Authorizing an application to access Twitter account data

- To access the Twitter API, you will need 4 things from the your Twitter App page.
- These keys are located in your Twitter app settings in the Keys and Access Tokens tab.

In [36]:

```
#import os to deal with files on the operating system
import os

#import pandas to deal with dataframes and more
import numpy as np
import pandas as pd

#import visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
import itertools

#import collections to deal with collections
import collections

#import tweepy to deal with the tweets
import tweepy as tweepy

#import nltk to deal with natural processing language
#we will study that in details in our Natural Processing course in AI for Business certificate
import nltk
from nltk.corpus import stopwords
import re
import networkx

#to filter warnings

import warnings
warnings.filterwarnings("ignore")

#setting some configurations for seaborn related plots
#setting the background style and font scale
#those are optional but makes the plots look nicer
sns.set(font_scale=1.5)
sns.set_style("whitegrid")
```

In [41]:

```
#required keys and tokens

ACCESS_TOKEN = ''
ACCESS_SECRET = ''
CONSUMER_KEY = ''
CONSUMER_SECRET = ''


auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

twitter_api = tweepy.API(auth)



# Nothing to see by displaying twitter_api except that it's now a
# defined variable

print(twitter_api)
```

```
<tweepy.api.API object at 0x7fdbb0269d30>
```

Search Twitter for Tweets

- Now you are ready to search Twitter for recent tweets!
- Start by finding recent tweets that use the #wildfires hashtag.
- You will use the .Cursor method to get an object containing tweets containing the hashtag #covid19.
- To create this query, you will define the:
  - Search term - in this case #covid19
  - the start date of your search
  - Remember that the Twitter API only allows you to access the past few weeks of tweets, so you cannot dig into the history too far.

In [3]:

```
# Define the search term and the date_since date as variables
search_words = "#covid19"
date_since = "2020-04-01"
```

In [5]:

```
#Below you use .Cursor() to search twitter for tweets containing the search term #wildfires.
#You can restrict the number of tweets returned by specifying a number in the .items() method.
#.items(5) will return 5 of the most recent tweets.

# Collect tweets
# Collect tweets
tweets = tweepy.Cursor(twitter_api.search,
              q=search_words,
              lang="en",
              since=date_since).items(5)

# Collect a list of tweets
[tweet.text for tweet in tweets]
```

Out[5]:

```
['RT @GregAbbott_TX: More glimmers of hope as we “safely” move forward and open up Texas while containing #COVID19.
\n\nTexas A&amp;amp;M, Texas Tech U…',
 "RT @MoHFW_INDIA: #IndiaFightsCorona:\n\nDon't harm our life-savers. Let's keep a positive attitude towards our heal
thcare service personnel,…",
 'RT @JosefinaVidalF: FM @BrunoRguezP: The aggression vs @EmbaCubaUS has been encouraged by the increasing hostile rh
etoric vs #Cuba involvin…',
 "RT @Laurie_Garrett: If you're not a @MSNBC watcher you may be missing @maddow 's recent focus on the meatpacking in
dustry as the driving fo…",
 'RT @GovWhitmer: I’ve said it before, and I’ll say it again — Michigan is an extraordinary place to live because of
the people who call it h…']
```

To Keep or Remove Retweets

In [6]:

```
new_search = search_words + " -filter:retweets"
new_search
```

Out[6]:

```
'#covid19 -filter:retweets'
```

In [7]:
```
tweets = tweepy.Cursor(twitter_api.search,
                       q=new_search,
                       lang="en",
                       since=date_since).items(5)

[tweet.text for tweet in tweets]
```

Out[7]:
```
["Masked man asking for money in the bank.\nThen - *panic* it's a robbery.\nNow - social distancing and masks are man
datory.\n#covid19 #masks",
 'Timeslots for #BHIVE #BUZZFEST is now confirmed Register @ https://t.co/N3u5RTyPIA\n@TVMohandasPai (https://t.co/N3
u5RTyPIA\n@TVMohandasPai) @sanjaynath… https://t.co/0OztpDkduA', (https://t.co/0OztpDkduA',)
 '#Opinion #Covid19 #DanielMartey Money Management lessons from COVID -19 https://t.co/ODpJIqHxJH (https://t.co/ODpJI
qHxJH) https://t.co/3NOL79P8yO', (https://t.co/3NOL79P8yO',)
 'Quick question @GovernorVA... if you pass "laws" when citizens are unable to properly assemble and address said la
w… https://t.co/KpgA2yYVKo', (https://t.co/KpgA2yYVKo',)
 'Tha Closet Studios recording studio will continue to remain closed due to the Covid 19. Our Graphic Design service
s… https://t.co/5Vu0UmnbHT'] (https://t.co/5Vu0UmnbHT'])
```

Who is Tweeting About Covid19?

In [8]:
```
tweets = tweepy.Cursor(twitter_api.search,
                           q=new_search,
                           lang="en",
                           since=date_since).items(5)

users_locs = [[tweet.user.screen_name, tweet.user.location] for tweet in tweets]
users_locs
```

Out[8]:
```
[['iambhanugupta', 'Bengaluru, India'],
 ['sheshagiri_pk', 'Bengaluru'],
 ['news_ghana', 'Accra, Ghana'],
 ['nova_hillbilly', 'Virginia, USA'],
 ['Quincie_Q', 'Texas, USA']]
```

Create a Pandas Dataframe From A List of Tweet Data

In [9]:

```
tweet_text = pd.DataFrame(data=users_locs,
                          columns=['user', "location"])
tweet_text
```

Out[9]:

	user	location
0	iambhanugupta	Bengaluru, India
1	sheshagiri_pk	Bengaluru
2	news_ghana	Accra, Ghana
3	nova_hillbilly	Virginia, USA
4	Quincie_Q	Texas, USA

