Wrangle report

Introduction

Data wrangling is a crucial and time consuming phase of any data analytics job. It is composed of many steps that need different amounts of effort to devote in order to be able to analyze and conclude interesting insights.

In this document, I will share with you a brief description on the wrangling efforts realized on the WeRateDogs twitter account data, that has been gathered from three different sources.

Gathering efforts:

1. Load the first file that was given by Udacity "`twitter-archive-enhanced.csv`" into a dataframe using read_csv(): that was straightforward.
2. Load programmatically a tsv file given the url. I needed more investigation in order to fully understand each step and I found it in: `https://www.adamsmith.haus/python/answers/how-to-download-a-csv-file-from-a-url-in-python`
3. Using Twitter API to download more data about the same tweets we have in the "`twitter-archive-enhanced.csv`" file.

   It took around a week of emails between me and the twitter team before they granted me developer access in order to create my own keys and tokens.

   The first attempt, I was able to put data downloaded through twitter API directly into a dataframe without saving it in a tweet-json.txt file in about 13 minutes. Also, When analyzing exceptions, I noticed a lot of "rate exceeding limit" and that's when I noticed I had just a few tweets in the dataframe. After some research, I found this: `https://stackoverflow.com/questions/21308762/avoid-twitter-api-limitation-with-tweepy` and added **wait_on_rate_limit=True** to the following:

   ```
   api = tweepy.API(auth, wait_on_rate_limit=True)
   ```

   The second attempt, I was able to put the content of all tweets in .txt file as a string variable. But I found it hard to exploit later. I should get back to this later in order to find a solution.

   The third attempt, I managed to create the .txt file with the necessary data in about 35 minutes. Then I put everything in a third dataframe using "read_json()".

Assessing efforts:

I opened all the files using libreoffice since I am on a ubuntu operating system. Then I started noticing many quality issues that I wrote down in the jupyter notebook. From the beginning I started classifying these issues as a preparation to the cleaning step in which some issues are preferably cleaned before others. Then tidiness issues were also noted.

When I commenced looking for issues programmatically, I noticed that the issues that could be represented by numbers (e.g. `There are 181 retweets in the file`) are more precise than before. Some of the programmatic issues I was looking for required more in depth internet research in order to make the code correct.

It is worth mentioning that I was able to find a lot more issues in the process but since we have a limited time to deliver the project, I decided to limit the investigation at that point.

Cleaning efforts:

First, I made copies of the three dataframes. Then, I decided to start by cleaning completeness issues, and I found out that I don't need those columns. However, before dropping them all, I decided to take advantage of the "retweet_status_id" column in order to delete all rows that are not original tweets. After that, I continued using the define-code-test strategy to clean all the remaining issues. Again, I did some research looking for ways to help me unblock some situations.

Visualization efforts:

I did three visualizations straightforwardly. But one took me some time and a lot of research in order to be able to make it happen.