Mining and Summarizing Public Opinion About the United States Southern Border Wall with Twitter Data

By
Dave Poortvliet

April, 2019

Mining and Summarizing Public Opinion About the United States Southern Border Wall with Twitter Data

By
Dave Poortvliet

A project submitted in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Information Systems

At
Grand Valley State University
April, 2019

Xinli Wang, PhD April 20, 2019

Table of Contents

Abstract

With heated debates in the media about the proposed southern border wall in the United States I wanted to better understand how the people living along the US/Mexico border really felt about the wall. My approach to answering this question was to mine large sets of twitter data from the cities along the border and analyze sentiment to understand if opinions on the wall were favorable, unfavorable or neutral. This paper will analyze the four different algorithms I used to determine sentiment and identify the benefits and drawbacks of each.

Introduction

The current president of the United States, Donald Trump, has proposed building a 2,000-mile wall along the southern border of the United States. To accomplish this he requested $5.6 billion from Congress. This proposal and request was debated in the news for several months and was the result of a 35-day government shutdown. Congress, the president, and news pundits all discussed their support or opposition to the wall, but I was curious how the individuals living along the southern border really felt. Their voices are important to hear since they live in the communities that will be impacted the most by the construction of a new wall.

My goal was to capture the sentiment from these individuals about the proposed border wall by mining large sets of Twitter data from cities along the U.S./Mexico border. I used the Twitter API and R to mine tweets and analyzed the sentiment of each tweet with 4 different algorithms to see which best told the story of those populations closest to the border.

This paper will summarize my technique for mining and cleaning Twitter data, define sentiment analysis, compare how each algorithm calculates sentiment, and review the pros and cons of each algorithm based on the type of data being analyzed.

Background and Related Work

There are many web pages, articles and tutorials describing the steps to conduct sentiment analysis. Most of these do an excellent job of providing code examples and instructions on conducting analysis. However, all lacked the details on how the sentiment algorithms work. None of the related work that I could find described the strengths and weaknesses of different algorithms based on the context being analyzed.

Program Requirements

The script I wrote in R searches Twitter for tweets based on a defined search term within a geographical radius for a given time period. The script removes duplicate tweets (retweets), removes special characters, converts the data into a vector, and stores the data into a data frame. The data frame is then run through 4 different sentiment analysis algorithms, the totals for each algorithm are summed and then compared.

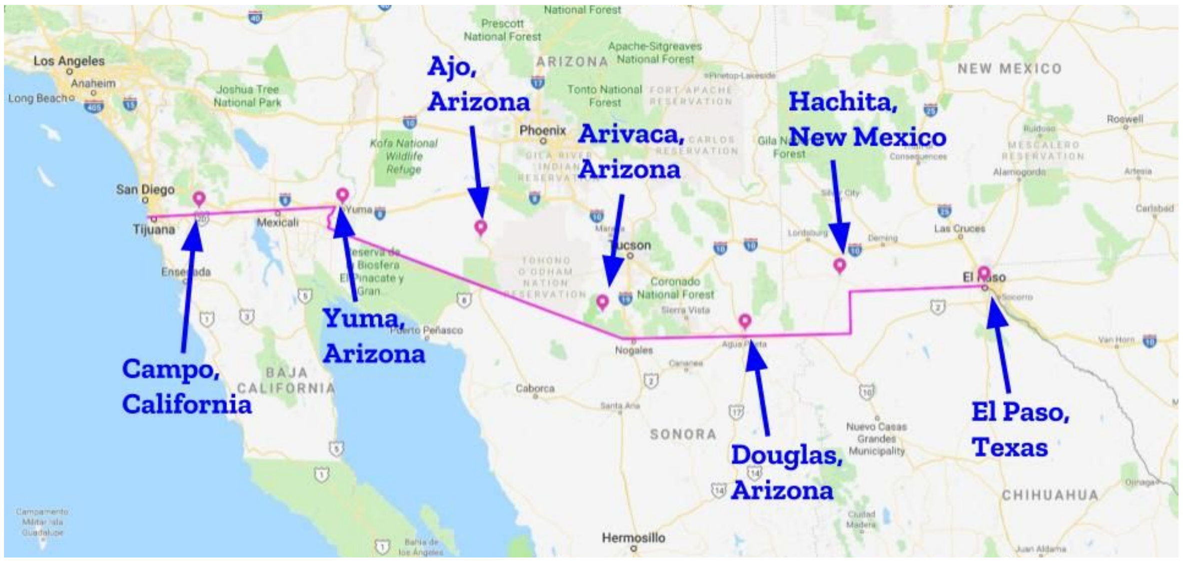
Implementation

The technologies I chose to analyze sentiment were: R, the Twitter API, and the Syuzhet R library. I was not familiar with any of these at the beginning of my project, but after my research it was clear that the R language was a top choice among statisticians and data miners for data analysis.

