wrangle_report

January 17, 2021

1 Wrangling Report

In this document I will walk through the wrangling process; the steps I have taken in gathering, assessing and cleaning data. These steps has been all performed in the wrangle_act.ipynb present in this directory. The file, referred to, is self contained and has detailed mark down cells that explain each step and gives an outline of the process. So while this report is not necessary for comprehension if you viewed and walked through the code it serves as a documentaion that gives a clean, code-free overview of the process and all performed steps.

1.1 Gathering:

I have gathered data from three different sources:

1. A local csv file `'twitter-archive-enhanced.csv'` containing all basic tweets' data, using pandas `read_csv()` function.

   This file was in a pretty comma seperated values format so it needed almost no effort other than getting it in place using pandas' `read_csv()`.

2. A file hosted on this link containing predictions for images in some of the tweets of the first file, using `requests` python library.

   To download this file programetically, I `imported` `requests` library and used the get method to store the file in a `response` object then extracted the file contents from the `content` attribute in that object to a text file.

   The next step was to read the file using `read_csv()` with tab dilemiter `sep='\t'`.

3. Twitter API for additional information about the tweets.

   For this step, I could not acquire a key from twitter as I could not get a developer's account. Yet, I have impleneted the piece of code that interacts with twitter api, send a request and recives a resopnse, and in the case of a successful response the data is stored in `recent_tweet_json.txt` file which is then used in the rest of the wragling process. And in the case of a failed attempt to dowload the requested data, the error messages are stored in `twitter_api_log_file.txt` and the wrangling process continues on with `tweet_json.txt` instead, which was previously downloaded from course instruction page.

   The check for whether the download was successful or not is implemented by comparing the number of lines in each file: `recent_tweet_json.txt` and `tweet_json.txt` and choosing the recently downloaded one `recent_tweet_json.txt` if it has as much or more lines as the `tweet_json.txt`.

1.2 Assessing:

After gathering data from the three sources some problems where immidietly obvious, e.g. tidiness issues of the data about tweets being spread between three different souces and quality issues of unsuitable datatypes. Some other prblems, on the other hand, needed further visual inspection; using pandas `DataFrame.head()`, `DataFrame.sample` and some plotting methods such as `DataFrame.plot()`, and programatic assessment mainly using `Dataframe.info()`, `Series.value_counts()`, `Series.unique()` and other helpful functions.

This step yielded an aggregate of the following problems:

1.2.1 Quality issues:

1. `timestamp` and `retweeted_status_timestamp` in `df_tweets` is not formatted as `datetime`.
2. all `id's`: `['tweet_id', 'retweeted_status_id', 'retweeted_status_user_id', 'in_reply_to_status_id', 'in_reply_to_user_id']` in `df_tweets` are should have the same `dtype`.
3. Zero `rating_denominator` not allowed. In fact it was a correction. **(Correct the rating values)**
4. Zero `rating_numerator` was not of a dog. **(Drop entry)**
5. *50/50 split* mistaken for a rating. **(Change)**
6. `Na` values in `df_tweets.name` has value `'None'` instead. **(Change to `Na`)**
7. `'a'` is not a name. **(Change to `Na`)**
8. `geo` column in `df_tweets_augment` has no values. **(Drop column)**
9. make the variable created from aggregating `['doggo', 'floofer','pupper', 'puppo']` a `'category'`.

1.2.2 Tidiness issues:

1. The `df_tweets_augment` and `df_tweets` both contain data about the same observational unit. **(Merge the two dataframes)**
2. The `df_images` has predictions about images in tweets in `df_tweets`. So they are the same observational unit ()tweets. **(Merge the relevant pieces in `df_images` to `df_tweets`)**
3. In `df_tweets` the columns:`['doggo', 'floofer','pupper', 'puppo']` form a list of values rather than variable names. **(`melt` within the frame)**

These were all the issues I have began with, to later within the cleaning process dicover another issue with data quality.

1.2.3 Quality issue discovered while cleaning:

10. Some tweets have, mistakenly, more than one dog stage at a time.

Which I mended in the next step, the cleaning step.

1.3 Cleaning:

After identifying all the issues, I have then turned to cleaning the data. I have cleaned them one by one on each on three steps:

1. ***Defining*** an approach to clean the issue.

2. ***Coding*** my approach to obtain clean data.
3. ***Testing*** if the cleaning code worked as expected and if the data is now as I intended it to be.

I have first resolved quality issues in the main tweet archive, using methods from pandas such as: `Series.astype()` to change data types, `Series.replace()` to replace unwanted values with `null`, and `DataFrame.append()` to add some data points after some corrections. As well as boolen indexing to find faulty entries and assigning the right values instead.

Then resolved tidiness issues and appended the relevant pieces of each dataframe together in one clean dataframe, using `merge` and `melt`.

2 Final Thoughts:

The data wrangling process, as I experienced it in this project, turned out to be very non-linear and entangled process. It did not go clearly from one step into the next, neither was a step all done at once.

Before I finished gathering all the data, I had already known some of the issues with it just by performing some relatively straitforward checks to see if it's imported correctly. And, nearly at the end of cleaning the data I have discovered a new issue which forced me to get back to assessing again.

So it was more of a cycle that I iterated through, and the more time I spent and the closer I looked the more I finer details I found and the cleaner the data was and easier to work with.