Predicting How People Vote From How They Tweet

A Mathematics Research Project submitted to
The Master of Arts in Teaching Program
of
Bard College

by
Rao Vinnakota

Annandale-on-Hudson, New York
May, 2019

Abstract

The project observes tweets concerning the midterm elections and uses the sentiment drawn from those tweets to predict voting trends.

Contents

1
Introduction

Donald Trump stunned the world when he was elected President of the United States. Trump ran an unorthodox campaign, specifically in regards to his use of social media. Trump used accounts on Twitter and Facebook as a voice to attack candidates, disseminate campaign information, and engage with voters.

Public polling in 2016 didn't predict the upset. In fact, it didn't come particularly close. The latest tracking polls before election day showed Clinton in the lead with a margin of anywhere from +1 to +6. Most pollsters believed that Trump would sweep strictly conservative areas, but Clinton would every state that really mattered. Obviously, that didn't happen

If you look through tweets following the election, Trump supporters claimed that victory was inevitable, and that Trump's social media interaction demonstrated it. true? Was Trump's potential victory present on social media for people to see?

That is idea my senior project explore. Can you mathematically evaluate opinions on social media to predict voting trends. I will gather tweets pertaining to the 2018 midterm elections and use the sentiment expressed in those tweets and attempt to predict voting results.

1.1 Political Polling - A Short Overview

The earliest iteration of polling was a straw poll conducted in Harrisburg, Pennsylvania which showed Andrew Jackson leading John Quincy Adams in the 1824 Presidential Election. Journalists from The *Harrisburg Pennsylvanian* asked local voters which they preferred. Jackson winning both the polls and the election created legitimized the poll. Polls continued to be local until the early 20th century when Literary Digest sent out postcards to reader asking for their pick of the presidential election. The Literary Digest correctly predicted 5 straight elections and opinion polls reached the national stage.

In the advertising boom of post-World War II, most polling shifted to telecommunications. With telephones, there was an ability to reach a wider audience, ask deeper questions. Pollsters could now build a profile of a voter, and forecast more accurate results. Accompanying the advance in polling was a focus on sampling. Pollsters couldn't call every single house in the United States. Instead, groups of people or "samples" that would accurately represent the country were called. Responses from the sample were extrapolated to cover the whole.

Today, the digital age and social media have made polling's barrier of entry low. A short scroll through Twitter will show hundreds of "opinion polls". Youtube videos' likes and dislikes function as an informal poll. The Algorithm will often promote videos that are more "well-liked". But, professional political polling is still done along the same lines. A population is sampled, called, and profiled. Profiling technique has improved. Today, forecasters like 538's Nate Silver, will build complex profiles of voters in various regions. But, the source of the information and the method of obtaining the information remains the same.

1.2 Social Media, Twitter, and Politics

The internet's first foray into politics was in 2008. Barack Obama established that it was acceptable for a candidate to leverage the internet as a resource, using the internet as a tool for both message and money. The website MyBarackObama.com helped Obama set records in grassroot donations and mobilizations.

Social websites from the forums of the early 2000s to social media platforms that are popular today are the most visited websites on the internet. These forums encourage conversation, including the discussion of politics. Especially for younger voters who don't own any landlines, these forums are the few places they express political opinions.

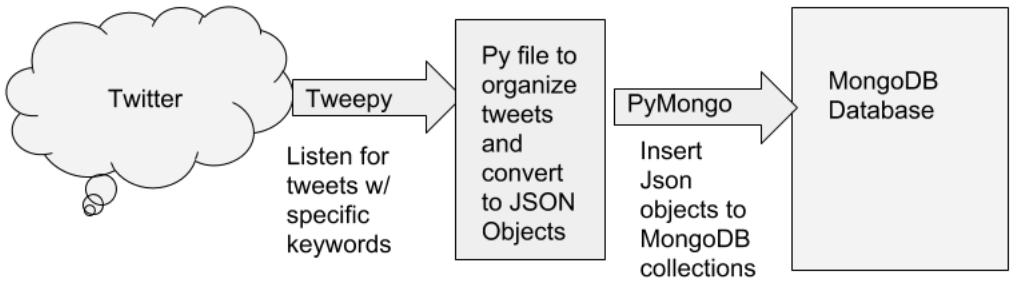
This project will specifically look at political opinions expressed on the microblogging platform Twitter. What sets it apart from blogs or other political forums is their ease of use. The longform opinions that are on forums and blogs are short and un-edited in Tweets. Twitter's mandatory message format - less than two hundred and eighty characters - makes sure posts are similar in style, and are far easier to analyze.

1.3 Big Data

A decade ago, attempting to gather, clean, and analyze millions of tweets would have been a fool's errand. It's with big data techniques and packages that this project is now possible. This presents my experiment with several advantages. Unlike traditional pollsters, I won't need to design a sample, but simply collect every tweet I can find. I also will capture a different audience.