Customizing Twitter Queries

- Let's see if anyone is talking about China and Covid19

In [10]:

```python
#Note that the code below creates a list that can be queried
#using Python indexing to return the first five tweets.

new_search = "covid19+CCP -filter:retweets"

tweets = tweepy.Cursor(twitter_api.search,
                   q=new_search,
                   lang="en",
                   since='2020-04-01').items(1000)

all_tweets = [tweet.text for tweet in tweets]
all_tweets[:5]
```

Out[10]:

```
['ABSOLUTELY IRRESPONSIBLE &amp; UNFORGIVABLE! #Trudeau WAS REPEATEDLY WARNED BY #CSIS OF #China CORNERING THE MARKET
ON… https://t.co/SzDV1tIi0g', (https://t.co/SzDV1tIi0g',)
 '@Jali_Cat @TheJusticeDept @WhiteHouse True that. This is becoming a communist controlled reaction. Also why track
 A… https://t.co/IhKzZqSFT6', (https://t.co/IhKzZqSFT6',)
 'ᴀᴜᴛᴡᴄɴAfter being threatened and insulted by #CCPChina, this is how #Australia responds. \n\n“Bring #Taiwan into th
e… https://t.co/q5ahorcAIL', (https://t.co/q5ahorcAIL',)
 '@RikJ7 @DrHaircut @joerogan Speculation will continue until the CCP is transparent about WHERE the COVID19 origins…
https://t.co/NT1d4I2Dx4', (https://t.co/NT1d4I2Dx4',)
 "@JonahDispatch There's viruses created and there's viruses exploited for geopolitical gain.\n\nCCP obviously was aw
ar… https://t.co/oHTCE5dUlt"] (https://t.co/oHTCE5dUlt"])
```

Remove URLs (links)

In [11]:

```python
#define a function to remove the urls
def remove_url(txt):
    """Replace URLs found in a text string with nothing
    (i.e. it will remove the URL from the string).

    Parameters
    ----------
    txt : string
        A text string that you want to parse and remove urls.

    Returns
    -------
    The same txt string with url's removed.
    """

    return " ".join(re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", txt).split())
```

In [12]:
```
all_tweets_no_urls = [remove_url(tweet) for tweet in all_tweets]
all_tweets_no_urls[:5]
```

Out[12]:
```
['ABSOLUTELY IRRESPONSIBLE amp UNFORGIVABLE Trudeau WAS REPEATEDLY WARNED BY CSIS OF China CORNERING THE MARKET ON',
 'JaliCat TheJusticeDept WhiteHouse True that This is becoming a communist controlled reaction Also why track A',
 'After being threatened and insulted by CCPChina this is how Australia responds Bring Taiwan into the',
 'RikJ7 DrHaircut joerogan Speculation will continue until the CCP is transparent about WHERE the COVID19 origins',
 'JonahDispatch Theres viruses created and theres viruses exploited for geopolitical gainCCP obviously was awar']
```

In [14]:
```
# Split the words from one tweet into unique elements
all_tweets_no_urls[0].lower().split()
```

Out[14]:
```
['absolutely',
 'irresponsible',
 'amp',
 'unforgivable',
 'trudeau',
 'was',
 'repeatedly',
 'warned',
 'by',
 'csis',
 'of',
 'china',
 'cornering',
 'the',
 'market',
 'on']
```

In [16]:

```python
# Create a list of lists containing lowercase words for each tweet
words_in_tweet = [tweet.lower().split() for tweet in all_tweets_no_urls]
words_in_tweet[:5]
```

Out[16]:

```
[['absolutely',
  'irresponsible',
  'amp',
  'unforgivable',
  'trudeau',
  'was',
  'repeatedly',
  'warned',
  'by',
  'csis',
  'of',
  'china',
  'cornering',
  'the',
  'market',
  'on'],
 ['jalicat',
  'thejusticedept',
  'whitehouse',
  'true',
  'that',
  'this',
  'is',
  'becoming',
  'a',
  'communist',
  'controlled',
  'reaction',
  'also',
  'why',
  'track',
  'a'],
 ['after',
  'being',
  'threatened',
  'and',
  'insulted',
  'by',
  'ccpchina',
  'this',
```

```
 'is',
 'how',
 'australia',
 'responds',
 'bring',
 'taiwan',
 'into',
 'the'],
['rikj7',
 'drhaircut',
 'joerogan',
 'speculation',
 'will',
 'continue',
 'until',
 'the',
 'ccp',
 'is',
 'transparent',
 'about',
 'where',
 'the',
 'covid19',
 'origins'],
['jonahdispatch',
 'theres',
 'viruses',
 'created',
 'and',
 'theres',
 'viruses',
 'exploited',
 'for',
 'geopolitical',
 'gainccp',
 'obviously',
 'was',
 'awar']]
```

Calculate and Plot Word Frequency

In [17]:

```
# List of all words across tweets
all_words_no_urls = list(itertools.chain(*words_in_tweet))

# Create counter
counts_no_urls = collections.Counter(all_words_no_urls)

counts_no_urls.most_common(15)
```

Out[17]:

```
[('the', 909),
 ('ccp', 511),
 ('to', 453),
 ('covid19', 387),
 ('of', 334),
 ('is', 306),
 ('a', 249),
 ('china', 220),
 ('and', 216),
 ('in', 196),
 ('for', 179),
 ('are', 119),
 ('that', 107),
 ('on', 101),
 ('this', 100)]
```

In [18]:

```
#create panda dataframe to analyze and plot the top 15 common words
clean_tweets_no_urls = pd.DataFrame(counts_no_urls.most_common(15),
                                    columns=['words', 'count'])

clean_tweets_no_urls.head()
```

