Recuperación y análisis de texto con R

Clase 2 - Educación Permanente FCS

Mag. Elina Gómez (UMAD)

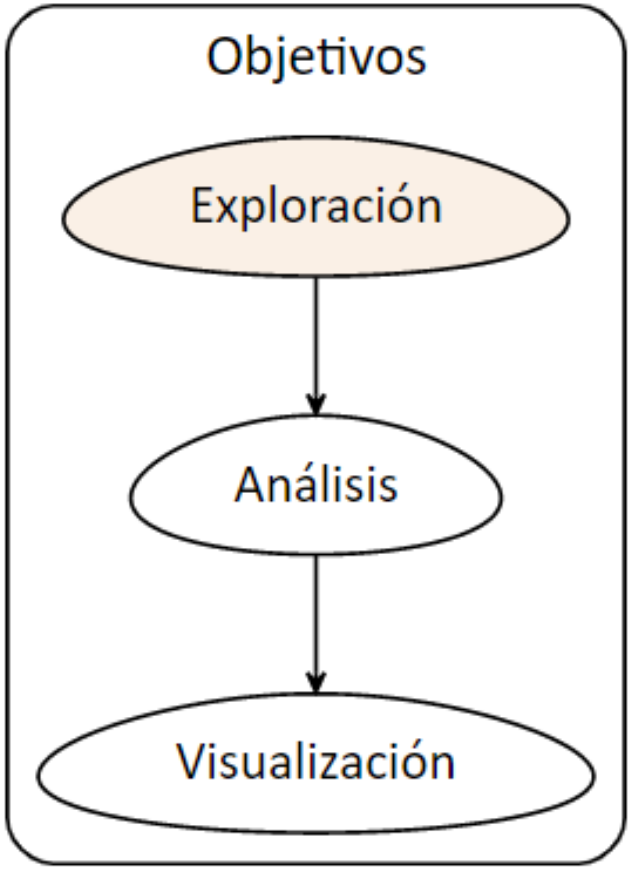
elina.gomez@cienciassociales.edu.uy

www.elinagomez.com

Mag. Gustavo Méndez Barbato

gustavo.mendez@cienciassociales.edu.uy

Objetivos de hoy



Objetivos de hoy

- Fuentes de datos

Fuentes de datos

Las fuentes de datos que vamos a ver son:

1. Recuperación de documentos en imagen o pdf (OCR)
2. Scraping web y parlamentario
3. Prensa digital
4. Audio
5. YouTube
6. APIs de redes sociales

1. Recuperación de documentos en imagen o pdf (OCR)

Existen diferentes librerías de R que nos permiten recuperar documentos en diferentes formatos:

- readtext
- pdftools

tesseract OCR

Tesseract es un motor de OCR (*reconocimiento óptico de caracteres*) para varios sistemas operativos. Es software libre, liberado bajo la licencia Apache, Versión 2.0 y su desarrollo es financiado por Google desde el 2006.

Acá se encuentra la documentación, cuenta con *más de 100 idiomas.*

tesseract OCR

Existe un paquete de R bien documentado que se llama *tesseract* y que cuenta con funciones que permiten el reconocimiento de caracteres incluso en español, descargando una base de entrenamiento del motor.

tesseract OCR:

Descargo un documento histórico del repositorio Internet Archive

```
##Chequear los idiomas disponibles
tesseract_info()
# Bajar por unicamente español para entrenar
tesseract_download("spa")
# asignar
(espanol <- tesseract("spa"))
#Probamos:
transcribopdf <- ocr("analesUruguay.pdf", engine = espanol)
```

tesseract OCR

La función *ocr_data()* devuelve una tabla dónde cada fila es una palabra con la confianza asociada a la misma y la ubicación exacta.

magick

El paquete *magick* complementa a *tesseract* en cuento a mejora de la calidad de las imagenes que sirven de input. Cuenta con varias funciones para mejorar la resolución, el color, contraste, espacios en blanco. Puede ser utilizado como paso previo.

2. Web scraping

¿Qué es web scraping?

Web scraping es una **técnica** para obtener datos no estructurado (etiquetas HTML) de una página web, a un formato estructurado de columnas y renglones, en los cuales se puede acceder y usar más fácilmente.

2. Web scraping

¿Para qué sirve Web scraping?

