Don't Take This Personally: Sentiment Analysis for Identification of "Subtweeting" on Twitter

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Noah Segal-Gould

Annandale-on-Hudson, New York
May, 2018

Abstract

The Oxford English Dictionary states that "subtweet" is defined as "(on the social media application Twitter) a post that refers to a particular user without directly mentioning them, typically as a form of furtive mockery or criticism." Following the rapid growth and adoption of social networking websites like Twitter, sentiment analysis has garnered much research interest in recent years. To computationally identify and categorize opinions expressed in text, sentiment analysis of figurative language such as irony and sarcasm has garnered even more recent interest. In this project, I will treat the identification of subtweets as a figurative language sentiment analysis problem and apply a novel approach for data collection and labeling, as well as Naive Bayes text classification. By identifying subtweets as they are posted online in real time, I will create a Twitter bot which archives and interacts with them. In addition to the paper, this project will be made available online with its source code and all data collected during its completion.

Contents

Dedication

I dedicate this senior project to @jack, who has willfully made numerous changes to Twitter which inevitably angered millions.

Acknowledgments

Thank you professors Sven Anderson, Keith O'Hara, and Rebecca Thomas for making this project possible through your combined efforts to teach and advise me. Thank you Benjamin Sernau '17 for enduring through three years of Computer Science courses with me and being a source of unending joy in my life. Thank you to Julia Berry '18, Aaron Krapf '18, and Zoe Terhune '18 for being my very best friends and giving me things worth caring about. Finally, thank you to my parents Tammy Segal and Emily Taylor for your constant support and patience throughout my four years at Bard College.

1
Introduction

1.1 Background

The news and social networking service Twitter had over 140 million active users who sent 340 million text-based Tweets to the platform every day by March of 2012 [1]. Since Twitter-founder Jack Dorsey sent the first Tweet in March of 2006 [2] social scientists, advertisers, and computer scientists have applied machine learning techniques to understand the patterns and structures of the conversations held on the platform. One such technique is sentiment analysis, which seeks to ascertain the opinions of bodies of text. Sentiment analysis techniques are often treated as classification problems which seek to place text into categories such as **positive**, **negative**, and **neutral**.

On Twitter, the most common way to publicly communicate with another user is to compose a tweet and place an "@" before the username of that user somewhere in the tweet (e.g. "How are you doing, @NoahSegalGould?"). Through this method, public discussions on Twitter maintain a kind of accountability: even if one were to miss the notification that they were mentioned in a tweet, one's own dashboard keeps a running list of their most recent mentions.

If an individual sought to disparage or mock another, they could certainly do so directly. But the targeted user would probably notice, and through the search funtions of the platform, anyone could see who has mentioned either their own or another's username. Instead, a phenomenon

persists in which users of the platform deliberately insult others in the vaguest way possible. Tweets of this kind are colloquially called "subtweets" and typically target a specific person but do not contain the username of that person.

All users do not necessarily possess the same exact definition of "subtweet." I trust the Oxford English Dictionary's "[tweet] that refers to a particular user without directly mentioning them, typically as a form of furtive mockery or criticism," however that definition is perhaps too restrictive. Some individuals believe subtweets abide by this definition, but others expand it to allow inclusion of others' real names (especially if that individual does not own a Twitter account), and some do not even require that a particular user be the target of the tweet. In this project, I implement a classifier which abides by a particularly loose definition in order to please as many parties as possible.

1.2 Motivation

The inspiration for this project came from interests I garnered taking courses within Bard College's Computer Science department as well as its Experimental Humanities concentration. The very first course I attended at Bard College was Professor Keith O'Hara's *Object-Oriented Programming with Robots*. It served as my first introduction to computer programming, and for my final project I created *Fuzzfeed*: a Twitter bot which generated fake *Buzzfeed* article titles. The following academic year, I wrote programs in Professor Collin Jennings' *Signs and Symbols: Patter Recognition in Literature and Code* and Professor Rune Olsen's *Cybergraphics* which analyzed and visualized topic models of poetry on Twitter. The first time I implemented sentiment analysis on my own was in Professor Gretta Tritch-Roman's *Mapping the 19th Century City*, for which I sought to analyze 1860s New York City newspapers for their sentiments toward immigration. Natural language processing struck me as entertaining and fruitful, so I chose to pursue it further.

By my Junior year, my friends and I used Twitter on a daily basis. In my free time, I made Twitter bots that utilized Markov chains to generate text based on corpora of their tweets. Data

collection became a passion of mine as I learned to appreciate the utility and acknowledge the deliberate limitations of web-based APIs. I taught myself web-scraping and utilized Python's *BeautifulSoup* library to programmatically acquire text from Bard College's official online course catalog as well as Twitter's web interface. These skills for programmatically interacting with the world wide web became useful resources during the completion of this project.

My peers introduced me to subtweeting, and I started to pay closer attention to tweets that followed the typical patterns of distanced criticism that subtweets were known for. Because some format seemed to exist which was popularly applied to produce the optimal subtweet, I pitched the concept of subtweet classification to my senior project adviser, Professor Sven Anderson, and I started work on this project in the Fall.

I was initially motivated to complete a senior project on this topic because I wanted to create something useful to my peers and also challenge their notions of public and private interactions on social networking applications like Twitter. Individuals I knew personally would take to the platform to complain indirectly about one another through their subtweets. Friends and I shared evenings debating on if a particular mutual friend's complaints were actually subtweets, and I wondered if that guess-work could be done by a program. I also wanted to challenge the hypocrisy of utilizing a service which presents itself as a public forum to speak in distinctly private ways. Toward this end, I decided the project would be in pursuit of the following goals: it would provide a framework for collecting examples of subtweets, train a classification algorithm using those examples, and finally utilize that classifier in real time to make tweets which were intended to be unseen by specific parties easily accessible to all parties. In presenting covertly hurtful content as obviously hurtful in a public fashion, perhaps I could promote a particular awareness that tweets posted by public accounts were indeed publicly accessible, and that Twitter's End User License Agreement (EULA) allowed for this kind of monitoring.

1.3 Changes in Data Acquisition

1.4 Sentiment Analysis

Sentiment analysis and opinion mining are typically used synonymously. Both terms were apparently coined in 2003, respectively in "Sentiment Analysis: Capturing Favorability Using Natural Language Processing"[6] and "Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews"[7]. In a commercial context, sentiment analysis and opinion mining are useful tools for anyone looking to sell a product. With the vast social networks available, these methods are becoming widespread. Instead of focus groups, opinion polls, and conduct surveys, sentiment analysis and opinion mining programs are increasingly applied to social networking websites to analyze the sentiments and opinions of users toward topics and products.

For various purposes, there exist numerous sentiment analysis and opinion mining methods. SemEval, which started as Senseval in 1998, holds evaluations in the form of competitions in which several tasks and subtasks are assigned. In 2015, SemEval tasks 10 [8] and 11 [9] focused respectively on sentiment analysis of figurative language on Twitter and sentiment analysis of Tweets in general. In 2016's task 4 [10], the focus was on sentiment analysis on Tweets, and again in 2017's task 4 [11] it was the same with the added challenge of working with Arabic-only Tweets. The subtasks assigned within a particular task are used as problems to which anyone can send a submission as a solution.

1.5 The Twitter API

1.6 Regular Expressions, N-Grams, & Tokenization

1.7 TF & TF-IDF

TF-IDF, or term frequency-inverse document frequency, is a statistical representation of how important a single word is for each document in a collection of documents.

1.8 Naive Bayes

Naive Bayes classifiers are probabilistic supervised learning models which make the "naive" assumption of independence between pairs of features being classified. Sentiment analysis is popularly performed through Naive Bayes.

1.9 Statistical Considerations

In tasks pertaining to text classification, like sentiment analysis, precision refers to the number of correctly labeled items which were labeled as belonging to the positive class and in fact did belong to that class (true positives) divided by the total number of elements which were labeled as belonging to the positive class including ones which were labeled positively either correctly or incorrectly. Recall, then, refers to the true positives divided by the total number of elements that actually belong to the positive class.

2
Implementation

2.1 Searching for Tweets Using the Twitter API

2.2 Cleaning the Data

2.3 Training the Classifier & K-Folds Cross-Validation

This is a process of splitting arrays into random train and test subsets according to Scikit Learn's

`sklearn.model_selection.train_test_split`

3
Results

3.1 Distributions & Datasets

For this project, I acquired and combined two datasets:

- Alec Go's Sentiment140 dataset [12]. It features 1,600,000 4-point scale classified (**highly negative** to **highly positive**) Tweets. 800,000 of these Tweets are classified as **positive** and 800,000 of them are classified as **negative**. This dataset was not tagged by humans. Instead, it was based entirely on emoticons present within individual Tweets.

- Ibrahim Naji's dataset [13], which features 1,578,628 Tweets. 790,185 of these Tweets are classified as **positive** and 788,443 of these Tweets are classified as **negative**. It was tagged by humans and on a binary (either **negative** or **positive**) scale.