Out[18]:

	words	count
0	the	909
1	ccp	511
2	to	453
3	covid19	387
4	of	334

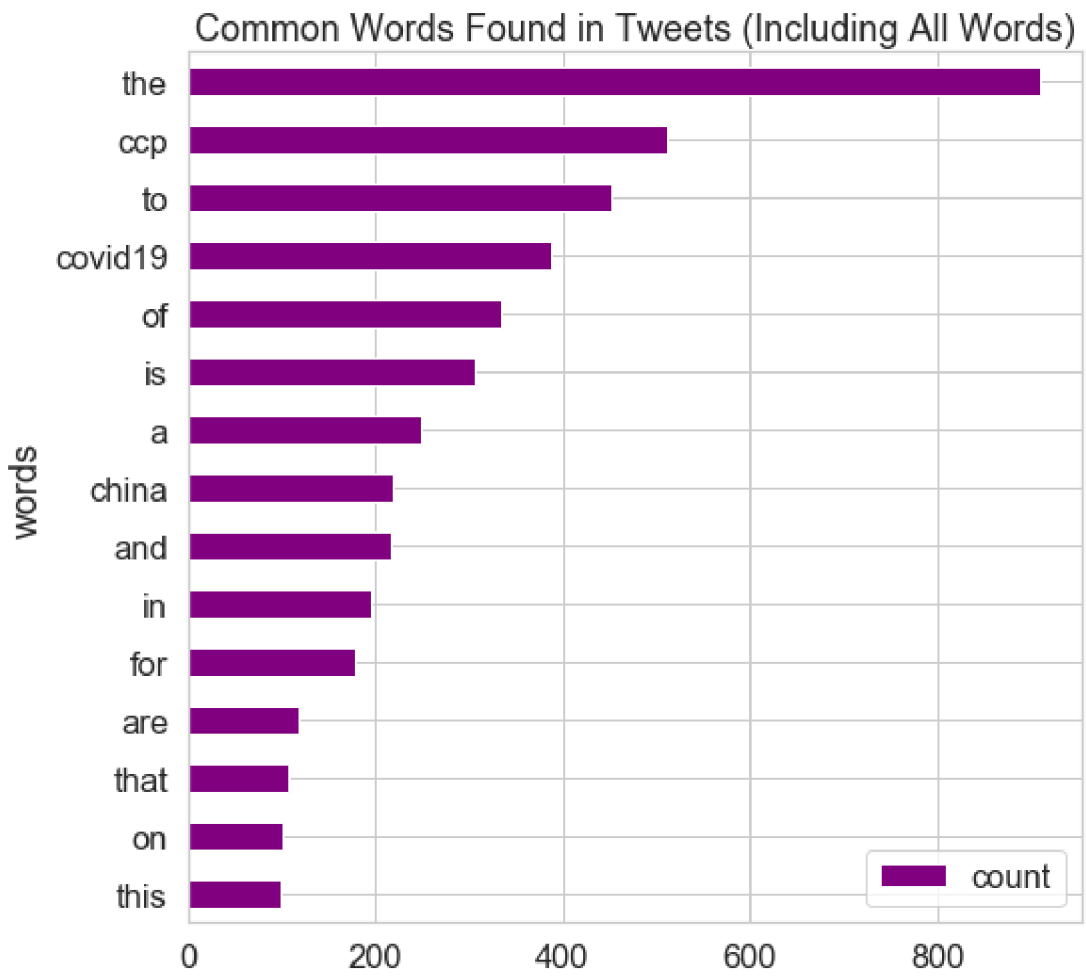
In [20]:

```
# Let's plot the graph of these words
fig, ax = plt.subplots(figsize=(8, 8))

# Plot horizontal bar graph
clean_tweets_no_urls.sort_values(by='count').plot.barh(x='words',
                      y='count',
                      ax=ax,
                      color="purple")

ax.set_title("Common Words Found in Tweets (Including All Words)")

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 [22]:

```
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

# View a few words from the set
list(stop_words)[0:10]
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/macpro/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[22]:

```
['be',
 'because',
 'out',
 'from',
 "should've",
 'not',
 "shouldn't",
 "weren't",
 'during',
 'these']
```

In [23]:

```
#Let's make sure if any of these words are in the tweets we have
words_in_tweet[0]
```

Out[23]:

```
['absolutely',
 'irresponsible',
 'amp',
 'unforgivable',
 'trudeau',
 'was',
 'repeatedly',
 'warned',
 'by',
 'csis',
 'of',
 'china',
 'cornering',
 'the',
 'market',
 'on']
```

In [24]:

```
# Remove stop words from each tweet list of words
tweets_nsw = [[word for word in tweet_words if not word in stop_words]
              for tweet_words in words_in_tweet]

tweets_nsw[0]
```

Out[24]:

```
['absolutely',
 'irresponsible',
 'amp',
 'unforgivable',
 'trudeau',
 'repeatedly',
 'warned',
 'csis',
 'china',
 'cornering',
 'market']
```

In [25]:

```
# Let's redo the count of most common words
all_words_nsw = list(itertools.chain(*tweets_nsw))

counts_nsw = collections.Counter(all_words_nsw)

counts_nsw.most_common(15)
```

Out[25]:

```
[('ccp', 511),
 ('covid19', 387),
 ('china', 220),
 ('world', 97),
 ('chinese', 95),
 ('amp', 89),
 ('virus', 84),
 ('us', 65),
 ('coronavirus', 65),
 ('communist', 43),
 ('wuhan', 43),
 ('people', 41),
 ('chinas', 37),
 ('ccpvirus', 37),
 ('also', 37)]
```

Creating a filtered dataframe

In [26]:

