

Analyze Word Frequency Counts Using Twitter Data and Tweepy in Python

Reference

- It is the example in this link just added more explanation
- <https://www.earthdatascience.org/courses/use-data-open-source-python/intro-to-apis/calculate-tweet-word-frequencies-in-python/>
(<https://www.earthdatascience.org/courses/use-data-open-source-python/intro-to-apis/calculate-tweet-word-frequencies-in-python/>)

Example objectives:

- Take a set of tweets and clean them, in order to analyze the frequency of words found in the tweets.
- Remove URLs from tweets.
- Clean up tweet text, including differences in case (e.g. upper, lower) that will affect unique word counts and removing words that are not useful for the analysis.
- Summarize and count words found in tweets.

Get Tweets Related to Climate

- When you work with social media and other text data, the user community creates and curates the content. This means there are NO RULES! This also means that you may have to perform extra steps to clean the data to ensure you are analyzing the right thing.
- Next, you will explore the text associated with a set of tweets that you access using Tweepy and the Twitter API. You will use some standard natural language processing (also known as text mining) approaches to do this.

```
In [6]: ▶ 1 #import os to deal with files on the operating system
2 import os
3
4 #import pandas to deal with dataframes and more
5 import pandas as pd
6
7 #import visualization libraries
8 import matplotlib.pyplot as plt
9 import seaborn as sns
10 import itertools
11
12 #import collections to deal with collections
13 import collections
14
15 #import tweepy to deal with the tweets
16 import tweepy as tweepy
17
18 #import nltk to deal with natural processing language
19 #we will study that in details in our Natural Processing course in AI fo
20 import nltk
21 from nltk.corpus import stopwords
22 import re
23 import networkx
24
25 #to filter warnings
26
27 import warnings
28 warnings.filterwarnings("ignore")
29
30 #setting some configurations for seaborn related plots
31 #setting the background style and font scale
32 #those are optional but makes the plots look nicer
33 sns.set(font_scale=1.5)
34 sns.set_style("whitegrid")
```

- The keys

```
In [7]: ▶ 1 #include the keys that you need for the twitter application on
2 #supllied for you as a twitter developer
3
4 access_token = '1219125691028930560-CZyXhFlgCpMM8rG11KwuYJaMoX7uNa'
5 access_secret = '0qSAXeOLmH9pKDPVFy2pQzbloaGxRcbb0JJgnkhq5F2d4'
6 consumer_key = 'HpU6B5BVTuwAfa5nYX1vAVxgD'
7 consumer_secret = 'Cgcs5YIHpIp5Pu5US3N0XAX8N4j1JgPQrE4aK8LYkM89nqeTQa'
8
9
10 auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
11 auth.set_access_token(access_token, access_secret)
12
13 #for more information on the tweepy.api refer to the following link
14 #it discusses more all the argument and parameters required
15 twitter_api = tweepy.API(auth, wait_on_rate_limit=True)
```

```
In [8]: 1 search_term = "#climate+change -filter:retweets"
        2
        3 tweets = tweepy.Cursor(twitter_api.search,
        4                             q=search_term,
        5                             lang="en",
        6                             since='2018-11-01').items(1000)
        7
        8 all_tweets = [tweet.text for tweet in tweets]
        9
        10 all_tweets[:5]
```

```
Out[8]: ['What a pleasure to be contributing to this wonderful tea magazine ☺my ar
         ticle was about growing #tea in the face of... https://t.co/eDB1udZndS', (htt
         ps://t.co/eDB1udZndS',)
         'YES! Why Investing in Women Helps Save the Planet\nEducation, female empo
         werment, and #climate-smart solutions can s... https://t.co/mDvTBBQLSo', (htt
         ps://t.co/mDvTBBQLSo',)
         'New study suggests governments and investors around the world should prio
         ritize small-scale, low carbon technologie... https://t.co/ueFzcj8UAW', (http
         s://t.co/ueFzcj8UAW',)
         '@business Here we go again, all of a sudden #climate change has wedded #c
         ovid-19 , as if distracting us from callin... https://t.co/zMIc9B81VP', (http
         s://t.co/zMIc9B81VP',)
         'Carbon Creed Issue 24 Thread: \n\n"A symposium of thoughts on climate cha
         nge"\n\nThis week the New York Times published... https://t.co/tOi5ERxJG6'
         (https://t.co/tOi5ERxJG6']
```

