Report

Wrangle and Analyze Data

From Michael Eibner

With Twitter Data from WeRageDogs

A Udacity Project

Project Overview and Introduction

Real-world data rarely comes clean. Using Python and its libraries, you will gather data from a variety of sources and in a variety of formats, assess its quality and tidiness, then clean it. This is called data wrangling. You will document your wrangling efforts in a Jupyter Notebook, plus showcase them through analyses and visualizations using Python (and its libraries) and/or SQL.
The dataset that you will be wrangling (and analyzing and visualizing) is the tweet archive of Twitter user @dog_rates, also known as WeRateDogs. WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog. These ratings almost always have a denominator of 10. The numerators, though? Almost always greater than 10. 11/10, 12/10, 13/10, etc. Why? Because "they're good dogs Brent." WeRateDogs has over 4 million followers and has received international media coverage.
WeRateDogs downloaded their Twitter archive and sent it to Udacity via email exclusively for you to use in this project. This archive contains basic tweet data (tweet ID, timestamp, text, etc.) for all 5000+ of their tweets as they stood on August 1, 2017. More on this soon.

Image via Boston Magazine

Project Motivation

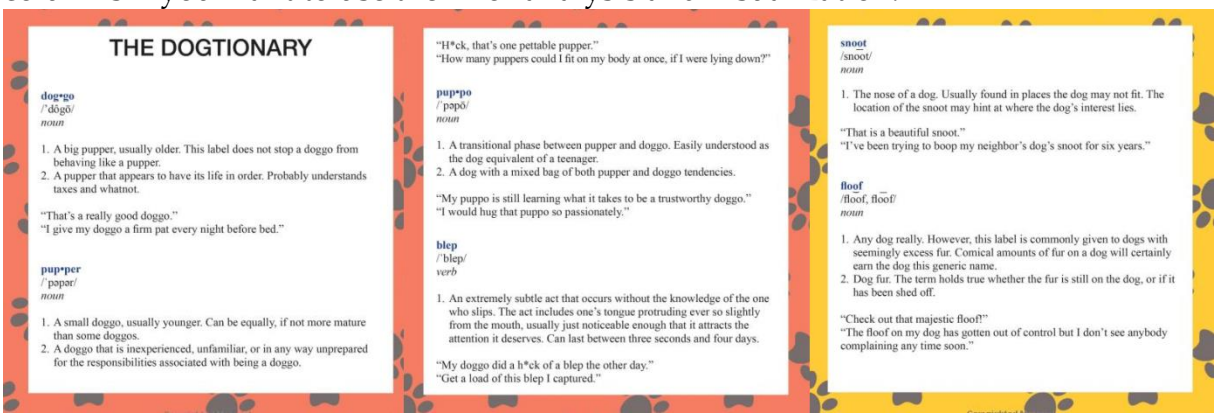
Context

Your goal: wrangle WeRateDogs Twitter data to create interesting and trustworthy analyses and visualizations. The Twitter archive is great, but it only contains very basic tweet information. Additional gathering, then assessing and cleaning is required for "Wow!"-worthy analyses and visualizations.
The Data
In this project, you will work on the following three datasets.
Enhanced Twitter Archive
The WeRateDogs Twitter archive contains basic tweet data for all 5000+ of their tweets, but not everything. One column the archive does contain though: each tweet's text, which I used to extract rating, dog name, and dog "stage" (i.e. doggo, floofer, pupper, and puppo) to make this Twitter archive "enhanced." Of the 5000+ tweets, I have filtered for tweets with ratings only (there are 2356).