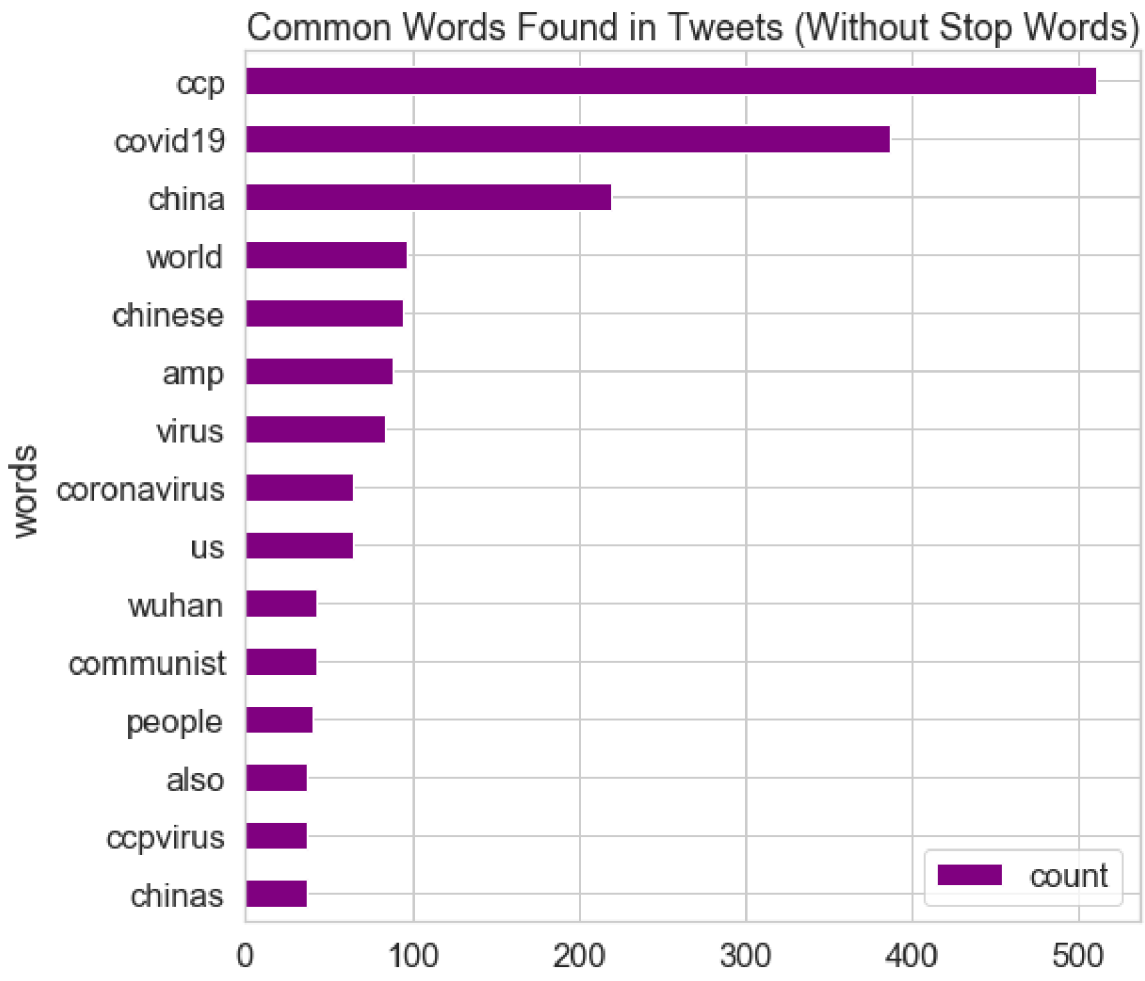
```
clean_tweets_nsw = pd.DataFrame(counts_nsw.most_common(15),
                                columns=['words', 'count'])

fig, ax = plt.subplots(figsize=(8, 8))

# Plot horizontal bar graph
clean_tweets_nsw.sort_values(by='count').plot.barh(x='words',
                       y='count',
                       ax=ax,
                       color="purple")

ax.set_title("Common Words Found in Tweets (Without Stop Words)")

plt.show()
```



Removing the search words (Collection words) from our dataframe

In [27]:
```
#remove the search terms we had used to assist frequency analysis
collection_words = ['ccp', 'covid19']
#tweets with no stop words and no collection words
tweets_nsw_nc = [[w for w in word if not w in collection_words]
                 for word in tweets_nsw]
```

Compare between the tweets of nostop words and the tweets of nostop words and no collections

In [28]:
```
tweets_nsw[0]
```

Out[28]:
```
['absolutely',
 'irresponsible',
 'amp',
 'unforgivable',
 'trudeau',
 'repeatedly',
 'warned',
 'csis',
 'china',
 'cornering',
 'market']
```

In [29]:
```
tweets_nsw_nc[0]
```

Out[29]:
```
['absolutely',
 'irresponsible',
 'amp',
 'unforgivable',
 'trudeau',
 'repeatedly',
 'warned',
 'csis',
 'china',
 'cornering',
 'market']
```

Calculate and Plot Word Frequency of Clean Tweets

In [32]:
```
#Let's plot the frequency analysis
# Flatten list of words in clean tweets
all_words_nsw_nc = list(itertools.chain(*tweets_nsw_nc))

# Create counter of words in clean tweets
counts_nsw_nc = collections.Counter(all_words_nsw_nc)

counts_nsw_nc.most_common(15)
```

Out[32]:
```
[('china', 220),
 ('world', 97),
 ('chinese', 95),
 ('amp', 89),
 ('virus', 84),
 ('us', 65),
 ('coronavirus', 65),
 ('communist', 43),
 ('wuhan', 43),
 ('people', 41),
 ('chinas', 37),
 ('ccpvirus', 37),
 ('also', 37),
 ('pandemic', 33),
 ('australia', 33)]
```

In [33]:

```
#Let's count the unique words too
len(counts_nsw_nc)
```

Out[33]: 4193

In [34]:

```
#Let's display the dataframe again from our cleaned up tweets
clean_tweets_ncw = pd.DataFrame(counts_nsw_nc.most_common(15),
                                columns=['words', 'count'])
clean_tweets_ncw.head()
```