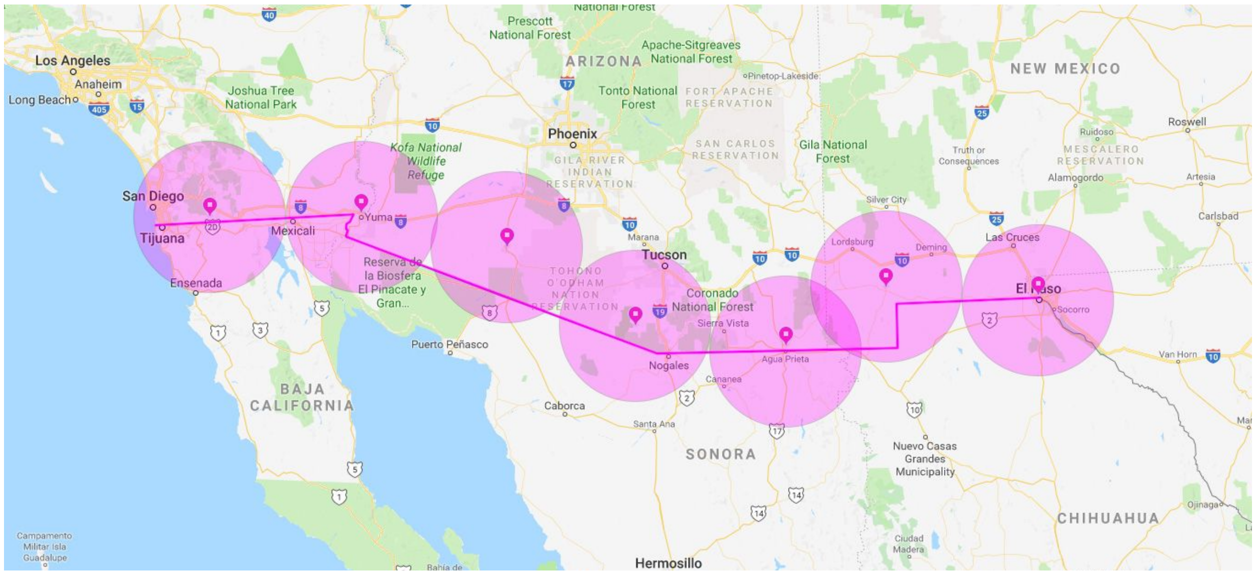
I did consider the Python program Tweepy to mine tweets, but the sentiment analysis with Tweepy seemed to be limited to a single algorithm called TextBlob.

I choose seven U.S. cities along the southern border. The pink line in the map below shows where the proposed border wall would be constructed.

- Campo, California
- Yuma, Arizona
- Ajo, Arizona
- Arivaca, Arizona
- Douglas, Arizona
- Hachita, New Mexico
- El Paso, Texas



I then determined collecting tweets within a 60-mile radius of each of these cities would encompass a geographical area covering the entire length of the proposed wall.



Connecting R and Twitter

I installed the twitteR package for R by running the command below, this allowed me to begin my data analysis of tweets.

```
install.packages("twitteR")
library(twitteR)
```

Next I entered my API keys in order to use the Twitter API. I assigned my keys to variable in R by using the `<-` syntax. The string `abcdefgh123456789` used below is a dummy key.

```
consumer_key <- 'abcdefgh123456789'
consumer_secret <- 'abcdefgh123456789'
access_token <- 'abcdefgh123456789'
access_secret <- 'abcdefgh123456789'
```

Once the consumer keys and access keys were assigned to variables I set up OAuth with following command. This validated my keys and authorized my use of the Twitter API.

```
setup_twitter_oauth(consumer_key, consumer_secret,
access_token, access_secret)
```

Searching Twitter

Now I could begin searching tweets using the function `searchTwitter`. To find tweets about GVSU, for example, I used the following command:

```
searchTwitter('gvsu')
```

By default the number of results returned for a given search is 25. To view more results I passed the `n` argument.

```
searchTwitter('gvsu' , n=50)
```

To find the most liked and shared tweets around a particular search term I used the argument `resultType='popular'`

```
searchTwitter('gvsu', resultType='popular')
```

To search within a specific date range I added the `since` and `until` arguments. Tweets available via the API only go back 6-9 days.

```
searchTwitter('gvsu' , since='2019-02-11', until='2019-02-15')
```

Searching tweets within a radius of a specified latitude and longitude is possible by adding the `geocode` argument. The example below searched for tweets within a 60-mile radius of Allendale, Michigan.

```
searchTwitter('gvsu' , geocode='42.974571,-85.954086,60mi')
```

For my research on the border wall, I used the search command below. This collected 3,200 tweets (Twitter's API limit) within a 60-mile radius of Campo, CA that contained the word 'border' and the word 'wall' between the dates of 1/4/2019 and 1/5/2019. I stored the results in the variable `campo_california`

```
campo_califonia <- searchTwitter('border wall' ,
since='2019-01-04', until='2019-01-05' ,
geocode='32.607522,-116.469890 , 60mi', n=3200)
```

Cleaning Mined Data

Now that I had a large data set of tweets I had to clean the tweets by removing retweets, links, tabs, spaces, and special characters so the sentiment algorithms can focus on the words and sentences in each tweet.

Applying the `strip_retweets` function to my original data set `campo_califonia` removed all tweets that had been reposted by other individuals, this removed duplicate tweets that would have skewed the sentiment analysis. I wrote these cleaned tweets to a new variable `campo_no_retweets` by using the following command:

```
campo_no_retweets <- strip_retweets(campo_califonia)
```

The resulting data returned when using the `searchTwitter` function contained much more information than just the text of each tweet. Tweets are accompanied with information about creation date, if the tweet was a reply to another tweet, how many times the tweet has been favorited, the originator's screen name, number of times the tweet has been retweeted, and the geolocation of the tweet with longitude and latitude. To see all the data returned I used the `head` command below. The `head` command in R will return the first several rows of a data set.

```
head(campo_no_retweets)
```

Sentiment algorithms will not be able to process this additional information, so we'll need to create a new data set that contains just the text of each tweet, essentially stripping out all the extra data. Twitter search results were easier to manipulate by putting them into tabular format, in R this is known as a data frame. The function `as.data.frame` makes this possible. I put the data set `campo_no_retweets` into the data frame `campo_df` by using the following command in R

```
campo_df <- as.data.frame(campo_no_retweets)
```

R stores data in temporary memory while the software is open. When R is closed the session is terminated and all data is wiped. While working in R I often saved data to an external file so I could use it later during my research. The function below writes a data frame to the `campo_df.csv` file. This file will be saved in the R working directory. Typing `getwd()` displayed the working directory so I could locate the file on my hard drive.