- Obtener datos de texto.
- Consolidar datos de redes sociales o extraer comentarios de usuarios/as.
- Precios de tiendas online, a través del análisis histórico de la competencia.
- Búsqueda en Google de diversas palabras clave.
- Etiquetas de imágenes, para clasificación de imágenes.

2. Scraping web y parlamentario

En el curso vamos a ver tres formas de Web scraping:

- Paquete *rvest*
- Paquete *speech* (Uruguay)
- Gdelt project

rvest

rvest es un paquete para scraping (raspado) y análisis web de Hadley Wickham.

Documentación

- Tutorial recomendado de Riva Quiroga (Chile)

https://programminghistorian.org/es/lecciones/introduccion-al-web-scraping-usando-r

¿Cómo usar rvest?

Para usar rvest, se requiere conocer las instrucciones en código, a las que llamaremos funciones, para para hacer las tareas más comunes en la extracción y manipulación de datos web.

- `read_html(«url»)` con esta función se crea un objeto que contiene todo el código o etiquetas HTML.
- `html_elements(«objeto html», «etiqueta css»)` se usa para seleccionar partes del objeto que contiene todo el código html. El segundo parámetros es la clase CSS que está relacionada con la sección que deseamos extraer.

¿Cómo usar rvest?

- `html_name()` obtiene los atributos html
- `html_text()` extrae el texto html
- `html_attr()` regresa los atributos específicos html (ej. href)
- `html_attrs()` obtiene los atributos html
- `html_table()` convierte una tabla html en una estructura de datos en R

Ejemplo rvest: texto

Descargo la extensión del SelectorGadget de Chrome e instalo y busco el nombre del nodo o elementos en una pagina que me interese scrapear

Ejemplo rvest: texto

```
library(rvest)
library(dplyr)

#Defino mi sitio html: Montevideo portal
mvdportal = read_html("https://www.montevideo.com.uy/index.html")

resumenes = mvdportal %>%
  html_elements(".text")%>% #defino los elementos que identifiqué con el SelectorGadget
  html_text()

titulares = mvdportal %>%
  html_elements("a")%>%
  html_text()
```

Ejemplo rvest: texto

Un ejemplo concreto para el caso uruguayo !

Ejemplo rvest: tabla

```
url <- 'https://en.wikipedia.org/wiki/R_(programming_language)'

url %>% read_html() %>%
        html_elements(css = '.wikitable') %>%
        html_table()
```

speech

The `speech` package

Nicolás Schmidt, Diego Luján, Juan Andrés Moraes

Description

Converts the floor speeches of Uruguayan legislators, extracted from the parliamentary minutes, to tidy data.frame where each observation is the intervention of a single legislator.

Installation

```
# Install speech from CRAN
install.packages("speech")

# The development version from GitHub:
if (!require("remotes")) install.packages("remotes")
remotes::install_github("Nicolas-Schmidt/speech")
```

speech

El paquete speech convierte los diarios de sesiones legisladorxs uruguayxs, en un marco de datos ordenado donde cada observación es la intervención de unx solx legisladorx.

Acá se encuentra la documentación del paquete con descripción de las funciones y argumentos.

speech

```
##Recomiendo instalar versión en desarrollo:

if (!require("remotes")) install.packages("remotes")
remotes::install_github("Nicolas-Schmidt/speech")

library(speech)
```

speech

```
url <- "http://bit.ly/35AUVF4"

sesion <- speech_build(file = url)
```

speech

```
#Función completa

sesion <- speech_build(file = url,
#url a pdf
compiler = FALSE,
#compila discursos de unx mismx legisladorx
quality = TRUE,
#aporta dos índices de calidad
add.error.sir = c("SEf'IOR"),
##forma errónea que lo que identifica a el/la legisladorx
rm.error.leg = c("PRtSIDENTE", "SUB", "PRfSlENTE"))
##identifica a el/la legisladorx que debe eliminarse
```

speech

Variables que incluye la tabla ordenada:

- legislator: nombre
- speech: discurso/s
- date: fecha de sesión
- id: identificador
- legislature: número de legislatura
- chamber: cámara del documento (representantes, senadores, asamblea general, comisión permanente)

Si quality es TRUE:

- index_1: index_1. Proporción del documento recuperado con respecto al original.
- index_2: index_2. Proporción del documento final en función del recuperado. Proporción del documento donde hay intervenciones de lxs legisladorxs.

