

Procesamiento de Lenguaje Natural con R



FOTH

Víctor Sanz, PhD

 victor.sanz@ugr.es

 @vsslledo

 VictorSuarezL



UNIVERSIDAD
DE GRANADA

Índice:

Miércoles 5 de feb

1. Repaso Tidyverse:

1. El universo tidyverse
2. Principales operaciones en tidyverse/dplyr

2. Obtención de datos:

1. API's
2. Web Scraping

3. Las características del lenguaje natural:

1. Expresiones Regulares
2. Bag of Words

Jueves 6 de feb

1. Exploración de datos textuales
2. Clasificación de temáticas
3. Análisis de datos
4. Análisis basado en redes

API de Redes Sociales

API: Application Programming Interface.

Existen dos métodos para acceder a los datos usando APIs:

1. **Stream API:**

- Proporciona un flujo de mensajes en tiempo real pueden filtrarse por una serie de características, siendo las más habituales: palabras clave, usuarios y localizaciones.
- También puede obtenerse una muestra aleatoria (~1%).
- Limitaciones: Sólo en tiempo real.

2. **REST API:**

- Proporciona información sobre el perfil de usuarios, timeline, lista de usuarios y seguidores, etc.
- Limitaciones: suele tener limitaciones de cantidad de mensajes o limitaciones temporales.

-> Ojo! Twitter ha sido probablemente una de las APIs más populares a la hora de analizar redes sociales. Debido a las nuevas políticas la mayoría de sus funcionalidades son de pago, sin embargo existen otras plataformas como Mastodon, Reddit, etc... con APIs para probar

Variables obtenidas

- **user_id:** ID del usuario.
- **status_id:** ID del status o tweet.
- **created_at:** Fecha de creación del tweet.
- **screenName:** Nombre de pantalla del usuario.
- **text:** Texto dentro del status o tweet.
- **source:** Clase de dispositivo.
- **reply_to_status_id:** ID del status al que responde.
- **reply_to_user_id:** ID del usuario al que responde.
- **reply_to_screen_name:** Nombre de pantalla al que responde.
- **is_retweet:** TRUE si el status o tweet ha sido retwuiteado.
- **favorite_count:** Número de veces el status o tweet ha sido marcado como favorito.
- **retweet_count:** Número de veces el status o tweet ha sido retwuiteado.



Características Stream API

Documentación: [link](#).

Es la manera más recomendada para extraer la información:

- 400 palabras clave
- 5,000 id de usuarios
- 25 location boxes¹

Problemas:

- Puede desconectarse con facilidad
- Es recomendable guardar la información obtenida con regularidad
- Programar script para reiniciarse cada hora



Características Rest API

Documentación: [link](#)

Es la manera de obtener mayor tipo de información:

- Timeline
- Información del usuario
- Lista de friends
- Lista de followers

Problemas:

- Las cantidad de búsqueda está limitada
- Retroceder en el tiempo es complicado

 Sampling bias?

Sampling bias?

- **Morstatter, F., Pfeffer, J., Liu, H., & Carley, K. M.** (2013). Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose. [Link](#)
- **González-Bailón, S., Wang, N., Rivero, A., Borge-Holthoefer, J., & Moreno, Y.** (2014). Assessing the bias in samples of large online networks. Social Networks. [Link](#)
- **Driscoll, K., & Walker, S.** (2014). Working within a black box: Transparency in the collection and production of big twitter data. International Journal of Communication. [Link](#)
- **Joseph, K., Landwehr, P. M., & Carley, K. M.** (2014). Two 1%s Don't make a whole: Comparing simultaneous samples from Twitter's Streaming API. En Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). [Link](#)
- **Steinert-Threlkeld, Z.** (2018). Twitter as Data (Elements in Quantitative and Computational Methods for the Social Sciences). Cambridge: Cambridge University Press. doi:10.1017/9781108529327

Sampling bias?

