textRank

jaume cloquell capo

21 de mayo de 2019

Text Rank

Comenzamos cargando los paquetes apropiados, que incluyen tidyverse para tareas generales, tidyverse para manipulaciones de texto, textrank para la implementación del algoritmo TextRank y finalmente rvest para raspar un artículo para usarlo como ejemplo. El github para el paquete textrank se puede encontrar https://github.com/bnosac/textrank

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.1.0     v purrr   0.3.0
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidytext)
library(textrank)
library(rvest)
```

```
## Loading required package: xml2
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:purrr':
##
##     pluck
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(lexRankr)
```

```
##
## Attaching package: 'lexRankr'
```

```
## The following object is masked from 'package:readr':
##
##     tokenize
```

Para mostrar este método he seleccionado al azar un artículo de nuestro diario nacional "elmundo" El cuerpo principal se selecciona utilizando los html_nodes.

a continuación cargamos el artículo en un tibble (ya que tidytext requería la entrada como data.frame). Comenzamos por tokenize según las frases, lo que se hace estableciendo token = "sentences" en unnest_tokens. La tokenización no siempre es perfecta con este tokenizador, pero tiene un número bajo de dependencias y es suficiente para este trabajo. Por último añadimos la columna de número de frase y cambiamos el orden de las columnas (textrank_sentences prefiere las columnas en un orden determinado).

```
article_sentences <- tibble(text = article) %>%
  unnest_tokens(sentence, text, token = "sentences") %>%
  mutate(sentence_id = row_number()) %>%
  select(sentence_id, sentence)
```

a continuación haremos un token de nuevo, pero esta vez para conseguir palabras. Al hacer esto, mantendremos la columna sentence_id en nuestros datos.

```
article_words <- article_sentences %>%
  unnest_tokens(word, sentence)
article_words
```

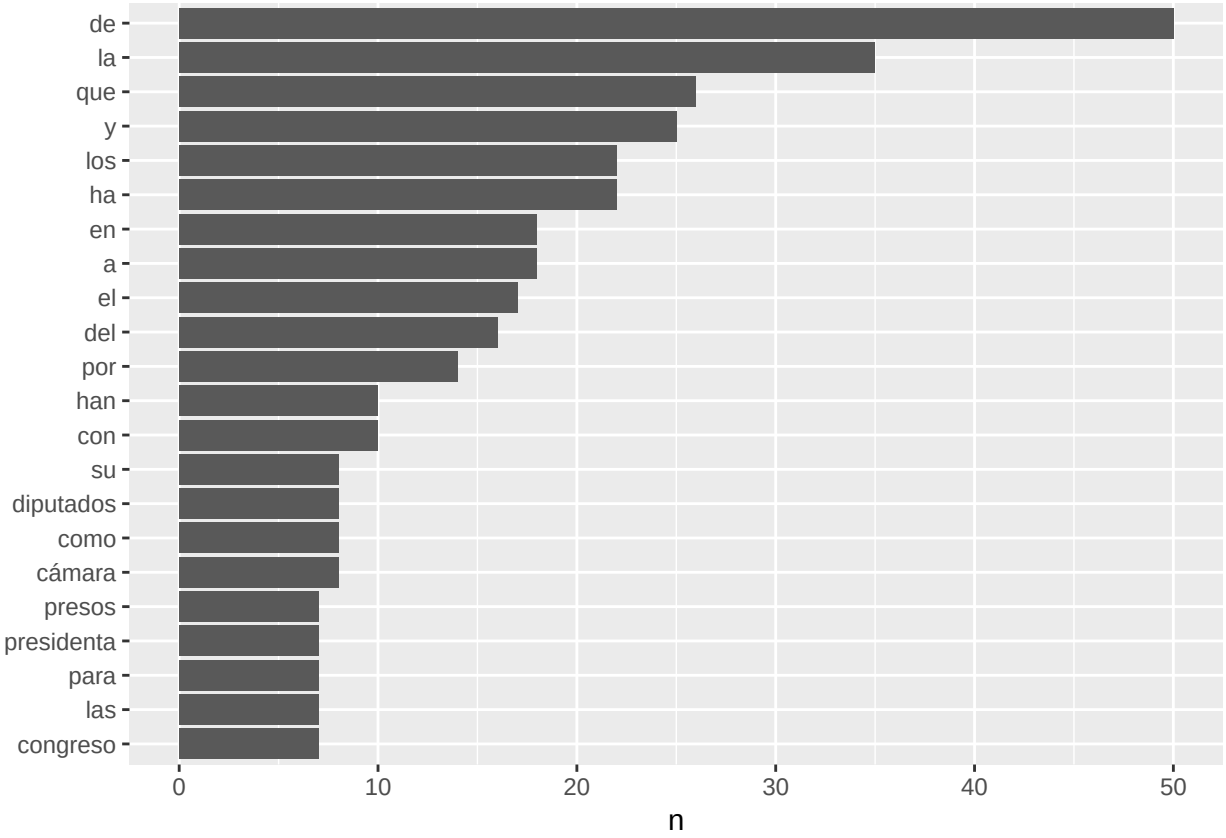
```
## # A tibble: 822 x 2
##    sentence_id word
##          <int> <chr>
##  1           1 xiii
##  2           1 legislatura
##  3           2 meritxell
##  4           2 batet
##  5           2 elegida
##  6           2 nueva
##  7           2 presidenta
##  8           2 del
##  9           2 congreso
## 10           2 de
## # ... with 812 more rows
```

```
article_words %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 365 x 2
##    word      n
##    <chr> <int>
##  1 de       50
##  2 la       35
##  3 que      26
##  4 y        25
##  5 ha       22
##  6 los      22
##  7 a        18
##  8 en       18
##  9 el       17
## 10 del      16
## # ... with 355 more rows
```

```
library(ggplot2)

article_words %>%
  count(word, sort = TRUE) %>%
  filter(n > 6) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```



ahora tenemos todas las entradas suficientes para la función textrank_sentences. Sin embargo, iremos un paso más allá y eliminaremos las palabras de stop en article_words ya que aparecerían en la mayoría de las frases y realmente no contienen ninguna información en sí mismas.

