Wrangle Report

Introduction

This is a report about my data wrangling project in the WeRateDogs(@dog_rates) Twitter data. The document will be divided in three sections, in wich I will describe my efforts in each process step: **Gathering**, **Assessment** and **Cleaning**.

Step 1: Gathering

This was the first step. The objective was to gather data from different sources and formats and store it in a readable way so it could be used later. Three different sources and formats were used:

- **Given File**
  This was the WeRateDogs twitter archive file. A CSV file containing more than 2300 tweets and some data about them, like the text, date, if the tweet was a retweet or not, and also some information previously extracted from the text like the rating numerator and denominator and the dog's stage. The file was manually downloaded from the given link.

- **Programmatically Downloaded File**
  This was the image predictions file, hosted in the Udacity's servers. A TSV file containing previously generated predictions about the dog's breed of each tweet in the archive file. With the given URL, the file was downloaded using the Python's *requests* library.

- **File Constructed Via API**
  This was the extra info file, built with Twitter API obtained data. A TXT file containing, in each line, a JSON object with the data provided by the API. Using the twitter archive's tweet id and the Tweepy library, some extra information like the retweet and favorite (like) counts could be gathered, stored and used to enrich the dataset and the future analyses that would be made on it.

Step 2: Assessment

This was the second step. The objective was to analyse the data (visually and programmatically) and identifiy quality and tidiness issues that would be cleaned later. The three tables obtained in the gathering step were analysed separately and the identified issues were documented.

The identified issues in ***tw_archive*** table are listed below. All the issues were fixed in the cleaning step.

Tidiness:

- One variable (dog stage) represented in four columns;
- Two variables (date and time) in one column called *timestamp*;

Quality:

- Some tweets were, actually, retweets (project key point);
- NULL values were represented as the word "None" in some columns;
- Some dogs were classified in more than one stage;
- Some dog's names were incorrectly extracted from the text;
- Some ratings were incorrectly extracted from the text;
- Unreadable content in the *source* column;

The identified issues in ***tw_extra_info*** table are listed below. The missing data issue couldn't be fixed because it wasn't possible to obtain the data from the API.

Quality:

- Some tweets had no images attached (project key point);
- Missing data in *place, contributors, coordinates* and *geo* columns;
- Invalid datatype in *id_str, in_reply_to_status_id_str, in_reply_to_user_id_str* and *quoted_status_id_str;*

The identified issues in ***predictions*** table are listed below. All the issues were fixed in the cleaning step.

Tidiness:

- Multiple prediction columns with different results for the same image;

Quality:

- No dogs were recognized in some images;
- Underscores used as spaces in the prediction columns;
- Lowercase and uppercase breeds in the prediction columns;

Step 3: Cleaning

The third and last step. The objective was to fix the issues identified in the assessment step. This was the hard-working part because it required a lot of research to solve the problems in the best possible way.

The first cleaned table was **tw_archive**. This was the most problematic one and the hardest to clean. The tidiness issues were fixed first, and the quality ones later.

Tidiness Issues:

1. The first tidiness issue, **one variable (dog stage) represented in four columns**, was solved by melting the columns into a single one called *stage.* The difficult part was to deal with the duplicated rows after melting the columns because they had to be carefully removed to prevent data loss. Some tweets had no stage in any of the columns, so the result were four identical rows with the "none" value in the *stage* column. Some others had more than one stage assigned, and that resulted in more than one row for the same *tweet_id* with different stages. And finally, the rows with only one stage assigned were also duplicated by the *melt* because they had the value "none" in the other stage columns. After carefully cleaning the "nones", the result was a single column, *stage*, containing the dog's stage.

2. The second tidiness issue, **two variables (date and time) in one column (*timestamp)***, was easier than the first one. The *timestamp* column was splitted in two new columns: *tweet_date* and *tweet_time*, with the corresponding content, in each one of them, and dropped later. At this point the tidiness issues were done.

Quality Issues:

1. The **retweets** were the first quality issue. As not considering retweets was a project's key point, they were cleaned by removing from the dataframe the rows where the *retweeted_status_id* column was not null.