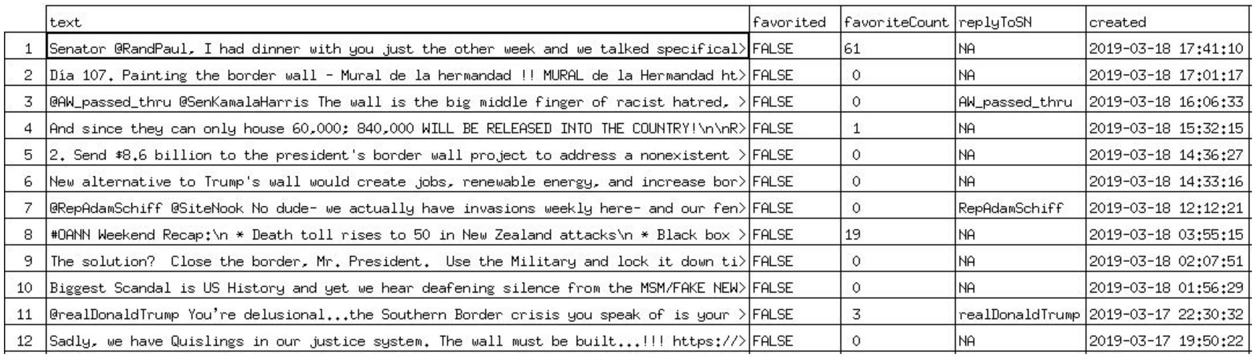
```
write.csv(campo_df, "campo_df.csv")
```

I imported the data back into R later from the .csv file by using the following command. This placed my data set into the variable `campo_df`

```
campo_df <-read.csv("campo_df.csv", header=TRUE)
```

To view the data frame in table format with rows and columns I used the the `View` command.

```
View(campo_df)
```



	text	favorited	favoriteCount	replyToSN	created
1	Senator @RandPaul, I had dinner with you just the other week and we talked specifical>	FALSE	61	NA	2019-03-18 17:41:10
2	Día 107. Painting the border wall - Mural de la hermandad !! MURAL de la Hermandad ht>	FALSE	0	NA	2019-03-18 17:01:17
3	@AW_passed_thru @SenKamalaHarris The wall is the big middle finger of racist hatred, >	FALSE	0	AW_passed_thru	2019-03-18 16:06:33
4	And since they can only house 60,000; 840,000 WILL BE RELEASED INTO THE COUNTRY!\n\nR>	FALSE	1	NA	2019-03-18 15:32:15
5	2. Send $8.6 billion to the president's border wall project to address a nonexistent >	FALSE	0	NA	2019-03-18 14:36:27
6	New alternative to Trump's wall would create jobs, renewable energy, and increase bor>	FALSE	0	NA	2019-03-18 14:33:16
7	@RepAdamSchiff @SiteNook No dude- we actually have invasions weekly here- and our fen>	FALSE	0	RepAdamSchiff	2019-03-18 12:12:21
8	#OANN Weekend Recap:\n * Death toll rises to 50 in New Zealand attacks\n * Black box >	FALSE	19	NA	2019-03-18 03:55:15
9	The solution? Close the border, Mr. President. Use the Military and lock it down ti>	FALSE	0	NA	2019-03-18 02:07:51
10	Biggest Scandal is US History and yet we hear deafening silence from the MSM/FAKE NEW>	FALSE	0	NA	2019-03-18 01:56:29
11	@realDonaldTrump You're delusional...the Southern Border crisis you speak of is your >	FALSE	3	realDonaldTrump	2019-03-17 22:30:32
12	Sadly, we have Quislings in our justice system. The wall must be built...!!! https://>	FALSE	0	NA	2019-03-17 19:50:22

I noted the text for each tweet was contained within the text column of the data frame. To view just the text column and disregard all the other data I used the `$` argument. I could see how this worked by using the following command:

```
head(campo_df $text)

RESULT
[1] "Senator @RandPaul, I had dinner with you just the other
week and we talked specifically about the border wall, where…
https://t.co/I5mN1Aq7z2"

[2] "Día 107. Painting the border wall - Mural de la hermandad
!! MURAL de la Hermandad https://t.co/RVrEP0M78s"

[3] "@AW_passed_thru @SenKamalaHarris The wall is the big
middle finger of racist hatred, a waste of money, and it won't
last long https://t.co/YibxpCZxO6"

[4] "And since they can only house 60,000; 840,000 WILL BE
RELEASED INTO THE COUNTRY!\n\nRefuse to be a victim.  Open
Carr… https://t.co/s0x2JyTZDA"

[5] "2. Send $8.6 billion to the president's border wall
project to address a nonexistent national emergency, on top of…
https://t.co/iE71OXe49E"
```

Next I needed to clean the text of each tweet by using the `gsub` function on only the text column of the data frame. This replaces all matches of a defined string. The # symbol is used to add a comment in the R console. Comments are not executed. I put the clean text of each tweet into a new variable `clean_tweet`

```
clean_tweet <- gsub("@\\w+", "", clean_tweet) #remove @
clean_tweet <- gsub("[[:punct:]]", "",clean_tweet) #remove
punctuation
clean_tweet <- gsub("[[:digit:]]", "", clean_tweet) #remove numbers
clean_tweet <- gsub("http\\w+", "", clean_tweet) #remove http
clean_tweet <- gsub("[ |\t]{2,}", "", clean_tweet) #remove tabs
clean_tweet <- gsub("^ ", "", clean_tweet) #remove leading spaces
clean_tweet <- gsub(" $", "", clean_tweet)      #remove trailing
spaces
```

I wanted to view the first few lines of the cleaned tweets in the new variable `clean_tweet` by using the `head` command again. I could see the additional information with each tweet had been discarded and the resulting text had been cleaned and was ready for sentiment analysis.

```
head(clean_tweet)

RESULT
[1] "Senator I had dinner with you just the other week and we
talked specifically about the border wall where"

[2] "Día Painting the border wall Mural de la hermandad MURAL
de la Hermandad"

[3] "The wall is the big middle finger of racist hatred a waste
of money and it won’t last long"

[4] "And since they can only house WILL BE RELEASED INTO THE
COUNTRY Refuse to be a victim Open Carr"

[5] "Send billion to the president’s border wall project to
address a nonexistent national emergency on top of"
```

Analyzing Sentiment

What is Sentiment Analysis?

