<http://blog.yhat.com/posts/rodeo-presidential-tweets.html>

I first got into data science almost out of coincidence because of a [side project](http://blog.yhat.com/posts/rodeo-presidential-tweets.html" \t "_blank). My friend and I originally came from computer science backgrounds, but we were fascinated at how much attention the presidential candidate tweets were getting; so we stumbled upon Twitter APIs, the difficulty of cleaning text, and data visualizations — except we didn’t know how to make visualizations without the drag and drop functionality of Tableau. Revisiting the project, I realized how making a word cloud is a fun introductory tutorial for those wanting to see the entire process of collecting, cleaning, and visualizing data in pure Python. So with this code, you too can create a word cloud in the shape of your favorite logo/image!

**Step 1: Import the necessary packages**

import numpy as np  
import matplotlib.pyplot as plt  
import re  
from twython import Twython  
from PIL import Image  
from wordcloud import WordCloud, STOPWORDS  
from IPython.display import Image as im

**Step 2: Collect the data**

The first step is to gain access to Twitter’s API by registering for an app to obtain the token keys for the API at [https://dev.twitter.com/](https://dev.twitter.com/" \t "_blank) and then creating a new app at [https://apps.twitter.com/](https://apps.twitter.com/" \t "_blank). Once the app has been created, head over to “Keys and Access Tokens” to get the API Key and API Secret.

[image 1] Figure 1: API Key and Secret for Twitter

Once the application has been created and approved, our next step is to connect to the Twitter API with the Twython package in order to extract the raw Tweets from our lucky, chosen user handle.

#Connect to Twitter  
APP\_KEY = "Dgh9WnBxXeLUcBYhXeohInazm"  
APP\_SECRET = "6fTM5hIYVbMGwVWoIoII7smrE3BexV6x8E9Xj7Ye6EJgw2LGA3"  
twitter = Twython(APP\_KEY, APP\_SECRET)

From there, we can now call the function on a user timeline. Note that Twitter limits how far one can go back on a user’s timeline, which is 3200 at this point in time for free tier. While not as simple as the other parts of creating the word cloud, the following block of code checks for the most recent id and uses that as a placeholder to get the next batch of 200 tweets.

#Get timeline   
user\_timeline=twitter.get\_user\_timeline(screen\_name='Nike',count=1) #get most recent id  
last\_id = user\_timeline[0]['id']-1  
for i in range(16):  
 batch = twitter.get\_user\_timeline(screen\_name='Nike',count=200, max\_id=last\_id)  
 user\_timeline.extend(batch)  
 last\_id = user\_timeline[-1]['id'] - 1

[image2] Figure 2: Our Output in Python

While this output is confusing, fortunately, there are resources to help break down the JSON tweet format.

[image3 Figure 3: Full Tweet JSON Format (Source: Raffi Krikorian)

The above image, although created in 2010, breaks down the JSON Tweet format and is an invaluable source for almost any Twitter project — such as time series, natural language processing, and geospatial projects.

Using Figure 3 as our guide, we can see that the “text” field of each tweet contains the information we need. Furthermore, it appears that our timeline data is a list of dictionaries, so we can iterate through the list and grab each value with the “text” key.

#Extract textfields from tweets  
raw\_tweets = []for tweets in user\_timeline:  
 raw\_tweets.append(tweets['text'])

[image4 Figure 4: Raw Tweet Fields

**Step 3: Clean all the data**

At this point, we have our raw data - all URL links, special characters, emojis, and extra white spaces. Unfortunately, these types of punctuation and words take the word out of a word cloud; so we need to deal with them on a case by case basis.

[image 5] Drake Alphabet

#Create a string form of our list of textraw\_string = ''.join(raw\_tweets)  
no\_links = re.sub(r'http\S+', '', raw\_string)  
no\_unicode = re.sub(r"\\[a-z][a-z]?[0-9]+", '', no\_links)  
no\_special\_characters = re.sub('[^A-Za-z ]+', '', no\_unicode)

[image 6] Figure 5: Words without special characters

Beyond removing special characters, we can see filer words like “and”, “the”, and “but” which don’t have much significance given their frequency in all sentences. And so we can get rid of these words, known as “stop words” by keeping them only if they are not part of the predefined words part of the wordcloud package. Additionally, we can also remove any straggler letters that might’ve remained by checking the length of each word and keeping it if it matches a certain length.

words = no\_special\_characters.split(" ")  
words = [w for w in words if len(w) > 2] # ignore a, an, be, ...  
words = [w.lower() for w in words]  
words = [w for w in words if w not in STOPWORDS]

[image 7 Figure 6: Clean words without fillers

**Step 4: Visualize the data**

Pick any of your favorite images to use as a background image! (Pro tip: works better with larger images with a white background)

[image 8 Figure 7: Nike Stencil

mask = np.array(Image.open('/Users/shsu/Downloads/nike.png'))

Create the word cloud instance (note that the [documentation](https://github.com/amueller/word_cloud/blob/master/wordcloud/wordcloud.py" \t "_blank) requires that the text be in a string format):

wc = WordCloud(background\_color="white", max\_words=2000, mask=mask)  
clean\_string = ','.join(words)  
wc.generate(clean\_string)

And finally show off our new image:

f = plt.figure(figsize=(50,50))  
f.add\_subplot(1,2, 1)  
plt.imshow(mask, cmap=plt.cm.gray, interpolation='bilinear')  
plt.title('Original Stencil', size=40)  
plt.axis("off")f.add\_subplot(1,2, 2)  
plt.imshow(wc, interpolation='bilinear')  
plt.title('Twitter Generated Cloud', size=40)  
plt.axis("off")  
plt.show()

[image 8 Stencil versus Cloud

Ta-da! We’ve now got our own word cloud image made from Twitter data with 100% pure Python code! You’re one project closer to Veni, Vedi, Vici-ing the data science world!

Thank you for taking the time to read my tutorial and feel free to leave a comment or connect on [LinkedIn](https://www.linkedin.com/in/stephenjhsu/" \t "_blank) as I’ll be posting more tutorials on data mining and data science.

References:

1. [Twitter App](https://apps.twitter.com/" \t "_blank) ()
2. [Twython documentation](https://twython.readthedocs.io/en/latest/" \t "_blank)
3. [Andreas Mueller Word Cloud Mask](https://github.com/amueller/word_cloud" \t "_blank)

<https://twitter.com/login?redirect_after_login=https%3A%2F%2Fdeveloper.twitter.com%2Fapps>

<https://twython.readthedocs.io/en/latest/>

<https://github.com/amueller/word_cloud>