Consideraciones

- La muestra aleatoria del 1% no es realmente aleatoria.
- Los hashtags, usuarios, topics menos populares son menos probables de entrar en la muestra.
- Para las búsquedas realizadas por keyword el sesgo no es tan importante.
- Las muestras pequeñas obtenidas a través de un conjunto de hashtags pueden estar sesgadas.
- Los datos obtenidos a través de la REST API están más sesgados que los obtenidos a través de la Streaming API.

Web Scraping

Web Scraping

¿En qué consiste el web scraping?

- Técnica para extraer datos de páginas web.
- Automatiza la recopilación de información.
- Utiliza código para "leer" el contenido HTML.
- Permite transformar datos no estructurados en estructurados.

Web Scraping

¿Para qué resulta útil el web scraping?

- Investigación académica: Recolección de grandes volúmenes de datos.
- Análisis de mercado: Seguimiento de precios, productos, tendencias.
- Periodismo de datos: Extracción de información de fuentes públicas.
- Automatización: Actualización de bases de datos de forma automática.

Web Scraping

¿Cómo funciona el web scraping?

- Enviar petición HTTP al servidor web (GET).
- Recibir el HTML de la página.
- Analizar el HTML usando selectores CSS o XPath.
- Extraer los datos deseados.
- Almacenar la información en CSV, JSON, bases de datos, etc.

Web Scraping

Consideraciones legales:

- Revisar el archivo robots.txt del sitio web.
 - Cumplir con los términos de servicio de la página.
 - Evitar sobrecargar servidores (buenas prácticas).
 - Consideraciones de propiedad intelectual y privacidad.
-

Herramientas populares para Web Scraping

- Python: BeautifulSoup, Scrapy, Selenium.
- R: rvest, httr, xml2.
- Javascript: Puppeteer, Cheerio.

Web Scraping

Un ejemplo usando **rvest**

Ejemplo: [link](#).

```
library(rvest)

# URL de la página web (usa tu propia URL si estás trabajando en local)
url ← "https://quotes.toscrape.com/" # Ejemplo de página similar

# Leer el contenido HTML de la página
webpage ← read_html(url)

# Extraer las citas utilizando los selectores CSS
quotes ← webpage %>%
  html_nodes(".quote .text") %>% # Selecciona el texto dentro de la clase 'quote'
  html_text() # Extrae el texto puro

# Ver las citas extraídas
print(quotes)
```

Web Scraping

Un ejemplo usando **rvest**

```
#> [1] "“The world as we have created it is a process of our thinking. It cannot be changed without chang
#> [2] "“It is our choices, Harry, that show what we truly are, far more than our abilities.”"
#> [3] "“There are only two ways to live your life. One is as though nothing is a miracle. The other is a
#> [4] "“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably s
#> [5] "“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absol
#> [6] "“Try not to become a man of success. Rather become a man of value.”"
#> [7] "“It is better to be hated for what you are than to be loved for what you are not.”"
#> [8] "“I have not failed. I've just found 10,000 ways that won't work.”"
#> [9] "“A woman is like a tea bag; you never know how strong it is until it's in hot water.”"
#> [10] "“A day without sunshine is like, you know, night.”"
```

Web Scraping

Un ejemplo usando **rvest**

```
webpage ← read_html(url)

# Extraer los nombres de los autores
authors ← webpage %>%
  html_nodes(".author") %>% # Selecciona los elementos con la clase 'author'
  html_text() # Extrae el texto de esos elementos

# Mostrar los autores extraídos
print(authors)
```

Web Scraping

Un ejemplo usando **rvest**

```
#> [1] "Albert Einstein" "J.K. Rowling" "Albert Einstein"
#> [4] "Jane Austen" "Marilyn Monroe" "Albert Einstein"
#> [7] "André Gide" "Thomas A. Edison" "Eleanor Roosevelt"
#> [10] "Steve Martin"
```

Trabajar con cadenas de texto



¿Para qué sirve?