text	rating_ numerator	rating_ denominator	name	doggo	floofer	pupper	puppo
This is Phineas. He's a mystical boy. Only ever appears in the hole of a donut. 13/10 https://t.co/MgUWQ76dJU	13	10	Phineas	None	None	None	None
This is Tilly. She's just checking pup on you. Hopes you're doing ok. If not, she's available for pats, snugs, boops, the whole bit. 13/10	13	10	Tilly	None	None	None	None
This is Archie. He is a rare Norwegian Pouncing Corgo. Lives in the tall grass. You never know when one may strike. 12/10 https://t.co	12	10	Archie	None	None	None	None
This is Darla. She commenced a snooze mid meal. 13/10 happens to the best of us https://t.co/tD36da7qLQ	13	10	Darla	None	None	None	None
This is Franklin. He would like you to stop calling him "cute." He is a very fierce shark and should be respected as such. 12/10 #BarkW	12	10	Franklin	None	None	None	None
Here we have a majestic great white breaching off South Africa's coast. Absolutely h*ckin breathtaking. 13/10 (IG: tucker_marlo) #Bar	13	10	None	None	None	None	None
Meet Jax. He enjoys ice cream so much he gets nervous around it. 13/10 help Jax enjoy more things by clicking below https://t.co/Zr4hWfAs1H https://t.co/tVJBRMnhxl	13	10	Jax	None	None	None	None
When you watch your owner call another dog a good boy but then they turn back to you and say you're a great boy. 13/10 https://t.co/	13	10	None	None	None	None	None
This is Zoey. She doesn't want to be one of the scary sharks. Just wants to be a snuggly pettable boatpet. 13/10 #BarkWeek https://t.c	13	10	Zoey	None	None	None	None
This is Cassie. She is a college pup. Studying international doggo communication and stick theory. 14/10 so elegant much sophisticat	14	10	Cassie	doggo	None	None	None
This is Koda. He is a South Australian deckshark. Deceptively deadly. Frighteningly majestic. 13/10 would risk a petting #BarkWeek ht	13	10	Koda	None	None	None	None
This is Bruno. He is a service shark. Only gets out of the water to assist you. 13/10 terrifyingly good boy https://t.co/u1XPQMl29g	13	10	Bruno	None	None	None	None
Here's a puppo that seems to be on the fence about something haha no but seriously someone help her. 13/10 https://t.co/BxvuXk0UC	13	10	None	None	None	None	puppo
This is Ted. He does his best. Sometimes that's not enough. But it's ok. 12/10 would assist https://t.co/f8dEDcrKSR	12	10	Ted	None	None	None	None
This is Stuart. He's sporting his favorite fanny pack. Secretly filled with bones only. 13/10 puppared puppo #BarkWeek https://t.co/y70c	13	10	Stuart	None	None	None	puppo

The extracted data from each tweet's text

I extracted this data programmatically, but I didn't do a very good job. The ratings probably aren't all correct. Same goes for the dog names and probably dog stages (see below for more information on these) too. You'll need to assess and clean these columns if you want to use them for analysis and visualization.



THE DOGTIONARY

dog•go
/ˈdôgō/
noun

1. A big pupper, usually older. This label does not stop a doggo from behaving like a pupper.
2. A pupper that appears to have its life in order. Probably understands taxes and whatnot.

"That's a really good doggo."
"I give my doggo a firm pat every night before bed."

pup•per
/ˈpəpər/
noun

1. A small doggo, usually younger. Can be equally, if not more mature than some doggos.
2. A doggo that is inexperienced, unfamiliar, or in any way unprepared for the responsibilities associated with being a doggo.

"H*ck, that's one pettable pupper."
"How many puppers could I fit on my body at once, if I were lying down?"

pup•po
/ˈpəpō/
noun

1. A transitional phase between pupper and doggo. Easily understood as the dog equivalent of a teenager.
2. A dog with a mixed bag of both pupper and doggo tendencies.

"My puppo is still learning what it takes to be a trustworthy doggo."
"I would hug that puppo so passionately."

blep
/ˈblep/
verb

1. An extremely subtle act that occurs without the knowledge of the one who slips. The act includes one's tongue protruding ever so slightly from the mouth, usually just noticeable enough that it attracts the attention it deserves. Can last between three seconds and four days.

"My doggo did a h*ck of a blep the other day."
"Get a load of this blep I captured."

snoot
/snoot/
noun

1. The nose of a dog. Usually found in places the dog may not fit. The location of the snoot may hint at where the dog's interest lies.

"That is a beautiful snoot."
"I've been trying to boop my neighbor's dog's snoot for six years."

floof
/floof, fl̄oof/
noun

1. Any dog really. However, this label is commonly given to dogs with seemingly excess fur. Comical amounts of fur on a dog will certainly earn the dog this generic name.
2. Dog fur. The term holds true whether the fur is still on the dog, or if it has been shed off.

"Check out that majestic floof!"
"The floof on my dog has gotten out of control but I don't see anybody complaining any time soon."

The Dogtionary explains the various stages of dog: doggo, pupper, puppo, and floof(er) (via the #WeRateDogs book on Amazon)

Additional Data via the Twitter API

Back to the basic-ness of Twitter archives: retweet count and favorite count are two of the notable column omissions. Fortunately, this additional data can be gathered by anyone from Twitter's API. Well, "anyone" who has access to data for the 3000 most recent tweets, at least. But you, because you have the WeRateDogs Twitter archive and specifically the tweet IDs within it, can gather this data for all 5000+. And guess what? You're going to query Twitter's API to gather this valuable data.

