# 607-Week9-Web_APIs

*Jose Mawyin*

*10/27/2019*

## Collecting Data using New York Times APIs and Sentiment Analysis

The New York Times web site provides a rich set of APIs, as described here: https://developer.nytimes.com/apis You'll need to start by signing up for an API key. Your task is to choose one of the New York Times APIs, construct an interface in R to read in the JSON data, and transform it into an R DataFrame.

This project contains the following Sections:

1. Data Collection
2. Data Analysis
3. Conclusions

### 1. Data Collection

I created a developers account at the New York Times (NYT) website and created a key connected to my account. This key will be used to access data from the NYT site.

```
API.Key <- "dk5upqVbE5VhXzCZwacqGB5O17iAyGKW"
```

Using the Article Search API I will download news article abstracts from the previous two years with the following search terms:

Term | Ecuador Begin Date | 2017-10-20 End Date | 2019-10-20

We then can generate a search string "MySearch.url" to use with the the NYT API.

```
term <- "ecuador" # Search Term
begin_date <- "20171020" #Begin Date
end_date <- "20191020"    #End Date
MySearch.url <- paste0("http://api.nytimes.com/svc/search/v2/articlesearch.json?q=",term,
                  "&begin_date=",begin_date,"&end_date=",end_date,
                  "&facet_filter=true&api-key=",API.Key, sep="")
```

Every search string will have a different number of hits or articles. We will find how many hits and using this number the number of search results pages for our chosen search string.

```
GetHits <- fromJSON(MySearch.url)
PageNumber <- round((GetHits$response$meta$hits[1] / 10)-1)
```

Then we can request the calculated number of search pages and load them into a list.

```
pages <- list()
for(i in 0:PageNumber){
  nytSearch <- fromJSON(paste0(MySearch.url, "&page=", i), flatten = TRUE) %>% data.frame()
  #message("Retrieving page ", i)
  pages[[i+1]] <- nytSearch
  Sys.sleep(10)
}
```

Combine the list of search results pages into a single data frame.

```
allNYTSearch <- rbind_pages(pages)
```

Let's do some data cleanup by only keeping rows of interest and changing the format of the datetime column from String to Date.

```
allNYTSearch$response.docs.pub_date <- as.POSIXct(allNYTSearch$response.docs.pub_date,format="%Y-%m-%d"
short_df <- select(allNYTSearch, response.docs.abstract, response.docs.pub_date, response.docs.word_cou

short_df$response.docs.pub_date <- as.Date(short_df$response.docs.pub_date)
```

The results are full of words that do not add anything to our analysis. The so called "stop words". Since our article is from a Spanish speaking country, we will also need to account for spanish "stop words"

```
data("stop_words")
spanish_stop_words <- bind_rows(stop_words,
                               data_frame(word = tm::stopwords("spanish"),
                                          lexicon = "custom"))
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

Let's divide the data set into groups based based on the year (2018 and 2019) using the "as.Date" function.

```
library(stringr)

NYT_2018 <- short_df %>% filter(between(response.docs.pub_date, as.Date("2018-01-01"), as.Date("2018-12
NYT_2019 <- short_df %>% filter(between(response.docs.pub_date, as.Date("2019-01-01"), as.Date("2019-10

dim(NYT_2018); dim(NYT_2019)
```

```
## [1] 275   4
```

```
## [1] 200   4
```

### 2. Data Analysis

The following two chunks will break up the news summaries into words. Will remove english and spanish "stop words". Then wil rank every word by a "sentiment" and finally will order the results on the frequency at which the words appear.

```
# get a list of words
NYT_Clean_2018 <- NYT_2018 %>%
  dplyr::select(response.docs.abstract) %>%
  unnest_tokens(word, response.docs.abstract) %>%
  anti_join(stop_words) %>%
  anti_join(spanish_stop_words) %>%
  filter(!word %in% c("del", "las", "la", "de", "tu", "noticias", "en" , "el", "lo", "esta", "sucediend
## Joining, by = "word"
## Joining, by = "word"
```

```
NYT_Clean_2018 %>%
  count(word, sort = TRUE) %>%
  top_n(15) %>%
  mutate(word = reorder(word, n))
```

```
## Selecting by n
```

```
## # A tibble: 16 x 2
##    word           n
##    <fct>      <int>
##  1 américa       24
##  2 latina        24
##  3 mundo         24
##  4 day           17
##  5 world         16
##  6 here’s        15
##  7 life          14
##  8 president     13
##  9 start         12
## 10 trump         12
## 11 loved         11
## 12 american      10
## 13 immigrants    10
## 14 countries      9
## 15 ecuador        9
## 16 national       9
```

```
# join sentiment classification to the abstract words
bing_word_counts_2018 <- NYT_Clean_2018 %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

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

```
bing_word_counts_2018 <- bing_word_counts_2018 %>%
  group_by(sentiment) %>%
  top_n(5) %>%
  ungroup() %>%
  mutate(word = reorder(word, n))
```

```
## Selecting by n
```

```
# get a list of words
NYT_Clean_2019 <- NYT_2019 %>%
  dplyr::select(response.docs.abstract) %>%
  unnest_tokens(word, response.docs.abstract) %>%
  anti_join(stop_words) %>%
  anti_join(spanish_stop_words) %>%
  filter(!word %in% c("del", "las", "la", "de", "tu", "noticias", "en" , "el", "lo", "esta", "sucediend
## Joining, by = "word"
## Joining, by = "word"
```

```
NYT_Clean_2019 %>%
  count(word, sort = TRUE) %>%
  top_n(15) %>%
  mutate(word = reorder(word, n))
```

```
## Selecting by n
```

```
## # A tibble: 18 x 2
##    word          n
##    <fct>     <int>
##  1 américa      23
##  2 latina       23
##  3 mundo        23
##  4 president    18
##  5 fuel         11
##  6 venezuela    11
##  7 week         11
##  8 wikileaks    11
##  9 day           9
## 10 here’s        9
## 11 million       9
## 12 protests      9
## 13 quito         9
## 14 united        9
## 15 ecuador       8
## 16 embassy       8
## 17 founder       8
## 18 maduro        8
```

```
# join sentiment classification to the abstract words
bing_word_counts_2019 <- NYT_Clean_2019 %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

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