3.2 Confusion Matrices

A confusion matrix is a table which visualizes the performance of an algorithm. In this case, I implemented a Naive Bayes classifier from Scikit Learn on my dataset and included in my results is a confusion matrix of the performance:

3.3 Most Informative Features

3.4 Statistical Analysis

4
Conclusion

Bibliography

[1] Twitter Inc., *Twitter turns six* (2012), available at `https://blog.twitter.com/official/en_us/a/2012/twitter-turns-six.html`.

[2] Jack Dorsey, *inviting coworkers* (2006), available at `https://twitter.com/jack/status/29`.

[3] Forsyth Alexander, *How to use Twitter activity to measure the effectiveness of your marketing* (2016), available at `https://www.ibm.com/blogs/business-analytics/how-to-use-twitter-activity-to-measure-the-effectiveness-of-your-marketing/`.

[4] Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan, *A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle*, Proceedings of the ACL 2012 System Demonstrations **ACL '12** (2012), 115–120.

[5] Twitter Inc., *Hateful conduct policy*, available at `https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy`.

[6] Tetsuya Nasukawa and Jeonghee Yi, *Sentiment Analysis: Capturing Favorability Using Natural Language Processing*, Proceedings of the 2nd International Conference on Knowledge Capture **K-CAP '03** (2003), 70–77, available at `http://doi.acm.org/10.1145/945645.945658`.

[7] Kushal Dave, Steve Lawrence, and David M. Pennock, *Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews*, Proceedings of the 12th International Conference on World Wide Web **WWW '03** (2003), 519–528, available at `http://doi.acm.org/10.1145/775152.775226`.

[8] Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov, *SemEval-2015 Task 10: Sentiment Analysis in Twitter*, SemEval@ NAACL-HLT (2015), 451–463.

[9] Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes, *SemEval-2015 Task 11: Sentiment Analysis of Figurative Language in Twitter*, Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015) (2015), 470–478.

[10] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov, *SemEval-2016 Task 4: Sentiment Analysis in Twitter*, SemEval@ NAACL-HLT (2016), 1–18.

[11] Sara Rosenthal, Noura Farra, and Preslav Nakov, *SemEval-2017 Task 4: Sentiment Analysis in Twitter*, Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) (2017), 502–518.

[12] Alec Go, Richa Bhayani, and Lei Huang, *Twitter Sentiment Classification using Distant Supervision*, CS224N Project Report, Stanford (2009), 12.

[13] Ibrahim Naji, *Twitter Sentiment Analysis Training Corpus (Dataset)* (2013), available at `http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/`.

Appendix A

Classifier Creator Program

segalgouldn Directory cleaning and file exporting. adb4c40 a minute ago

1 contributor

2177 lines (1655 sloc) | 53.2 KB

Using Scikit-Learn and NLTK to build a Naive Bayes Classifier that identifies subtweets

In all tables, assume:

- "❶" represents a single hashtag
- "❷" represents a single URL
- "❸" represents a single mention of username (e.g. "@noah")

Import libraries

```
%matplotlib inline
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction import text
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.externals import joblib
from os.path import basename, splitext
from random import choice, sample
from nltk.corpus import stopwords
from string import punctuation
from pprint import pprint
from glob import glob

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

import scipy.stats
import itertools
import enchant
import nltk
import json
import re
```

Set up some regex patterns

```
hashtags_pattern = re.compile(r'(\#[a-zA-Z0-9]+)')
```

```
urls_pattern = re.compile(r'(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]|\(([^\s()<>]+|(
```

```
at_mentions_pattern = re.compile(r'(?<=^|(?<=[^a-zA-Z0-9-\.]))@([A-Za-z0-9_]+)')
```

Prepare English dictionary for language detection

```
english_dict = enchant.Dict("en_US")
```

Use NLTK's tokenizer instead of Scikit's

```
tokenizer = nltk.casual.TweetTokenizer(preserve_case=False, reduce_len=True)
```

Prepare for viewing long text in CSVs and ones with really big and small numbers

```
pd.set_option("max_colwidth", 1000)
```

```
pd.options.display.float_format = "{:.4f}".format
```

Load the two data files

Only use tweets with at least 50% English words

Also, make the mentions of usernames, URLs, and hashtags generic

```
def load_data(filename, threshold=0.5):
    data = [(hashtags_pattern.sub("❶",
             urls_pattern.sub("❷",
             at_mentions_pattern.sub("❸",
             t["tweet_data"]["full_text"])))
             .replace("\u2018", "'")
             .replace("\u2019", "'")
             .replace("\u201c", "\"")
             .replace("\u201d", "\"")
             .replace("&quot;", "\"")
             .replace("&amp;", "&")
             .replace("&gt;", ">")
             .replace("&lt;", "<"))
            for t in json.load(open(filename))
            if t["tweet_data"]["user"]["lang"] == "en"
            and t["reply"]["user"]["lang"] == "en"]
    new_data = []
    for tweet in data:
        tokens = tokenizer.tokenize(tweet)
        english_tokens = [english_dict.check(token) for token in tokens]
        percent_english_words = sum(english_tokens)/len(english_tokens)
        if percent_english_words >= threshold:
            new_data.append(tweet)
    return new_data
```

```
subtweets_data = load_data("../data/other_data/subtweets.json")
```

```
non_subtweets_data = load_data("../data/other_data/non_subtweets.json")
```

Show examples

```
print("Subtweets dataset example:")
print(choice(subtweets_data))
```

```
Subtweets dataset example:
This little girls to weird for me pure retard
```

```
print("Non-subtweets dataset example:")
print(choice(non_subtweets_data))
```

```
Non-subtweets dataset example:
TESTED: "The Golf Infomercial" Wedge Test

Do golf infomercial wedges really work?

VIEW RESULTS: ❷ ❷
```

Find the length of the smaller dataset

```
smallest_length = len(min([subtweets_data, non_subtweets_data], key=len))
```

Cut both down to be the same length

```
subtweets_data = subtweets_data[:smallest_length]
```

```
non_subtweets_data = non_subtweets_data[:smallest_length]
```

```
print("Smallest dataset length: {}".format(len(subtweets_data)))
```

```
Smallest dataset length: 7837
```

Prepare data for training

```
subtweets_data = [(tweet, "subtweet") for tweet in subtweets_data]
```

```
non_subtweets_data = [(tweet, "non-subtweet") for tweet in non_subtweets_data]
```

Combine them

```
training_data = subtweets_data + non_subtweets_data
```

Create custom stop words to include generic usernames, URLs, and hashtags, as well as common English first names

```
names_lower = set([name.lower() for name in open("../data/other_data/first_names.txt").read().split("\n")])
```

```
generic_tokens = {"❶", "❷", "❸"}
```

```
stop_words = text.ENGLISH_STOP_WORDS | names_lower | generic_tokens
```

Build the pipeline

```
sentiment_pipeline = Pipeline([
    ("vectorizer", TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                   ngram_range=(1, 3),
                                   stop_words=stop_words)),
    ("classifier", MultinomialNB())
])
```