2. The second quality issue, the **null values represented by "none"**, was also easy to fix. The "none" values were just replaced by *np.nan.*

3. Third quality issue: **Dogs classified in more than one stage**. This took some time because some tweets had, actually, more than one stage in the text, but only one related to the dog (like this, for example: "*Here's a puppo participating in the #ScienceMarch. Cleverly disguising her own doggo agenda."*) and some had more than one stage because there were more than one dog in the image, and the dogs were classified in different stages. To solve this, all the cases were manually analysed and fixed. The ones with more than one stage in the text but only one related to the dog were fixed by removing the incorrect stage and keeping the right one. For the ones with more than one stage assigned because there was more than one dog in the image and the dogs were classified in different stages, the stage was discarded, i.e. set as *np.nan*, because only one stage was allowed per observation.

4. The fourth quality issue were the **invalid names**. It was clear that this issue was caused by an incorrect extraction of the name from the tweet text. Probably the used algorithm expected all the tweets to be like "*This is Dido. She's playing the lead role in "Pupper Stops to Catch Snow Before Resuming Shadow Box with Dried Apple."* So the dog's name would be right after "This is". It wasn't the case, and because of this a lot of names were incorrectly extracted. The solution was use the *extract* method, with a regular expression, to re-extract the names from the texts considering some different patterns observed in the texts.

5. Fifth quality issue: **incorrect ratings.** As the invalid names, this was also caused by an incorrect extraction of the rating from the text. Some rating numerators had decimals, some texts had more than one rating, and some didnt even have a rating, but had something that match the pattern of a rating and so it was extracted like one. As with the names, the solution was to use a regular expression to re-extract the ratings from the text, considering the points previously mentioned. To do this, the method *extractall* was used because it returns all the matches of the given pattern and not only the first one as done by the *extract* method. Since some texts had two ratings, the dataframe returned by the *extractall* method was unstacked and the ratings were joined with a "|" (pipe) so they got like this: *"10/10|7/10"*. Then, a new column named *rating_count* was created, and it's content was the *len()* of the s*plit()* method applied in the mentioned column. With this new column it was possible to analyse the tweets with more than one rating to see if the extracted ratings were real ones or not. As done with the stages, if the tweet had more than one different rating because there were more than one dog in the image, the rating was discarded, i.e. set as 0/0. If not, the rating was manually fixed.

6. The sixth, and final quality issue of this table, was the **unreadable content in the *source* column**. For this one, the *sub()* method of the Python's *re* libary, with a regular expression, was applied in the column to remove the undesired tags from the text and make it more readable.

The next table was the **tw_extra_info**. Since no tidiness issues were identified, only the quality ones were fixed.

Quality Issues:

1. The first quality issue were the **tweets with no images attached**. Cleaning them was another project key point, and so, that was done by removing from the dataframe the rows where the *extended_entities* column was NULL. According to the definition of the 'Tweet Objects' found in the API documentation, if the tweet had at least one image there would be an 'extended_entities' object.

2. The second quality issue: **Invalid datatype in *id_str, in_reply_to_status_id_str, in_reply_to_user_id_str* and *quoted_status_id_str* columns**. An easy one. Using the *astype()* method, the columns were converted to string. As they had some NaN values, it was necessary to replace the 'nan' (string) content with *np.nan* after the conversion.

And finally, the last one. The **predictions** table. Only one tidiness issue and three quality issues were identified.

Tidiness Issue:

1. The unique tidiness issue identified was the **multiple prediction columns with different results for the same image** one. To fix this, with a function applied in the prediction columns, the breed with the highest confidence was set in a new column named *breed* if a dog was recognized in the image. If not, the *breed* was *np.nan*.

Quality Issues:

1. The first quality issue was **images not classified as dogs in any prediction**. In some predictions, the neural network identified some object in the image instead of the dog itself. But, in some cases, it happened with all the predictions so it was impossible to know the dog's breed, wich was the most important information of the table. The solution was to remove those cases from the dataset using the *breed* column previously created.

2. Second quality issue: **Underscores used as spaces in dog's breeds.** Only for better visualization of the breeds in future analyses. It was fixed by replacing the underscores with a space.

3. Third and final quality issue: **Lowercase breeds**. Also for better visualization in future analyses. Some breeds were written in lowercase (e.g. *groenendael* and *toy terrier*) and had to be fixed. It was done by applying the *str.title()* method in the *breed* column.

Conclusion

As my first Python/data wrangling project it was very challenging, and also exciting, for me to do this. As I said before, it required a lot of research and, of course, some creativity on my part, to solve the problems that appeared during the development of this project. Every step had it's difficulties, maybe potencialized by my lack of experience with the language, but I think the most important here was that I was able to solve everything and that made me very proud of myself. It was an awesome first experience and I'm sure it will help me a lot in my future projects.