Out[34]:

	words	count
0	china	220
1	world	97
2	chinese	95
3	amp	89
4	virus	84

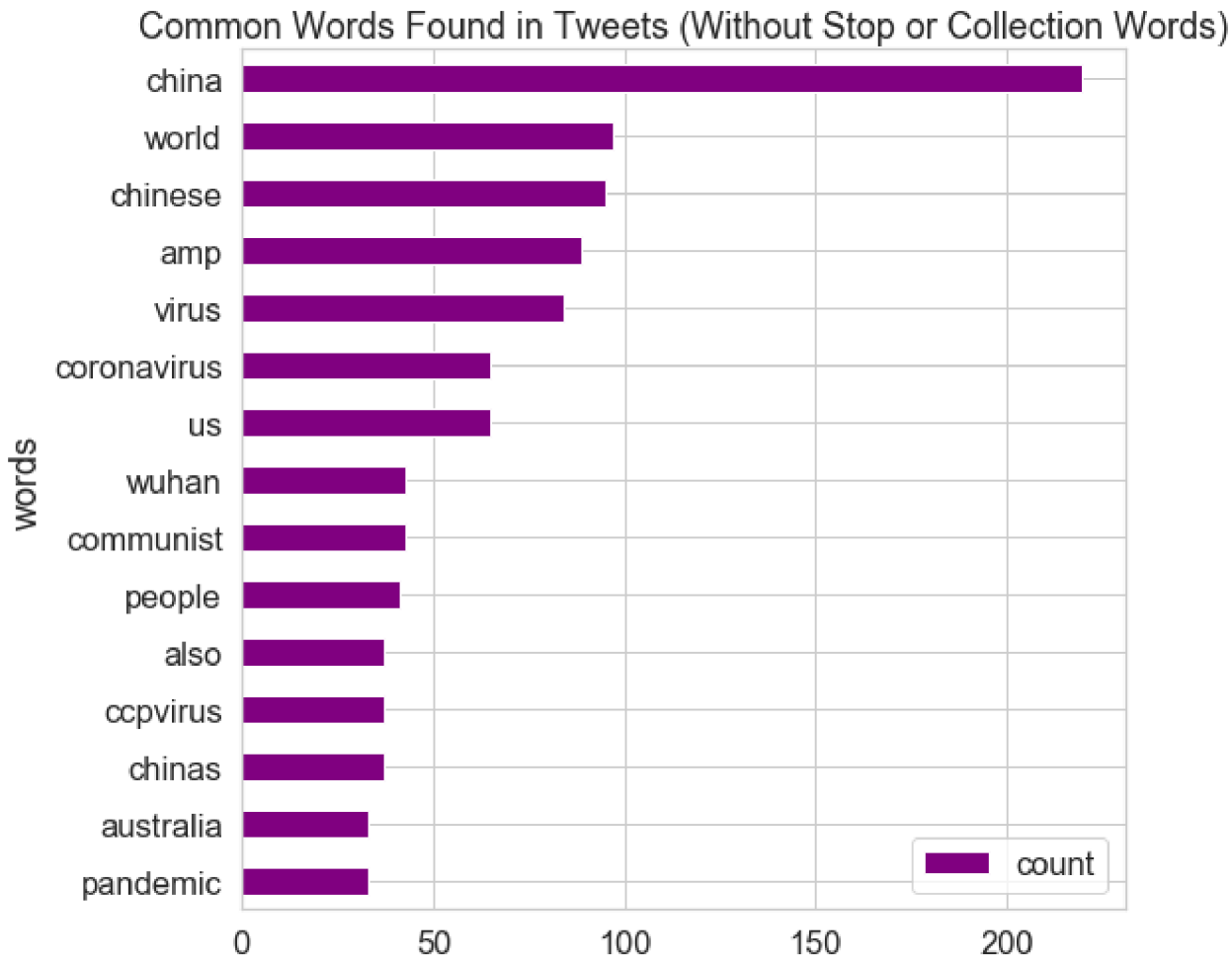
In [35]:

```
fig, ax = plt.subplots(figsize=(8, 8))

# Plot horizontal bar graph
clean_tweets_ncw.sort_values(by='count').plot.barh(x='words',
                      y='count',
                      ax=ax,
                      color="purple")

ax.set_title("Common Words Found in Tweets (Without Stop or Collection Words)")

plt.show()
```



Tweet Extraction

- This method involves using an extractor object from a certain twitter
- our developer account only allows 200 queries at a time

In [42]:

```
def twitter_setup():
    """
    Utility function to setup the Twitter's API
    with our access keys provided.
    """
    # Authentication and access using keys:
    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

    # Return API with authentication:
    api = tweepy.API(auth)
    return api
```

In [50]:

```python
# We create an extractor object:
extractor = twitter_setup()

# We create a tweet list as follows:
tweets = extractor.user_timeline(screen_name="MarshaBlackburn", count=200)
print("Number of tweets extracted: {}.\n".format(len(tweets)))

# We print the most recent 5 tweets:
print("5 recent tweets:\n")
for tweet in tweets[:5]:
    print(tweet.text)
    print()
```

```
Number of tweets extracted: 200.

5 recent tweets:

RT @MarshaBlackburn: China is a reckless and dangerous communist regime.

We must hold them accountable. https://t.co/9qTqYvHgoq (https://t.co/9qTqYvHgoq)

FBI documents confirmed:

Politically driven prosecutions,

Abuse of power,

And the Corrupting presence of the… https://t.co/VJaFUpLBcs (https://t.co/VJaFUpLBcs)

China is a reckless and dangerous communist regime.

We must hold them accountable. https://t.co/9qTqYvHgoq (https://t.co/9qTqYvHgoq)

The Democrats failed at impeachment and will stop at nothing to bring down @realDonaldTrump. https://t.co/Hcy0yt8alD
 (https://t.co/Hcy0yt8alD)

We were giving the @NCAA a chance to get their act together, but I think their leadership is weak. They are tentati…
https://t.co/ldj2oL3bUY (https://t.co/ldj2oL3bUY)
```

In [51]:

```
# We create a pandas dataframe as follows:
data = pd.DataFrame(data=[tweet.text for tweet in tweets], columns=['Tweets'])

# We display the first 10 elements of the dataframe:
display(data.head(10))
```

	Tweets
0	RT @MarshaBlackburn: China is a reckless and d...
1	FBI documents confirmed:\n \nPolitically drive...
2	China is a reckless and dangerous communist re...
3	The Democrats failed at impeachment and will s...
4	We were giving the @NCAA a chance to get their...
5	28.3 million mail-in ballots that have gone mi...
6	RT @SenateCommerce: BREAKING: Today @SenatorWi...
7	RT @DARPA: CBS's Catherine Herridge talks to D...
8	It is paramount that as tech companies utilize...
9	RT @SenTomCotton: The circumstantial evidence ...

