Wrangle Data Report

Ryan Johnson

February 5, 2019

Here is a short summary of how the data wrangling process was performed in the "WeRateDogs" twitter data project as part of the Data Analysis Nano Degree(DAND) from Udacity.

Students in the DAND program learned to wrangle data in the following way;

1. Gather the data
2. Assess the data
3. Clean the data
4. Visualize and Analyze the data

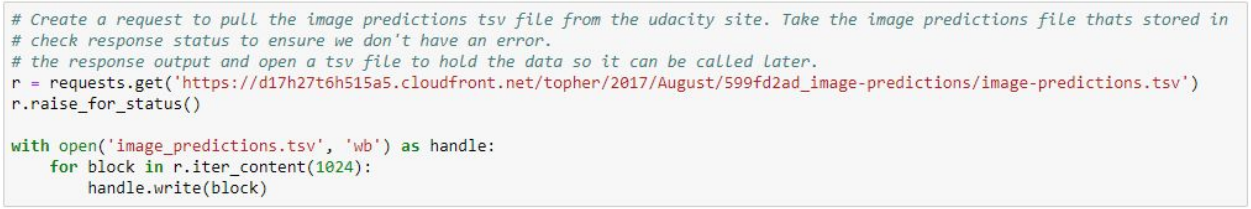
I will walk through how I approached each of these steps.

Gathering the Data

The gathering portion of this project involved the collection of three different datasets from three different sources. Each source required a different method of acquiring and importing the data.

The first data set was a csv file that was available for download on udacity's site called twitter_archive_enhanced.csv. This by far was the largest data set with 2356 entries.

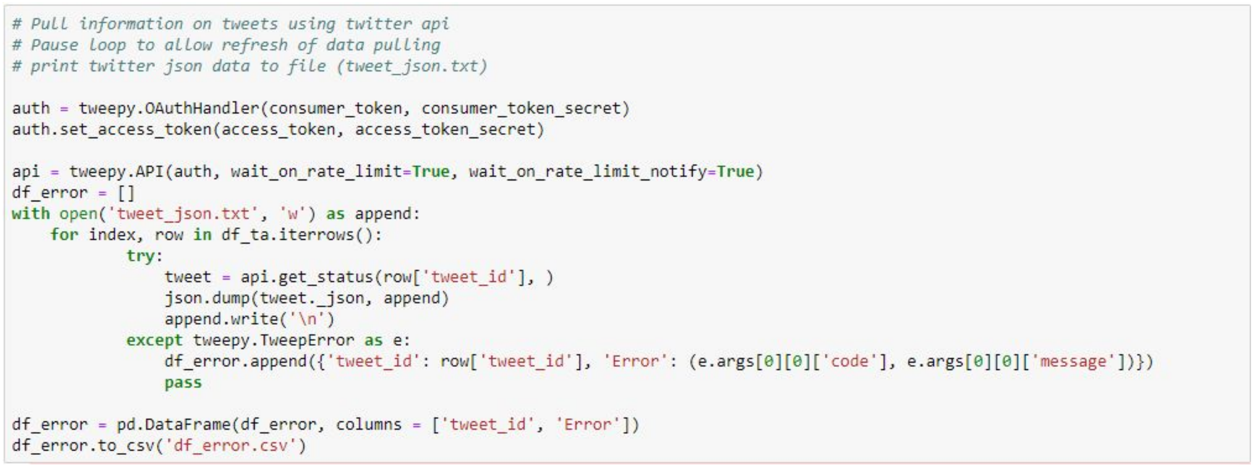
The second data set was a list of predictions of the dogs breed from a cloud site. It was pulled using pythons Requests library. It was a tsv file that required us to read it into a dataframe correctly to use it. A snippet of the code for pulling the file is shown below.



```
# Create a request to pull the image predictions tsv file from the udacity site. Take the image predictions file thats stored in
# check response status to ensure we don't have an error.
# the response output and open a tsv file to hold the data so it can be called later.
r = requests.get('https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv')
r.raise_for_status()

with open('image_predictions.tsv', 'wb') as handle:
    for block in r.iter_content(1024):
        handle.write(block)
```