Remove URLs (links)

- The tweets above have some elements that you do not want in your word counts. For instance, URLs will not be analyzed in this lesson. You can remove URLs (links) using regular expressions accessed from the re package.
- Re stands for regular expressions. Regular expressions are a special syntax that is used to identify patterns in a string.
- While this lesson will not cover regular expressions, it is helpful to understand that this syntax below:
- `([^0-9A-Za-z \t])(\w+://\S+)`
- Tells the search to find all strings that look like a URL, and replace it with nothing – "".
- It also removes other punctuation including hashtags - #.
- `re.sub` allows you to substitute a selection of characters defined using a regular expression, with something else.
- In the function defined below, this line takes the text in each tweet and replaces the URL with "" (nothing): `re.sub("([^\0-9A-Za-z \t])(\w+://\S+)", "", tweet`

```

In [9]: 1 def remove_url(txt):
2         """Replace URLs found in a text string with nothing
3         (i.e. it will remove the URL from the string).
4
5         Parameters
6         -----
7         txt : string
8             A text string that you want to parse and remove urls.
9
10        Returns
11        -----
12        The same txt string with url's removed.
13        """
14
15        return " ".join(re.sub("([^\0-9A-Za-z \t])|(\w+:\/\/\S+)", "", txt).s

```

- After defining the function, you can call it in a list comprehension to create a list of the clean tweets.

```

In [10]: 1 all_tweets_no_urls = [remove_url(tweet) for tweet in all_tweets]
2         all_tweets_no_urls[:5]

```

```

Out[10]: ['What a pleasure to be contributing to this wonderful tea magazine my arti
cle was about growing tea in the face of',
'YES Why Investing in Women Helps Save the PlanetEducation female empowerm
ent and climatesmart solutions can s',
'New study suggests governments and investors around the world should prio
ritize smallscale low carbon technologie',
'business Here we go again all of a sudden climate change has wedded covid
19 as if distracting us from callin',
'Carbon Creed Issue 24 Thread A symposium of thoughts on climate changeThi
s week the New York Times published']

```

Text Cleanup - Address Case Issues

- Capitalization is also a challenge when analyzing text data. If you are trying to create a list of unique words in your tweets, words with capitalization will be different from words that are all lowercase.

```

In [11]: 1 # Note how capitalization impacts unique returned values
2         ex_list = ["Dog", "dog", "dog", "cat", "cat", ",","]
3
4         # Get unique elements in the list
5         set(ex_list)
6         {'', 'Dog', 'cat', 'dog'}

```

```

Out[11]: {'', 'Dog', 'cat', 'dog'}

```

- To account for this, you can make each word lowercase using the string method `.lower()`. In the code below, this method is applied using a list comprehension.

```
In [13]: 1 # Note how capitalization impacts unique returned values
2 words_list = ["Dog", "dog", "dog", "cat", "cat", ",","]
3
4 # Make all elements in the list lowercase
5 lower_case = [word.lower() for word in words_list]
6
7 # Get all elements in the list
8 lower_case
9
```

```
Out[13]: ['dog', 'dog', 'dog', 'cat', 'cat', ',',']
```

- Now all of the words in your list are lowercase. You can again use `set()` function to return only unique words.

```
In [14]: 1 # Now you have only unique words
2 set(lower_case)
```

```
Out[14]: {'', 'cat', 'dog'}
```

Create List of Lower Case Words from Tweets

- Right now, you have a list of lists that contains each full tweet and you know how to lowercase the words.
- However, to do a word frequency analysis, you need a list of all of the words associated with each tweet. You can use `.split()` to split out each word into a unique element in a list, as shown below.

```
In [15]: 1 # Split the words from one tweet into unique elements
        2 all_tweets_no_urls[0].split()
```

```
Out[15]: ['What',
          'a',
          'pleasure',
          'to',
          'be',
          'contributing',
          'to',
          'this',
          'wonderful',
          'tea',
          'magazine',
          'my',
          'article',
          'was',
          'about',
          'growing',
          'tea',
          'in',
          'the',
          'face',
          'of']
```