Image Predictions File

One more cool thing: I ran every image in the WeRateDogs Twitter archive through a **neural network** that can classify breeds of dogs*. The results: a table full of image predictions (the top three only) alongside each tweet ID, image URL, and the image number that corresponded to the most confident prediction (numbered 1 to 4 since tweets can have up to four images).

tweet_id	jpg_url	img_num	p1	p1_conf	p1_dog	p2	p2_conf	p2_dog	p3	p3_conf	p3_dog
892177421306343426	https://pbs.twimg.cor	1	Chihuahua	0.323581	TRUE	Pekinese	0.0906465	TRUE	papillon	0.0689569	TRUE
891815181378084864	https://pbs.twimg.cor	1	Chihuahua	0.716012	TRUE	malamute	0.078253	TRUE	kelpie	0.0313789	TRUE
891689557279858688	https://pbs.twimg.cor	1	paper_towel	0.170278	FALSE	Labrador_retriever	0.168086	TRUE	spatula	0.0408359	FALSE
891327558926688256	https://pbs.twimg.cor	2	basset	0.555712	TRUE	English_springer	0.22577	TRUE	German_short-haired_pointer	0.175219	TRUE
891087950875897856	https://pbs.twimg.cor	1	Chesapeake_Bay_retriever	0.425595	TRUE	Irish_terrier	0.116317	TRUE	Indian_elephant	0.0769022	FALSE
890971913173991426	https://pbs.twimg.cor	1	Appenzeller	0.341703	TRUE	Border_collie	0.199287	TRUE	ice_lolly	0.193548	FALSE
890729181411237888	https://pbs.twimg.cor	2	Pomeranian	0.566142	TRUE	Eskimo_dog	0.178406	TRUE	Pembroke	0.0765069	TRUE
890609185150312448	https://pbs.twimg.cor	1	Irish_terrier	0.487574	TRUE	Irish_setter	0.193054	TRUE	Chesapeake_Bay_retriever	0.118184	TRUE
890240255349198849	https://pbs.twimg.cor	1	Pembroke	0.511319	TRUE	Cardigan	0.451038	TRUE	Chihuahua	0.0292482	TRUE
890006608113172480	https://pbs.twimg.cor	1	Samoyed	0.957979	TRUE	Pomeranian	0.0138835	TRUE	chow	0.00816748	TRUE
889880896479866881	https://pbs.twimg.cor	1	French_bulldog	0.377417	TRUE	Labrador_retriever	0.151317	TRUE	muzzle	0.0829811	FALSE
889665388333682689	https://pbs.twimg.cor	1	Pembroke	0.966327	TRUE	Cardigan	0.0273557	TRUE	basenji	0.00463323	TRUE
889638837579907072	https://pbs.twimg.cor	1	French_bulldog	0.99165	TRUE	boxer	0.00212864	TRUE	Staffordshire_bullterrier	0.00149818	TRUE
889531135344209921	https://pbs.twimg.cor	1	golden_retriever	0.953442	TRUE	Labrador_retriever	0.0138341	TRUE	redbone	0.00795775	TRUE

Tweet image prediction data

So, for the last row in that table:
tweet_id is the last part of the tweet URL after "*status/*"
→ **https://twitter.com/dog_rates/status/889531135344209921**
p1 is the algorithm's #1 prediction for the image in the tweet → **golden retriever**
p1_conf is how confident the algorithm is in its #1 prediction → **95%**
p1_dog is whether or not the #1 prediction is a breed of dog → **TRUE**
p2 is the algorithm's second most likely prediction → **Labrador retriever**
p2_conf is how confident the algorithm is in its #2 prediction → **1%**
p2_dog is whether or not the #2 prediction is a breed of dog → **TRUE**
etc.
And the #1 prediction for the image in that tweet was spot on:

(Original Text from Udacity)

Gathering:

Twitter Archive:
I downloaded the 'twitter-archive-enhanced.csv' file from Udacity and read it with pandas.read_csv. Then I was able to create a data frame and called it 'df_archive'.

Image Predictions:
The second data frame I created was the 'df_img_pred'. I got it by downloading it programmatically. It was a little bit trickier creating a data frame from a tsv file.

Twitter API:
This one was the hardest data to gather. At first, I had to create a Twitter developer account and then I got Consumer Keys and Access Tokens. With the Tweepy libraty I downloaded the Twitter JSON data. Then I extracted a list of tweet IDs from the 'twitter-archive-enhanced.csv' file.
I recorded a text file, called 'tweet_json.txt' and from this file, I created the third data frame, called 'df_twitter'.

Assessing Data:

Archive Dataframe

With the .head() function, I was able to get an overview about the file.
Because I wanted to know how many data I got, I used the .info() function.
In the third step I was able to analyze that every text has a hyperlink.
I checked if there are duplicates in the 'tweet_id' column.
The rating system from WeRateDogs is special and I wanted to know how special, so I checket it's numerator and denominator.
In the next step I checked if all dogs have names. And obviously this is not the case.
I checked the source column, too, but , because I'm a beginner, I was not able to work with this data.