2
Building a Data Pipeline

Gathering a large and varied data set is essential to this project. The most efficient method is to create a data pipeline that runs autonomously. This particular pipeline made use of the Twitter API to gather Tweets and a cloud server running MongoDB to store them. The pipeline creation process can be summarized as follows:



1. Building a tweet streamer to "listen" to tweets based on keywords

2. Preprocessing the tweets

3. Storing processed tweets in a database.

The pipeline was built in python3. The code for it can be found in Appendix A

2.1 Streaming Tweets

There are two ways to gather tweets using Tweepy (the Python Twitter API). The first is to query a profile or a search term. This method is not preferred for several reasons. First, Twitter has put a hard cap on the number of tweets each query can return at one time. This means that querying will be in-exact and time consuming loop. Next, Twitter's search function will naturally promote tweets that have more engagement. This means that tweets with more likes and retweets, comments, and followers will get preference, which results in a biased data set.

The other way to gather tweets is to stream tweets. A Tweepy stream monitors live twitter content and returns all tweets that contain a given set of keywords. Streaming has the disadvantage of gathering only live tweets - it does not search tweets from before it started running. But, streaming has the advantage of gathering any tweets that meet the keyword requirement, irrelevant of engagement.

The code written to stream tweets can be found in Appendix A. The streamer is a abstracted in the Twitter API. To use it, the user must fill in several methods. The methods defined here are on_connect, on_error, and on_data. The first two are trivial. On_connect is used to connect, and confirms once the stream is live. If the stream crashes - which can happen because of several reasons - on_error will return exactly which error caused the stream to malfunction.

The last method on_data is where the tweets are received and processed. Each tweet is received as a dictionary. The keys are attributes of the tweet (i.e. date and time, text, user, etc.). From there any processing is up to the user; the tweet's text can be printed, the tweets can be prepared for entry to a data base, etc. More on that later.

2.2 Databases

Before describing how the database for this project works, it's important to have a basic grasp of databases. A database is a structure that stores data, usually in the form of documents or objects.

2.2.1 MongoDB and NoSQL

For years most databases had been operated and queried using SQL or Structured Query Language. This language worked off the idea that documents were stored in tables. It's organization made it incredibly efficient. A drawback of the efficiency was that queries to the databases ended up being unnecessarily complicated.

NoSQL databases - as inferred from the name.- databases that aren't operated using SQL. They also lack the table structure of SQL databases. MongoDB is an open-source database that stores objects as JSONS. It looks something like this;

Each database will contain a variety of collections. These collections are where the objects themselves are actually stored. Having multiple collection allows the user to organize their database. Each object also has a series of attributes. These attributes work like a tagging system and are used as search terms.

2.3 Adding Tweets to a MongoDB

The code written in Appendix A shows the actions for adding the streamed tweets to a MongoDB database. It uses commands provided by the Python MongoDB API known as PyMongo.

All mongo actions take place in the on_data method. First, the mongo client is initialized. The client is what interacts with the database. The tweet which is stored as a dictionary is converted to json using the json python package. Then, tweet is added to a collection in the database.

Appendix A
Data Collection Code

The following code is used to gather tweets for various elections. To change which election is being monitored, the keywords need to be changed. To change the storage destination, change the past for the variable MONGO_HOST.

```
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
from pymongo import MongoClient
import json
import sys



MONGO_HOST= 'mongodb://localhost/miscdb'  # assuming you have mongoDB installed locally
                                          # and a database called 'twitterdb'
ckey = "pF5W6BaHFteEYdkq38cgZ755g"
csecret = "AwnBHQTmvVr6cMSYKm89vQOXMO3sPyoGvhXMaow3zlg2G74vLq"
atoken = "2646419088-gtzEn525GUUtTlwpegdZGd8dGLWl8m2IGq9jk9y"
asecret = "iQbg8ZfJ8KkI06n0lwkCVVprtkzsIp6h1RMXP5LN3Z6o2"

class StreamListener(StreamListener):
    #This is a class provided by tweepy to access the Twitter Streaming API.

    def on_connect(self):
        # Called initially to connect to the Streaming API
        print("You are now connected to the streaming API.")

    def on_error(self, status_code):
        # On error - if an error occurs, display the error / status code
        print('An Error has occured: ' + repr(status_code))
        return False

    def on_data(self, data):
        #This is the meat of the script...it connects to your mongoDB and stores the
    tweet
        try:
            client = MongoClient(MONGO_HOST)
```

```
33
34              # Use twitterdb database. If it doesn't exist, it will be created.
35              db = client.miscdb
36
37              # Decode the JSON from Twitter
38              datajson = json.loads(data)
39
40              #grab the 'created_at' data from the Tweet to use for display
41              created_at = datajson['created_at']
42
43              #print out a message to the screen that we have collected a tweet
44              print("Tweet collected at " + str(created_at) + " " + datajson['text'])
45
46              #insert the data into the mongoDB into a collection called twitter_search
47              #if twitter_search doesn't exist, it will be created.
48              db.misc.insert(datajson)
49          except Exception as e:
50             print(e)
51
52
53 auth = OAuthHandler(ckey, csecret)
54 auth.set_access_token(atoken, asecret)
55 #input = input("Which districts would you like to observe? Please separate districts by a
       comma")
56 #keywords = list(input.split(','))
57 #keywords = ["Antonio Delgado", "John Faso", "NY19", "@DelgadoforNY19", "@RepJohnFaso"]
58 keywords = ["#midterms", "#vote", "#bluewave", "#redwave", "#democrats", "#elections", "#
       gop", "#usa"]
59 twitterStream = Stream(auth, StreamListener())
60 twitterStream.filter(track=keywords)
61 '''
62
63 '''
```