K-Folds splits up and separates out 10 training and test sets from the data, from which the classifier is trained and the confusion matrix and classification reports are updated

```
def confusion_matrices(training_data, num_folds=10):
    text_training_data = np.array([row[0] for row in training_data])
    class_training_data = np.array([row[1] for row in training_data])
    kf = KFold(n_splits=num_folds, random_state=42, shuffle=True)

    cnf_matrix_test = np.zeros((2, 2), dtype=int)
    cnf_matrix_train = np.zeros((2, 2), dtype=int)

    test_reports = []
    train_reports = []
    for i, (train_index, test_index) in enumerate(kf.split(text_training_data)):

        text_train, text_test = text_training_data[train_index], text_training_data[test_index]
        class_train, class_test = class_training_data[train_index], class_training_data[test_index]

        sentiment_pipeline.fit(text_train, class_train)

        predictions_test = sentiment_pipeline.predict(text_test)
        predictions_train = sentiment_pipeline.predict(text_train)

        cnf_matrix_test += confusion_matrix(class_test, predictions_test)
        cnf_matrix_train += confusion_matrix(class_train, predictions_train)

        print("Test Data Iteration {}:".format(i+1))

        test_report = classification_report(class_test, predictions_test, digits=4)
        test_reports.append(test_report)
        print(test_report)

        print(("Test Data Null Accuracy: {:.4f}\n"
               .format(max(pd.value_counts(pd.Series(class_test)))/float(len(class_test)))))
        print(("Test Data Accuracy: {:.4f}\n"
               .format(accuracy_score(class_test, predictions_test))))
        print("="*53)

        print("Train Data Iteration {}:".format(i+1))

        train_report = classification_report(class_train, predictions_train, digits=4)
        train_reports.append(train_report)
        print(train_report)

        print(("Train Data Null Accuracy: {:.4f}\n"
               .format(max(pd.value_counts(pd.Series(class_train)))/float(len(class_train)))))
        print(("Train Data Accuracy: {:.4f}\n"
               .format(accuracy_score(class_train, predictions_train))))
        print("="*53)

    def reports_mean(reports):
        reports_lists_of_strings = [report.split("\n") for report in reports]
        reports = [[[float(e) for e in report_string[2][16:].split()],
                    [float(e) for e in report_string[3][16:].split()],
                    [float(e) for e in report_string[5][16:].split()]]
                   for report_string in reports_lists_of_strings]
        mean_list = np.mean(np.array(reports), axis=0).tolist()
        print("              precision    recall  f1-score   support")
        print()
        print("non-subtweet     {0:.4f}    {1:.4f}    {2:.4f}      {3:d}".format(mean_list[0][0],
                                                                                 mean_list[0][1],
```

```
                                                                    mean_list[0][2],
                                                                    int(mean_list[0][3])))
        print("    subtweet     {0:.4f}    {1:.4f}    {2:.4f}      {3:d}".format(mean_list[1][0],
                                                                    mean_list[1][1],
                                                                    mean_list[1][2],
                                                                    int(mean_list[1][3])))
        print()
        print(" avg / total     {0:.4f}    {1:.4f}    {2:.4f}      {3:d}".format(mean_list[2][0],
                                                                    mean_list[2][1],
                                                                    mean_list[2][2],
                                                                    int(mean_list[2][3])))
        print()
        print("="*53)

    print("Test Data Averages Across All Folds:")
    reports_mean(test_reports)
    print("Train Data Averages Across All Folds:")
    reports_mean(train_reports)
    return {"Test": cnf_matrix_test, "Train": cnf_matrix_train}
```

```
%%time
cnf_matrices = confusion_matrices(training_data)
cnf_matrix_test = cnf_matrices["Test"]
cnf_matrix_train = cnf_matrices["Train"]
```

```
Test Data Iteration 1:
              precision    recall  f1-score   support

non-subtweet     0.7338    0.6431    0.6855       793
    subtweet     0.6758    0.7613    0.7160       775

 avg / total     0.7052    0.7015    0.7006      1568

Test Data Null Accuracy: 0.5057

Test Data Accuracy: 0.7015

=====================================================
Train Data Iteration 1:
              precision    recall  f1-score   support

non-subtweet     0.9907    0.9806    0.9856      7044
    subtweet     0.9808    0.9908    0.9858      7062

 avg / total     0.9857    0.9857    0.9857     14106

Train Data Null Accuracy: 0.5006

Train Data Accuracy: 0.9857

=====================================================
Test Data Iteration 2:
              precision    recall  f1-score   support

non-subtweet     0.6940    0.6324    0.6618       789
    subtweet     0.6584    0.7176    0.6867       779

 avg / total     0.6763    0.6747    0.6742      1568

Test Data Null Accuracy: 0.5032

Test Data Accuracy: 0.6747

=====================================================
Train Data Iteration 2:
              precision    recall  f1-score   support

non-subtweet     0.9908    0.9786    0.9847      7048
```

```
    subtweet     0.9789    0.9909    0.9849      7058

 avg / total     0.9848    0.9848    0.9848     14106

Train Data Null Accuracy: 0.5004

Train Data Accuracy: 0.9848

=====================================================
Test Data Iteration 3:
              precision    recall  f1-score   support

non-subtweet     0.7021    0.6866    0.6943       769
    subtweet     0.7047    0.7196    0.7121       799

 avg / total     0.7034    0.7034    0.7033      1568

Test Data Null Accuracy: 0.5096

Test Data Accuracy: 0.7034

=====================================================
Train Data Iteration 3:
              precision    recall  f1-score   support

non-subtweet     0.9869    0.9829    0.9849      7068
    subtweet     0.9829    0.9869    0.9849      7038

 avg / total     0.9849    0.9849    0.9849     14106

Train Data Null Accuracy: 0.5011

Train Data Accuracy: 0.9849

=====================================================
Test Data Iteration 4:
              precision    recall  f1-score   support

non-subtweet     0.7313    0.6355    0.6800       801
    subtweet     0.6651    0.7562    0.7077       767

 avg / total     0.6989    0.6945    0.6936      1568

Test Data Null Accuracy: 0.5108

Test Data Accuracy: 0.6945

=====================================================
Train Data Iteration 4:
              precision    recall  f1-score   support

non-subtweet     0.9907    0.9802    0.9854      7036
    subtweet     0.9805    0.9908    0.9856      7070

 avg / total     0.9856    0.9855    0.9855     14106

Train Data Null Accuracy: 0.5012

Train Data Accuracy: 0.9855

=====================================================
Test Data Iteration 5:
              precision    recall  f1-score   support

non-subtweet     0.7078    0.6560    0.6809       779
    subtweet     0.6828    0.7322    0.7067       788

 avg / total     0.6952    0.6943    0.6939      1567

Test Data Null Accuracy: 0.5029
```

```
Test Data Accuracy: 0.6943

==================================================
Train Data Iteration 5:
              precision    recall  f1-score   support

non-subtweet     0.9871    0.9829    0.9849      7058
    subtweet     0.9829    0.9871    0.9850      7049

 avg / total     0.9850    0.9850    0.9850     14107

Train Data Null Accuracy: 0.5003

Train Data Accuracy: 0.9850

==================================================
Test Data Iteration 6:
              precision    recall  f1-score   support

non-subtweet     0.6836    0.6583    0.6707       758
    subtweet     0.6906    0.7145    0.7023       809

 avg / total     0.6872    0.6873    0.6870      1567

Test Data Null Accuracy: 0.5163

Test Data Accuracy: 0.6873

==================================================
Train Data Iteration 6:
              precision    recall  f1-score   support

non-subtweet     0.9874    0.9846    0.9860      7079
    subtweet     0.9845    0.9873    0.9859      7028

 avg / total     0.9860    0.9860    0.9860     14107

Train Data Null Accuracy: 0.5018

Train Data Accuracy: 0.9860

==================================================
Test Data Iteration 7:
              precision    recall  f1-score   support

non-subtweet     0.7003    0.6285    0.6625       751
    subtweet     0.6876    0.7525    0.7185       816

 avg / total     0.6937    0.6930    0.6917      1567

Test Data Null Accuracy: 0.5207

Test Data Accuracy: 0.6930

==================================================
Train Data Iteration 7:
              precision    recall  f1-score   support

non-subtweet     0.9860    0.9852    0.9856      7086
    subtweet     0.9851    0.9859    0.9855      7021

 avg / total     0.9855    0.9855    0.9855     14107

Train Data Null Accuracy: 0.5023

Train Data Accuracy: 0.9855

==================================================
Test Data Iteration 8:
              precision    recall  f1-score   support
```

```
non-subtweet     0.7342    0.6429    0.6855       812
    subtweet     0.6612    0.7497    0.7027       755

 avg / total     0.6990    0.6943    0.6938      1567

Test Data Null Accuracy: 0.5182

Test Data Accuracy: 0.6943

=====================================================
Train Data Iteration 8:
              precision    recall  f1-score   support

non-subtweet     0.9906    0.9795    0.9850      7025
    subtweet     0.9799    0.9908    0.9853      7082

 avg / total     0.9852    0.9852    0.9852     14107

Train Data Null Accuracy: 0.5020

Train Data Accuracy: 0.9852

=====================================================
Test Data Iteration 9:
              precision    recall  f1-score   support

non-subtweet     0.7321    0.6429    0.6846       829
    subtweet     0.6472    0.7358    0.6886       738

 avg / total     0.6921    0.6867    0.6865      1567

Test Data Null Accuracy: 0.5290

Test Data Accuracy: 0.6867

=====================================================
Train Data Iteration 9:
              precision    recall  f1-score   support

non-subtweet     0.9919    0.9796    0.9857      7008
    subtweet     0.9801    0.9921    0.9861      7099

 avg / total     0.9860    0.9859    0.9859     14107

Train Data Null Accuracy: 0.5032

Train Data Accuracy: 0.9859

=====================================================
Test Data Iteration 10:
              precision    recall  f1-score   support

non-subtweet     0.7060    0.6799    0.6927       756
    subtweet     0.7116    0.7361    0.7236       811

 avg / total     0.7089    0.7090    0.7087      1567

Test Data Null Accuracy: 0.5175

Test Data Accuracy: 0.7090

=====================================================
Train Data Iteration 10:
              precision    recall  f1-score   support

non-subtweet     0.9870    0.9849    0.9859      7081
    subtweet     0.9848    0.9869    0.9859      7026

 avg / total     0.9859    0.9859    0.9859     14107

Train Data Null Accuracy: 0.5019
```

```
Train Data Accuracy: 0.9859

=====================================================
Test Data Averages Across All Folds:
              precision    recall  f1-score   support

non-subtweet     0.7125    0.6506    0.6798      783
    subtweet     0.6785    0.7376    0.7065      783

 avg / total     0.6960    0.6939    0.6933      1567

=====================================================
Train Data Averages Across All Folds:
              precision    recall  f1-score   support

non-subtweet     0.9889    0.9819    0.9854      7053
    subtweet     0.9820    0.9890    0.9855      7053

 avg / total     0.9855    0.9854    0.9854      14106

=====================================================
CPU times: user 1min 8s, sys: 1.76 s, total: 1min 10s
Wall time: 1min 12s
```