Sentiment analysis is the measurement of positive and negative language. It is used to evaluate written words and sentences to determine if the overall meaning is favorable, unfavorable, or neutral for a given subject. There are several algorithms today that can process sentiment on very large data sets to determine public opinions on products, services, and political campaigns. Running sentiment analysis on a regular basis can shed light on changes in public opinion.

Preparing Sentiment Algorithms

The Syuzhet R package in R allowed me to compare four sentiment algorithms: syuzhet (default), bing, afinn, and nrc.

I installed the Syuzhet library by running the following command in R:

```
installed.packages("syuzhet")
library("syuzhet")
```

To analyze tweets I first needed to convert the clean text of tweets into a vector. A vector is a basic data structure in R that contain elements of the same type, in this case, the elements will be words and sentences. Once R can recognize a tweet as a vector the sentiment algorithms can analyze and score each word and sentence independently. To convert tweets into a vector I used the `as.vector` function and put the collection of words and sentences into a new data frame `words_df`

```
words_df <- as.vector(clean_tweet)
```

Comparing Sentiment Algorithm Calculations

Syuzhet Algorithm

The Syuzhet lexicon was developed in the Nebraska Literary Lab. The name "Syuzhet" comes from the Russian Formalists Victor Shklovsky and Vladimir Propp who divided narrative into two components, the "fabula" (the text) and the "syuzhet" (the order). The Syuzhet algorithm is used for analyzing literary works and focuses on how the elements of the text are organized and assigns a sentiment score for each word ranging from -1 to 1 using fractions.

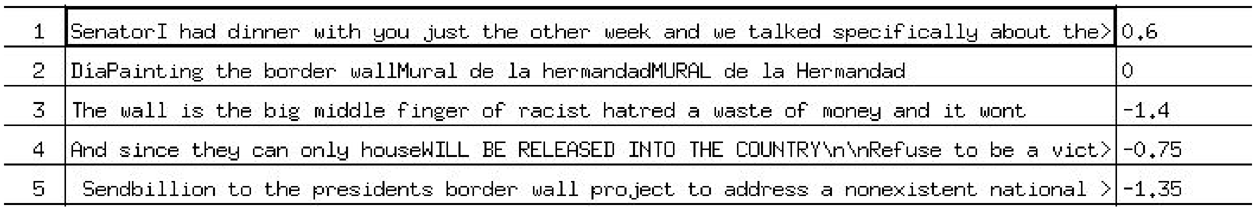
To extract the sentiment values for each tweet I used the `get_sentiment` function on the `words_df` data frame and placed the values into a new variable `syuzhet_values`

```
syuzhet_values <- get_sentiment(words_df)
```

Next I combined the original clean tweets with the syuzhet sentiment scores using the `cbind function` and viewed the results. The last column in the data frame summed the scores from each word in the tweet. A negative number indicated the tweet contained unfavorable sentiment about the terms ‘border wall’ while a positive number indicated a favorable sentiment. Zero indicated a neutral sentiment.

```
syuzhet_campo <- cbind(clean_tweet, syuzhet_values)

View(syuzhet_campo)
```



1	SenatorI had dinner with you just the other week and we talked specifically about the>	0.6
2	DiaPainting the border wallMural de la hermandadMURAL de la Hermandad	0
3	The wall is the big middle finger of racist hatred a waste of money and it wont	-1.4
4	And since they can only houseWILL BE RELEASED INTO THE COUNTRY\n\nRefuse to be a vict>	-0.75
5	Sendbillion to the presidents border wall project to address a nonexistent national >	-1.35

Bing Algorithm

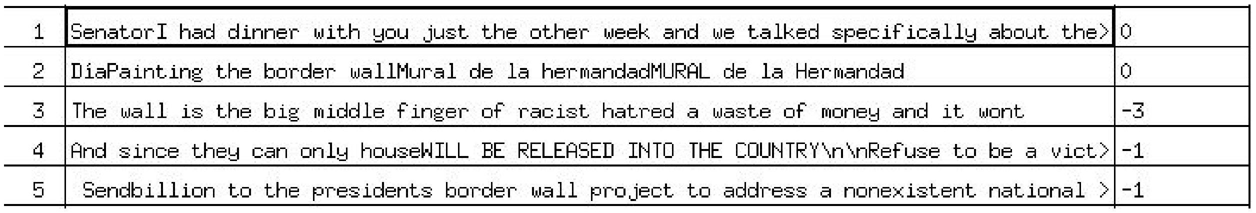
The Bing lexicon was developed by Minqing Hu and Bing Liu. Hu and Bing categorized words into two main types: facts and opinions. This lexicon focuses on the sentiment of opinion words. The original goal for this algorithm was to detect fake product reviews. The algorithm assigns a -1 to negative words and +1 to positive words as defined in the Hu and Bing lexicon. Running the following command invoked the bing algorithm and gave me a total for each tweet.

```
bing_values <- get_sentiment(words_df, method="bing")
```

Once again I combined the original tweet with the bing sum values and viewed the results.

```
bing_campo <- cbind(clean_tweet, bing_values)

View(bing_campo)
```



1	SenatorI had dinner with you just the other week and we talked specifically about the>	0
2	DiaPainting the border wallMural de la hermandadMURAL de la Hermandad	0
3	The wall is the big middle finger of racist hatred a waste of money and it wont	-3
4	And since they can only houseWILL BE RELEASED INTO THE COUNTRY\n\nRefuse to be a vict>	-1
5	Sendbillion to the presidents border wall project to address a nonexistent national >	-1