The third data set was a request to query each of the tweets we already had using Twitter's API and Tweepy. The Tweepy library returned a json data set that I could then parse through and append the data to a pandas data frame. This was then saved as a csv file to keep from having to make requests against Twitters framework which was very time consuming as they only allowed ~1k requests every 10-15min. With >2300 requests it did take some time to collect all the additional information needed. A snippet of the code for query Twitter is included below.



```python
# Pull information on tweets using twitter api
# Pause loop to allow refresh of data pulling
# print twitter json data to file (tweet_json.txt)

auth = tweepy.OAuthHandler(consumer_token, consumer_token_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
df_error = []
with open('tweet_json.txt', 'w') as append:
    for index, row in df_ta.iterrows():
            try:
                tweet = api.get_status(row['tweet_id'], )
                json.dump(tweet._json, append)
                append.write('\n')
            except tweepy.TweepError as e:
                df_error.append({'tweet_id': row['tweet_id'], 'Error': (e.args[0][0]['code'], e.args[0][0]['message'])})
                pass

df_error = pd.DataFrame(df_error, columns = ['tweet_id', 'Error'])
df_error.to_csv('df_error.csv')
```

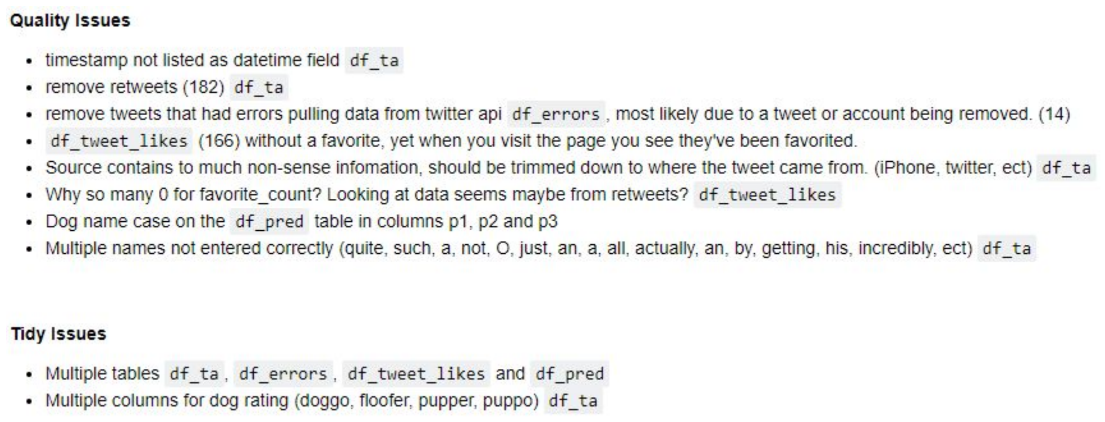
Assessing the Data

The second part of my data wrangling adventure directed me to visually assess the data from the 3 data frames created. Using Jupyter Notebooks I programmatically assessed the data using pandas built in functions such asl df.info(), df.head(), df.describe(), df.value_counts(), ect…

The datasets were assessed using the criteria of quality and tidiness. Quality issues consisted of typos, data inconsistencies, missing information, missing records, etc. Tidiness issues consisted of multiple tables, two data types being in one column, multiple columns for one classification, etc. As data issues were found that fell into one of these categories, they corrected as best as possible.

Through the data wrangling process we might find issues later in the process that would encourage you to come back to the assessing step, while I did find other issues later on I left them to be corrected at another time.

A list of the issues I did find in the Assessing the Data step are listed below.



Quality Issues

- timestamp not listed as datetime field `df_ta`
- remove retweets (182) `df_ta`
- remove tweets that had errors pulling data from twitter api `df_errors` , most likely due to a tweet or account being removed. (14)
- `df_tweet_likes` (166) without a favorite, yet when you visit the page you see they've been favorited.
- Source contains to much non-sense infomation, should be trimmed down to where the tweet came from. (iPhone, twitter, ect) `df_ta`
- Why so many 0 for favorite_count? Looking at data seems maybe from retweets? `df_tweet_likes`
- Dog name case on the `df_pred` table in columns p1, p2 and p3
- Multiple names not entered correctly (quite, such, a, not, O, just, an, a, all, actually, an, by, getting, his, incredibly, ect) `df_ta`

Tidy Issues

- Multiple tables `df_ta` , `df_errors` , `df_tweet_likes` and `df_pred`
- Multiple columns for dog rating (doggo, floofer, pupper, puppo) `df_ta`

Cleaning the Data

In the cleaning step all the quality and tidiness issues are corrected. Cleaning the data followed a format of define the issue, code to correct and test to ensure cleaning the data didn’t cause other issues. The cleaning steps were performed using python tools such as merge, melt and some functions I created to assist in the process.

At the end of the Gathering, Assessing and Cleaning phases I went from a dataset of ~2300 to a dataset of ~1800. All cleaning steps can be find in my wrangle_act.ipynb file.

Conclusion

Data wrangling allows for the collection, assessment and cleaning of data to give yourself or team members working with you, a set of data that is in a format so you can analyze and visualize it, draw conclusions and share with others for business or enjoyment purposes.