

Wrangle and Analyze Report

1 Data Wrangling Report

prepared by Debarghya Biswas

1.1 Gathering

In the gathering step, we are asked to gather 3 different tables using 3 different ways: 1. `twitter-archive-enhanced.csv`: this file is obtained by downloading it manually from the udacity site. 2. `image-predictions.tsv`: this file is obtained by downloading it programmatically using the python's requests library 3. `tweet_json.txt`: this file is obtained by calling the twitter API for each tweet, using the tweet id as a parameter, which we obtained from the `twitter-archive-enhanced.csv` data. Instead of using the Requests library, we use the Tweepy library which provides a higher-level way to get the data from the API.

After that, we make sure that all the data are available in the same directory as the `wrangle-act.ipynb`

1.2 Assessing

In the assessing step, we try to gather some quality and tidiness issues from the data we have collected in the gathering step. At the start, I try to assess each file separately before moving to other files. In the end, I try to find an issue related to redundancy by assessing all the data in the 3 files together.

Note: 1. In the below, the `tweet_api_data` refer to the data obtained from the `tweet_json.txt`

1.2.1 Quality

`tweet_api_data` findings:

1. Have unrelated columns like the user, favorited, retweeted which is based on the personal user information who obtains the data from API. This is found by looking through the columns inside the `tweet_json`.
2. Have a column in which all data is null: contributors, coordinates, geo, place, quoted status id, and quoted status id str This is found by calling the `info()` method, which indicates the number of non-null data in each column.
3. Source column contain the whole HTML tag ``. This is obtained by looking through the data in each column.

4. possibly_sensitive and possibly_sensitive_appealable column are all zero. This is initially obtained by looking through several rows in the data, after which I started getting a little bit suspicious since it seems like all the data in those 2 columns are always 0. This suspicion is then confirmed by using the value_counts() function.

image_predictions table

1. p1, p2, p3 whitespace usage is not standardized, some uses -, some uses _ some are capitalized, some are not. This is obtained by looking through the data.

twitter-archive-enhanced.csv findings:

1. the dog name may not be accurate, some names are a or an.
2. the dog name has None string that should be a null instead.
3. Column name doggo, floofer, puppo, and pupper has value either None or its column name.
4. Some rating is wrong, it is marked by the denominator is not 10. One of the texts of the tweets is below: > "This is Darrel. He just robbed a 7/11 and is in a high-speed police chase. Was just spotted by the helicopter 10/10".

However, the captured rating was 7/11 instead of 10/10 5. Some rating_numerator also captures wrong data, spotted by having the value less than 10. 6. the timestamp should be put as DateTime instead of a string

1.2.2 Tidiness

1. the p1_dog, p2_dog, p3_dog contain redundancy since the p1, p2, p3 is not unique throughout the row in image_predictions
2. created_at / timestamp, source, text, in_reply_to_status_id, in_reply_to_user_id are duplicated in tweet_api_data table and twitter_archive table
3. tweet_api_data should be part of twitter_archive table
4. Entities data seems to contain image information which is already contained in the twitter archive data, like the image_url and extended URL
5. Extended entities column contains duplicate information of the entities column
6. Some tweets in the tweet_api_data are retweets. We can obtain the real tweet data for such a case using the retweeted_status column.

1.3 Cleaning

In this step we are trying to clean the data, however, due to time constraints and the project's recommendation, we will only clean a few of the issues we mentioned above, instead of all of them.

1. The first thing we fix is the fact that some tweets data in the tweet_api_data are retweets. It is actually a combination of tidiness and quality issues. It is included in the tidiness since because of this, one row could contain multiple data (the data of the retweeted status and the data of the real status). It can also be included as a quality issue since if the data users use this data carelessly, they could have wrong analysis by calculating using the retweeted status's data, instead of the real status, which may have different statistics. One example would be using the retweeted status to determine the number of favorite counts.

To clean it, we first capture all the `retweeted_status` data, store it in a different data frame, call it `temp`. After that, we drop rows that contain the `retweeted_status` in the `tweet_api_data`. It is done by removing all rows whose `retweeted_status` column is not null. After that, since currently, all rows must have a null value for the `retweeted_status` column, we can safely drop the column. Lastly, we append the rows in the `temp` variable into the `tweet_api_data` table.

2. The `p1_dog`, `p2_dog`, `p3_dog` contains redundancy since the `p1`, `p2`, `p3` is not unique throughout the row in `image_predictions`. This is a quality issue. What I mean in the above is that Husky could appear in any row in `p1`, `p2`, `p3` and the value for `p1_dog`, `p2_dog`, and `p3_dog` would be the same if the classifier name, the `p1/p2/p3` is the same.

To clean it, we first create an empty data frame to store the mapping between `p` and `p_dog`. After that, we store all `px` and `px_dog` combinations in that data frame and remove the duplicate row using all columns as its comparison. Furthermore, we check for duplicate in the `p` variable, which returns us an empty list. It shows that the same `p` will give us the same `p_dog`, as we assume previously. After that, we remove the `p1_dog`, `p2_dog`, and `p3_dog` column from the `image_prediction` table.

3. There are some columns in `tweet_api_data` and `twitter_archives` which are redundant as they are different columns who have a similar meaning. To solve this, we can drop the `timestamp`, `source`, `text`, `in_reply_to_status_id`, `in_reply_to_user_id` in the `twitter_archives`
4. `tweet_api_data` should be part of `twitter_archive`. This is a tidiness issue. We should merge the `tweet_api_data` table to the `twitter_archive` table, joining on `tweet_id` and `id`.
5. There is some more redundancy in the form of the column whose values are all empty or unrelated to the data, such as `user`, `favorited`, `retweeted`, which are based on the user profile who get the API. We should drop those columns, such as `user`, `favorited`, `retweeted`, `contributors`, `coordinates`, `geo`, `place`, `quoted status id`, and `quoted status id str`.
6. Column name `doggo`, `floofer`, `puppo`, and `pupper` has value either `None` or its column name in the `twitter_archives_clean` table. We should change those value to be boolean, taking value `True` if the column is the column name, `False` otherwise.
7. `possibly_sensitive` and `possibly_sensitive_appealable` contain the same value for all row in `twitter_archives`. We should drop the `possibly_sensitive` and `possibly_sensitive_appealable` column
8. `Source` column contain the whole HTML tag ``. To clean it, we can use a regular expression to extract only the `href` part of the link
9. The name of the dog are marked as `None` instead of `nan` in `twitter_archives_clean` table. We should replace `None` to `nan`
10. Some dog has named `a`, `an`, and the in `twitter_archives_clean` table. This must be a mistake. In order to increase the quality of the data, we should replace `'a'`, `'an'`, `'the'` to `nan`
11. Some rows have invalid ratings, with rating numerator less than 10 or denominator not equal to 10. To solve this, we can either develop a better function to extract the rating from the status or we can drop rows with such occurrences.