See the most informative features

How does "MultinomialNB.coef_" work?

```
def most_informative_features(pipeline, n=10000):
    vectorizer = pipeline.named_steps["vectorizer"]
    classifier = pipeline.named_steps["classifier"]

    class_labels = classifier.classes_

    feature_names = vectorizer.get_feature_names()

    top_n_class_1 = sorted(zip(classifier.coef_[0], feature_names))[:n]
    top_n_class_2 = sorted(zip(classifier.coef_[0], feature_names))[-n:]

    return {class_labels[0]: pd.DataFrame({"Log Probability": [tup[0] for tup in top_n_class_1],
                                           "Feature": [tup[1] for tup in top_n_class_1]}),
            class_labels[1]: pd.DataFrame({"Log Probability": [tup[0] for tup in reversed(top_n_class_2)],
                                           "Feature": [tup[1] for tup in reversed(top_n_class_2)]})}
```

```
%%time
most_informative_features_all = most_informative_features(sentiment_pipeline)
```

```
CPU times: user 1.34 s, sys: 39.3 ms, total: 1.38 s
Wall time: 1.38 s
```

```
most_informative_features_non_subtweet = most_informative_features_all["non-subtweet"]
```

```
most_informative_features_subtweet = most_informative_features_all["subtweet"]
```

```
final_features = most_informative_features_non_subtweet.join(most_informative_features_subtweet,
                                                             lsuffix=" (Non-subtweet)",
                                                             rsuffix=" (Subtweet)")
final_features.to_csv("../data/other_data/most_informative_features.csv")
final_features.head(25)
```

	Feature (Non-subtweet)	Log Probability (Non-subtweet)	Feature (Subtweet)	Log Probability (Subtweet)
0	! ! &	-12.6618	.	-7.5300
1	! ! (	-12.6618	,	-7.9193
2	! !)	-12.6618	"	-8.0928
3	! ! .	-12.6618	people	-8.3903
4	! ! 100	-12.6618	?	-8.4594
5	! ! 15	-12.6618	don't	-8.5588
6	! ! 3	-12.6618	like	-8.5889
7	! ! 5	-12.6618	just	-8.6754
8	! ! 8am	-12.6618	i'm	-8.6969
9	! ! :)	-12.6618	!	-8.9031
10	! ! ;)	-12.6618	it's	-8.9727
11	! ! absolutely	-12.6618	...	-9.0431
12	! ! amazing	-12.6618	you're	-9.0488
13	! ! ask	-12.6618	:	-9.0704
14	! ! awesome	-12.6618	know	-9.0928
15	! ! big	-12.6618	twitter	-9.1443
16	! ! bite	-12.6618	friends	-9.1650
17	! ! close	-12.6618	time	-9.2879
18	! ! collection	-12.6618	want	-9.2923
19	! ! come	-12.6618	u	-9.3004
20	! ! don't	-12.6618	really	-9.3518
21	! ! enter	-12.6618	shit	-9.3699
22	! ! epic	-12.6618	good	-9.4017
23	! ! extremely	-12.6618	think	-9.4155
24	! ! family	-12.6618	make	-9.4225

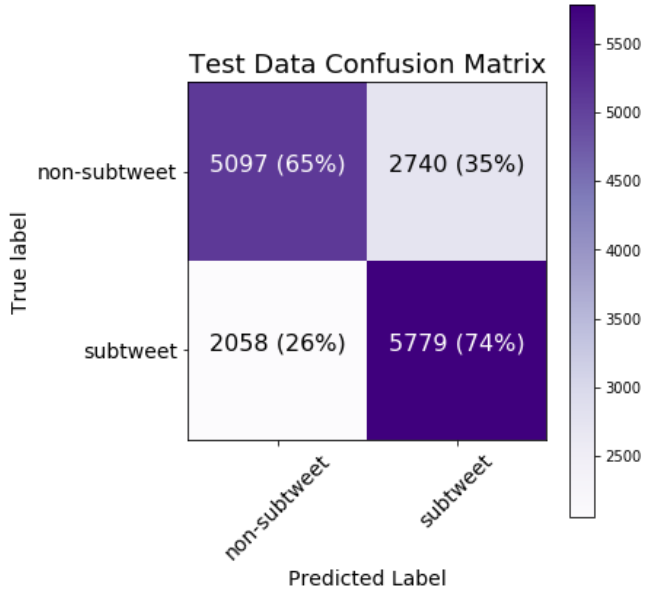
Define function for visualizing confusion matrices

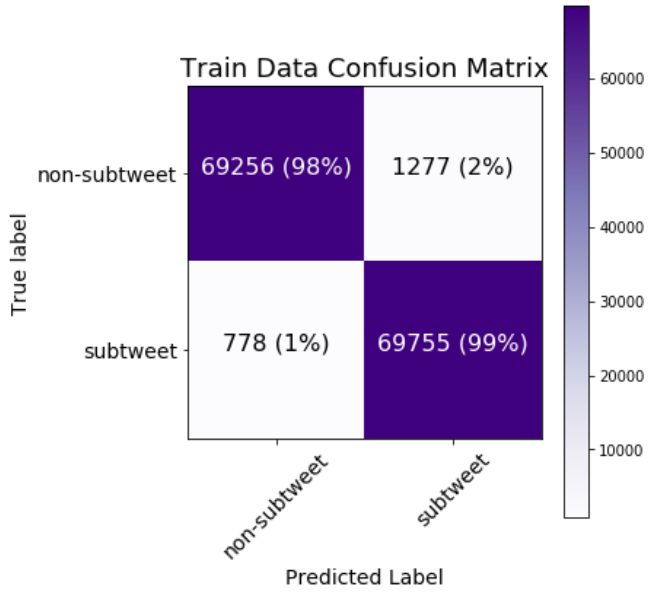
```
def plot_confusion_matrix(cm, classes=["non-subtweet", "subtweet"],
                          title="Confusion Matrix", cmap=plt.cm.Purples):

    cm_normalized = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation="nearest", cmap=cmap)
    plt.colorbar()

    plt.title(title, size=18)

    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45, fontsize=14)
    plt.yticks(tick_marks, classes, fontsize=14)

    thresh = cm.max() / 2.
```

```
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, "{} ({:.0%})".format(cm[i, j], cm_normalized[i, j]),
                 horizontalalignment="center", size=16,
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()

    plt.ylabel("True label", fontsize=14)
    plt.xlabel("Predicted Label", fontsize=14)
```

Show the matrices

```
np.set_printoptions(precision=2)

plt.figure(figsize=(6, 6))
plot_confusion_matrix(cnf_matrix_test, title="Test Data Confusion Matrix")

plt.figure(figsize=(6, 6))
plot_confusion_matrix(cnf_matrix_train, title="Train Data Confusion Matrix")

plt.show()
```





Update matplotlib style

```
plt.style.use("fivethirtyeight")
```

Save the classifier for another time

```
joblib.dump(sentiment_pipeline, "../data/other_data/subtweets_classifier.pkl");
```

Print tests for the classifier

```
def process_tweets_for_testing(filenames):
    dataframes = {}
    for filename in filenames:
        username = splitext(basename(filename))[0][:-7]
        dataframes[username] = {}

        user_df = pd.read_csv(filename).dropna()
        user_df["Text"] = user_df["Text"].str.replace(hashtags_pattern, "❶")
        user_df["Text"] = user_df["Text"].str.replace(urls_pattern, "❷")
        user_df["Text"] = user_df["Text"].str.replace(at_mentions_pattern, "❸")
        user_df["Text"] = user_df["Text"].str.replace("\u2018", "'")
        user_df["Text"] = user_df["Text"].str.replace("\u2019", "'")
        user_df["Text"] = user_df["Text"].str.replace("\u201c", "\"")
        user_df["Text"] = user_df["Text"].str.replace("\u201d", "\"")
        user_df["Text"] = user_df["Text"].str.replace("&quot;", "\"")
        user_df["Text"] = user_df["Text"].str.replace("&amp;", "&")
        user_df["Text"] = user_df["Text"].str.replace("&gt;", ">")
        user_df["Text"] = user_df["Text"].str.replace("&lt;", "<")

        predictions = sentiment_pipeline.predict_proba(user_df["Text"])[:, 1].tolist()
        user_df["SubtweetProbability"] = predictions

        dataframes[username]["all"] = user_df

        scores = user_df[["SubtweetProbability"]].rename(columns={"SubtweetProbability": username})

        dataframes[username]["scores"] = scores
        dataframes[username]["stats"] = scores.describe()

    return dataframes
```

Load the CSV files

```
filenames = glob("../data/data_for_testing/friends_data/*.csv")
```

```
%%time
dataframes = process_tweets_for_testing(filenames)
```

```
CPU times: user 9.09 s, sys: 153 ms, total: 9.24 s
Wall time: 9.52 s
```

