Project: WeRateDogs Data wrangling

Introduction

In the course of this project, we wrangled **WeRateDogs** Twitter data to create interesting and trustworthy analyses and visualizations. To achieve this, we gathered data from Twitter's API, assessed, cleaned and stored it into files. In this document, we will be clearly explaining the logic behind our wrangling efforts. The document includes five sections: Project's files tree, Data gathering, Data assessment, Data cleaning and Data storing.

Project's files tree

Our work includes three folders:

data: Data before, during, and, after wrangling. The data is stored in text and database files. More on these files in the next sections.

src: Jupyter notebook file that contains our source code, wrangle_act.ipynb

report: Project's reports. These are wrangle_report.pdf to explain wrangling efforts and act_report.pdf for Analysis and visualization.

Figure 1: Files tree

Data gathering

Three pieces of data has been gathered:

WeRateDogs archive

The data was accessible as a downloadable file, twitter_archive_enhanced.csv. So, we just downloaded and stored it into **data** folder. Then, we loaded it in **archive** Dataframe.

Tweets' image predictions

The data has been downloaded on the internet as image-predictions.tsv and loaded in **image_predictions** Dataframe.

Retweet count and favorite count

We gathered the data in two steps. First of all, we queried Twitter's API with Python's library Tweepy, retrieved the data in json format and wrote it to tweet_json.txt. We stored line by line each tweet's retweet count and favorite count, in the file (in append mode). Each line is stored as a dictionary-like object. The tweet's id is the key and a string that contains the retweet count and the favorite count separated by space is the value. Also, we write into the file. Figure 2 shows how the file looks like.

As a second step, we read tweet_json.txt line by line and stored each tweet's id, retweet count and favorite count in **tweet_json** Dataframe.

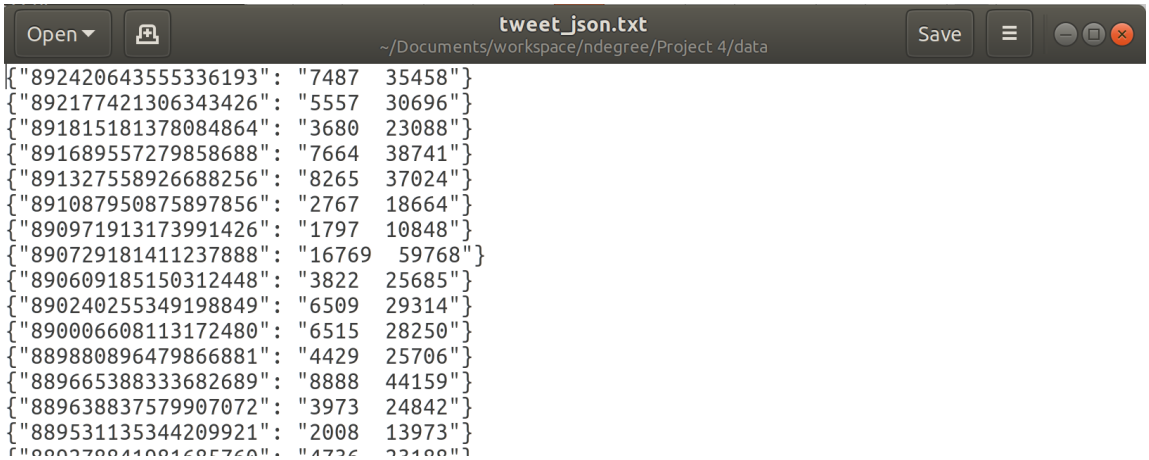

Figure 2: tweet_json.txt content

Data assessment

We detected five (5) tidiness issues and twelve (12) quality issues. Tidiness issues has been detected visually (with Google sheets). However, quality issues have been detected both visually (with Google sheets) and programmatically (Pandas). Issues related to wrong names in **archive** table were first detected visually and then we looked for patterns to find the remaining one. We actually, noticed that most of wrong names are lowercase. So, we pulled them out. After detecting issues, we document them. Figure 2 and Figure 3 show tidiness and quality issues detected.