puy

- Es posible combinar con el paquete *puy* para recuperar el dato del partido político al que pertenece

speech App

- Existe una Shiny de speech que permite descargar de forma tabulada las sesiones sin escribir código: https://bancodedatos-fcs.shinyapps.io/shiny_speech/

3. Prensa digital

Monitor de prensa

- Existe un monitor de prensa (en Twitter) que permite descargar http://137.184.138.178
- Desarrollada por Leandro Domínguez, Guillermo Eijo y Sebastian Felix en el marco del proyecto de grado "Análisis de publicaciones sobre seguridad ciudadana en redes sociales" (FING-Udelar) - Agosto 2022
- Acumula desde enero 2009. Tiene tres módulos: Indicadores, Entidades y Cluster.

Proyecto Gdelt

El proyecto GDELT cuenta con *una base de datos global de la sociedad que monitorea las noticias de impresas y web del mundo desde casi todos los rincones de cada país en más de 100 idiomas e identifica las personas, ubicaciones, organizaciones, temas, fuentes, emociones, recuentos, citas, imágenes y eventos que impulsan nuestra sociedad global cada segundo de cada día, creando una plataforma abierta y gratuita para la informática en todo el mundo.*

Proyecto Gdelt

Existe un paquete de R llamado gdeltr2 que no se encuentra bien documentado pero que cuenta con mucho potencial. Las consultas a la base pueden hacerse también desde Big Query de Google y procesamiento posterior en R.

Proyecto Gdelt

- GDELT Events Database [EVENTS]: Global Events, 1979 to present.
- GDELT Global Knowledge Graph [GKG] : GDELT's Knowledge Graph, April 2013 to present.
- GDELT Full Text API [Full Text API]: Full text search for all monitored sources within a 24 hour window. Output includes raw data, sentiment, and word counts.
- GDELT Visual Knowledge Graph VGKG: Google Cloud Vision API output for every indexed piece of GKG media.

Proyecto Gdelt

Proyecto reciente en Argentina usando Gdelt para obtener noticias sobre antivacunismo.

gdelt2

Tutorial

Instalación:

```
devtools::install_github("hadley/devtools")
devtools::install_github("hafen/trelliscopejs")
devtools::install_github("abresler/gdeltr2")
```

gdelt2

El mode *ArtList* recupera todo los artículos que tienen esa mención en un determinado tiempo. Está restringido a 250 resultados y 52 semanas. Para hacer búsquedas combinadas: ‘ “Lacalle Pou” covid’

```
articulos  = gdeltr2::ft_v2_api(
  terms = c("Lacalle Pou"),
  modes = c("ArtList"),
  visualize_results = F,
  timespans = "55 days",
  source_countries = "UY")
```

gdelt2

El mode *TimelineVol* recupera una métrica diaria de la intensidad del volumen de los artículos que coinciden con una búsqueda específica.

```
intensidad = gdeltr2::ft_v2_api(
  terms = c("Lacalle Pou"),
  modes = c("TimelineVol"),
  visualize_results = F,
  timespans = "55 days",
  source_countries = "UY"
)
```

gdelt2

El mode *TimelineVol* recupera una métrica diaria de la intensidad del volumen de los artículos que coinciden con una búsqueda específica. El mode *TimelineVolInfo* es igual pero con información anexa y desagregada para cada artículo.

```
intensidad = gdeltr2::ft_v2_api(
  terms = c("Lacalle Pou"),
  modes = c("TimelineVol"),
  visualize_results = F,
  timespans = "55 days",
  source_countries = "UY"
)
```

gdelt2

El mode *TimelineTone* recupera el tono (positivo y negativo) de los artículos que coniciden con la búsqueda, por día.

```
tono_diario = gdeltr2::ft_v2_api(
  terms = c("Lacalle Pou"),
  modes = c("TimelineTone"),
  visualize_results = F,
  timespans = "30 days",
  source_countries = "UY"
)
```

gdelt2

El mode *ToneChart* recupera el tono (positivo y negativo) de los artículos que coniciden con la búsqueda, por artículo.

```
prueba4 = gdeltr2::ft_v2_api(
  terms = c("Lacalle Pou"),
  modes = c("ToneChart"),
  visualize_results = F,
  timespans = "30 days",
  source_countries = "UY"
)
```