Show a random table

```
chosen_username = choice(list(dataframes.keys()))
dataframes[chosen_username]["all"].sort_values(by="SubtweetProbability", ascending=False).head(5)
```

	Text	Date	Favorites	Retweets	Tweet ID	SubtweetProbability

	Text	Date	Favorites	Retweets	Tweet ID	SubtweetProbability
2092	I hate when people overuse emojis	2015-06-26 13:01:35	0	0	614478624197091328	0.8579
2137	Also you don't need to resort to social media 24/7 to complain about your very privileged life ¯_(ツ)_/¯	2015-06-15 17:24:46	1	0	610558590278070272	0.8443
2151	When I try to be supportive and caring I get ignored and then I'm told I'm not being supportive or caring ¯_(ツ)_/¯	2015-06-13 08:44:07	0	0	609702789896372224	0.8366
2134	What he doesn't know (unless he stalks my twitter which I know he does) is that I have fake accounts following all his social media	2015-06-15 17:26:41	0	0	610559074820861953	0.8177
1510	If you don't have tweet notifications turned on for me are we really friends	2016-07-14 14:21:21	1	0	753655639465922560	0.8076

Prepare statistics on tweets

```
tests_df = pd.concat([df_dict["scores"] for df_dict in dataframes.values()], ignore_index=True)
```

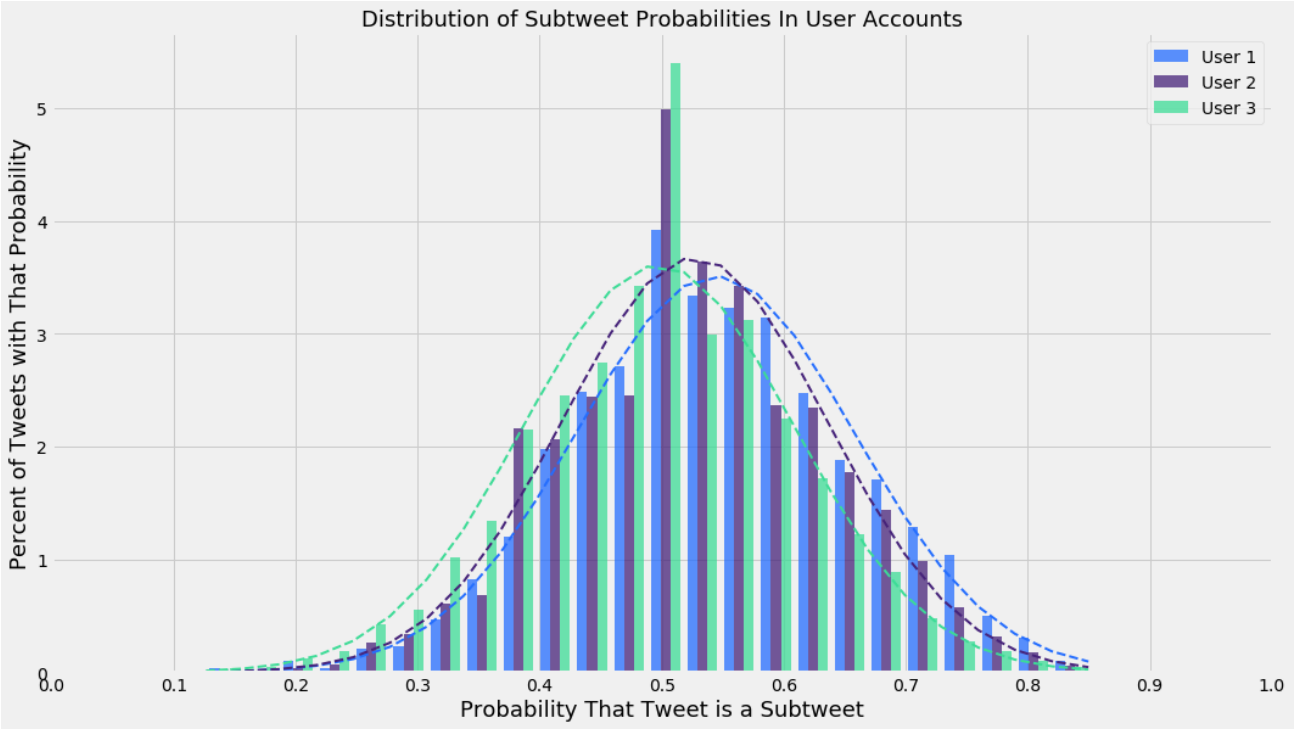
```
tests_df.describe()
```

	adhaardesai	akrapf96	generatedtext	gothodile	juliaeberry	kayleesue	keithohara
count	621.0000	2640.0000	2066.0000	3488.0000	4356.0000	1939.0000	1169.0000
mean	0.4996	0.5086	0.5438	0.5270	0.5187	0.4976	0.4388

	adhaardesai	akrapf96	generatedtext	gothodile	juliaeberry	kayleesue	keithohara
std	0.1059	0.1150	0.1136	0.1086	0.1023	0.1106	0.0981
min	0.1981	0.0953	0.1266	0.1626	0.1522	0.0566	0.1497
25%	0.4291	0.4304	0.4669	0.4538	0.4492	0.4260	0.3733
50%	0.4971	0.5037	0.5417	0.5217	0.5180	0.4981	0.4379
75%	0.5670	0.5847	0.6213	0.5982	0.5843	0.5669	0.5016
max	0.8457	0.8579	0.8497	0.8749	0.8674	0.8766	0.8157

Plot a histogram with three random users

```
random_choices = sample(list(dataframes.values()), 3)
scores = [df_dict["scores"][df_dict["scores"].columns[0]].tolist()
          for df_dict in random_choices]

fig = plt.figure(figsize=(16, 9))
ax = fig.add_subplot(111)

n, bins, patches = ax.hist(scores,
                           bins="scott",
                           color=["#256EFF", "#46237A", "#3DDC97"],
                           density=True,
                           label=["User 1", "User 2", "User 3"],
                           alpha=0.75)

stats = [df_dict["stats"][df_dict["stats"].columns[0]].tolist()
         for df_dict in random_choices]

line_1 = scipy.stats.norm.pdf(bins, stats[0][1], stats[0][2])
ax.plot(bins, line_1, "--", color="#256EFF", linewidth=2)

line_2 = scipy.stats.norm.pdf(bins, stats[1][1], stats[1][2])
ax.plot(bins, line_2, "--", color="#46237A", linewidth=2)

line_3 = scipy.stats.norm.pdf(bins, stats[2][1], stats[2][2])
ax.plot(bins, line_3, "--", color="#3DDC97", linewidth=2)

ax.set_xticks([float(x/10) for x in range(11)], minor=False)
ax.set_title("Distribution of Subtweet Probabilities In User Accounts", fontsize=18)
ax.set_xlabel("Probability That Tweet is a Subtweet", fontsize=18)
ax.set_ylabel("Percent of Tweets with That Probability", fontsize=18)

ax.legend()

plt.show()
```



Plot a histogram with all of them

First, get some statistics

```
new_tests_df = pd.concat([df_dict["scores"].rename(columns={df_dict["scores"].columns[0]:"SubtweetProbability"})
                          for df_dict in dataframes.values()], ignore_index=True)

new_tests_df_stats = new_tests_df.describe()
```

Then view them

```
new_tests_df_stats
```

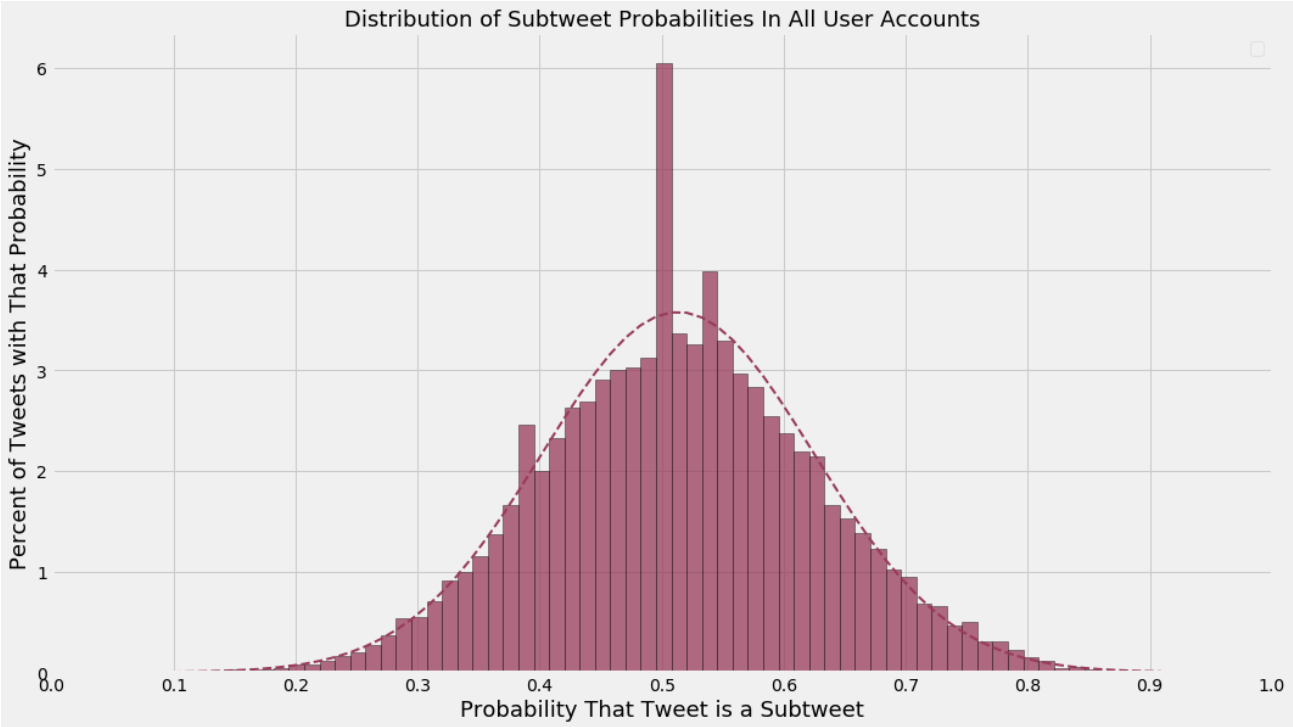
	SubtweetProbability
count	28632.0000
mean	0.5133
std	0.1115
min	0.0566
25%	0.4385
50%	0.5093
75%	0.5860
max	0.9091