Afinn Algorithm

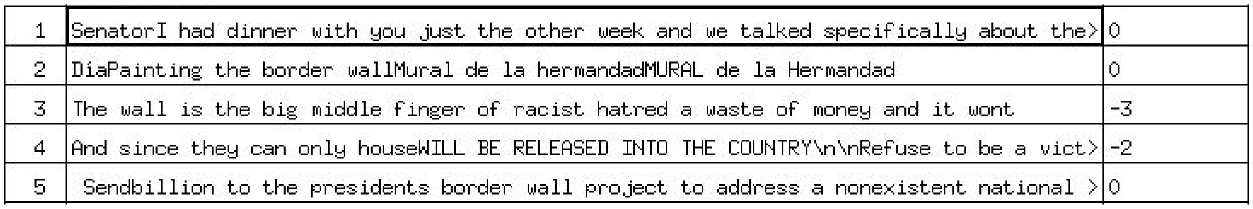
The AFINN algorithm uses a list of English terms manually scored with values between -5 (negative) and +5 (positive) by Finn Årup Nielsen. This lexicon was originally developed to detect obscene words.

```
    afinn_values <- get_sentiment(words_df, method="afinn")
```

I combined the original tweet with the Afinn sum values and viewed the results.

```
    afinn_campo <- cbind(clean_tweet, afinn_values)

    View(bing_afinn)
```



1	SenatorI had dinner with you just the other week and we talked specifically about the>	0
2	DiaPainting the border wallMural de la hermandadMURAL de la Hermandad	0
3	The wall is the big middle finger of racist hatred a waste of money and it wont	-3
4	And since they can only houseWILL BE RELEASED INTO THE COUNTRY\n\nRefuse to be a vict>	-2
5	Sendbillion to the presidents border wall project to address a nonexistent national >	0

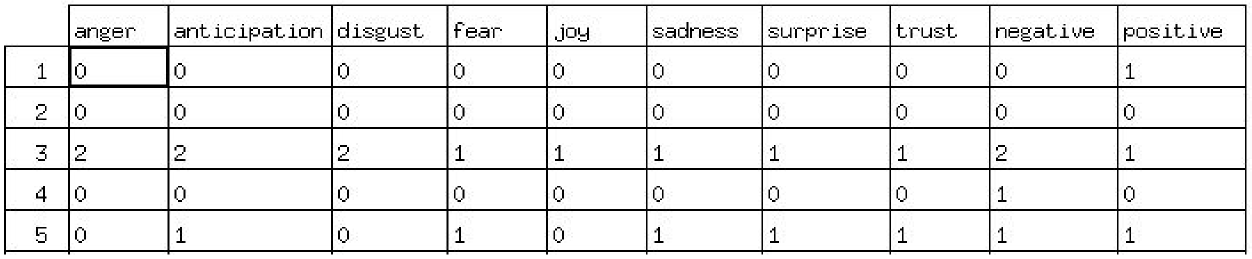
NRC Algorithm

The NRC Emotion Lexicon is a list of English words that have been mapped to eight emotions via crowdsourcing: anger, fear, anticipation, trust, surprise, sadness, joy, and disgust. The NRC algorithm then provides an overall sentiment score for the tweet based on a tally of the emotions present.

To view how this is accomplished I used the `get_nrc_sentiment` on the `words_df` data frame and then used the `View` command to see the breakdown of emotions for each tweet.

```
nrc_values <- get_nrc_sentiment(words_df)

View(emotions_df)
```



	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0
3	2	2	2	1	1	1	1	1	2	1
4	0	0	0	0	0	0	0	0	1	0
5	0	1	0	1	0	1	1	1	1	1

The output shows the count of words in the tweet that are associated with each emotion. In the example above, the third tweet has 2 words associated with anger, 2 with anticipation, 2 with disgust, 1 with fear, 1 with joy, 1 with sadness, 1 with surprise, and 1 with trust. The `get_nrc_sentiment` function then assigns the tweet with an overall positive or negative score by summing the number of positive emotions or negative emotions present.

Syuzhet Bing, Afinn, and NRC Scoring

Having each algorithm analyze the same tweet shed some light on exactly how each calculates a sentiment score.

Syuzhet

```
The border wall is stupid, racist and a big waste of money
 0    0     0   0    -1       0      0  0  0    -1  0    1     =   -1
```

Bing

```
The border wall is stupid, racist and a big waste of money
 0    0     0   0    -1      -1      0  0  0    -1  0    0     =   -3
```

Afinn

```
The border wall is stupid, racist and a big waste of money
 0    0     0   0    -2      -3      0  0  1    -1  0    0     =   -5
```

NRC

```
The border wall is stupid, racist and a big waste of money
 0    0     0   0    -1      -1      0  0 .25 -.75  0   .6     =   -1.9
```

Even though each algorithm used a different scale for analyzing sentiment, they all used a negative number as unfavorable sentiment, a positive number as favorable sentiment and zero as neutral sentiment.

To determine the overall sentiment score with each algorithm for the original data set `words_df` I summed the values. Why not average the scores? The neutral words scored with 0 would bring the overall sentiment down. For example:

```
The border wall is stupid, racist and a big waste of money
 0    0     0   0    -1       0      0  0  0    -1  0    1
```

Summing the sentiment score for the sentence above would result in a -1, while averaging the scores would result in a -0.08. The result of averaging the sentiment (-0.08) could likely be rounded to 0 (neutral) and this tweet is more accurately representing unfavorable sentiment about the border wall.

I summed the total values for each algorithm by using the `sum` function in R:

```
sum(syuzhet_values)
     -3.9
sum(bing_values)
     -10
sum(afinn_values)
     -9
sum(nrc_values)
     -1
```

The above indicates that the Syuzhet, Bing, Afinn and NRC algorithms all indicated an unfavorable sentiment around the the terms ‘border wall’ for the sample tweets I pulled.

