Twitter Sentiment Analysis part1

January 24, 2021

[1]:
```
import pandas as pd
import json
import credentials
import tweepy_custom
from monkeylearn import MonkeyLearn
import os
import re
```

1 Tweepy API

to help make the process faster, and more modular I made a separate py file containing classes relevant to my use of tweepy.

[2]:
```
tweepy = tweepy_custom
```

[3]:
```
client = tweepy.TwitterClient()
tweets = client.search(query='peloton',count=100)
```

[4]:
```
df = tweepy.convert_df(tweets)
```

[5]:
```
if not os.path.isdir(r'C:\Users\windows\BLOG\twitter api\clean_tweets'):
    os.mkdir(r'C:\Users\windows\BLOG\twitter api\clean_tweets')
else:
    print('clean_tweets dir exists')

if not os.path.isdir(r'C:\Users\windows\BLOG\twitter api\unclean_tweets'):
    os.mkdir(r'C:\Users\windows\BLOG\twitter api\unclean_tweets')
else:
    print('unclean_tweets dir exists')
```

```
clean_tweets dir exists
unclean_tweets dir exists
```

[6]:
```
if not os.path.isfile('unclean_tweets/tweets'):
    df.to_csv('unclean_tweets/tweets',encoding ='utf8')
    print('file created')
else:
    print('file exists, try another filename')
```

1.1 Inspecting the data

[7]:
```
tweets = df.copy()
tweets.head()
```

[7]:
```
                                                text  retweet_count  \
0          @jgudzik @PhillyD https://t.co/7rSYceHwjg              0
1  @SherylNYT Do people in Scranton not have a Pe…              0
2  https://t.co/iSRf0wc6jO Biden Peloton Raises S…              0
3  Get $100 off accessories when you buy the a #P…              2
4  Is Joe Biden's Peloton bike really a cybersecu…              3

   favorite_count          created_at
0               0 2021-01-20 23:59:44
1               0 2021-01-20 23:59:14
2               0 2021-01-20 23:59:12
3               2 2021-01-20 23:58:48
4              15 2021-01-20 23:58:41
```

[8]:
```
tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   text            100 non-null    object
 1   retweet_count   100 non-null    int64
 2   favorite_count  100 non-null    int64
 3   created_at      100 non-null    datetime64[ns]
dtypes: datetime64[ns](1), int64(2), object(1)
memory usage: 3.2+ KB
```

[9]:
```
tweets['retweet_count'].agg(['mean','median','max'])
```

[9]:
```
mean       164.62
median       0.00
max       1768.00
Name: retweet_count, dtype: float64
```

[10]:
```
tweets['favorite_count'].agg(['mean','median','max'])
```

[10]:
```
mean       1.35
median     0.00
max       56.00
Name: favorite_count, dtype: float64
```

1.2 Preprocessing data

Using the ascii table to remove any emojis or text that may effect the analysis. https://theasciicode.com.ar/

[11]:
```
tweets['text_c'] = tweets['text'].str.replace('\n','')
```

[12]:
```
def acii (string):
    for c in string:
        if ord(c)<32 or ord(c)>122:
            string =  string.replace(c,'')
    return string
```

[13]:
```
tweets['text_c'] = tweets['text_c'].apply(acii)
```

1.2.1 Removing adverts

On social media there's bound to be adverts, some are more sinister and tricky to identify other use phrases such as 'Giveaway!' or 'buy now'. I want to remove these as they are usually tonally positive.

remove HTTPS, get x amount, win, prize, less than 3

[14]:
```
patterns = [r'[Gg][iI][Vv][Ee][Aa][Ww][Aa][Yy]',
            r'[Ww][Ii][Nn]',
            r'[Gg][Ee][Tt]\s[$£][0-9]*',
            r'[Pp][Rr][Ii][Zz][Ee]',
            r'BUY NOW',
            r'buy now',
           r'when you buy']
mask = {}
for i,p in enumerate(patterns):
    mask[i] = tweets['text_c'].str.contains(p)
```