In [52]:

```python
# We print info from the first tweet:
print(tweets[0].id)
print(tweets[0].created_at)
print(tweets[0].source)
print(tweets[0].favorite_count)
print(tweets[0].retweet_count)
print(tweets[0].geo)
print(tweets[0].coordinates)
print(tweets[0].entities)
```

```
1256065847900110849
2020-05-01 03:40:09
Twitter for iPhone
0
472
None
None
{'hashtags': [], 'user_mentions': [{'screen_name': 'MarshaBlackburn', 'indices': [3, 19], 'name': 'Sen. Marsha Blackb
urn', 'id_str': '278145569', 'id': 278145569}], 'symbols': [], 'urls': [], 'media': [{'media_url_https': 'https://pb
s.twimg.com/media/EWuUF3LXgAEZWGB.jpg', 'sizes': {'medium': {'w': 1080, 'h': 972, 'resize': 'fit'}, 'thumb': {'w': 15
0, 'h': 150, 'resize': 'crop'}, 'small': {'w': 680, 'h': 612, 'resize': 'fit'}, 'large': {'w': 1080, 'h': 972, 'resiz
e': 'fit'}}, 'type': 'photo', 'source_status_id': 1255998604235268096, 'expanded_url': 'https://twitter.com/MarshaBla
ckburn/status/1255998604235268096/video/1', 'id': 1254888853149810688, 'indices': [106, 129], 'source_user_id_str':
'278145569', 'id_str': '1254888853149810688', 'display_url': 'pic.twitter.com/9qTqYvHgoq', 'source_user_id': 27814556
9, 'media_url': 'http://pbs.twimg.com/media/EWuUF3LXgAEZWGB.jpg', 'source_status_id_str': '1255998604235268096', 'ur
l': 'https://t.co/9qTqYvHgoq'}]}
```

In [53]:

```
# We add relevant data:
data['len']  = np.array([len(tweet.text) for tweet in tweets])
data['ID']   = np.array([tweet.id for tweet in tweets])
data['Date'] = np.array([tweet.created_at for tweet in tweets])
data['Source'] = np.array([tweet.source for tweet in tweets])
data['Likes']  = np.array([tweet.favorite_count for tweet in tweets])
data['RTs']    = np.array([tweet.retweet_count for tweet in tweets])
# Display of first 10 elements from dataframe:
display(data.head(10))
```

	Tweets	len	ID	Date	Source	Likes	RTs
0	RT @MarshaBlackburn: China is a reckless and d...	129	1256065847900110849	2020-05-01 03:40:09	Twitter for iPhone	0	472
1	FBI documents confirmed:\n \nPolitically drive...	139	1256009369826852870	2020-04-30 23:55:43	Twitter for iPhone	1433	657
2	China is a reckless and dangerous communist re...	108	1255998604235268096	2020-04-30 23:12:57	Twitter Media Studio	1028	472
3	The Democrats failed at impeachment and will s...	116	1255983894630973440	2020-04-30 22:14:30	Twitter for iPhone	1060	430
4	We were giving the @NCAA a chance to get their...	140	1255977262740094978	2020-04-30 21:48:09	Twitter for iPhone	125	48
5	28.3 million mail-in ballots that have gone mi...	116	1255961810081140737	2020-04-30 20:46:44	Twitter for iPhone	674	390
6	RT @SenateCommerce: BREAKING: Today @SenatorWi...	140	1255961377715388418	2020-04-30 20:45:01	Twitter for iPhone	0	36
7	RT @DARPA: CBS's Catherine Herridge talks to D...	140	1255961130024960001	2020-04-30 20:44:02	Twitter Web App	0	37
8	It is paramount that as tech companies utilize...	140	1255950332947030017	2020-04-30 20:01:08	Twitter for iPhone	97	33
9	RT @SenTomCotton: The circumstantial evidence ...	140	1255944164795133955	2020-04-30 19:36:37	Twitter for iPhone	0	831

Visualization and basic statistics

Averages and popularity

In [54]:

```
# We extract the mean of lenghts:
mean = np.mean(data['len'])

print("The lenght's average in tweets: {}".format(mean))
```

```
The lenght's average in tweets: 127.62
```

In [55]:

```
# We extract the tweet with more FAVs and more RTs:

fav_max = np.max(data['Likes'])
rt_max  = np.max(data['RTs'])

fav = data[data.Likes == fav_max].index[0]
rt  = data[data.RTs == rt_max].index[0]

# Max FAVs:
print("The tweet with more likes is: \n{}".format(data['Tweets'][fav]))
print("Number of likes: {}".format(fav_max))
print("{} characters.\n".format(data['len'][fav]))

# Max RTs:
print("The tweet with more retweets is: \n{}".format(data['Tweets'][rt]))
print("Number of retweets: {}".format(rt_max))
print("{} characters.\n".format(data['len'][rt]))
```

```
The tweet with more likes is:
Let’s quarantine China!
Number of likes: 21676
23 characters.

The tweet with more retweets is:
RT @WhiteHouse: President @realDonaldTrump offered his condolences to Americans across the south who just endured dea
dly tornadoes and seve…
Number of retweets: 9477
140 characters.
```

