Wrangle Report

Table of Contents

Introduction

The purpose of this project is to retrieve insightful information from a carefully curated, though still messy, dataset, the WeRateDogs Twitter archive. In the process I had a chance to practice and experience data wrangling techniques which include gathering data via Twitter API, cleaning data, storing cleaned dataframes, and visualizing some aspects of the data to make a point. The goal aims at getting hands-on the data cleansing process to prepare data for further interesting and trustworthy analyses.

In addition to the base dataset, an enhanced Twitter archive, which is a csv file available via download. We also need to retrieve additional data via Twitter API and the image predictions tab-delimited data which is stored online via an URL with the assistance of the requests module.

Data Gathering

Enhanced Twitter Archive:

- Served essential info, such as tweet id, timestamp, tweet text, rating (numerator & denominator), and name, etc.

Additional data via Twitter API:

- Authentication required Twitter access token/secret and customer key/secret.

- Retrieved data and stored as a JSON file;
- Retrieving data via Twitter API and Python in Jupyter was a challenge. It became a great learning experience to finally figure out and get data.

Image Predictions File:

- It took me some time to figure out how to interact with the requests module and use what it provided.

Retrieving Tweets archives via Twitter API is an essential skill in data engineering. Although it wasn't new to me because I used to work on a Spark Sentiment Analysis project in Scala based on Tweets this task is still a challenge. In previous project, I retrieved Tweets in four other environment, Eclipse, JetBrains IntelliJ, from Linux Bash and from a Windows shell as well. They shared a lot in common. This project asked us to retrieve Tweets in Jupyter and via Python and we need to first store Tweets in JSON format. Even with the help on Twitter API syntax the process still took me two days to realize that Twitter secrets and keys are still required in the tweet retrieval process. Otherwise, we won't get any valid tweet ids.
Similar to many other issues, the moment I confirmed the root cause. I wondered why it took me so long to figure out such a straightforward issue yet how come the obstacle looked so tough before its solution was revealed?

During Data Assessing

After I gathered the data, I used the following tools to identify quality and tidiness issues.
To get an idea about the data, I apply these Pandas methods

- on DataFrame: .sample(), .head(), .info(), columns, .describe(), count(),
- on DataFrame column: .unique(), .value_counts, .nunique(), .type(),

To spot missing values, erroneous datatypes and other quality issues.

- On DataFrame: .isna(axis=0|1).sum(),
- On DataFrame column: .notnull().count(),

Issues – in summary:

Besides missing values or mislabeled data, I performed these tasks:

- **convert data types,**
- **rename column names,**
- **replace characters such as underscore, and**
- **exclude retweets to better prepare the data for analyses**.

Wrangle Report

I learned that cleansing decision, such as the right data types, number of columns, etc., really depend on what we're gonna do with the data. For example, date and time can be string or datetime. The right data type really depends on what the data will be playing in analyses. Mislabeled issues, i.e. Issues on values, need visual assessment and often require domain knowledge or at least the context information. For example, invalid names depend on visual inspection. Whether a name is valid or not may be addressed systematically via lexicon comparison. It's required to possess some basic understanding on the meaning of the data.

Quality Issues

- Tweet_id, timestamp, sources, and dog_stages need to convert to suitable data types for next operations.
- Dog names may not be valid, such as "the", "a", "actually", and "infuriating" etc.
- Missing values in the dog stages column as "None"
- Identified 181 retweets and excluded them from the analysis.
- Numerators beyond its denominator 10 are no mistake as we know. It attracted spotlights and successfully brought WeRateDogs in public attention. Nothing wrong about a success story.
- Values in p1, p2 and p3 (Image Predications) were not all initial capitalized. Applied .title() to uniform the cases.

Tidiness Issues:

- Tweet_id is a unique value for each tweet. Before merging three DataFrames we need to rename column name.
- Joined four dog stages into one.
- Dropped columns which were not in alignment with the goal.

In the WeRateDogs Twitter Archive DataFrame,

- Replace faulty names or incorrect names
  Use a dictionary to define a map. Flexible for future expansion.
- Exclude retweets because they are not the original and introduce redundancy
- Convert the timestamp column from a string to DateTime objects
- Date and Time columns should be Timestamp to fit time-related operations.
- Combine the Dog Stages into one column named Stage
- Remove unwanted columns and rearrange columns for easier reads
- Convert tweet_id data type from integer (numpy.int64) or float to String (str)
- rating_numerator & rating_denominator should be float, rather than integers.

In the TWEET API Online dataframe

- Rename the "id" column to "tweet_id" to match the other 2 DataFrames
- convert tweet_id column from a number to a string value
- Exclude retweet from the Twitter API Online DataFrame
- remove unwanted columns from the dataframe

In the Image Predictions DataFrame

- Replace the underscore in the p1, p2, and p3 columns
- Initial-capitalizing p1, p2 and p3 values
- Convert "tweet_id" column to String

In Data Cleaning

I performed the following tasks on Pandas column to prepare the data for analyses:

- astype(str) to convert data type,
- Used lambda function to manipulate values: apply(lambda time: time.strftime('%m-%d-%Y'))

Use an inner join, i.e. Pandas merge(), to combine DataFrames into a single one
After three cleaned DataFrames were merged I performed three additional fixes in verification.

- Dropped two redundant columns: "unnamed : 0" and 'quoted_status_permalink'
- Converted the "tweet_id" column to str
- Apply replace() and title()

Other issue

“False Negative” vs. “True Negative” requires human intervention.

When peeking at the images among the 4 least rated tweets I notice that only 1 tweet is actually a dog. The rest of them are not dogs at all. Although it’s no surprise they are in the least popular tweets because they are no dog the finding still led me to seek for the next 10 tweets with the lowest rating. The rating is expanded from 1 to 3 where many tweets were rated 3. I pick the first one which made it the 10th entity and dropped the rest of rated 3 entities.

I looked into these lowest 10 rated tweets and found that only two of them were classified as dogs with more than 70% confident level (CL). What caught my attention is this dog classified as “Paper Towel” but is really a Maltese lying on the floor mat with the same color as his hair. The confident level of its prediction is only 32.80% though. The visual inspect of the picture reveals the fact.

Because of this, I expanded my filter criteria. In addition to the “Image classified as Dog with CL > 70%” rule I added another rule, “Image not classified as Dog yet with CL < 30%”. I did get

“Paper Towel” this time which is a “False Negative” but the criteria also included a tweet which was classified as Tick but it looked like a Cricket. Its confident level is so low as 24.25%.

I would like to summarize my research on this issue as follows:

- “False Negative” and “True Negative” demand visual inspection.
- Human inspection is needed before more adaptive algorithms are discovered.

Summary

While we invest efforts in developing algorithms and systematically approaching to automate the task is always a goal we need to be aware of retaining sufficient amount of information to avoid undermine the trustworthy of the result. It is often a challenge to find a balance between underfit and overfit

In many cases there is no right or wrong on the data itself. Cleaning decision and efforts are based on how to better prepare the data for next operations and to smooth the process while preserving enough information to reach trustworthy conclusion.

Next Steps

One of my To-Dos is to explore how to systematically spawn multiple Jupyter cells in current notebook. I can create a new Jupyter cell, Markdown or Code, get_ipython().set_next_input(s).

But, my attempt to spawn two or more cells failed. I think this is an interesting technique to possess. I researched the nbformat module and its v4 class. Since I’m running out of time on my program I’ll stop here and move on.