Project: Wrangle and Analyze Data

Data wrangling

First thing to do in this project is "Data Wrangling", which consists of "Data Gathering", "Data Assessing", and "Data Cleaning". Oftentimes, the raw data is rarely clean. Data wrangling is a process to prepare datasets for easy access and analysis.

Data Gathering

Data needed for this project is gathered from three sources, including:

1. The WeRateDogs™ Twitter archive that is stored in a file called 'twitter-archive-enhanced.csv'.
2. The tweet image predictions by a neural network. This file is hosted on Udacity's servers and downloaded programmatically using the Requests library.

3. Using the tweet IDs in the WeRateDogs™ Twitter archive in 1. to **query the Twitter API for each tweet's JSON data** using Python's Tweepy library and store each tweet's entire set of JSON data in a file called tweet_json.txt file.

Three dataframes df1, df2, and df3 are generated for data gathered from the three aforementioned sources, respectively.

Libraries used in data gathering

- Pandas
- Requests
- IO
- TweetPy
- Json

Import csv file

Dataframe df1 was created to store data from "**twitter-archive-enhanced.csv**". The data was imported using pandas function called "**read_csv()**".

Download file from Udacity's server

Dataframe df2 was created to store data in "**image-predictions.tsv**" downloaded programmatically from url of the host's server using requests' function called "**get(url)**". The data was decoded and then imported and stored in df2 using pandas.read_csv() in the same way as df1 but the "**io.SringIO()**" was used to produce string object from the encoded data. Note that the data file is tab-separated.

Query Twitter API

To query Twitter data, TweetPy was used. First, API object for access to Twitter API was created.

Twitter's API has a rate limit. Rate limiting is used to control the rate of traffic sent or received by a server. Instead of getting tweet status one by one using **api.get_status()**, optionally, **api.statuses_lookup()** to get a chunk of 100 tweets at time may come in handy. This may save a lot of running time!

After querying each chunk of tweet ids, returned tweet's JSON data was extracted by **json.dump()** and written to "**tweet_json.txt**" file with each JSON data on its own line.

Data Assessing

Before cleaning, it is essential to assess data to inspect what to clean. In this process, two issues are concerned:

- data **quality** issue
- data **tidiness** issue

First, visual assessment is performed by looking through each table df1, df2, and df3 by opening the data in spreadsheet. Then, use **df.info()** to programatically assess datatypes of each variables. Check if there is duplicated tweet by **df.duplicated()**. Use **series.value_counts()** for some variables to see their values and counts.

Data Cleaning

Data cleaning was performed to solve the issues found in data assessing process. Each issue was solved by three steps; **Define** how to solve the issue, **Code** to program the solving, and **Test** if the issue was solved. Note that prior to cleaning, all dataframes were copy from the original ones.

Some functions used in cleaning process are:

- drop(), dropna()
- str.extract() with regular expression
- str.split()
- str.contains()
- str.lower()
- astype()
- rename()
- merge()

Data Storing

Clean data was finally saved to 'twitter_archive_master.csv' file using **pandas.to_csv()** function.