Time series

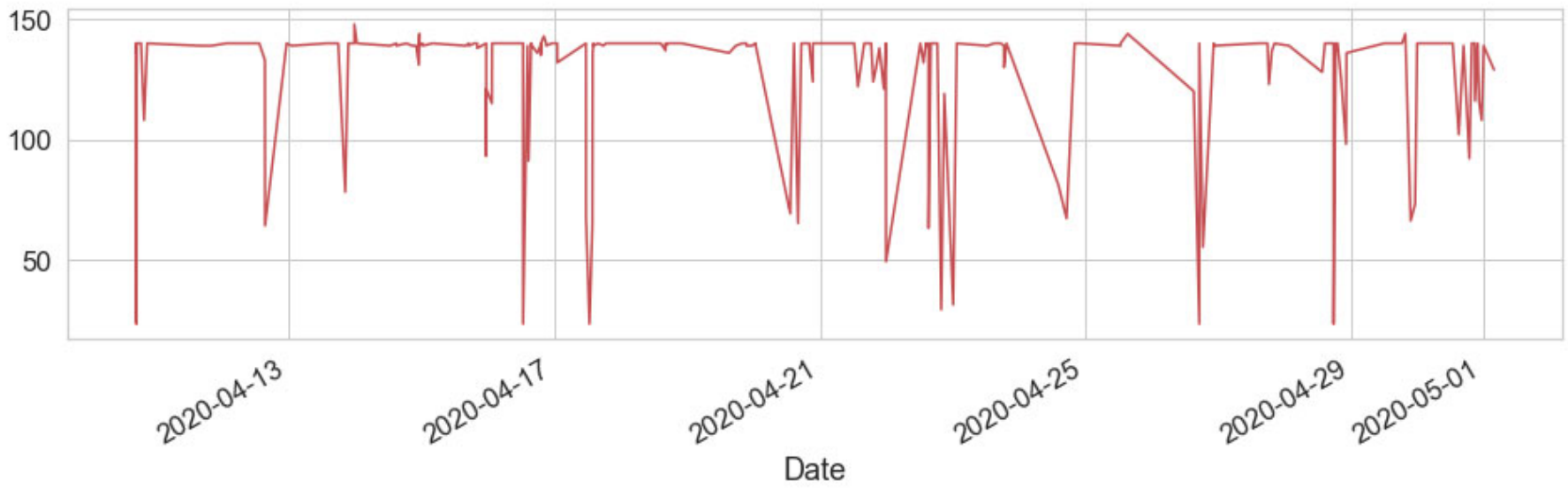
- Pandas has its own object for time series. Since we have a whole vector with creation dates, we can construct time series respect tweets lengths, likes and retweets.
- The way we do it is:

In [56]:

```
# We create time series for data:
tlen = pd.Series(data=data['len'].values, index=data['Date'])
tfav = pd.Series(data=data['Likes'].values, index=data['Date'])
tret = pd.Series(data=data['RTs'].values, index=data['Date'])
```

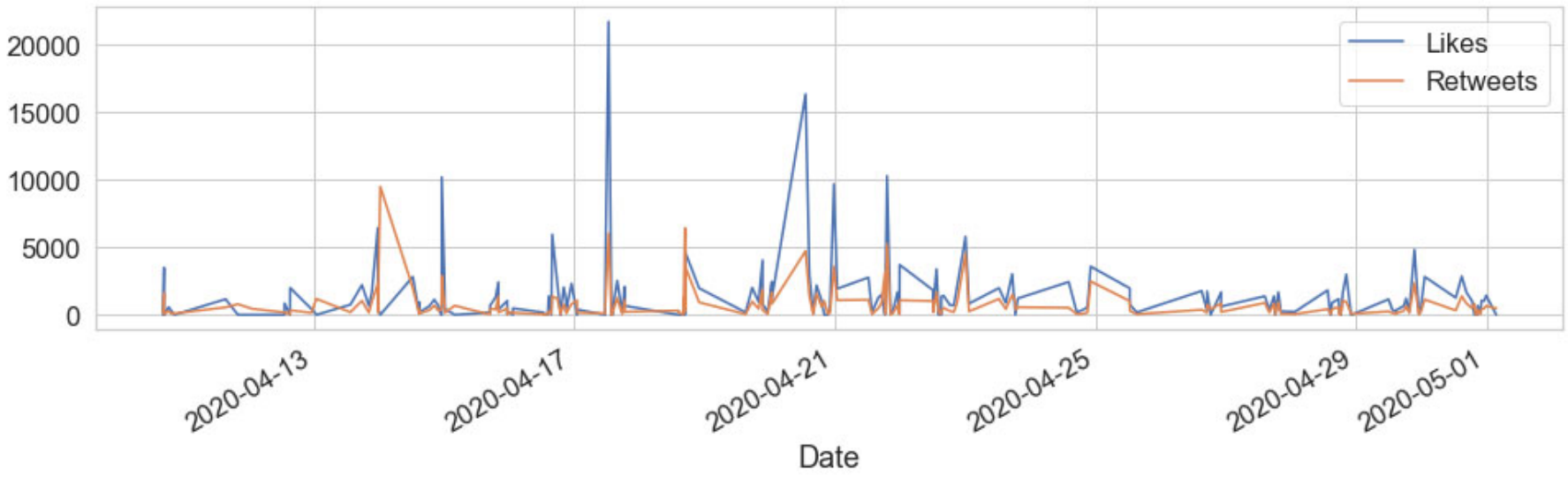
In [57]:

```
# Lenghts along time:
tlen.plot(figsize=(16,4), color='r');
```



In [58]:

```
# Likes vs retweets visualization:
tfav.plot(figsize=(16,4), label="Likes", legend=True)
tret.plot(figsize=(16,4), label="Retweets", legend=True);
```