Now plot

```
fig = plt.figure(figsize=(16, 9))
ax = fig.add_subplot(111)

n, bins, patches = ax.hist(new_tests_df["SubtweetProbability"].tolist(),
```

```
                            bins="scott",
                            color="#983B59",
                            edgecolor="black",
                            density=True,
                            alpha=0.75)

line = scipy.stats.norm.pdf(bins, new_tests_df_stats["SubtweetProbability"][1],
                               new_tests_df_stats["SubtweetProbability"][2])

ax.plot(bins, line, "--", color="#983B59", linewidth=2)


ax.set_xticks([float(x/10) for x in range(11)], minor=False)
ax.set_title("Distribution of Subtweet Probabilities In All User Accounts", fontsize=18)
ax.set_xlabel("Probability That Tweet is a Subtweet", fontsize=18)
ax.set_ylabel("Percent of Tweets with That Probability", fontsize=18)

ax.legend()

plt.show()
```

```
No handles with labels found to put in legend.
```



Statisitics on training data

Remove mentions of usernames for these statistics

```
training_data = [(tweet[0]
                  .replace("❶", "")
                  .replace("❷", "")
                  .replace("❸", "")) for tweet in training_data]
```

Lengths

```
length_data = [len(tweet) for tweet in training_data]
```

```
length_data_for_stats = pd.DataFrame({"Length": length_data, "Tweet": training_data})
```

```
# length_data_for_stats = length_data_for_stats[length_data_for_stats["Length"] <= 280]
```

```
# length_data_for_stats = length_data_for_stats[length_data_for_stats["Length"] >= 5]
```

```
length_data = length_data_for_stats.Length.tolist()
```

Top 5 longest tweets

```
length_data_for_stats.sort_values(by="Length", ascending=False).head()
```

	Length	Tweet
8887	281	This Tweet does not endorse the use of Nazi Symbols in any form! I think the image which has been published on social media and MSM is a day or two old. It conjures up strong emotions for many people, My question is simple what meaning do you think is being conveyed by the image?
2198	281	I need to learn how to do this. I ask "how can I help" a lot because I genuinely want to make things better for friends , but this *can* put a burden back upon those who are suffering. Sometimes it may be best to just have exuberant and fearless compassion the same way a pet does
1531	281	hi! I'm not normally v personal like this and I probably won't be at least for a v long time but I thought I'd share this \nwhile I was scrolling on Twitter today I had like a sudden impulse to just dump all my thoughts about what id been reading and seeing and so far it actually-
10533	281	Some people are undecided about testing on animals. Understandable. There's so much propaganda and secrecy about it. Here's a quick test though, & you're answer should tell you. What would you do if some man came to your house & squirted disinfectant in your beautiful dog's eyes?
10521	281	Enthralled by Raja Shiv Chhatrapati, a well mounted magnum opus on life of the Maratha warrior at Red Fort. Vividly brought out his philosophies, struggles, inspiration from mother Jijayee & penchant for gender equality through well conceived music, dance & dialogues. A must see!

Top 5 shortest tweets

```
length_data_for_stats.sort_values(by="Length", ascending=True).head()
```

	Length	Tweet
7699	1	A
3473	2	no
5896	2	uh
6676	2	i-
2038	2	Ha

Tweet length statistics

```
length_data_for_stats.describe()
```

	Length
count	15674.0000
mean	106.8089
std	73.8680
min	1.0000
25%	48.0000
50%	87.0000
75%	150.0000
max	281.0000

Punctuation

```
punctuation_data = [len(set(punctuation).intersection(set(tweet))) for tweet in training_data]
```

```
punctuation_data_for_stats = pd.DataFrame({"Punctuation": punctuation_data, "Tweet": training_data})
```

Top 5 most punctuated tweets

```
punctuation_data_for_stats.sort_values(by="Punctuation", ascending=False).head()
```

	Punctuation	Tweet
8957	11	Going to go ahead and crown myself the absolute emperor of finding things on menus that sound interesting, deciding I would like to try them, then being told "I'm sorry sir, that's actually not available..."\n\n[then why the @#$% is it ON YOUR MENUUUUUUUU--]
6725	9	4-yo: DADDEEEEEE!? LET'S PLAY!\nMe: Ok, baby. \n4yo: you play w/ her. put a dress on her DADDEEEEEE. \nMe: Ok. *puts doll in dollhouse*\n4yo: SHE DOESN'T GO THERE!!
11718	9	Self-employed people: have you ever turned to social media to call out a client who is many weeks/months delinquent on a payment? \n(Obviously, you're probably burning a bridge with that move, but if they don't pay...)
13365	9	Billboard Hot 100: (-3) Tell Me You Love Me, [19 weeks]. *peak: *
11845	9	Tucker Carlson Tonight & TFW you're asking about America\nbut you're scolded it's really about Israel ...\n \nTucker: "What is the American national security interest ... in Syria?"\n\nSen. Wicker(R): "Well, if you care about Israel ..." \n\nThat was the exact question & answer\nShocking

Tweets punctuation statistics

```
punctuation_data_for_stats.describe()
```

	Punctuation
count	15674.0000
mean	1.9168

	Punctuation
std	1.5787
min	0.0000
25%	1.0000
50%	2.0000
75%	3.0000
max	11.0000

Stop words

```
stop_words_data = [len(set(stopwords.words("english")).intersection(set(tweet.lower())))
                   for tweet in training_data]
```

```
stop_words_data_for_stats = pd.DataFrame({"Stop words": stop_words_data, "Tweet": training_data})
```

Top 5 tweets with most stop words

```
stop_words_data_for_stats.sort_values(by="Stop words", ascending=False).head()
```

	Stop words	Tweet
0	8	I don't yet have adequate words to do so, but someday I wanna write about the beautiful dance which happens in Google docs between a writer & a good editor working simultaneously towards a deadline. When it's working, it's a beautiful dance—though no one really sees it.
9063	8	Honestly yea i fucked up but all of you are trash asf and your opinions mean nothing to me because mother fucker i can fix shit but yall are to close minded to see.
9035	8	The role of DAG Rod Rosenstein will be an Oscar winner in the future film about the Trump presidency. I'd like the story of the first few months to be told through the eyes of the bewildered Sean Spicer.
9038	8	Done watching 'Hacksaw Ridge'. If there's one thing I learned from that movie, it is simply, Have Faith in God.
9039	8	I feel people who can't celebrate or at the very least respect Cardi B's success have never watched the grind from the ground up. They can't understand that her work ethic has gotten her where she is now. You don't have to stand for what's she's about but she's worked for it

Top 5 tweets with fewest stop words

```
stop_words_data_for_stats.sort_values(by="Stop words", ascending=True).head()
```

	Stop words	Tweet
3632	0	...
8290	0	24
11925	0	FUCK
10940	0	78 ... !

