Capstone project: report

Jacopo Malatesta

22/2/2020

Contents

Goals of the project

This small research aims at gaining insight into the Twitter content concerning the 2020 Democratic Party presidential primaries. The Twitter discourse around this topic has come under a lot of scrutiny in recent times due to the often aggressive behavior of both the candidates and their supporters.

The goal of this research is to analyze Twitter data concerning the four main Democratic candidates in order to highlight the underlying emotion and sentiment beneath their tweets and to shed light on the most frequent topics and issues. The candidates whose Tweets were analyzed are Bernie Sanders, Elizabeth Warren, Pete Buttigieg and Joe Biden.

Choice of the subject matter

The choice of this topic is partly driven by a long-standing interest in American politics and culture, but more importantly by a desire to deepen my knowledge of digital analysis methods and social media in general. After my master degree, I plan on working in the communication and marketing field and in order to be successful in this area acquiring some basic skills in the analysis of social media data is essential. Though I realize that in order to land a good marketing job many more skills need to be added to my talent stack, this small project might turn out to be a good stepping stone in my professional development.

Main features of the projects

The overall structure of this work was inspired by Céline Van den Rul's article "A Guide to Mining and Analysing Tweets with R", which also served as a source of inspiration for many of the lines of codes that were used.

This project consists of three separate steps: collecting Twitter data, analyzing the content and sentiment of its text and visualizing the results.

The data collection has been carried out by scraping Twitter data with an R studio package called "Rtweet"; the analysis and visualization parts instead required the use of basic text and sentimental analysis tools, such as the syuzhet and tidytext packages. Naturally, more common packages such as rio, dplyr and ggplot2 were used as well throughout the project.

Technical overview

In this part I'll go through each step of the project by commenting the scripts I wrote. However, due to the length of some of the chunks of codes, only some lines will be embedded and commented. Readers interested in having a deeper look at the codes can inspect the complete scripts in the scripts folder.

Packages used

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rtweet)
library(rio)
library(tidytext)
library(quanteda)
```

```
## Package version: 1.5.2
```

```
## Parallel computing: 2 of 4 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

```
##
## Attaching package: 'quanteda'
```

```
## The following object is masked from 'package:rio':
##
##     convert
```

```
## The following object is masked from 'package:utils':
##
##     View
```

```
library(ggplot2)
library(syuzhet)
```

```
##
## Attaching package: 'syuzhet'
```

```
## The following object is masked from 'package:rtweet':
##
##     get_tokens
```

Data collection

The candidates' tweets were retrieved with the `get_timelines` function from the Rtweet package, which allows users to scrape the most recent 3200 tweets from specific accounts.

The argument n allows me to set the number of tweets to scrape, in this case it was 1000 for each candidate.

```
overall_tweets <- get_timelines(c("JoeBiden", "PeteButtigieg", "ewarren", "BernieSanders"), n = 1000)
```

```
tweets <- import(here::here("output","tweets.csv"))
```

The data were then saved as comma-separated values files, as this is one of the most universal formats available for storing text and data.

Text pre-processing and text analysis

The text analysis was only conducted on organic tweets, that is tweets that are not retweets.

```
tweets_organic <- tweets[tweets$is_retweet==FALSE, ]
```

The first step consisted in removing unwanted text elements, such as http elements, hashtag mentions, "amp" elements, and punctuation. In order to achieve this, the gsub function was used. This function searches for a pattern and replaces it with a character string.

```
tweets_organic$text <- gsub("https\\S*", "", tweets_organic$text)

tweets_organic$text <- gsub("@\\S*", "", tweets_organic$text)

tweets_organic$text <- gsub("amp", "", tweets_organic$text)

tweets_organic$text <- gsub("[\r\n]", "", tweets_organic$text)

tweets_organic$text <- gsub("[[:punct:]]", "", tweets_organic$text)
```

The next step was using the tidytext package to remove stop words from the words in the text column.