In [59]:

```python
# We obtain all possible sources:
sources = []
for source in data['Source']:
    if source not in sources:
        sources.append(source)

# We print sources list:
print("Creation of content sources:")
for source in sources:
    print("* {}".format(source))
```

```
Creation of content sources:
* Twitter for iPhone
* Twitter Media Studio
* Twitter Web App
```

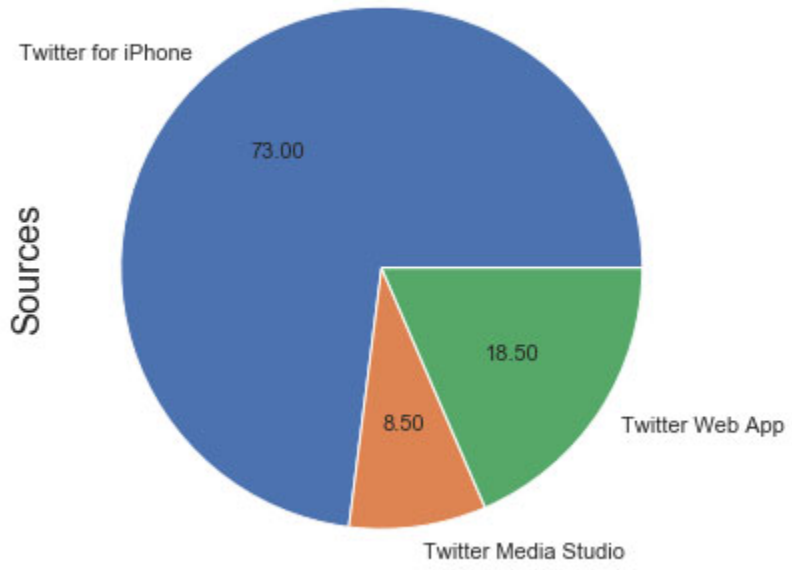
In [60]:

```
#Let's count number of sources to create a pie chart
# We create a numpy vector mapped to labels:
percent = np.zeros(len(sources))

for source in data['Source']:
    for index in range(len(sources)):
        if source == sources[index]:
            percent[index] += 1
            pass

percent /= 100

# Pie chart:
pie_chart = pd.Series(percent, index=sources, name='Sources')
pie_chart.plot.pie(fontsize=11, autopct='%.2f', figsize=(6, 6));
```



Sentiment Analysis

- used to determine whether the tweets retrieved are of a positive or negative attitude

In [62]:

```python
from textblob import TextBlob
import re

def clean_tweet(tweet):
    '''
    Utility function to clean the text in a tweet by removing
    links and special characters using regex.
    '''
    return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())

def analize_sentiment(tweet):
    '''
    Utility function to classify the polarity of a tweet
    using textblob.
    '''
    analysis = TextBlob(clean_tweet(tweet))
    if analysis.sentiment.polarity > 0:
        return 1
    elif analysis.sentiment.polarity == 0:
        return 0
    else:
        return -1
```

In [63]:

```
# We create a column with the result of the analysis:
data['SA'] = np.array([ analize_sentiment(tweet) for tweet in data['Tweets'] ])

# We display the updated dataframe with the new column:
display(data.head(10))
```

	Tweets	len	ID	Date	Source	Likes	RTs	SA
0	RT @MarshaBlackburn: China is a reckless and d...	129	1256065847900110849	2020-05-01 03:40:09	Twitter for iPhone	0	472	-1
1	FBI documents confirmed:\n \nPolitically drive...	139	1256009369826852870	2020-04-30 23:55:43	Twitter for iPhone	1433	657	1
2	China is a reckless and dangerous communist re...	108	1255998604235268096	2020-04-30 23:12:57	Twitter Media Studio	1028	472	-1
3	The Democrats failed at impeachment and will s...	116	1255983894630973440	2020-04-30 22:14:30	Twitter for iPhone	1060	430	-1
4	We were giving the @NCAA a chance to get their...	140	1255977262740094978	2020-04-30 21:48:09	Twitter for iPhone	125	48	-1
5	28.3 million mail-in ballots that have gone mi...	116	1255961810081140737	2020-04-30 20:46:44	Twitter for iPhone	674	390	-1
6	RT @SenateCommerce: BREAKING: Today @SenatorWi...	140	1255961377715388418	2020-04-30 20:45:01	Twitter for iPhone	0	36	0
7	RT @DARPA: CBS's Catherine Herridge talks to D...	140	1255961130024960001	2020-04-30 20:44:02	Twitter Web App	0	37	0
8	It is paramount that as tech companies utilize...	140	1255950332947030017	2020-04-30 20:01:08	Twitter for iPhone	97	33	0
9	RT @SenTomCotton: The circumstantial evidence ...	140	1255944164795133955	2020-04-30 19:36:37	Twitter for iPhone	0	831	0

Analyzing the results

- To have a simple way to verify the results, we will count the number of neutral, positive and negative tweets and extract the percentages.

In [64]:

```
# We construct lists with classified tweets:
pos_tweets = [ tweet for index, tweet in enumerate(data['Tweets']) if data['SA'][index] > 0]
neu_tweets = [ tweet for index, tweet in enumerate(data['Tweets']) if data['SA'][index] == 0]
neg_tweets = [ tweet for index, tweet in enumerate(data['Tweets']) if data['SA'][index] < 0]
```

In [65]:

```
# We print percentages:

print("Percentage of positive tweets: {}%".format(len(pos_tweets)*100/len(data['Tweets'])))
print("Percentage of neutral tweets: {}%".format(len(neu_tweets)*100/len(data['Tweets'])))
print("Percentage de negative tweets: {}%".format(len(neg_tweets)*100/len(data['Tweets'])))
```

```
Percentage of positive tweets: 32.5%
Percentage of neutral tweets: 41.5%
Percentage de negative tweets: 26.0%
```

Conclusion

- The users tweets so far have been mostly Neutral in Sentiment.
- Her tweets do not appear to cause alarm at this time.

In [ ]: