Twitter Data Wrangling

Introduction

Quoting from Udacity's page
Real-world data rarely comes clean. Using Python and its libraries, you will gather data from a variety of sources and in a variety of formats, assess its quality and tidiness, then clean it. This is called data wrangling. You will document your wrangling efforts in a Jupyter Notebook, plus showcase them through analyses and visualizations using Python (and its libraries) and/or SQL.

Goals

The goal of this project is to wrangle WeRateDogs Twitter data to create interesting and trustworthy analyses and visualizations. The Twitter archive is great, but it only contains very basic tweet information. Additional gathering, then assessing and cleaning is required for "*Wow!*"-worthy analyses and visualizations.

Data Wrangling

To begin the project, we will start with these processes below. There are 3 big steps of data wrangling, which are

Data Gathering

In this project, we gathered data from several data sources

- Downloaded csv that contains some information about @dog_rates tweets (twitter-archive-enhanced-2.csv)
- Twitter's API to get some additional data that is not included in the csv file (such as retweets and favorite counts). Twitter's API requires authentication to access it. You can follow this guide on how to connect to a developer account and get an app set up
- Dogs' image prediction that contains the prediction of dog's breed using 3 different algorithms (image-predictions-3)

Once you have your Twitter account and Twitter app set up, the following code, which is provided in the Getting started portion of the Tweepy documentation, will create an API object that you can use to gather Twitter data.

```
# connect to Twitter API
consumer_key = 'HIDDEN'
consumer_secret = 'HIDDEN'
access_token = 'HIDDEN'
access_secret = 'HIDDEN'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

# set wait limit is True
api = tweepy.API(auth_handler = auth,
                 wait_on_rate_limit = True,
                 wait_on_rate_limit_notify = True)
```

- You can get a consumer key, consumer secret key, access token, and access secret token in your Twitter app. Those are secret information, nobody can see it
- Since Twitter's API has a limitation of pulling the data, we can set the wait limit to avoid a failure of pulling the data

After connecting to the API, you can call the API and also store the additional information by looping each tweet_id in our downloaded csv file

```
# Query Twitter's API for JSON data for each tweet ID in the Twitter archive
count = 0
fails_dict = {}
start = timer() # set timer

# Save each tweet's returned JSON as a new line in a .txt file
with open('tweet_json.txt', 'w') as outfile:
    # This loop will likely take 20-30 minutes to run because of Twitter's rate limit
    for tweet_id in tweet_ids:
        count += 1
        print(str(count) + ": " + str(tweet_id))
        try:
            tweet = api.get_status(tweet_id, tweet_mode='extended')
            print("Success")
            json.dump(tweet._json, outfile)
            outfile.write('\n')
        except tweepy.TweepError as e:
            print("Fail")
            fails_dict[tweet_id] = e
            pass

end = timer() # end timer
print(end - start)
print(fails_dict)
```

- If the data is succeeded to be read, it will be dumped to our output file
- If it fails, it will be skipped and we can see the error reason because we store it as well in dictionary

We now can extract retweet count and favorite count columns. After that we can join all 3 data sources into one table using tweet_id. The details of the Data Gathering process can be found in the python script.

Data Assessing

In this phase, we assessed all the data including rows and columns if there seemed to be some issues found. We can start with checking the data type or structure. After assessing the data, we found some potential issues that belong to each category

1. **Completeness**
    a. in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id, and retweeted_status_timestamp have many missing values
    b. expanded_urls also has missing values. I think url is not relevant to be analyzed
    c. all columns related to dog predictions have missing values
2. **Tidiness**
    a. name, p1, p1_conf, p1_dog, p2, p2_conf, p2_dog, p3, p3_conf, p3_dog columns' names are not representative
    b. floofer, pupper, puppo, and doggo are separated into many columns
3. **Quality**
    a. dog that is categorized into more than 1 dog type (example: both doggo and pupper) → I realized this issue during the cleaning
    b. 'None' value instead of null in floofer, pupper, puppo, and doggo
    c. some tweets are retweets, we only need original tweet (example: tweet_id = 760153949710192640)
    d. some tweets have no image, we want to get tweets with image(s) (example: tweet_id = 886267009285017600)
    e. timestamp column shouldn't be in string, img_num should be int, ['p1_dog','p2_dog','p3_dog'] should be not object
    f. inconsistency of using "-" or "_" and inconsistency of lowercase and uppercase in p1, p2, and p3 columns
    g. incorrect value because of hardcoded values (example: tweet_id = 832088576586297345, 740373189193256964) and some rows have denominator more than 10
    h. names of dog are unclear because of hardcoded values and also null values are filled as 'None'
    i. some columns are not relevant for analysis (expanded_urls, in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id, retweeted_status_timestamp, floofer, pupper, puppo, and doggo)

Data Cleaning

After defining all potential issues, we clean the data one by one. The details of code for each issue can be found in python script. Once all issues have been cleaned, the cleaned data is stored into csv named 'twitter_archive_master.csv'. The final data is

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1981 entries, 0 to 2330
Data columns (total 19 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            1981 non-null   int64
 1   timestamp           1981 non-null   datetime64[ns, UTC]
 2   text                1981 non-null   object
 3   dog_name            1339 non-null   object
 4   retweet_count       1981 non-null   int64
 5   favorite_count      1981 non-null   int64
 6   jpg_url             1981 non-null   object
 7   img_num             1981 non-null   int64
 8   first_pred_group    1981 non-null   object
 9   first_pred_conf     1981 non-null   float64
 10  is_first_pred_dog   1981 non-null   int64
 11  second_pred_group   1981 non-null   object
 12  second_pred_conf    1981 non-null   float64
 13  is_second_pred_dog  1981 non-null   int64
 14  third_pred_group    1981 non-null   object
 15  third_pred_conf     1981 non-null   float64
 16  is_third_pred_dog   1981 non-null   int64
 17  dog_stage           304 non-null    object
 18  final_rating        1981 non-null   float64
dtypes: datetime64[ns, UTC](1), float64(4), int64(7), object(7)
memory usage: 309.5+ KB
```

We now have a table with 18 columns and 1981 rows. This table contains raw and clean data that can be used for analysis in the next phase