Firstly, I split the text column into different words with the unnest_tokens function.

```
tweets_cleaned <- tweets_organic %>%
  select(text) %>%
  unnest_tokens(word, text)
```

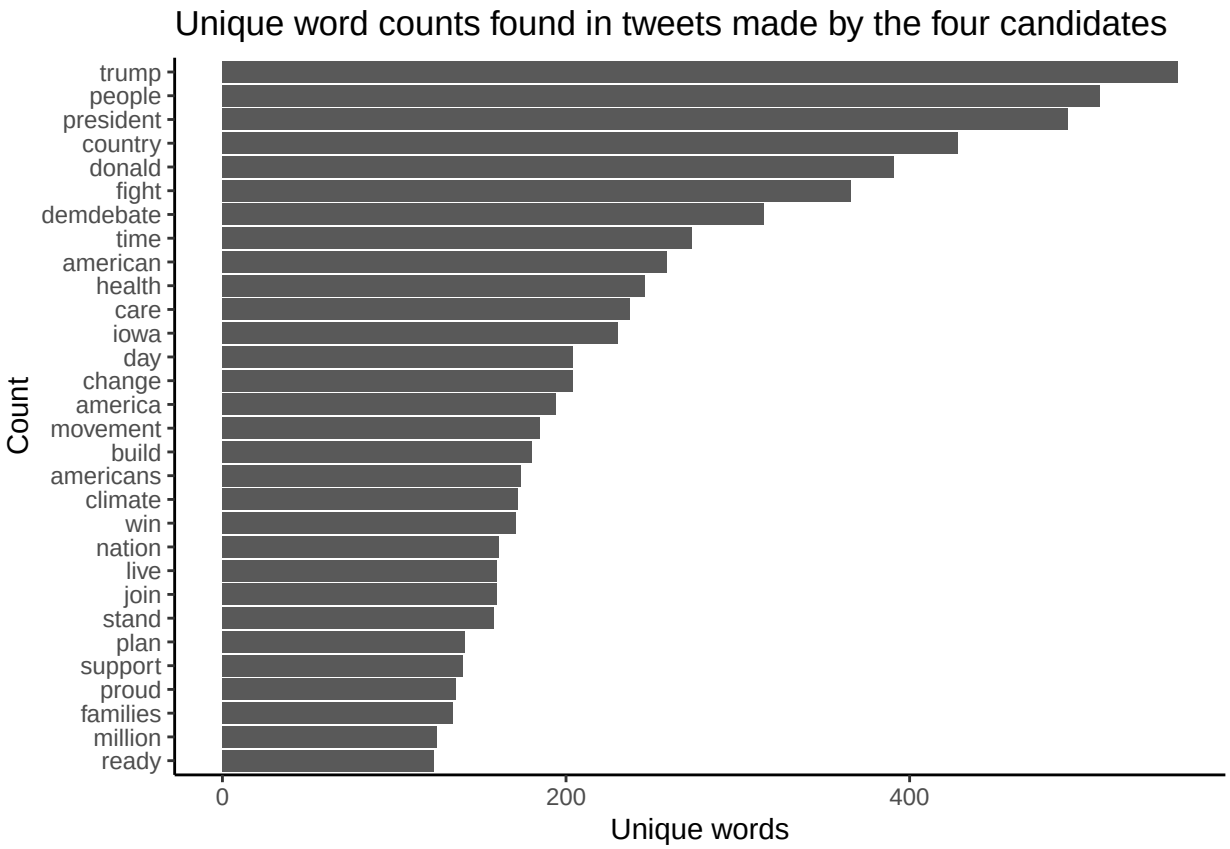
Secondly, I removed the stop words with an anti_join. An anti_join gives us the observations in our original dataset (in this case, the words from tweets_cleaned) that do not exist in the other table (the stop words).

```
tweets_cleaned <- tweets_cleaned %>%
  anti_join(stop_words)

## Joining, by = "word"
```

However, there are still some words left in there that need to be removed. In order to do so, I created a custom list (actually a vector) which I then turned into a dataframe.

```
custom_list <- c("aapi2020", "ve", "II", "Il", "ll", "lI", "â","iâ", "weâ", "canâ", "itâ", "caign", "hsl

custom_df <- tibble(joinColumn = custom_list)

tweets_cleaned <- tweets_cleaned %>%
  anti_join(custom_df,
            by = c("word" = "joinColumn"))
```

After getting rid of the stop words, I briefly analyzed the text by counting the most frequent words in the candidates' tweets. I will delve deeper into these results in the final part of this report.



