Udacity Data Analyst Nanodegree

Project 7 - Wrangle and Analyse Data

Wrangle Report on “WeRateDogs” Twitter Dataset

by **Dilip Rajkumar**

After so many requests... here you go.

Good dogg. 420/10

9:52 PM - 28 Nov 2015

9,330 Retweets 25,607 Likes

94 9.3K 26K

WeRateDogs™
@dog_rates

Your Only Source for Professional Dog Ratings STORE: @ShopWeRateDogs | IG, FB & SC: WeRateDogs | MOBILE APP: @GoodDogsGame Business: dogratingtwitter@gmail.com

Introduction

In this report we will describe the wrangling efforts on the **WeRateDogs™** twitter dataset. WeRateDogs is a famous twitter handle which rates dog in an unusual and humorous manner(They even rated Snoop Dogg, 420/10).

Data wrangling consists of:

- Gathering data
- Assessing data
- Cleaning data

Gathering

The first step in the Data Wrangling process is acquiring or *gathering* the data. *Without Gathering we would have no data*. The data was acquired from the following sources as per the Udacity Project Rubric.

- The WeRateDogs Twitter archive (twitter_archive_enhanced.csv). This csv file file was provided to us and we read it manually.
- The tweet image predictions (image_predictions.tsv) file, classifies the breed of the dog present in each image using a neural network. This file hosted on Udacity's servers, was downloaded programmatically using the **Requests** library.
- Using the tweet IDs in the WeRateDogs Twitter archive, we queried the Twitter API for each tweet's JSON data using Python's **Tweepy** library and stored each tweet's entire JSON data of each tweet in a file called tweet_json.txt file. Each tweet's JSON data was written to its own line and read line by line, into a pandas DataFrame with tweet ID, retweet count, and favorite (i.e. "like") count.
- All the above data was imported into the programming environment (Jupyter Notebook)

Assessing

After gathering each of the above pieces of data, the next step was to assess them visually and programmatically for quality and tidiness issues. The following quality issues and tidiness issues were detected and documented.

Quality

Completeness, Validity, Accuracy, Consistency => a.k.a *content issues*

twitter_archive dataset

- tweet_id , in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id should be *string* instead of *float* or *integer* as we are not performing any arithmetic operations on them.
- timestamp should be datetime instead of object (string)
- Tweet with more than one rating in the text description, sometimes have the first occurence erroneously used for the rating.
- numerator_rating erroneously ignores decimal and does not display full float value.
- Many Tweets don't have any image URLs
- In several columns null objects are non-null (None has to be set to NaN)
- Name column has invalid names i.e '*a*', '*an*','*the*',etc.. which typically begin in lowercase
- Some of the invalid names for certain tweet IDs have valid names in the text field, which can be manually replaced.
  - 770414278348247044 : Name *Al* instead of *Al Cabone*,
  - 776201521193218049 : Name *O* instead of *O'Malley*
- Tweet Sources are difficult to read.
- Extra characters after '&' in text column for certain tweet IDs.

- Dataset contains retweets. We only want original ratings (no retweets) that have images.

image_predictions dataset

- Missing values from images dataset (2075 rows instead of 2356)
- Some tweet_ids have the same jpg_url
- Some tweets are have 2 different tweet_ids one redirect to the other

tweet_data json dataset

- Some tweet_IDs could not be queried from the tweepy API query.

Tidiness

Untidy data => a.k.a *structural issues*

- A gender column extracted from the text columns in **twitter_archive** was deemed necessary.
- Dog stage variable (*doggo, floofer, pupper, puppo*) is in four columns needs to be melted into a single column as per tidy data format.
- Merge **tweet_data** and **image_predictions** dataframes into single **twitter_archive** dataframe
- Prediction results in the **image_predictions** dataframe from the various prediction algorithms are spread across multiple columns which can be optimized to a few columns.

Cleaning

Cleaning our data is the third step in data wrangling. It is where we fixed the quality and tidiness issues that we identified in the assess step. We used the two types of cleaning, the *manual* and *programmatic* .The manual was not recommended except for issues that were one-off occurrences. The Programmatic Cleaning followed the pattern of:

- Define: where we define the problem
- Code: where we write the code to fix the problem
- Test: where we test our code to see if the problem was fixed.

A copy of the dataset was always made to test the changes before applying to the main dataset. Since all Quality and Tidiness issues were not spotted in the initial assessment, many reassessments were done as we discovered new issues while cleaning the data.

Challenges and Learnings:

This project was the most challenging and exciting I did as an aspiring data analyst. The API query calls made using **Tweepy** took so long to run. The presence of ~ 30 columns in all the 3 datasets was very confusing initially.Fortunately the Tweet data dictionaries helped in understanding the significance of every variable in the dataset. It was also challenging and fun to learn about **regular expressions** which was used in string extraction in our wrangling process. Many tweets were not about dogs which was difficult to identify an filter out.

Conclusion

In today's world of Data analytics, data wrangling has become an indispensable skill since so much of the world's data isn't clean. If we analyzed, visualized, or modelled our data before wrangling, we would be facing unpleasant consequences like making faulty analytics, wasting time and missing out on cool insights and visualizations.

Future work:

There were many variables from the json data gathered from the Tweepy API query which were discarded in our wrangling process. Investigating some of these variables in detail could help us in making better analytics of this WeRateDogs dataset.

References

Many API documentations and stackoverflow answers came extremely handy, few of them are:

1. https://docs.python.org/3/library/re.html
2. https://bokeh.pydata.org/en/latest/docs/user_guide.html#userguide
3. https://pandas.pydata.org/pandas-docs/stable/generated/pandas.core.groupby.DataFrameGroupBy.agg.html
4. https://stackoverflow.com/questions/29919306/find-the-column-namewhich-has-the-maximum-value-for-each-row
5. The tweepy page - http://docs.tweepy.org/en/v3.2.0/api.html#API and of course
6. WeRateDogs^{TM} Official Twitter Page - https://twitter.com/dog_rates/
7. https://regexone.com/