Image Prediction Dataframe:
Like in the Archive Data, I got with .head() and .info() an overview about the tables.

Twitter API Dataframe:
With .head() and .info() I got an overview from the tables.
Then I checked the 'tweet_id' column if there were duplicates.

I collected the following issues:

1. Wrong Dog Names:
The 'names' column had wrong data, obviously "the" is not a dog name for instance.
That's a quality issue.

2. & 3. Wrong Type:
The 'tweet_id' columns in the Archive and the Image Predictions Dataframes were an integer. IDs should always be strings.
That's a quality issue.

4. Missing Data:
The 'expanded_urls' column is missing data. Without it, we're not able to see where the tweet comes from.

5. Too much of Information:
The 'text' column contains strings and URLs, this should be in separated columns.
That's a tidiness issue.

6. Wrong Type:
For future analysis, the 'timestamp' column should be a datetime.
That's a quality issue.

7. Duplicates:
In the 'jpg_url' column there are duplicates.
That's a quality issue.

8. Not consistent Format:
The name of the dog breeds in the columns 'p1', 'p2', and 'p3', are not even formatted.
That's a quality issue.

9. Not necessary Columns:
The columns 'doggo', 'floofer', 'pupper', and 'puppo', are not necessary. Most of the data in it is 'None'.
That's a tidiness issue.

10. Separated Dataframes:
The Archive Dataframe, Image Prediction Dataframe and the Twitter Dataframe are 3 separated Tables. This is not necessary.
That's a tidiness issue.

11. Confusing Rating System:
The rating denominator should always be 10, but it hast different numbers, too. Even in the rating numerator, there're overwhelming outliers.

Cleaning Data:

Issue 1:
Every dog's name is capitalized. Therefore, every wrong data in the 'name' column is not capitalized. I replaced all strings not-capitalized with the word 'None'. So, we got only the right names and the 'None's.

Issue 2 & 3:
I selected the 'tweet_id' column from the Archive and from the Image Prediction Dataframes and setted it as a string. As a string, it's easier to handle the data or merge it to other dataframes.

Issue 4:
Instead of dropping the rows with missing data from the 'expanded_urls' column, I selected all rows with non-missing data by pd.isnull() and recreated the dataframe 'archive_clean'.

Issue 5:
At first I created two empty lists for the new columns, one for the texts and one for the URLs. With a for loop I splitted all text columns and added them to their specific list. Then, the 'text' column got its texts without URLs back and the 'text_url' column got added to the 'archive_clean' dataframe with the .concat() function.

Issue 6:
To handle the data correctly, I changed the type from object / string to datetime and defined the format as YYYY-MM-DD to make it easier to analyze it.

Issue 7:
We want to work with unique images, so the duplicates had to disappear. In this case I dropped the duplicates out of the dataframe.

Issue 8:
The 'p1-3' columns weren't equal formatted, sometimes capitalized, sometimes '-' instead of '_' and sometimes space between the words. To get an even formation, I replaced all '-' and spaces to underscores and setted every word to lowercase. Because writing it all down for every single column would be too much lines of code, I used a function, called 'column_formatting'. Then I called the function for every column to get a clean result.

Issue 9:
At first, I created a variable for all 4 unnecessary columns and replaced all 'None' with empty strings. In the second step I added the values of all 4 columns together and stored it as 'dog_list'. Some rows had multiple values, I called them 'double_stager'. With .value_counts() I found out that there only are 'doggopupper', 'doggopuppo' , and 'doggofloofer'. On the next step, I created the new 'dog_stage' column with pd.DataFrame() from the 'dog_list'.

Issue 10:
We only want to work with original data. It means, the retweet and replies columns are not necessary. At first I had to got rid of the rows with retweets, so I recreated the dataframe where the 'retweet_status_id', 'retweet_status_user_id', and the 'retweet_status_timestamp' are null, because the original data has no values there. Because I'm lazy, I dropped the retweet, replies and the unnecessary columns from the Issue 9 in the same line of code by using .drop().

Issue 11:
The Archive, Image Prediction, and Twitter API Dataframes should be merged in one single Dataframe. Because all 'tweet_id' columns were formatted to strings, it's possible to use the .merge() function. I didn't wanted to lose data, so I used a left join.

Issue 12:
The rating system was confusing. I was able to detect that there were decimal numbers in the 'text' columns but the type was integer. Therefore, I changed the type as a float. Because the denominator wasn't the same every row, the rating system was resulting wrong scores. To get a fair rating, I created a new column by dividing the numerator through the denominator.

Storing Data:

I stored the data as a .csv file, called 'twitter_archive_master.csv'.