wrangle_report

October 12, 2020

1 Wrangling WeRateDogs twitter data - overview

`Christine Shuttleworth, 12st of October 2020`

Introduction

WeRateDogs Twitter data contains some very entertaining data rating dogs and their images, that are being sent by their owners. The data avaiable comes from three sources: firstly a csv archive file which contains most of the data from the twitter archive of WeRateDogs. Secondly, some extra data that was extracted via the Twitter API and dog image predictions from a neural network outcome tsv file.

Load twitter_archive_enhanced.csv and learn about the data

[1034]:
```
df_ta = pd.read_csv('twitter-archive-enhanced.csv')
```

There were not real problems reading in the csv file. Looking at the data however, some very obvious structural problems and qualitiy problems became apparent pretty quickly. They are listed in the assessment section below.

Request data from the twitter API and load it into a dataframe

Some additional data like retweet count and favorite count, where not extracted with the twitter archive. These I extracted using the twitter API. This step was by far the most difficult. Setting up a Twitter developer account and getting log in credentials was the first step. Then I looked for a way to access the API without hard coding the credentials into the Jupyter notebook. There are several ways to do this, see below. I finally decided to use the dotenv method. I used two methods to extract the extra data: once writing only the wanted data to a csv file or secondly writing the all the data available through the API to a text file using JSON dump. Using the second method, turned out to be more useful, as I found out that I then had extra data to look at, which I did not realise I wanted at the start.

https://developer.twitter.com/en/docs/labs/tweets-and-users/quick-start/get-tweets

[1040]:
```
##Using .env file and python-dotenv to keep access token safe
#pip install -U python-dotenv

#import os
#from pathlib import Path  # Python 3.6+ only
env_path = Path('.') / '.env'
load_dotenv(dotenv_path=env_path)
```

```
consumer_key = os.getenv("TWAPIKEY")
consumer_secret = os.getenv("TWAPISECRETKEY")

#use tweepy to access twitter API with OAuth2

auth = tw.AppAuthHandler(consumer_key, consumer_secret)

#Other option to store passkey safely:
#1. could use a python .config file and the config library to store access␣
 ↪token e.g. with wikiart API
#response = requests.get(f'https://www.wikiart.org/en/Api/2/login?
 ↪accessCode={cfg.twitter['api_key']}&secretCode={cfg.
 ↪twitter['api_secret_key']')

#2. secure storage of access details with yaml
#import yaml

#with open("config.yml", 'r') as ymlfile:
#    cfg = yaml.safe_load(ymlfile)

#print(cfg[api_creds'access_code'])
#print(cfg[api_creds'secret_code'])

#3.using magic command to access variables in .env
#%env
##Get, set, or list environment variables.

##Usage:

#%env: lists all environment variables/values
#%env var: get value for var
#%env var val: set value for var
#%env var=val: set value for var
#%env var=$val: set value for var,

##using python expansion if possible
```

[1047]:
```
## Extracting all data available through the twitter archive using the twitter␣
 ↪API and writing the returned JSON to a text file.
def check_hashtag(tw_json):
    try:
        hashtags = tw_json['entities']['hashtags'][0]['text']
    except IndexError:
        hashtags = np.nan
    return hashtags
```

```
def check_jpg_url(tw_json):
    try:
        jpg_url = tw_json['entities']['media'][0]['url']
    except KeyError:
        jpg_url = np.nan
    return jpg_url

def check_expanded_url(tw_json):
    try:
        expanded_url = tw_json['extended_entities']['media'][0]['expanded_url']
    except KeyError:
        expanded_url = np.nan
    return expanded_url

j=0

df_json = pd.DataFrame(columns = ['tweet_id', 'favorite_count',␣
 ↪'retweet_count','hashtags','jpg_url_json', 'expanded_url'])

with open ('tweet_json.txt', 'r') as file:
    for line in file:
        tw_json = json.loads(line)
        j+=1

        hashtags = check_hashtag(tw_json)
        jpg_url = check_jpg_url(tw_json)
        expanded_url = check_expanded_url(tw_json)

        df_json = df_json.append({'tweet_id':tw_json['id'], 'favorite_count':
 ↪tw_json['favorite_count'], 'retweet_count':tw_json['retweet_count'], \
                                  'hashtags':hashtags,'jpg_url_json':jpg_url,␣
 ↪'expanded_url':expanded_url}, ignore_index=True)

print(f'Total lines in dataset:{j}')
```

```
Total lines in dataset:2331
```

Request data from URL and load .tsv file into dataframe

Finally, the results of a neural network prediction of all the dogs images in the twitter archive was ingested using the requests library. There was no problem there.

[1049]:
```
url='https://d17h27t6h515a5.cloudfront.net/topher/2017/August/
 ↪599fd2ad_image-predictions/image-predictions.tsv'
response = requests.get(url)
#response.content

with open('image-predictions.tsv', 'wb') as file:
```

```
    file.write(response.content)

df_pre = pd.read_csv('image-predictions.tsv', delimiter='\t')
```