- Of course, you will notice above that you have a capital word in your list of words.
- You can combine `.lower()` with `.split()` to remove capital letters and split up the tweet in one step.
- Below is an example of applying these methods to the first tweet in the list.

```
In [16]: 1 # Split the words from one tweet into unique elements
          2 all_tweets_no_urls[0].lower().split()
```

```
Out[16]: ['what',
          'a',
          'pleasure',
          'to',
          'be',
          'contributing',
          'to',
          'this',
          'wonderful',
          'tea',
          'magazine',
          'my',
          'article',
          'was',
          'about',
          'growing',
          'tea',
          'in',
          'the',
          'face',
          'of']
```

- To split and lower case words in all of the tweets, you can string both methods .lower() and .split() together in a list comprehension.

```
In [17]: 1 # Create a List of Lists containing lowercase words for each tweet
2 words_in_tweet = [tweet.lower().split() for tweet in all_tweets_no_urls]
3 words_in_tweet[:2]
```

```
Out[17]: [['what',
'a',
'pleasure',
'to',
'be',
'contributing',
'to',
'this',
'wonderful',
'tea',
'magazine',
'my',
'article',
'was',
'about',
'growing',
'tea',
'in',
'the',
'face',
'of'],
['yes',
'why',
'investing',
'in',
'women',
'helps',
'save',
'the',
'planeteducation',
'female',
'empowerment',
'and',
'climatesmart',
'solutions',
'can',
's']]
```

Calculate and Plot Word Frequency

- To get the count of how many times each word appears in the sample, you can use the built-in Python library collections, which helps create a special type of a Python dictionary.
- The collection.Counter object has a useful built-in method most_common that will return the most commonly used words and the number of times that they are used.
- To begin, flatten your list, so that all words across the tweets are in one list. Note that you could flatten your list with another list comprehension like this: all_words = [item for sublist in tweets_nsw for item in sublist]
- However, it is actually faster to use itertools to flatten the list as follows.


```
In [18]: 1 # List of all words across tweets
2 all_words_no_urls = list(itertools.chain(*words_in_tweet))
3
4 # Create counter
5 counts_no_urls = collections.Counter(all_words_no_urls)
6
7 counts_no_urls.most_common(15)
```

```
Out[18]: [('climate', 758),
 ('the', 732),
 ('change', 559),
 ('to', 467),
 ('of', 367),
 ('and', 330),
 ('a', 249),
 ('is', 246),
 ('in', 236),
 ('on', 179),
 ('we', 157),
 ('for', 138),
 ('climatechange', 110),
 ('are', 105),
 ('that', 96)]
```

- Based on the counter, you can create a Pandas Dataframe for analysis and plotting that includes only the top 15 most common words.