gdelt2

Últimos términos, lugares, personas, cosas, de los últimos 15 minutos a nivel mundial.

```
ultimo = gdeltr2::ft_trending_terms()
```

gdelt2

Tablas de inestabilidad con variables *'instability'*, *'tone'*, *'protest'*, *'conflict'*,*'artvolnorm'* . Es posible visualizar gráficos.

```
inestabilidad_zona <-
  gdeltr2::instability_api_locations(
    location_ids = c("UY"),
    use_multi_locations = c(T, F),
    variable_names = c('instability', 'tone', 'protest', 'conflict','artvolnorm'),
    time_periods = c('daily'),
    nest_data = F,
    days_moving_average = NA,
    return_wide = T,
    return_message = T,
    visualize = T
  )
```

gdelt2

Por último, recuperar temas pre-calasificados con AA (IA), hay 59840.

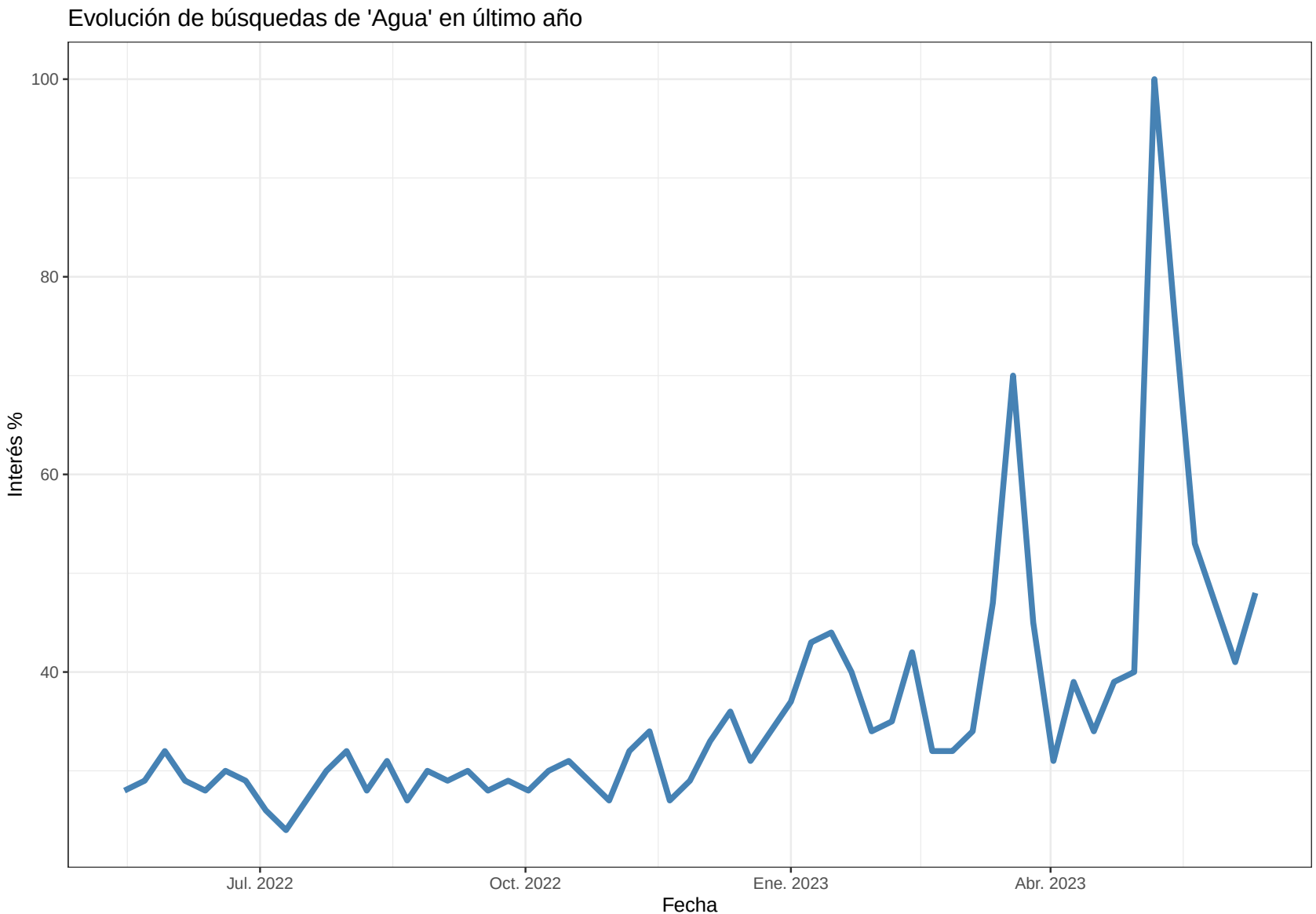
```
##cargo códigos de temas
df_gkg <-
  gdeltr2::dictionary_ft_codebook(code_book = "gkg")


tema =  ft_v2_api(gkg_themes = "WB_2901_GENDER_BASED_VIOLENCE",modes = c("Artlist"),
                  visualize_results = F,
                  timespans = "55 days")
```