```
bing_word_counts_2019 <- bing_word_counts_2019 %>%
  group_by(sentiment) %>%
  top_n(5) %>%
  ungroup() %>%
  mutate(word = reorder(word, n))
```

```
## Selecting by n
```

Let’s plot and compare side by side the sentiment analysis of the news article abstracts of the years 2018 vs 209 connected to the search term “ecuador”

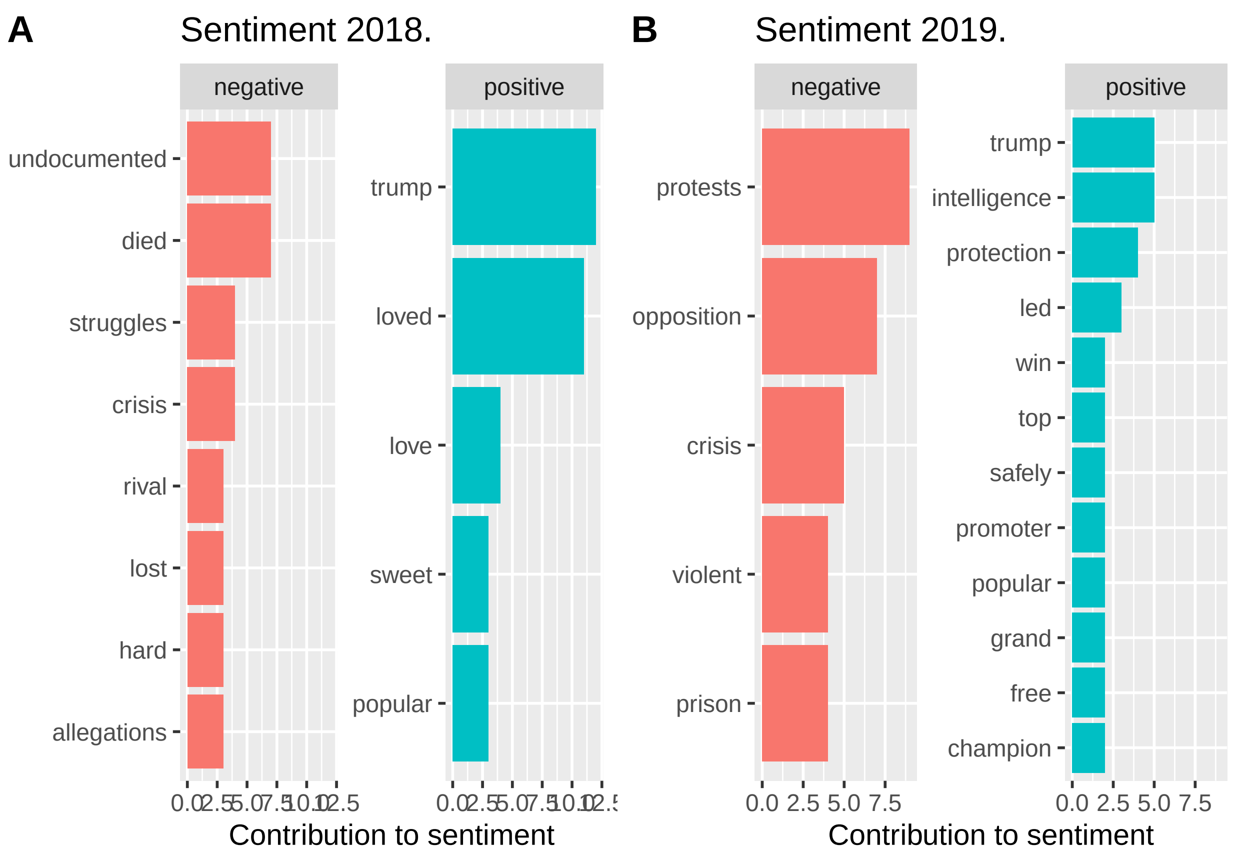
```r
library(cowplot)
```

```
##
## ********************************************************
## Note: As of version 1.0.0, cowplot does not change the
##   default ggplot2 theme anymore. To recover the previous
##   behavior, execute:
##   theme_set(theme_cowplot())
## ********************************************************
```

```r
Y2019 <- ggplot(bing_word_counts_2019,aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(title = "Sentiment 2019.",
       y = "Contribution to sentiment",
       x = NULL) + coord_flip()

Y2018 <- ggplot(bing_word_counts_2018,aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(title = "Sentiment 2018.",
       y = "Contribution to sentiment",
       x = NULL) + coord_flip()

plot_grid(Y2018, Y2019, labels = "AUTO")
```



## 3. Conclusions

We have shown the qualitavely difference in sentiment in Ecuador news between the years 2018 and 2019. This matches the local context of the country in which the year 2018 was a year when the immigration of undocumented immigrants from Venezuela into Ecuador was a major concern. This compares to the year 2019 when the country suffered a series of protest due to the reduction of fuel subsidies that hit the economically deprived segment of the Ecuadorian population.