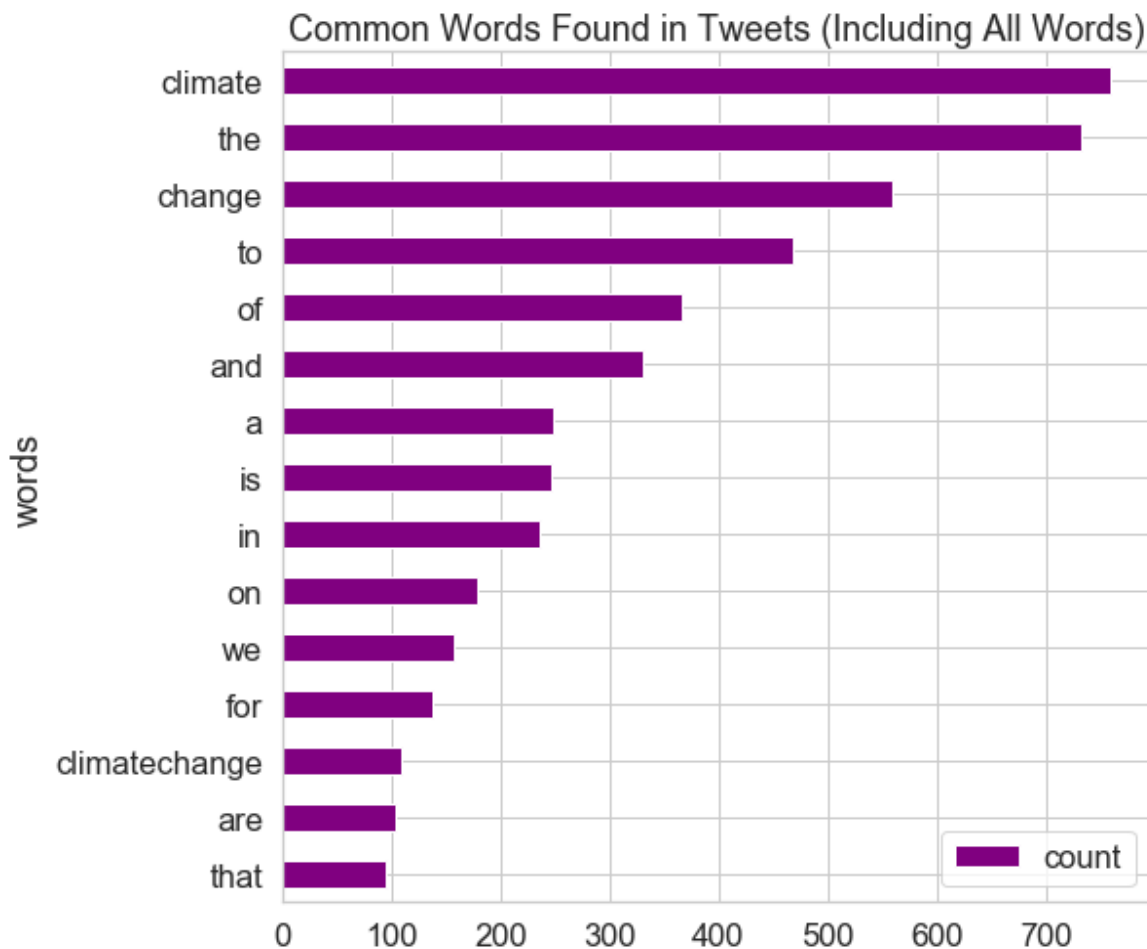
```
In [19]: 1 clean_tweets_no_urls = pd.DataFrame(counts_no_urls.most_common(15),
2                                             columns=['words', 'count'])
3
4 clean_tweets_no_urls.head()
```

```
Out[19]:
```

	words	count
0	climate	758
1	the	732
2	change	559
3	to	467
4	of	367


- Using this Pandas Dataframe, you can create a horizontal bar graph of the top 15 most common words in the tweets as shown below.

```
In [20]: 1 fig, ax = plt.subplots(figsize=(8, 8))
2
3 # Plot horizontal bar graph
4 clean_tweets_no_urls.sort_values(by='count').plot.barh(x='words',
5                                     y='count',
6                                     ax=ax,
7                                     color="purple")
8
9 ax.set_title("Common Words Found in Tweets (Including All Words)")
10
11 plt.show()
```




Remove Stopwords With nltk

- In addition to lowercase words, you may also want to perform additional clean-up, such as removing words that do not add meaningful information to the text you are trying to analysis.
- These words referred to as “stop words” and include commonly appearing words such as who, what, you, etc.
- The Python package nltk, commonly used for text analysis, provides a list of “stop words” that you can use to clean your Twitter data.

In [21]:  1 nltk.download('stopwords')

```
[nltk_data] Downloading package stopwords to  
[nltk_data]   C:\Users\szaki5\AppData\Roaming\nltk_data...  
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[21]: True

In [23]:  1 stop_words = set(stopwords.words('english'))
2
3 *# View a few words from the set*
4 list(stop_words)[0:10]

Out[23]: ['what',
"aren't",
'while',
"mustn't",
'wouldn',
'has',
'should',
'that',
'my',
"you'd"]

- Notice that the stop words provided by nltk are all lower-case. This works well given you already have converted all of your tweet words to lower case using the Python string method `.lower()`.
- Next, you will remove all stop words from each tweet. First, have a look at the words in the first tweet below.

In [24]: 1 words_in_tweet[0]