Data cleaning

After assessing the data, we first make a copy in three tables: **archive_clean**, **image_predictions_clean**, and **tweet_json_clean**. Then, we define how to fix issues detected , we write code to fix them and we test the code. We end up with two tables: **archive_clean** with sixteen (16) columns and **image_predictions_clean** with twelve (12) columns. Let's dive into what we did.

Cleaning tidiness issues

Tidiness issues have been cleaned according to the rules of tidy data. We

- kept in the **archive_clean** table, only rows of the tweets which images are in table **image_predictions_clean** in order to have consistent data;
- merged **tweet_json** table columns retweet_count and favorite_count to **archive_clean** table so as each observational units to form a table;
- melted doggo, floofer, pupper, puppo columns into one single column stage. After storing data in stage column, we identified two wrong stages on the image below: To fix this issue we use the Dogtionary. Thus, we replaced *doggopupper* by puppo, *doggofloofer* by floofer and *doggopuppo* by doggo;
- replaced the column name img_num by best_prediction for clarity purpose.

Tidiness

1. Tweets to keep in the **`archive`** table are those in **`image_predictions`** table.
2. Retweet count and favorite count in **`tweet_json`** table should in **`archive`** table.
3. Column `text` in **`archive`** table contains URLs that should be removed since there is a column **expanded_urls** that can direct to the tweet.
4. Dog Stages `doggo, floofer, pupper, puppo` should be stored in `stage` column in **`archive`** table.
5. `img_num` is not an appealing variable name since it corresponds to the best predictions.

Figure 3: Tidiness issues

Quality

`archive` table

1. `timestamp` data type should be datatime.
2. `expanded_urls`, missing values
3. `name` **Zoey** instead of **my**
4. `name` **Al Cabone** instead of **Al**
5. `name` **O'Malley** instead of **O**. Check single letter names.
6. `name` **Quizno** instead of **his**. Seems not to be a dog.
7. **a, not, one, an, just, very, actually, such, the, this, unacceptable,....** instead of **None** or names. These names seems to be lower case.
8. `source` column as Text-formatted string instead HTML-formatted string for clarity.
9. Some `rating numerators` and `denominators` don't match with rating in text column.
10. `in_reply_to_user_id`, `in_reply_to_status_id`, `retweeted_status_id`, `retweeted_status_user_id` in **string** instead **float**. Missing values.
11. `retweeted_status_timestamp` data type should be a datatime.

`image_predictions` table

12. `img_num` data type should be categorical and it values more appealing

Figure 4: Quality issues

```
Entrée [45]: archive_clean.stage.value_counts()
```

```
Out[45]: 1741
pupper          210
doggo            65
puppo            23
doggopupper      11
floofer           7
doggofloofer      1
doggopuppo        1
Name: stage, dtype: int64
```

Figure 5: Wrong stages highlighted

Cleaning quality issues

Quality issues have been cleaned regarding data quality dimensions.

Completeness issues: These are missing records. See issues 2 and 10 on Figure 4 ;

Validity issues: These are data type issues. See issues 1, 10 and 11 on Figure 4 ;

Accuracy issues: These are wrong names and text formatting issues. See issues 3, 4, 5, 6, 7, 8, 9 and 112 on Figure 4 .

Data storing

Storing cleaned data

After cleaning data, we stored it in two files:

- twitter_archive_master.csv for **archive_clean** DataFrame;
- image_predictions_master.csv for **image_predictions_clean** DataFrame.

Twitter API credentials

In order to hide Twitter API keys, secrets, and tokens we store them in a SQLite database called WeRateDogs.db

Conclusion

Throughout this document, we presented our data wrangling efforts. We are looking for ways to improve what have done so far. So, any remark and suggestion are welcomed. In the second report, act_report.pdf, we will our analysis and visualization efforts.