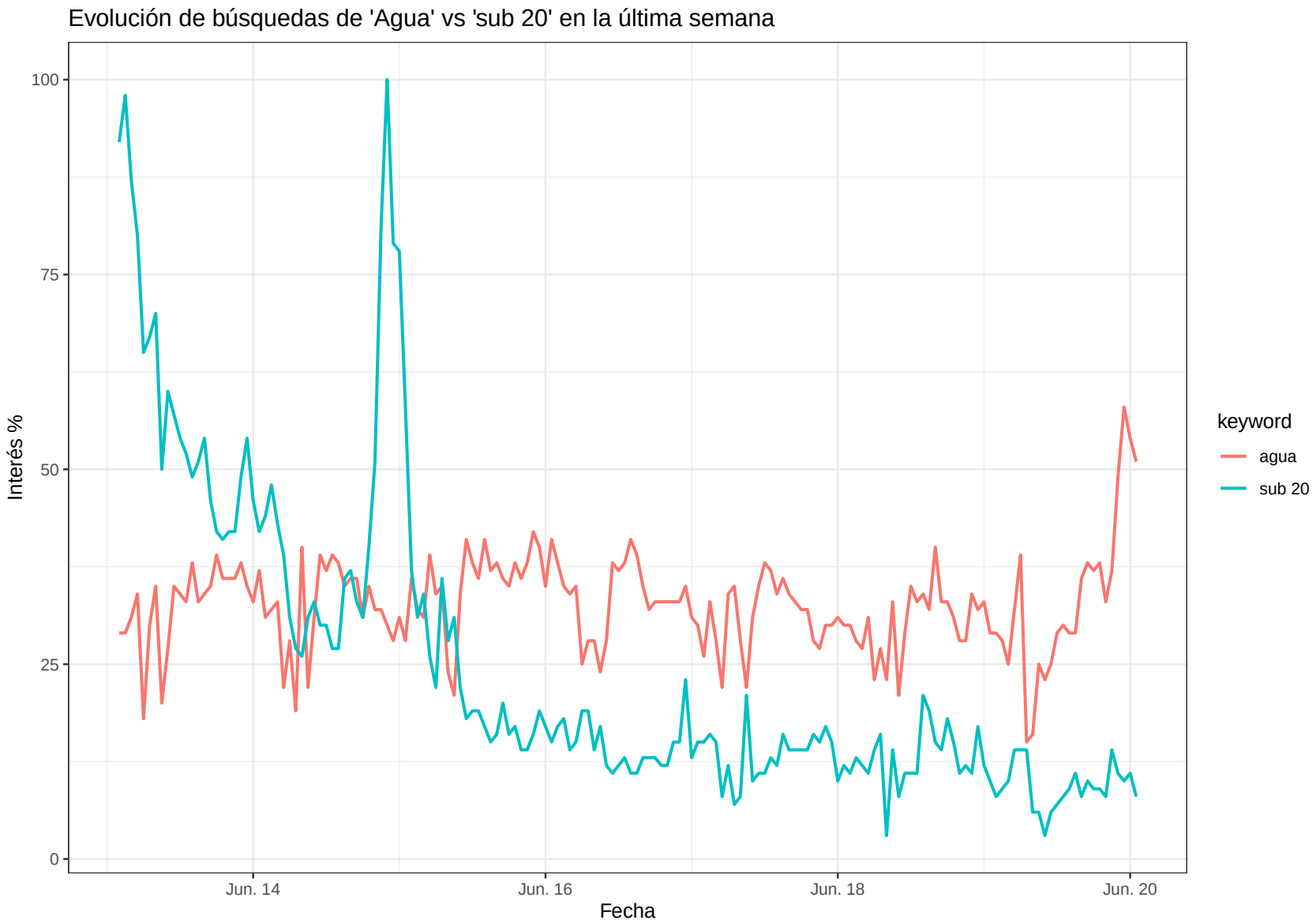
gtrendsR

- El paquete gtrendsR
- Permite realizar búsquedas de los términos más buscados en Google, proporciona una métrica propia para saber el volumen de búsqueda asociado.
- Permite análisis longitudinales, por países,departamentos, etc.