```
Out[24]: ['what',
          'a',
          'pleasure',
          'to',
          'be',
          'contributing',
          'to',
          'this',
          'wonderful',
          'tea',
          'magazine',
          'my',
          'article',
          'was',
          'about',
          'growing',
          'tea',
          'in',
          'the',
          'face',
          'of']
```

- Below, you remove all of the stop words in each tweet.
- The list comprehension below might look confusing as it is nested.
- The list comprehension below is the same as calling:

for all_words in words_in_tweet:

```
    for word in all_words:
        #remove stop words
```

- Now, compare the words in the original tweet to the words in the tweet after the stop words are removed:

```
In [27]: 1 # Remove stop words from each tweet List of words
        2 tweets_nsw = [[word for word in tweet_words if not word in stop_words]
        3                  for tweet_words in words_in_tweet]
        4
        5 tweets_nsw[0]
```

```
Out[27]: ['pleasure',
          'contributing',
          'wonderful',
          'tea',
          'magazine',
          'article',
          'growing',
          'tea',
          'face']
```

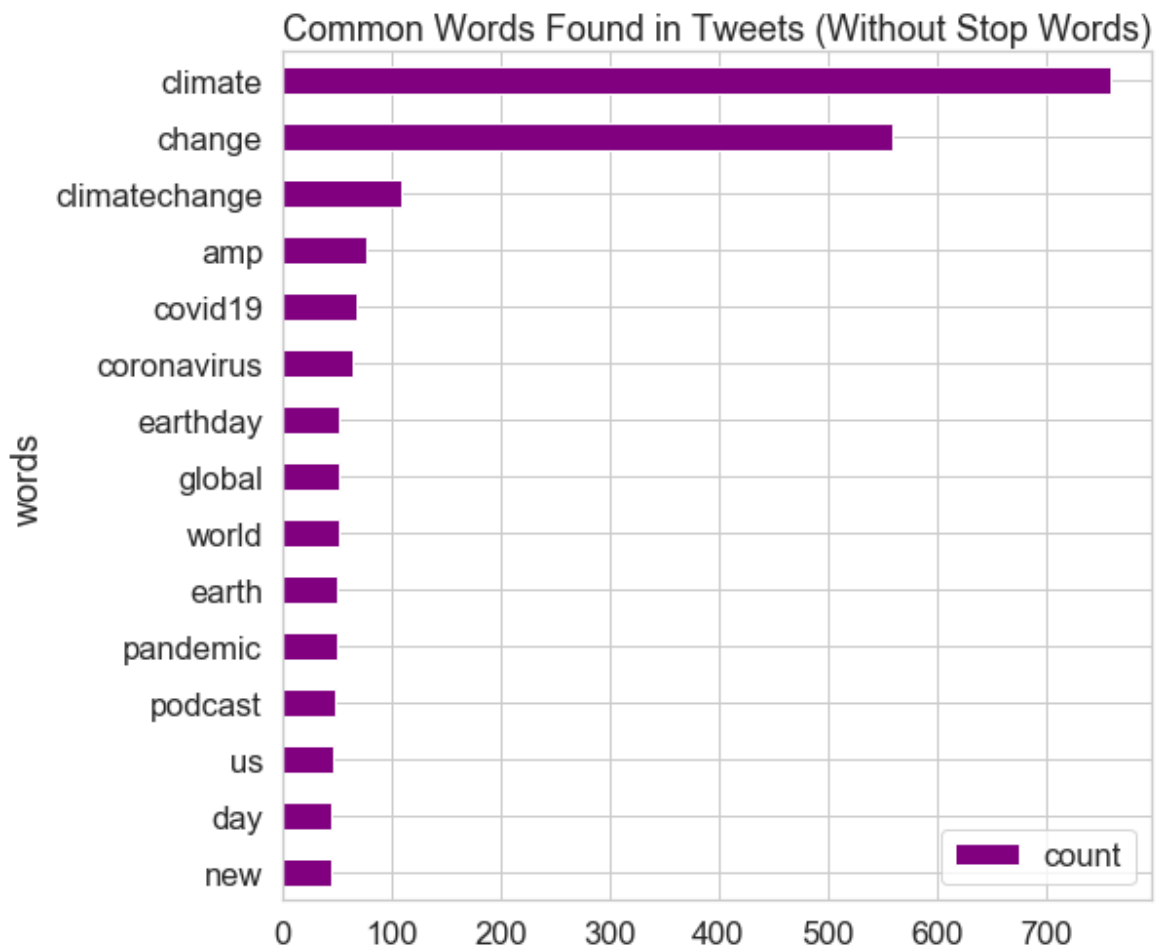
- Again, you can flatten your list and create a counter to return the most commonly used words and the number of times that they are used.