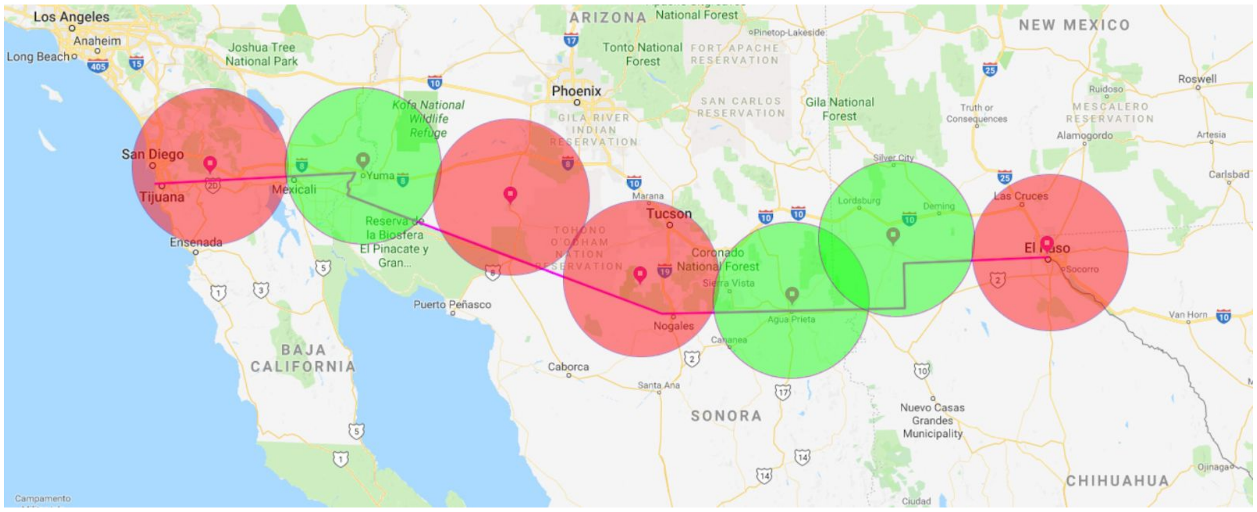
Results

Using the Bing algorithm on the 268,800 tweets I collected from 7 border cities over 11 days yielded the sentiment results below. Running this large data set through each algorithm, and spot checking several dozen tweets I felt the Bing algorithm evaluated tweets the best. I evaluate the pros and cons of each algorithm in the next section of this paper.

I was curious if sentiment changed before and after the government shutdown so I split the data up accordingly.

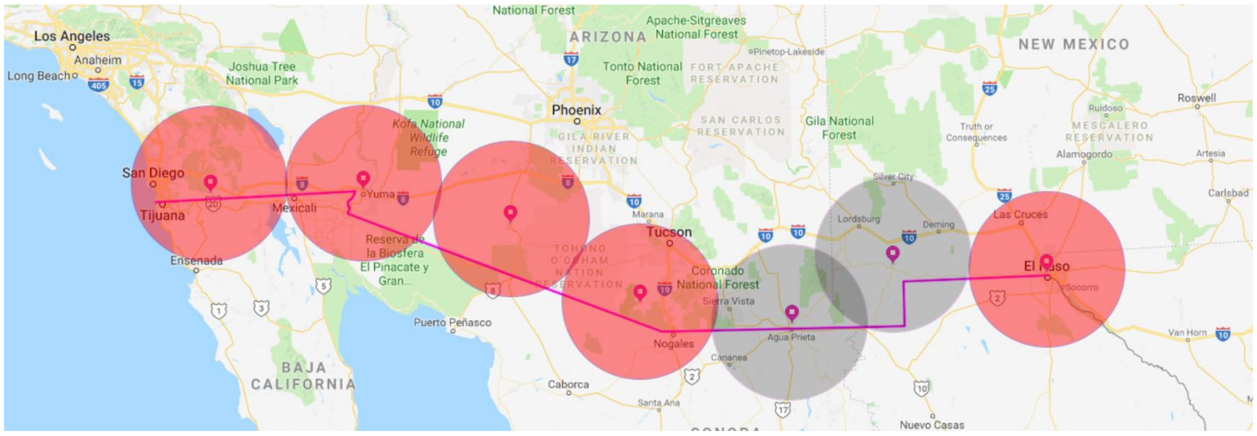
During the U.S. Government Shutdown

During the U.S. government shutdown sentiments around the term ‘border wall’ were varied. The sentiment analysis showed that 4 of the border cities showed unfavorable sentiment around the terms ‘border wall’ while 3 cities showed favorable sentiment around the same terms.



After the U.S. Government Shutdown

After the U.S. government shutdown attitudes around the term ‘border wall’ shifted toward unfavorable. The 4 border cities that showed unfavorable sentiment around the terms ‘border wall’ remained unfavorable while the 3 cities that showed favorable sentiment shifted to negative or neutral feelings.



Evaluation

Pros and Cons of Sentiment Algorithms

One of the most interesting outcomes of my project was that each algorithm produced different results. Researching the origin of each algorithm helped me understand their benefits and drawbacks for analyzing different material.