	Stop words	Tweet
1796	0	fuck u

Tweets stop words statistics

```
stop_words_data_for_stats.describe()
```

	Stop words
count	15674.0000
mean	7.1515
std	1.3116
min	0.0000
25%	7.0000
50%	8.0000
75%	8.0000
max	8.0000

Unique words

```
unique_words_data = [len(set(tokenizer.tokenize(tweet))) for tweet in training_data]
```

```
unique_words_data_for_stats = pd.DataFrame({"Unique words": unique_words_data, "Tweet": training_data})
```

```
# unique_words_data_for_stats = unique_words_data_for_stats[unique_words_data_for_stats["Unique words"] >= 2]
```

```
unique_words_data = unique_words_data_for_stats["Unique words"].tolist()
```

Top 5 tweets with most unique words

```
unique_words_data_for_stats.sort_values(by="Unique words", ascending=False).head()
```

	Tweet	Unique words
13936	GIVE AWAY!\n\nThe rules are really easy, all you have to do is :\n1. Must be following me (i check) \n2. RT and fav this tweet\n3. tag your mutuals/anyone\n4. only 1 winner! \n5. i ship worldwide;) \n\nit ends in 8th May 2018 or when this tweet hit 2k RT and like!\n\nGood luck! ❤️	59
4881	got into a tepid back nd forth w/ a uknowwhoAJ+columnist bc i said they steal their "hot takes" from blk twitter & alike. wallahi my bdeshi ass did not sign up 4 this app to be called asinine by a 30yrold pakistani whos whole politics is Post Colonial Memes for Oriental Minded T-	57
7013	Crazy how wrong u can be about someone. A girl I graduated w/ was always doing drugs& got pregnant at 16. I assumed she'd end up being a loser but it turn out she now has 4 beautiful kids& is making over $4,500/month just off of child support payments from the 3 different dads	57

	Tweet	Unique words
4992	Got into an argument w/ someone I went to HS w/ & I would js like to repeat again tht I cannot wait to stunt on all the ppl who were bitches to me in HS @ our reunion. Catch me rollin up w/ my sexy ass gf, a nice car, a bomb body & the career of my dreams as a big fuck u to them	55
11542	Thought I'd bring this back... and no, I'm not talking about myself here. I wish just once I'd be so bored with my life that I'd find the time to bash people/celebs I don't like.. I mean if I despise someone THAT much, why still watch his/her every move? 🤦‍♀️	55

Top 5 tweets with fewest unique words

```
unique_words_data_for_stats.sort_values(by="Unique words", ascending=True).head()
```

	Tweet	Unique words
6106	Annoying	1
2525	Bitch	1
12087	Chandler	1
14559	Yes yes yes yes yes yes	1
14442	Hello\n	1

Tweets unique words statistics

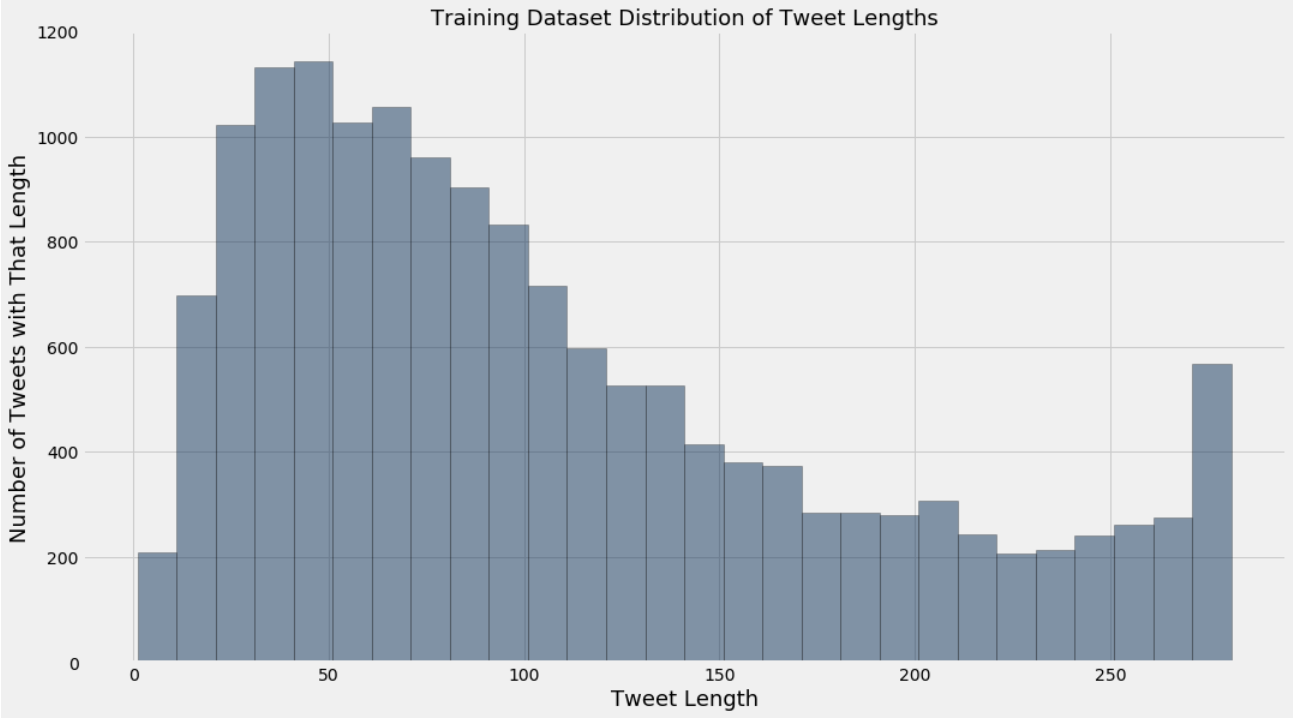
```
unique_words_data_for_stats.describe()
```

	Unique words
count	15674.0000
mean	19.2412
std	11.9298
min	1.0000
25%	10.0000
50%	17.0000
75%	27.0000
max	59.0000

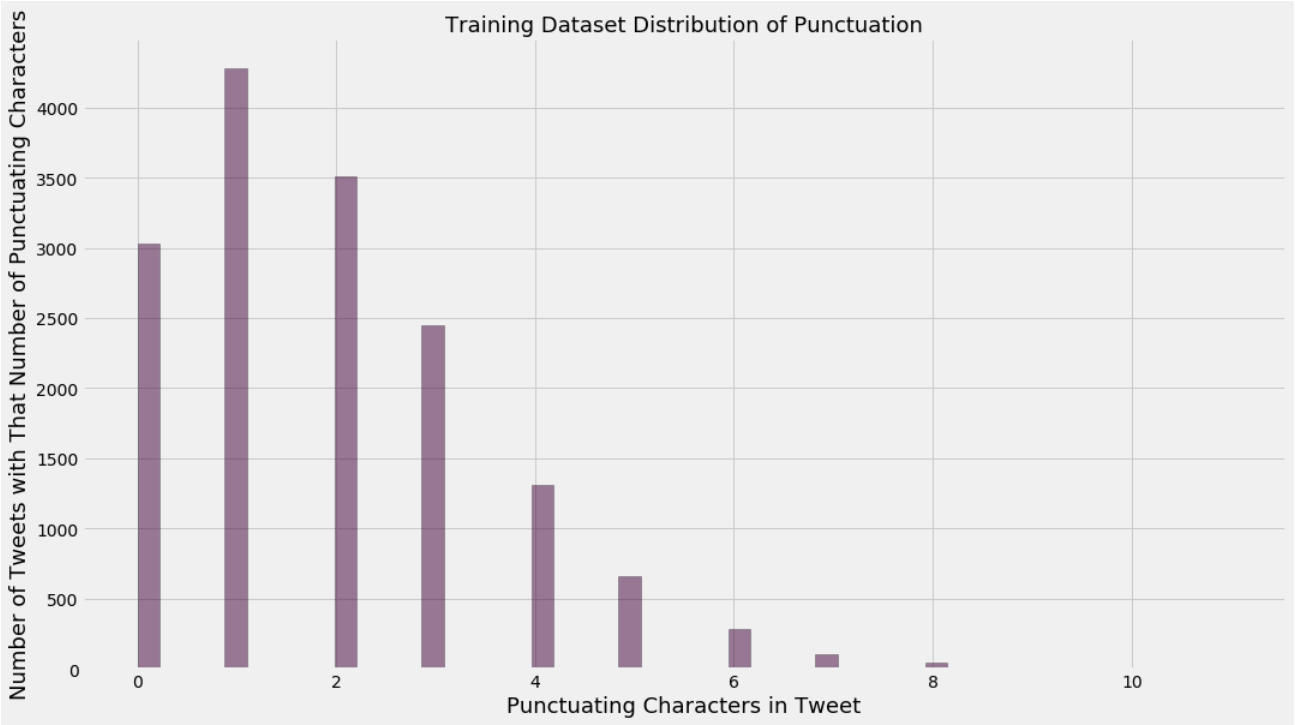
Plot them

```
length_mean = length_data_for_stats.describe().Length[1]
length_std = length_data_for_stats.describe().Length[2]

fig = plt.figure(figsize=(16, 9))
ax = fig.add_subplot(111)

n, bins, patches = ax.hist(length_data,
                           bins="scott",
                           edgecolor="black",
                           # density=True,
                           color="#12355b",
```