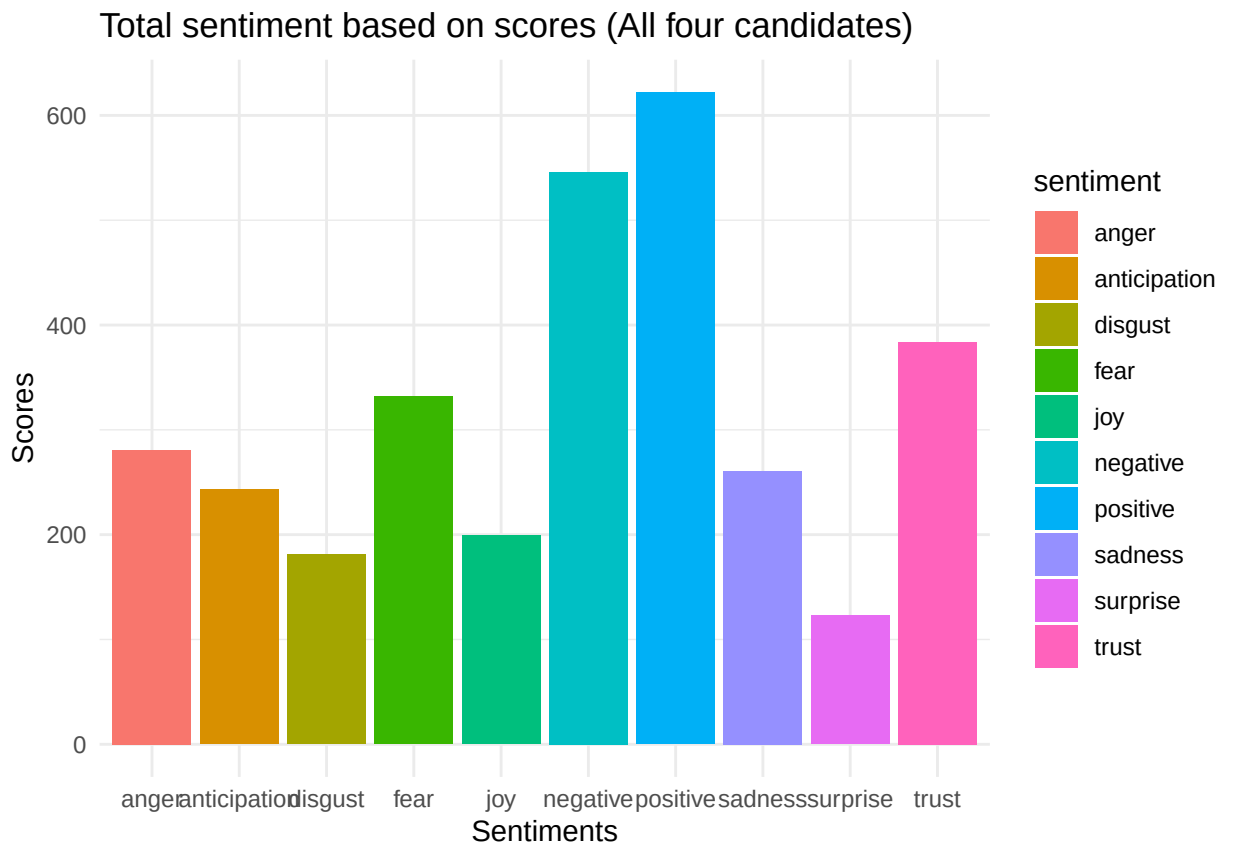
Sentiment analysis

In order to conduct some basic sentiment analysis, I converted the sets of words from UTF-8 to ASCII. Afterwards, I used the `get_nrc_sentiment` function from the syuzhet package, which calls the NRC sentiment dictionary to calculate the presence of eight different emotions and their corresponding valence in a text file.

```
tweets_cleaned <- iconv(tweets_cleaned, from="UTF-8", to="ASCII", sub="")

tweets_sentiment <- get_nrc_sentiment((tweets_cleaned))
sentimentscores_tweets <- data.frame(colSums(tweets_sentiment[,]))

names(sentimentscores_tweets) <- "Score"

sentimentscores_tweets <- cbind("sentiment"=rownames(sentimentscores_tweets),sentimentscores_tweets)

rownames(sentimentscores_tweets) <- NULL
```