Syuzhet

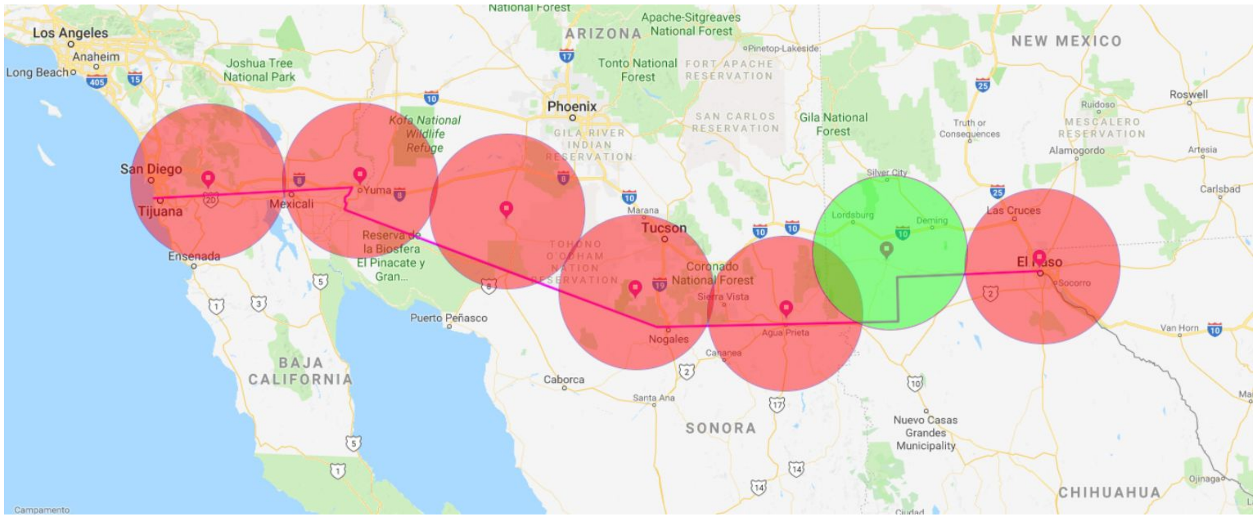
The lexicon used for this algorithm is based on the work from Russian formalists who analyzed the order and meaning of words in literature.

Pro

Works well for finding themes in books and determining the distinct linguistic style of a given author

Con

Does not work well on short microposts like tweets since the algorithm looks at order of words. Tweets are too short to determine themes.



Total sentiment score from Syuzhet algorithm for all tweets gathered from 1/21/19 - 2/1/19

Bing

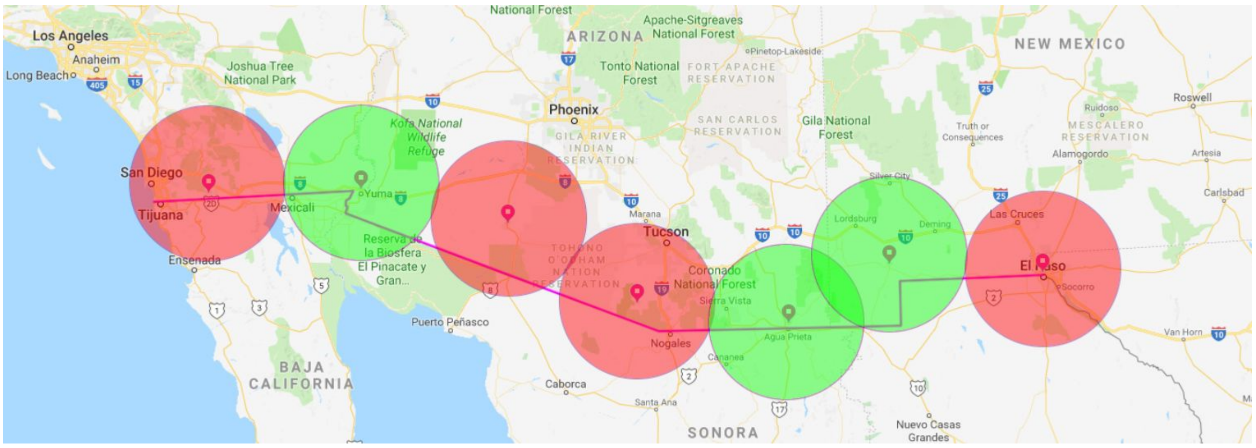
Originally created to detect fake product reviews on web sites. Minqing Hu and Bing Liu focused their work on “opinion” words.

Pro

Worked well for analyzing tweets. Due to the small lexicon less words were scored, however more words were scored more accurately.

Con

Ignored the order of words, so words like ‘very’ or ‘extremely’ in front of a positive or negative word were ignored. The original positive or negative word would have the same score regardless of the adjective in front of it.



Total sentiment score from Bing algorithm for all tweets gathered from 1/21/19 - 2/1/19

Afinn

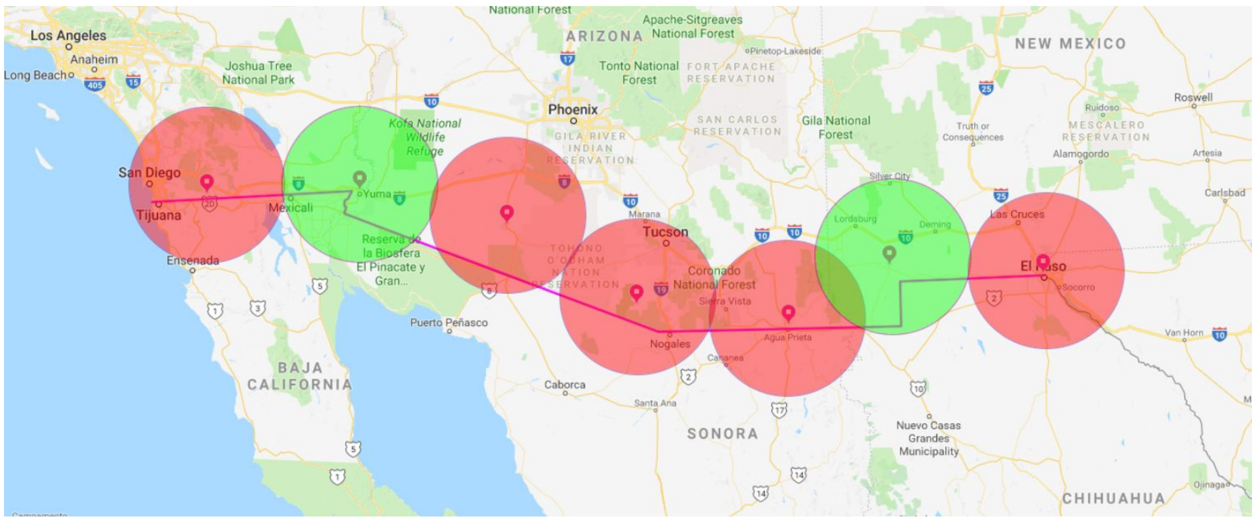
Created by Finn Årup Nielsen originany to detect obscenities and then expanded to determine sentiment.