```
In [28]: 1 all_words_nsw = list(itertools.chain(*tweets_nsw))
          2
          3 counts_nsw = collections.Counter(all_words_nsw)
          4
          5 counts_nsw.most_common(15)
```

```
Out[28]: [('climate', 758),
          ('change', 559),
          ('climatechange', 110),
          ('amp', 78),
          ('covid19', 69),
          ('coronavirus', 65),
          ('world', 52),
          ('global', 52),
          ('earthday', 52),
          ('earth', 51),
          ('pandemic', 50),
          ('podcast', 49),
          ('us', 48),
          ('day', 46),
          ('new', 45)]
```

Then, you can create the Pandas Dataframe and plot the word frequencies without the stop words.

```
In [29]: 1 clean_tweets_nsw = pd.DataFrame(counts_nsw.most_common(15),
2                                     columns=['words', 'count'])
3
4 fig, ax = plt.subplots(figsize=(8, 8))
5
6 # Plot horizontal bar graph
7 clean_tweets_nsw.sort_values(by='count').plot.barh(x='words',
8                                                    y='count',
9                                                    ax=ax,
10                                                    color="purple")
11
12 ax.set_title("Common Words Found in Tweets (Without Stop Words)")
13
14 plt.show()
```



Remove Collection Words

- In addition to removing stopwords, it is common to also remove collection words. Collection words are the words that you used to query your data from Twitter.
- In this case, you used “climate change” as a collection term. Thus, you can expect that these terms will be found in each tweet. This could skew your word frequency analysis.
- Below you remove the collection words - climate, change, and climatechange - from the tweets through list comprehension

```
In [30]: 1 collection_words = ['climatechange', 'climate', 'change']
        2 tweets_nsw_nc = [[w for w in word if not w in collection_words]
        3                     for word in tweets_nsw]
```

- Compare the words in first tweet with and without the collection words.

```
In [31]: 1 tweets_nsw[0]
```

```
Out[31]: ['pleasure',
          'contributing',
          'wonderful',
          'tea',
          'magazine',
          'article',
          'growing',
          'tea',
          'face']
```

```
In [32]: 1 tweets_nsw_nc[0]
```

```
Out[32]: ['pleasure',  
          'contributing',  
          'wonderful',  
          'tea',  
          'magazine',  
          'article',  
          'growing',  
          'tea',  
          'face']
```

Calculate and Plot Word Frequency of Clean Tweets

- Now that you have cleaned up your data, you are ready to calculate and plot the final word frequency results.
- Using the skills you have learned, you can flatten the list and create the counter for the words in the tweets.

```
In [34]: 1 # Flatten list of words in clean tweets  
2 all_words_nsw_nc = list(itertools.chain(*tweets_nsw_nc))  
3  
4 # Create counter of words in clean tweets  
5 counts_nsw_nc = collections.Counter(all_words_nsw_nc)  
6  
7 counts_nsw_nc.most_common(15)
```

```
Out[34]: [('amp', 78),  
          ('covid19', 69),  
          ('coronavirus', 65),  
          ('world', 52),  
          ('global', 52),  
          ('earthday', 52),  
          ('earth', 51),  
          ('pandemic', 50),  
          ('podcast', 49),  
          ('us', 48),  
          ('day', 46),  
          ('new', 45),  
          ('science', 41),  
          ('environment', 38),  
          ('crisis', 35)]
```

- To find out the number of unique words across all of the tweets, you can take the len() of the object counts that you just created.