Finally, I visualized the scores for each emotion with the ggplot2 package.

```
ggplot(data=sentimentscores_tweets,aes(x=sentiment,y=Score))+
  geom_bar(aes(fill=sentiment),stat = "identity")+
  theme(legend.position="none")+
  xlab("Sentiments")+ylab("Scores")+
  ggtitle("Total sentiment based on scores (All four candidates)")+
  theme_minimal()
```



Problems and shortcomings

This section of the report will go through the main challanges and struggles I have encountered in my work, as well its failures and shortcomings.

Lack of a permanent dataset

As explained in the previous section, the Twitter data have been collected with the `get_timelines` function from the R tweet dataset. This function retrieves the most recent tweets shared by any specified Twitter account. This feature, albeit useful, lead to the first setback I suffered in this work: since it always retrieves the most recent tweets, this function gets different tweets depending on when it is run.

As a result, in this project I wasn't able to produce a fixed and unchanging dataset. So, for example, if one year from now other users were to run my lines of code to scrape the candidates' tweets, they would not get the same tweets that I am working on today. Therefore the reproducibility of my work and analysis is compromised.

As far as I know, the Rtweet package package doesn't allow users to use the `get_timelines` function with a specific time interval (the "from" and "until" arguments are only available for the search_tweets function).

Removing stop words

Removing stop words turned out to be more challenging than expected. This was mostly due to the fact that the `anti_join(stop_words)` function did not remove all the stop words I intended to leave out. However,

after a couple of quick Google searches I found out a way to add some extra stop words, by inserting them manually into a dataframe and then removing from the set of words via an `anti_join`.

Commenting the results

Retweets and favorites

The table below portrays a quite unambiguous picture as to which candidate is the most popular on Twitter. Bernie Sanders largely outscores his three peers both on retweets and favorites.

I've added the kable() function from the knitr library to display a better-looking table.

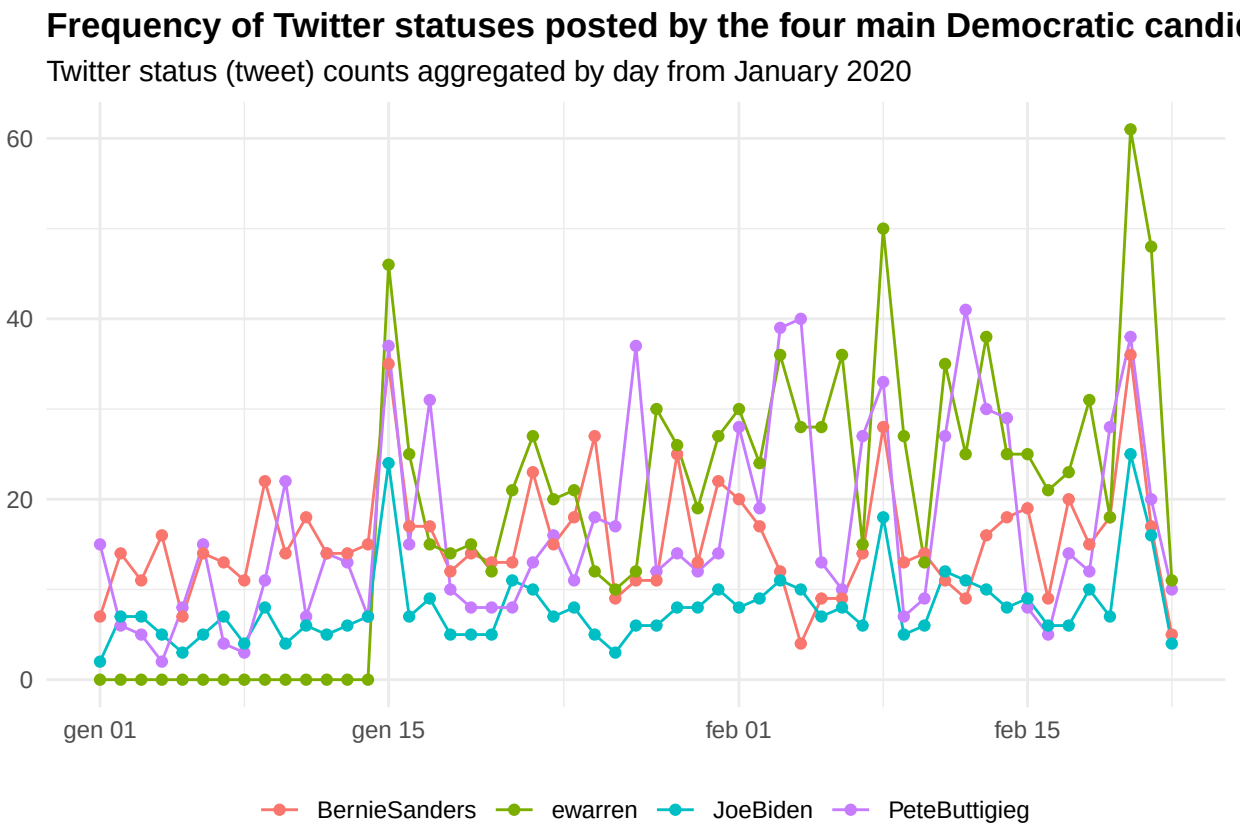
screen_name	average_retweet	average_favorite
BernieSanders	3706.649	20172.747
JoeBiden	1877.926	9312.048
ewarren	1190.044	5861.417
PeteButtigieg	539.007	3378.148