Pro

Lexicon contains slang which lends itself well for analyzing tweets.

Con

Some oddities in scoring, words like ‘big’ were rated positive. Word order was ignored.



Total sentiment score from Afinn algorithm for all tweets gathered from 1/21/19 - 2/1/19

NRC

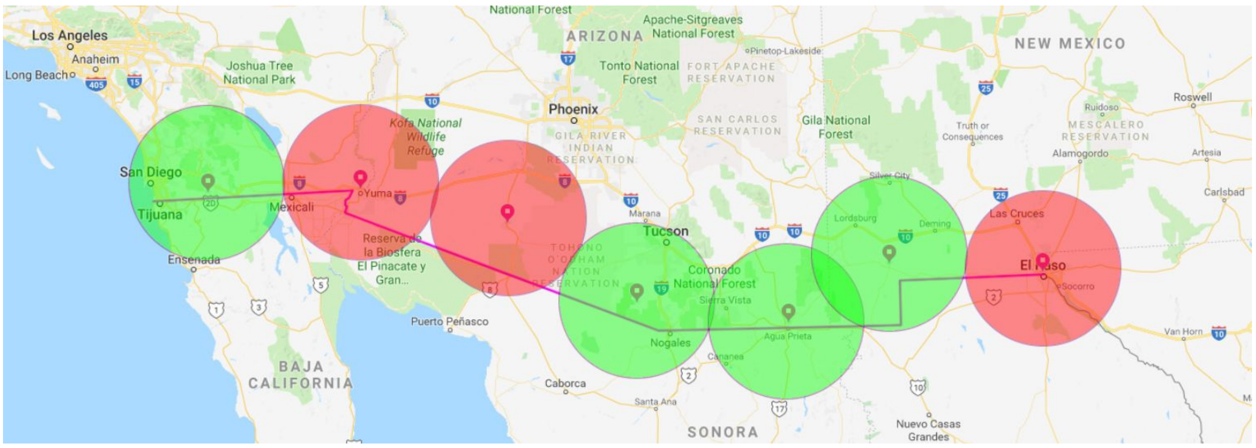
Crowdsourced lexicon that asks users to associate an emotion to a word. Emotions are grouped into positive or negative sentiment.

Pro

A very large lexicon that is updated regularly

Con

Nearly all tweets analyzed were scored positive. Oddities in words scored positive like chairman, big and fireball



Total sentiment score from NRC algorithm for all tweets gathered from 1/21/19 - 2/1/19

Reflection

I am new to the programming language R, as a novice I am sure there are ways to improve my code. I also struggled writing a perfect regular expression that cleaned tweets properly, after much trial and error I believe I arrived at a good solution, but again, I am sure I overlooked efficiencies.

Sentiment is inherently subjective from person to person. An individual that is typically neutral on a specific subject might have a bad day and tweet something negative. This is why it is important to gather a large data set. The populations within a 60 mile radius of Ajo, Arizona and Hachita, New Mexico were under 60,000. On several days during my research these geographic locations produced less than 100 tweets. Further research would need to be conducted to see if these small data sets could have been skewed by a small number of individuals.

I found no sentiment algorithms that could detect sarcasm. This seems to be a major hurdle for sentiment analysis.

Conclusions and Future Work

I was curious how the individuals living along the southern border of the United States felt about the proposed border wall. By mining tweets and analyzing sentiment from cities along the U.S./Mexico border I was able to determine that sentiment around the border wall overall was unfavorable.

When considering sentiment analysis for your brand, service or political campaign it’s important to know how each algorithm works and which is the best tool for the data you are analyzing.

A custom lexicon around the context being analyzed could be produced for further accuracy. For example, if customers use the word ‘sick’ in a favorable way to describe a product, a custom lexicon could be produced that ranks ‘sick’ as a positive word.

Coupling demographic data with sentiment analysis may lend more insight into the sentiment of certain words. Knowing a tweet is from an individual that is 15 years old versus 50 years old may help determine if a word like ‘sick’ is favorable or unfavorable.

Bibliography

Finn Årup Nielsen, "A new ANEW: evaluation of a word list for sentiment analysis in microblogs", Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages. Volume 718 in CEUR Workshop Proceedings: 93-98. 2011 May. Matthew Rowe, Milan Stankovic, Aba-Sah Dadzie, Mariann Hardey (editors)

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60

Nielsen, and Finn Årup. “A New ANEW: Evaluation of a Word List for Sentiment Analysis in Microblogs.” ArXiv.org, 15 Mar. 2011, arxiv.org/abs/1103.2903

Indurkhya, Nitin, and Frederick J. Damerau. Handbook of Natural Language Processing. Chapman & Hall/CRC, 2010.

Gentry, Jeff. “TwitteR v1.1.9.” TwitteR Package | R Documentation, www.rdocumentation.org/packages/twitteR/versions/1.1.9

Jockers, Matthew. Introduction to the Syuzhet Package, 13 Dec. 2017, cran.r-project.org/web/packages/syuzhet/vignettes/syuzhet-vignette.html

Mohammad, Saif. NRC Word-Emotion Association Lexicon, www.saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm