of advantages. Some of the D3.js wordcloud visualization advantages are, user friendly, the transition can be followed, and new word updates or subtract in real time. You have the all control on word font sizes and styles. Everything is happening interactively. Some of the Python matplotlib wordcloud library advantages are, easy to generate plots, could be save to a file in periodically to keep record of word clouds. Both visualizations helped me to achieve different kind of goals.

3.6 Wordcloud Analyst

After analyzing my data, I generated many graphs. However, I found a few of them useful for my analysis. Below I have explained them one by one:

Figure 10: User @realDonaldTrump

Donald Trump has 34.1K total tweets on his record and Figure:10 shows his last 3200 tweets before 12/06/2016. As wee see word 'Hillary' shows up after 1000 tweets, and it is the most used one by 2,200 tweets. After 2,600 tweets it shows up in smaller fonts, means user @realDonaldTrump used word 'Hillary' less frequently.

Figure 11: User @HillaryClinton

Hillary Clinton has 9.8K total tweets on his record and Figure:11 shows her last 3200 tweets before 12/6/2016. As we see, word 'Donald' shows up after 1,000 tweets and it is popular after 2,400 tweets then by 3,200 tweets its frequency drops.

Filter "Ghana", There where general election at December 7, 2016

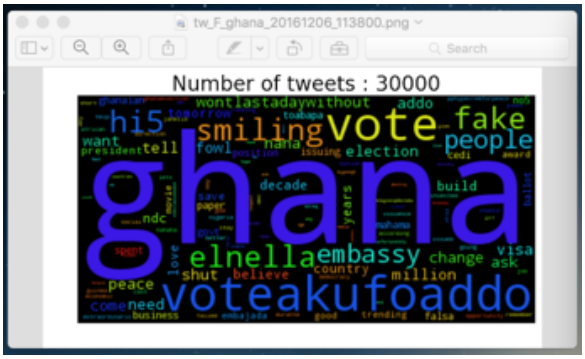

Figure 12: Dec 6, 8:25am-4:38pm

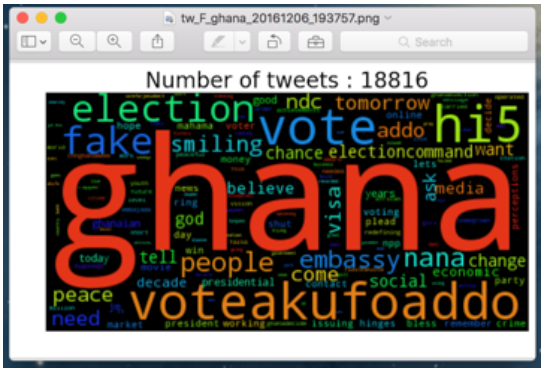

Figure 13: Dec 6, 5:22pm-12:37am

Presidential Candidate
*John Draman **Mahama**
Ivor Greenstreet
Nana **Akufo-Addo** (addo , voteakufoaddo)
Pae Kwesi Nduon
Edward Mahama
Nana Konadu A. R
Jacob Osei Yeboah

Figure 14: Dec 7, 10:40am-3:11pm **Figure 15**

These figures show how people's tweets wary before and on the election day. As you see from the Figures, Nana Akufo-Addo (voteakufoaddo) was one of the most tweeted one between candidates, and he won the election.

4 Conclusion

In this paper my aim was to reach my goals, and I could be able to do so. For example, I aimed to get data from twitter and generate the most used word visualization. I was able to get real time new tweets and most recent 3200 tweets for selected users and created wordclouds.

Secondly, I wanted to find historical twitter data for 2016 Presidential Election day. However, twitter has one-week restriction for old tweets, but if you have tweet ID's you can retrieve older tweets also. 2016 Election day there was more than 40 million tweets. I found tweet ID's for 20 Million tweets for this day, but it was too slow process to retrieve each tweet by their tweet id one by one.

Next, I wanted to create Word Cloud to analyze twitter social media for selected filter. I was able to do that in many different ways.

During my studies I came across with some difficulties. I tried to use Python Multithreading, but it is so complex. It needs improvement, I wasn't able use it to process tweets. However, I used Python Multiprocessing and it increased the performance of processing tweets significantly.

If I compare Python and C, I could easily say C OpenMP and C MPI are more powerful, but Python program is really easy to write code and find libraries to use for all kind of needs. Like Tweepy, it is a great open-source library to access to the Twitter API, heavily relies on it and has great Streaming API support.

To conclude, my study only shows that it is possible to find out what social media talks about a subject by just using a specific filter, and this can open many doors. With so many users around the globe, Twitter is a very good source for political and social studies, as well as advertisement and marketing studies. If we could be able to use the current and historical data and show the results visually, we can get impromptu results for many major events, and may predict the actual results. Therefore, I believe using Twitter data has a promising future.

References

[1] Wikipedia The free encyclopedia. *Twitter*, 2016. https://en.wikipedia.org/wiki/Twitter#/.

[2] David Sayce. *10 Billions Tweets? number of tweets per day?*, 2010 (Updated March 2016). http://www.dsayce.com/social-media/10-billions-tweets/.

[3] Mike Isaac and Sydney Ember. *For Election Day Influence, Twitter Ruled Social Media*, 2016. http://www.nytimes.com/2016/11/09/technology/for-election-day-chatter-twitter-ruled-social-media.html?_r=0.

[4] GET statuses/user timeline (Twitter Official Website). *Twitter Developer Documentation*, 2016. https://dev.twitter.com/rest/reference/get/statuses/user_timeline/.

[5] Weiai Xu. *Mining Twitter User Profile on Python*, 2014. http://social-metrics.org/slideshare-tutorial-1/.

[6] Daniel Forsyth. *Analyzing a NHL Playoff Game With Twitter*, 2014. http://www.danielforsyth.me/analyzing-a-nhl-playoff-game-with-twitter/.

[7] Gregory Saxton. *Using Python to Grab Twitter User Data*, 2014. http://social-metrics.org/twitter-user-data/.

[8] Marco Bonzanini. *Mining Twitter Data with Python (Part 1: Collecting data)*, 2015. https://marcobonzanini.com/2015/03/02/mining-twitter-data-with-python-part-1/.

[9] Joe Whitfield-Seed. *Animated d3 word cloud*, 2016. http://bl.ocks.org/jwhitfieldseed/9697914/.