Frequency

The graph below plots the frequency in tweets by the four Democratic candidates. Here are some interestin takeaways:

1. The noticeable spike in frequency occurring around January the 14th coincides with the seventh Democratic debate that took place in Des Moines, Iowa the same day.
2. A second spike in frequency occured approximately between February the 4th and February the 7th. Over this period the Iowa caucus controversy and the Democratic debate took place.
3. The last increase in frequency portrayed in the graph is most likely due to the ninth Democratic debate and to the Nevada caucuses of February the 22nd.
4. In all these peaks Elizabeth Warren appears to outdo her opponents in terms of frequency of tweets. Joe Biden, on the other hand, has the least account account among the four.

```
tweets %>%
  filter(created_at > "2020-01-01") %>%
  group_by(screen_name) %>%
  ts_plot("days") +
  geom_point() +
  theme_minimal() +
  theme(
    legend.title = element_blank(),
    legend.position = "bottom",
    plot.title = element_text(face = "bold")) +
  labs(
    x = NULL, y = NULL,
    title = "Frequency of Twitter statuses posted by the four main Democratic candidates",
    subtitle = "Twitter status (tweet) counts aggregated by day from January 2020"
  )
```



Most frequent words

The text analysis I conducted revealed the most frequent words in the tweets shared by the four candidates. On the basis of the results displayed on page 5 of this report and in the scripts in the GitHub repository, the following conclusions can be drawn:

1. Unsurprisingly, "Trump" is the most recurrent word in the dataset. This result corroborates the impression that defeating Trump is perceived as the Democratic Party's priority in this election.
2. Many words in the dataset are mostly rethorical and non-ideological ("America", "people", "time", etc.). Others instead refer to actual policies and issues. Among the latter we find both "health" and "care" as the most recurrent, signalling the increased importance that the topic of health care insurance has assumed in the American public debate. This result provides some evidence in support of the thesis that the Sanders campaign in 2016 brought progressive policies at the forefront of the public discourse.
3. "Climate" ranks quite high as well, thanks to the attention this issue received on the global scale in the last months.
4. Bernie Sanders is the only candidate to have "war" and "workers" in his 30 most frequent words.
5. Joe Biden is the only candidate whose top 30 most recurrent words include "gun" and "violence".
6. Words missing from the top 30 are just as interesting as the ones included: "gender", "female", "immigration", "race" and "police" are all absent.