[15]:
```
#Combine the masks for each pattern
for i in mask:
    if not i ==0:
        m = m | (mask[i-1] | mask[i])
    else:
        m= mask[0]
```

[16]:
```
tweets[m]
```

[16]:

	text	retweet_count	\
3	Get $100 off accessories when you buy the a #P…	2	
6	The Complete Home Gym Giveaway Featuring Pelot…	0	
50	Enter now to win a brand-new Peloton Bike, ret…	0	

	favorite_count	created_at	\

```
3                2 2021-01-20 23:58:48
6                0 2021-01-20 23:57:42
50               0 2021-01-20 23:50:16

                                                  text_c
3   Get $100 off accessories when you buy the a #P…
6   The Complete Home Gym Giveaway Featuring Pelot…
50  Enter now to win abrand-new Peloton Bike, reta…
```

Dropping ads by index.

[17]:
```
tweets.drop(index = tweets[m].index,inplace=True)
```

1.2.2 Removing web links

I initially thought that any link would mean the tweet was an advert, however, that's not the case. Instead of removing any tweet with a weblink, I will just replace it with an empty string.

[18]:
```
link_p = '(https*://\S*)'
tweets['text_c'] =tweets['text_c'].str.replace(link_p,'')
```

1.2.3 Removing tweets with less that 3 words

[19]:
```
three_or_more = r'.+\s+.+\s+.+\s+.+'

large_tweets =tweets['text_c'].str.contains(three_or_more)
small_tweets = (large_tweets ==False)
tweets.drop(index=tweets[small_tweets].index,inplace=True)

tweets =tweets.reset_index()
tweets.drop(columns ='index',inplace=True)
```

1.3 Preping and exporting clean data of sentiment analysis

Lucky this data has been simple to clean. I think partially because I had already formatted what attributes I needed in the tweepy_custom.py file. By creating a class to filter only the full text this stopped any retweets and therefore any duplicates. It also allowed me to create a class to return only the necessary keys in the format of a dataframe.

[20]:
```
cleaned_tweets = tweets.copy()
cleaned_tweets =cleaned_tweets.reset_index()
cleaned_tweets.drop(columns ='text',inplace=True)
cleaned_tweets.rename(columns={'text_c':'text'},inplace=True)
```

1.3.1 Exporting clean csv file.

[21]:
```
cleaned_tweets.to_csv('clean_tweets/tweet_data_cleaned',encoding ='utf8')
```

1.3.2 Formatting to JSON for MonkeyLearn

Monkey learn requires that you either pass your text data as a list or as a JSON. The benefit of formatting it to a JSON is that it allows for an extra variable **'external_id'**, this will allow me to match my results back to the dataframe.

[22]:
```
data = cleaned_tweets[['text','index']]
#using reset_index to create an external_id, this way i can match the data back␣
 ↪up with id
data = data.rename(columns={'index':'external_id'})
data = data.applymap(str)

#Loading it to JSON
data_json = data.to_json(orient = 'records')
data_json =json.loads(data_json)
```

1.4 MonkeyLearn

[ ]:
```
ml = MonkeyLearn('63b9ee718f78913ef76792fecd3e4b08da783f63')
data = ["This is a great tool!"]
model_id = 'cl_pi3C7JiL'
result = ml.classifiers.classify(model_id, data_json)
```

[ ]:
```
results = result.body
results[0].keys()
```

[ ]:
```
result_data = pd.DataFrame(results)
```

[ ]:
```
result_data['tag_name'] = [r['classifications'][0]['tag_name']for r in results]
result_data['confidence'] = [r['classifications'][0]['confidence']for r in␣
 ↪results]
result_data.drop(columns={'classifications'},inplace=True)
```

[28]:
```
result_data['tag_name']
```

[28]:
```
0     Negative
1      Neutral
2      Neutral
3      Neutral
4     Negative
5      Neutral
6     Negative
7      Neutral
8      Neutral
9      Neutral
10    Negative
11     Neutral
Name: tag_name, dtype: object
```

1.4.1 Exporting results as csv

[ ]:
```
result_data.to_csv('results/sentiment_results_twitter',encoding ='utf8')
```