Wrangle Report

January 14, 2022

Mikhail Gorbunov

Table of Contents

Data Gathering **3**

Cleaning Data **4**

Storing Data **5**

1. Data Gathering

We used three archives of data in the Jupyter Notebook “wrangle_act.ipynb” (bellow simply notebook):

Udacity provided the first archive, “WeRateDogs,” Twitter archive via direct link: https://d17h27t6h515a5.cloudfront.net/topher/2017/August/59a4e958_twitter-archive-enhanced/twitter-archive-enhanced.csv and it has 2356 records. For usability purposes, the notebook checks the presence of the file in the workspace and, if the file doesn’t exist, automatically downloads it.

The second archive, The Tweet Image predictions, also was provided by Udacity via direct link: https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv and it has 2075 records. According to the requirements, the notebook also automatically downloads it directly to the workspace using the Requests library.

To obtain the third archive: the tweet_json, the notebook queried the Twitter API for each Id in the first archive using the requests library. The author used the Tweepy library as suggested by Udacity but had some difficulties obtaining user-level permissions required by Twitter API 1.1. After some research, the author found another way to query all needed data alongside with some additional benefits: to use Twitter API 2.0 with the requests library:

Big Thanks to https://www.kaggle.com/andrewedward37/data-collection-with-twitter-api-v2 for the guide on how to create basic requests. For simplicity of the process, the functions make_request and pull_all_the_data, alongside auxiliary functions, were created. The business logic of the above functions includes the following:

- Splitting all querying ids into batches (in the notebook – 50, but it's adjustable)
- Queering the API: one batch per request.
- Tracking the current quantity of requests within a given time interval. If needed, the function pauses the process and then continues it automatically.
- The function saves the data after every successful response and restarts with the last saved position if it's needed
- Rising the exception if something wrong with the request
- Workaround the known Pandas bug ("https://github.com/pandas-dev/pandas/issues/20608#") with high int values

All above allows gathering all needed data within 1-2 minutes (depending on the settings) and is easy to fit Twitter's limit: 300 requests within 15 minutes. Compare it with the suggested by Udacity way with 15 -20 minutes running time and the risk to be banned.

This archive got 2328 records.

Comparing the number of records between the first and last archives, some difference was noted: the number of records in the first archive is higher than in the second archive. This means that not all records in the first archive got the matching records in the last archive. Then

all ids of such records were found, and then some of them were queried directly to Twitter (to exclude the bugs in the functions), but the result was null. The author concluded that tweets with these ids already don't exist.

2. Assessing Data

First, all three datasets were merged to the master_df data frame for a better understanding of the dependencies between them. During assessing were found bellow quality and tidiness issues:

2. 1 Quality issues:

1. All columns related to the df_ipr data frame have fewer non-null values than others. It is caused by less quantity of rows. (2075 compared to 2356)
2. The expanded_urls column has fewer non-null values than it should be (2297 instead of 2356)
3. Some tweets IDs in df_ta were wrong or deleted in Twitter. (see "#find all missed id's"). It caused null values in all columns related to tweet_json_df data frame (text_y, lang, etc.)
4. Some records (with non-null values in columns like 'in_reply_to_status_id') are not original tweets as required
5. Some of the columns have only a few non-null rows. (geo)
6. Some of the rows have all three non-dog values in p1, p2, p3 dog columns
7. The timestamp column has a string type.
8. The column rating_numerator has a minimum of 0, whereas the valid minimum is 10 (" Almost always greater than 10"). And the maximum is over 1700 that is not good too.
9. The column rating_denominator: it should be 10, but the min is 0, max is 170.
10. Columns doggo, floofer, pupper, puppo have string None values. They should be replaced on NaN.

2.2 Tidiness issues:

1. Dataset has a lot of redundant columns such as in_reply_to_status_id and in_reply_to_user_id_x, id and tweet_id, and conversation_id, etc.
2. doggo, floofer, upper, and puppo it's one categorical variable. It should be one column instead of 4.
3. values in public_metrics are dictionaries. It should be converted to several columns like reply_count, like_count, etc
4. Columns rating_denominator and rating_numerator should be combined into one variable

3. Cleaning Data

The copy of original data created: copy_of_master_df. Then were performed bellow steps to clean the data:

- Dropped all rows with null values in column jpg_url. (all rows without images)
- Dropped all rows with null values in the column expanded_urls (issue 1)
- Dropped all rows with null values in column lang for deleting all rows with id's where Twitter responded null in data (tweets already don't exist) (issue 2)
- Dropped all rows with non-null values in columns in_reply_to_status_id and retweeted_status_id (we need only original tweets) (issue 3)
- Dropped redundant and unuseful columns (including timestamp because we have the same data in the column createad_at in the right format) (issue 5, 7, tidiness issue #1)
- Dropped all rows with three False values in one row in columns p1_dog, p2_dog, p3_dog (records where prediction is not a dog) (issue 6)
- Columns rating_numerator and rating_denumerator were cleaned from not valid values and then cast into one variable. (Quality Issue #8 and #9, tidiness issue 4)
- Columns doggo, floofer, pupper, puppo were combined into one variable alongside with replacement string ‘none’ value on NumPy.nan (Quality Issue #10 and tidiness issue 2)
- The column ‘public_metrics’ was converted into 4 variables (likes, retweets, etc) (Tidiness issue #3)

4. Storing Data

After the above steps, the clean dataset was stored in the file twitter_archive_master.csv