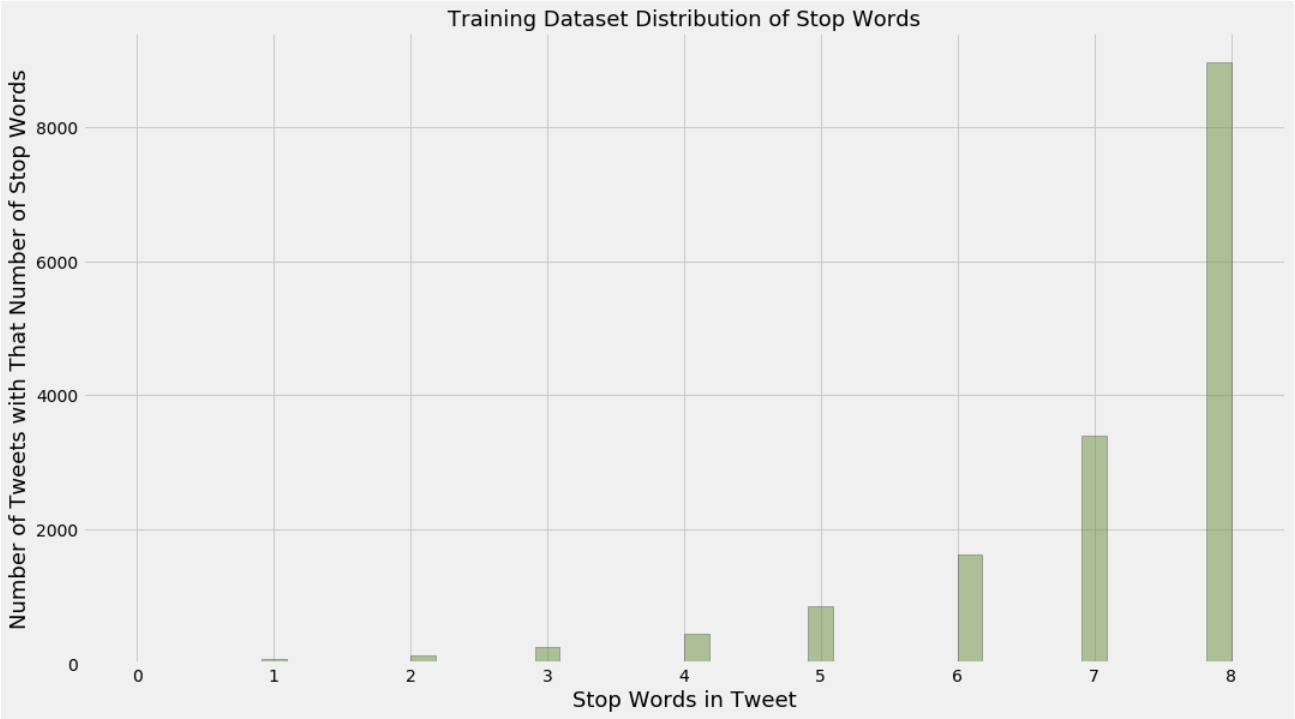
```
                           alpha=0.5)

# length_line = scipy.stats.norm.pdf(bins, length_mean, length_std)
# ax.plot(bins, length_line, "--", linewidth=3, color="#415d7b")

ax.set_title("Training Dataset Distribution of Tweet Lengths", fontsize=18)
ax.set_xlabel("Tweet Length", fontsize=18);
ax.set_ylabel("Number of Tweets with That Length", fontsize=18);

plt.show()
```



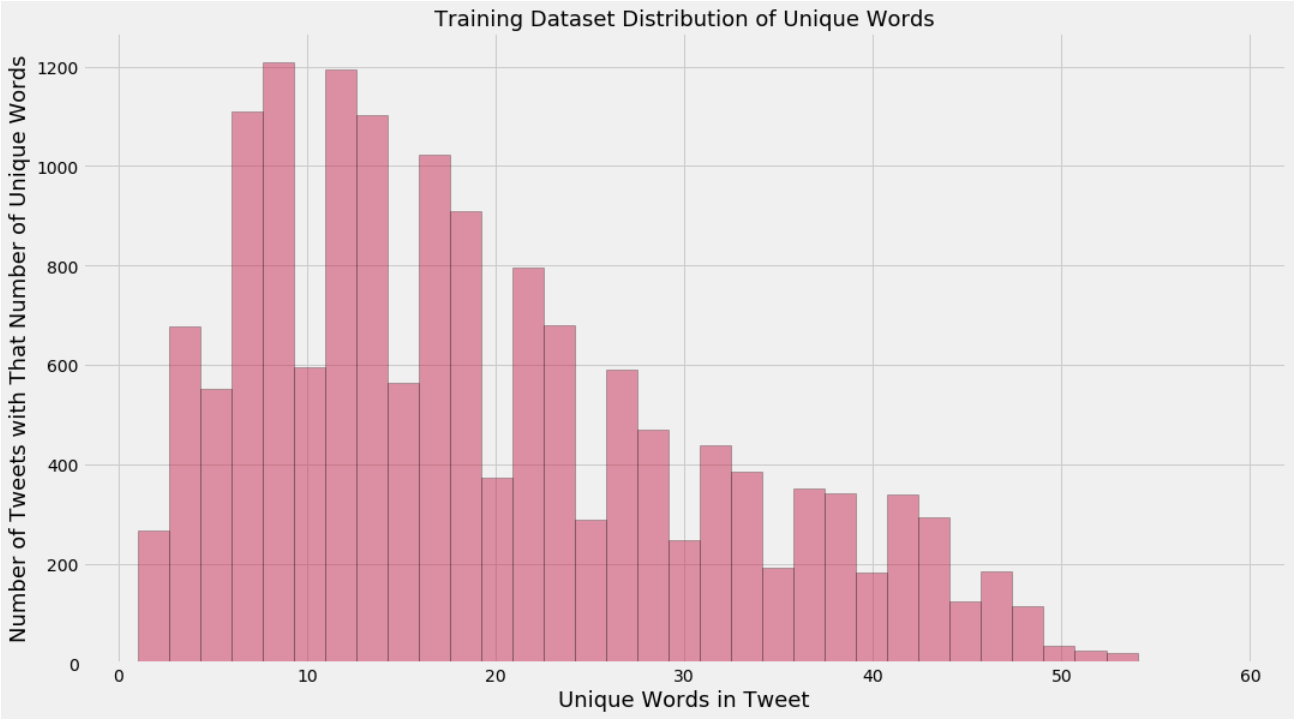

```
punctuation_mean = punctuation_data_for_stats.describe().Punctuation[1]
punctuation_std = punctuation_data_for_stats.describe().Punctuation[2]

fig = plt.figure(figsize=(16, 9))
ax = fig.add_subplot(111)

n, bins, patches = ax.hist(punctuation_data,
                           bins="scott",
                           edgecolor="black",
                           # density=True,
                           color="#420039",
                           alpha=0.5)

# punctution_line = scipy.stats.norm.pdf(bins, punctuation_mean, punctuation_std)
# ax.plot(bins, punctution_line, "--", linewidth=3, color="#673260")

ax.set_title("Training Dataset Distribution of Punctuation", fontsize=18)
ax.set_xlabel("Punctuating Characters in Tweet", fontsize=18)
ax.set_ylabel("Number of Tweets with That Number of Punctuating Characters", fontsize=18)

plt.show()
```




```
stop_words_mean = stop_words_data_for_stats.describe()["Stop words"][1]
stop_words_std = stop_words_data_for_stats.describe()["Stop words"][2]

fig = plt.figure(figsize=(16, 9))
ax = fig.add_subplot(111)

n, bins, patches = ax.hist(stop_words_data,
                           bins="scott",
                           edgecolor="black",
                           # density=True,
                           color="#698f3f",
                           alpha=0.5)

# stop_words_line = scipy.stats.norm.pdf(bins, stop_words_mean, stop_words_std)
# ax.plot(bins, stop_words_line, "--", linewidth=3, color="#87a565")

ax.set_title("Training Dataset Distribution of Stop Words", fontsize=18)
ax.set_xlabel("Stop Words in Tweet", fontsize=18)
ax.set_ylabel("Number of Tweets with That Number of Stop Words", fontsize=18)

plt.show()
```




```
unique_words_mean = unique_words_data_for_stats.describe()["Unique words"][1]
unique_words_std = unique_words_data_for_stats.describe()["Unique words"][2]

fig = plt.figure(figsize=(16, 9))
ax = fig.add_subplot(111)

n, bins, patches = ax.hist(unique_words_data,
                           bins="scott",
                           edgecolor="black",
                           # density=True,
                           color="#ca2e55",
                           alpha=0.5)

# unique_words_line = scipy.stats.norm.pdf(bins, unique_words_mean, unique_words_std)
# ax.plot(bins, unique_words_line, "--", linewidth=3, color="#d45776")

ax.set_title("Training Dataset Distribution of Unique Words", fontsize=18)
ax.set_xlabel("Unique Words in Tweet", fontsize=18)
ax.set_ylabel("Number of Tweets with That Number of Unique Words", fontsize=18)

plt.show()
```


Training Dataset Distribution of Unique Words
Number of Tweets with That Number of Unique Words
0
200
400
600
800
1000
1200
0
10
20
30
40
50
60
Unique Words in Tweet