```
In [35]: 1 len(counts_nsw_nc)
```

```
Out[35]: 3744
```

- Last, you can create the Pandas Dataframe of the words and their counts and plot the top 15

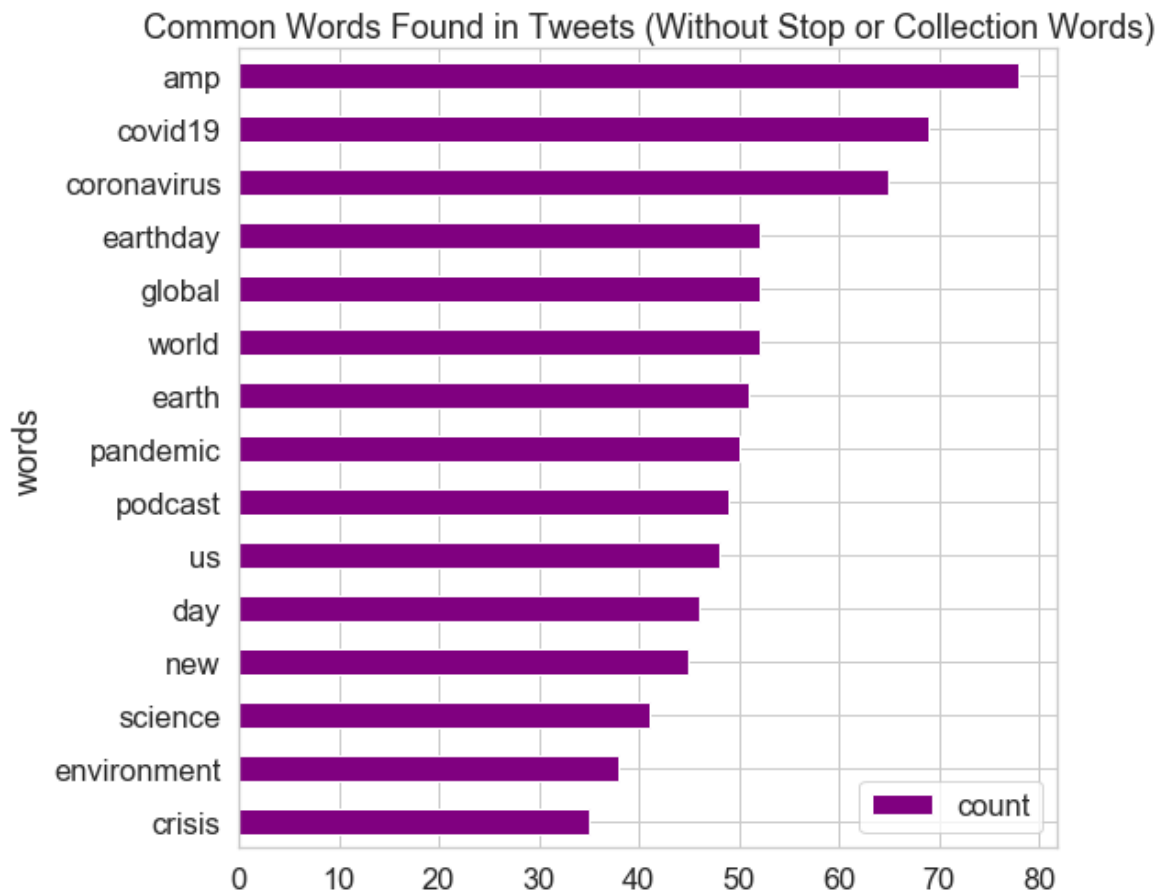
most common words from the clean tweets (i.e. no URLs, stop words, or collection words).

```
In [36]: ▶ 1 clean_tweets_ncw = pd.DataFrame(counts_nsw_nc.most_common(15),  
2                                           columns=['words', 'count'])  
3 clean_tweets_ncw.head()
```

Out[36]:

	words	count
0	amp	78
1	covid19	69
2	coronavirus	65
3	world	52
4	global	52

```
In [37]: 1 fig, ax = plt.subplots(figsize=(8, 8))
2
3 # Plot horizontal bar graph
4 clean_tweets_ncw.sort_values(by='count').plot.barh(x='words',
5 y='count',
6 ax=ax,
7 color="purple")
8
9 ax.set_title("Common Words Found in Tweets (Without Stop or Collection W
10
11 plt.show()
```



```
In [ ]: 1
```