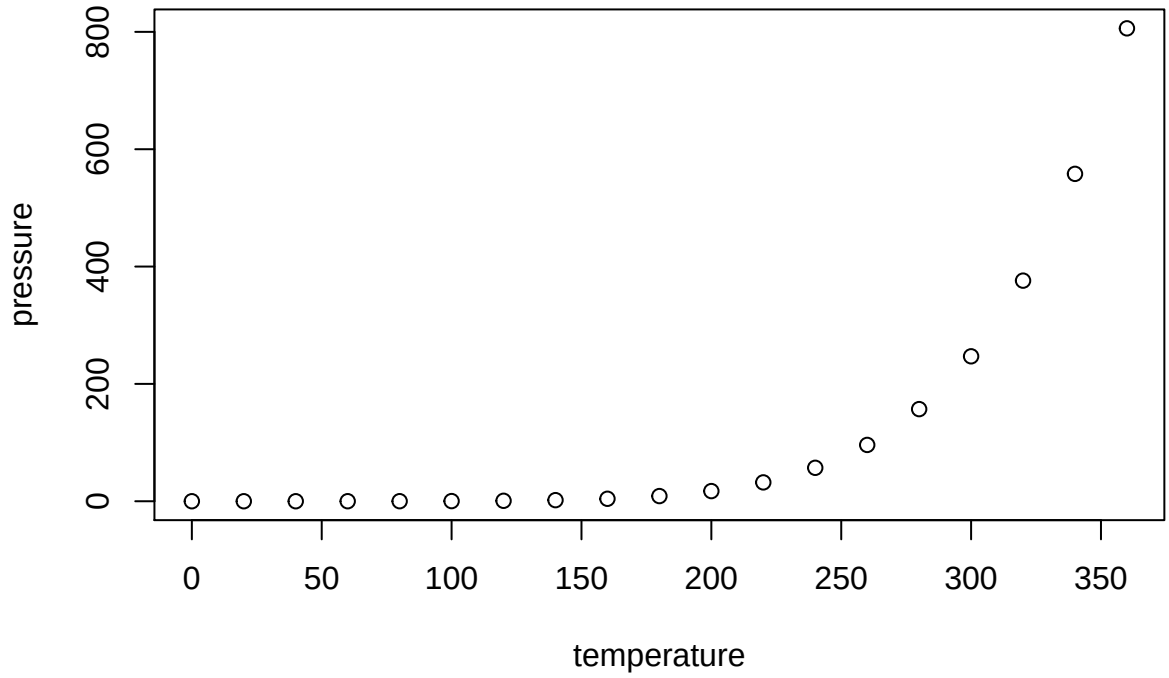
Sentiment analysis

Things this project taught me

This project turned out to be useful in more ways than one. There are many things I learned by working on it that I will treasure in the future:

- **Learning to use rtweet**: in recent years I developed an interest in Twitter, so I had a natural curiosity in learning to use this package. Rtweet is just as useful as it is easy to use (although, as I mentioned in previous sections, I had some issues with the `get_timelines` function). This project allowed me to delve deeper into its characteristics and functions. Equally important, I got to familiarize as well with its limitations, such as the inability of the search_tweets function to retrieve tweets older than 9 days, or the fact that Twitter caps the number of search results returned to 18,000 every 15 minutes.
- **Learning the basics of text analysis**: before this course and this project, I only had a vague idea as to what text and sentiment analysis were. Although I'm still far from being proficient, I now have a better grasp on these topics. In particular, I am glad to have learned about stop words and how to remove them.Most importantly, I had the chance to learn these techniques hands-on, instead of just studying them on textbooks. However, I remain skeptical about sentiment analysis and the very arbitrary way in which it assigns emotion and sentiment scores to words regardless of their context.
- **Learning R functions not related to text analysis**: I have also had the opportunity to learn about useful R functions which I had never seen before, such as `gsub` and `anti_join`. Plus, I've got to refresh my knowledge of famous packages like ggplot2 and dplyr.
- **That I have to improve my stress management skills**: despite it being quite simple, this project took a toll on my psychological well-being these last weeks. This is probably due to the fact that I have only been coding for some months now, so I am still not used to spend many hours reading line of codes on a screen. Anyway, this project provided a reminder that if I intend to pursue a career in a busy corporate environment I'd better work on my stress management skills.

You can also embed plots, for example:


pressure
0
200
400
600
800
0
50
100
150
200
250
300
350
temperature