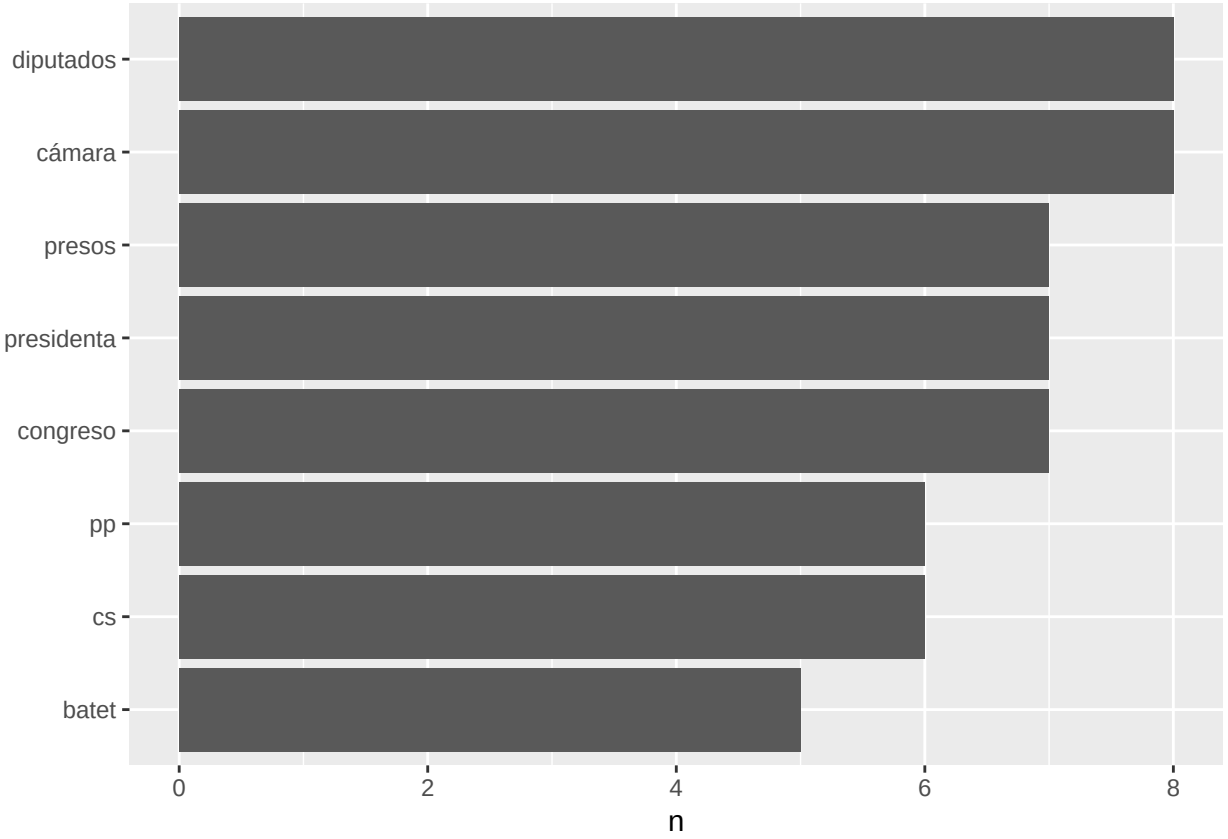
```
article_words <- article_words %>%
  anti_join(data_frame(word = stopwords(kind = "es")), by = "word")
```

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

Si volvemos a visualizar el gráfico anterior podemos observar como hemos eliminado las palabras que no aportaban valor a las frases, tales como los artículos y conjunciones.

```
article_words %>%
  count(word, sort = TRUE) %>%
  filter(n > 4) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
```

```
  geom_col() +
  xlab(NULL) +
  coord_flip()
```



Ejecutamos el algoritmo TextRank sólo requiere 2 entradas.

Un marco de datos con frases Un data.frame con tokens (en nuestro caso palabras) que forman parte de cada frase. Así que estamos listos para correr

```
article_summary <- textrank_sentences(data = article_sentences,
                                      terminology = article_words)
article_summary
```

```
## Textrank on sentences, showing top 5 most important sentences found:
##   1. meritxell batet, elegida nueva presidenta del congreso de los diputados  cámara alta.
##   2. ahora entendemos por qué los golpistas querían una presidenta del congreso como ésta y un presi
##   3. "los independentistas han utilizado la cámara para dar un mitin y la presidenta del congreso ha
##   4. batet ha dado por hecho que todos los parlamentarios han prometido la carta magna acomodándose a
##   5. hoy hemos tenido que soportar que algunos diputados utilizaran la expresión presos políticos y s
```

Si bien el método de impresión es bueno, podemos extraer la información para un buen análisis posterior. La información sobre las frases se almacena en frases. Incluye la información article_sentences más la puntuación de textrank calculada. Si miramos el artículo a lo largo del tiempo, sería interesante ver dónde aparecen las frases importantes. En el gráfico siguiente podemos observa que frases poseenmayor score.

```
article_summary[["sentences"]] %>%
  ggplot(aes(textrank_id, textrank, fill = textrank_id)) +
  geom_col() +
  theme_minimal() +
  scale_fill_viridis_c() +
```

```
  guides(fill = "none") +
  labs(x = "Sentence",
       y = "TextRank score")
```