Documentación: [link](#)

- Manejo de cadenas de texto de forma sencilla y eficiente.
- Sintaxis clara y consistente con el tidyverse.
- Facilita tareas de limpieza, transformación y extracción de texto.
- Optimizado para trabajar con datos textuales en análisis de datos y minería de texto.



Funciones más comunes:

- Búsqueda y detección de patrones (`str_detect`, `str_which`)
- Extracción de texto específico (`str_extract`, `str_extract_all`)
- Sustitución y limpieza de texto (`str_replace`, `str_replace_all`)
- División y unión de texto (`str_split`, `str_c`)
- Manipulación de longitud y formato (`str_trim`, `str_pad`, `str_to_lower`, `str_to_upper`)



Act. 1:

En el archivo `covid_19.rds` os he dejado una muestra de tweets sobre la conversación que tuvo lugar a principios del Covid. ¿Cómo podríamos saber cuáles son las personas más mencionadas en esos tweets?

Act. 2:

¿Y cuáles son los hashtags más usados?



Act. 1:

```
library(stringr)

read_rds("./data/covid_19.rds") %>%
  select(screen_name, text, created_at) → df

df %>%
  filter(str_detect(text, "@")) %>%
  mutate(text = str_extract_all(text, "@\\w+", simplify = F)) %>%
  unnest(text) %>%
  count(text, sort = T)
```

StringR

```
#> # A tibble: 574 × 2
#>   text                n
#>   <chr>             <int>
#> 1 @realDonaldTrump    57
#> 2 @RealJamesWoods    45
#> 3 @LivePDDave1       31
#> 4 @DonaldJTrumpJr    16
#> 5 @RealCandaceO      15
#> 6 @Heat_Miser2       14
#> 7 @SKYRIDER4538      14
#> 8 @ScottFordTVGuy    12
#> 9 @Julietknows1      11
#> 10 @TomFitton        11
#> # i 564 more rows
```

TidyText

TidyText

¿Cuáles son las principales ventajas de TidyText?

- Estructura ordenada y limpia (formato tidy, una palabra por línea)
- Integración con el ecosistema de tidyverse
- Facilita el análisis de NLP (Tokenización, EDA, stopwords, análisis de sentimiento, topic model)
- Optimizado para el procesamiento de grandes volúmenes de datos
- Compatible con otros paquetes textdata, quanteda, tm, wordcloud

TidyText

Primeros pasos con TidyText: unnest_tokens

```
library(tidytext)
library(tidytext)

df <- tibble(text = c("Text analysis with R", "tidytext is useful"))

df %>%
  unnest_tokens(word, text) # Divide el texto en palabras
```

```
#> # A tibble: 7 × 1
#>   word
#>   <chr>
#> 1 text
#> 2 analysis
#> 3 with
#> 4 r
#> 5 tidytext
```

TidyText

Primeros pasos con TidyText: stopwords

```
data("stop_words")  
  
df %>%  
  unnest_tokens(word, text) %>%  
  anti_join(stop_words)
```

```
#> # A tibble: 3 × 1  
#>   word  
#>   <chr>  
#> 1 text  
#> 2 analysis  
#> 3 tidytext
```

TidyText

Primeros pasos con TidyText: **get_sentiments**

```
df %>%  
  unnest_tokens(word, text) %>%  
  inner_join(get_sentiments("bing")) %>%  
  # inner_join(get_sentiments("nrc"))  
  count(sentiment, sort = TRUE)
```

```
#> # A tibble: 1 × 2  
#>   sentiment      n  
#>   <chr>      <int>  
#> 1 positive      1
```

TidyText

Primeros pasos con TidyText: **get_sentiments**

```
df %>%  
  unnest_tokens(ngram, text, token = "ngrams", n = 2)
```

```
#> # A tibble: 5 × 1  
#>   ngram  
#>   <chr>  
#> 1 text analysis  
#> 2 analysis with  
#> 3 with r  
#> 4 tidytext is  
#> 5 is useful
```


TidyText