Wrangle Report

By Ryan Reardon

Introduction

In this report I'll describe my wrangling effort in the data wrangling section of the Wrangle and Analyze Data Project: WeRateDogs Twitter Account.

Data wrangling consists of:

- Gathering Data
- Assessing Data
- Cleaning Data

Gathering Data

Gathering data for this project is composed of three data sets:

- The WeRateDogs Twitter archive. This file was given by Udacity to download manually: twitter_archive_enhanced.csv.
- The tweet image predictions, i.e., what breed of dog (or another object, animal, etc.) is present in each tweet according to a neural network. This file (image_predictions.tsv) is hosted on Udacity's servers and downloaded programmatically using the Requests library and the following URL: https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv.
- Each tweet's retweet count and favorite ("like") count at minimum, and any additional data you find interesting. Using the tweet IDs in the WeRateDogs Twitter archive, query the Twitter API for each tweet's JSON data using Python's Tweepy library and stored each tweet's entire set of JSON data in a file called tweet_json.txt file. Each tweet's JSON data written to its own line.

The Three data sets were combined using the following methods:

Manually: First data set, I downloaded the twitter_archive_enhanced.csv file manually given by Udacity and saved in the same folder as wrangle_act. IPYNB file to open in a Jupyter Notebook. Using pandas, I read in the CSV file to the Jupyter Notebook.

Programmatically: Second data set, using Requests library and the following URL: https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv , I downloaded the image_predictions.tsv file from Udacity servers. Using pandas, I read in TSV file to the Jupyter Notebook

Query Twitter API: Third data set, I queried a twitter API to get a JSON object using tweet_id from twitter_archive_enhanced.csv using Python's Tweepy library and stored in a list and saved it to text file called tweet_json.txt and loaded the file with columns of interest into Jupyter Notebook.

Assessing Data

I assessed data using the following methods:

- **Visual Assessment**: printing the dataframe in Jupyter Notebook looking for inconsistencies and by opening each file individually and reviewing the file.
- **Programmatic Assessment:** by using methods in pandas; for example, info, value_counts, duplicated etc.

Then I detected and documented quality and tidiness issues prior to merging the data into one file by tweet_id with an image and a rating.

Quality: empty values, incorrect datatypes, incorrect ratings due a similar statement like 24/7 when there's a real rating in the text etc., duplicate images, and 3 separate dog breed predictions and confidence ratings that should be in one column.

Tidiness: extra columns not needed for analysis, proper column headers, and doggo, pupper, puppo, and floofer columns combined into one column.

Cleaning Data

Cleaning data is divided into three parts:

- **Define:** This where we define cleaning plan in writing and convert the quality and tidiness assessments into cleaning tasks.
- **Code:** Translate what we defined into code.
- **Test:** Test your data set to make sure your cleaning code worked

I created a copy of the original data frames and changed the tweet_id to string to merge the three data sets. I decided to merge the data sets first, because I thought it would be easier to wrangle if they were joined first.

Removed the retweets, duplicates, NA's in the jpg_url column and removed columns that wouldn't be used for analysis.

Melted the dog stages into one column, which was super challenging. Especially when a few of the observations has more than one stage and need to be concatenated together. After, I changed the "None" value to no_stage and dropped the individual dog stages.

In the gather process, I noticed large numerators based on other values in the tweet, incorrect ratings that were fractional ratings in the tweet too. First, I updated the numerator and denominator rating columns as a float data type. Then, I updated fractional ratings in tweet to the correct numerator rating. Next, I deleted the tweets without real ratings and finally to correct those extremely large ratings that were the result of multiple dogs in one image; I created single rating column, which multiplied the numerator/denominator by a factor of 10. Last, I deleted the rating numerator and denominator columns, since we had a single rating column.

The data had missing names of the dogs or erroneous names; for example, "a" or "the" etc. I did notice quite a few tweets had "This is", "Meet", "Say hello to", "Here we have", or "named" that had these strange names. So, to code for this issue I stripped the name from the tweet if it had one of these conditions, if it did not, the name was updated to unidentified.

The data set had three dog breed predictions and their percentage of accuracy in three columns. To condense, the dog breed predictions in one column and the percentage accuracy in one column. We searched for the true level confidence added to a list and dog breed prediction to a list and made two columns out of these lists and if the breed truly couldn't be recognized it was marked as "unidentified" with a confidence prediction of "0". After the columns were combined the p1, p2, p3, p1_conv, p2_conf, and p3_conf were deleted.

Another issue was the source data with div brackets from the html coding around URL data. I used re findall to extract the source data; for example, twitter for iPhone etc. and filled the NaN values from one image with ratings.

Last, I deleted the columns that I wasn't going to use for analysis, updated data types, and renamed columns with a more readable column header.

Conclusion

Data wrangling is challenging. I found it daunting, from gathering the best data for analysis, detecting and documenting and assessing our data and recording the quality and tidiness issues that needed to be resolved. Cleaning data wasn't the easiest, but python did make it faster with the many packages available to assist in the process. I needed to iterate through the process a few times, where I consistently went back to assessing the data for any quality tidiness issues I missed, that can be cleaned.

Data wrangling is a core skill for data analysis as most data sets in the world aren't tidy.   Performing analysis and visualization prior to wrangling data will produce inaccurate results, miss out on insights, and our recommendation could have financial or reputational impact.

In closing, data wrangling remains the fundamental building block than enables visualizations and statistical modeling.   Only through data wrangling can data be made useful. Performing data wrangling tasks effectively and efficiently is fundamental to becoming an data analyst in any industry.