- Es útil para analizar intereses/preocupaciones de las personas lo cual nos puede dar información del ámbito *privado*, trascendiendo o complementando con los mensajes emitidos de carácter público (rrss)

gtrendsR: ejemplo con análisis del tema agua

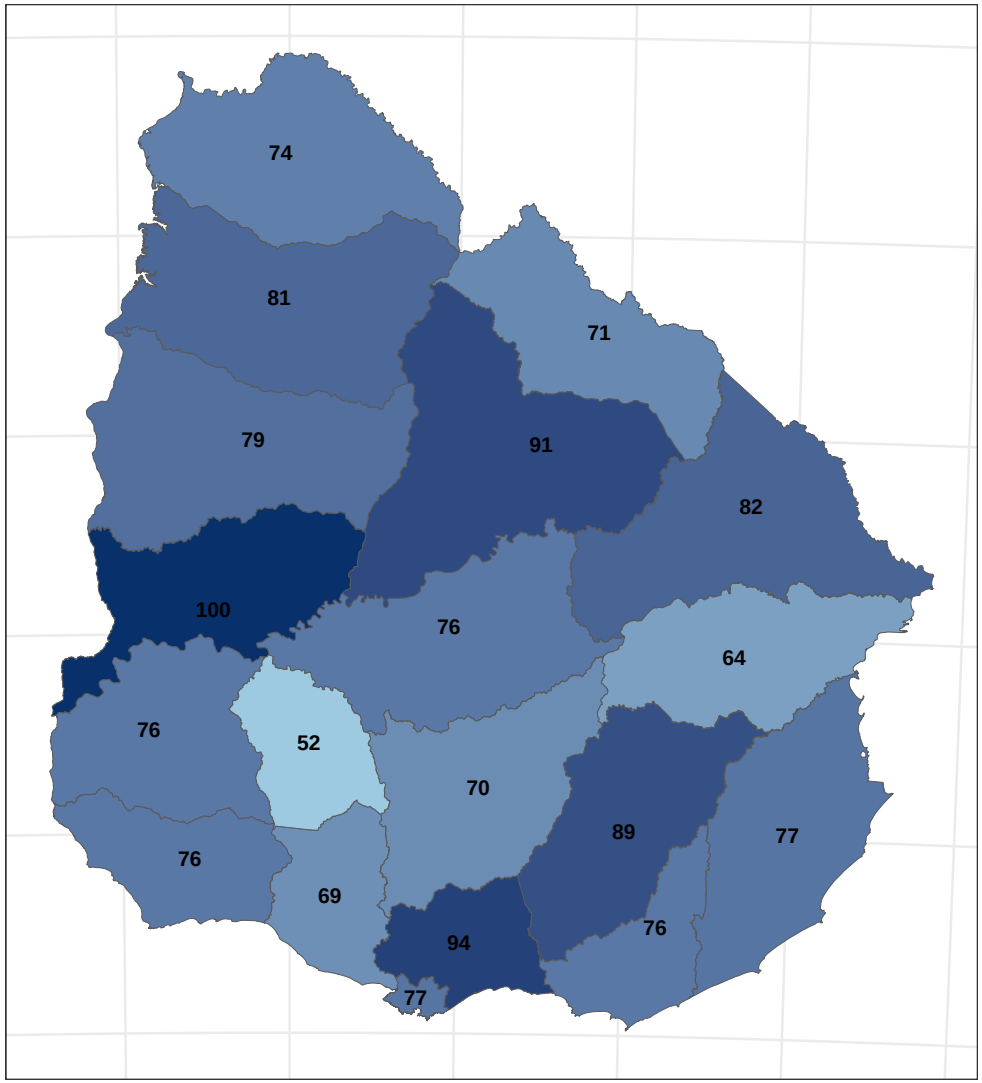


gtrendsR: ejemplo con análisis del tema agua

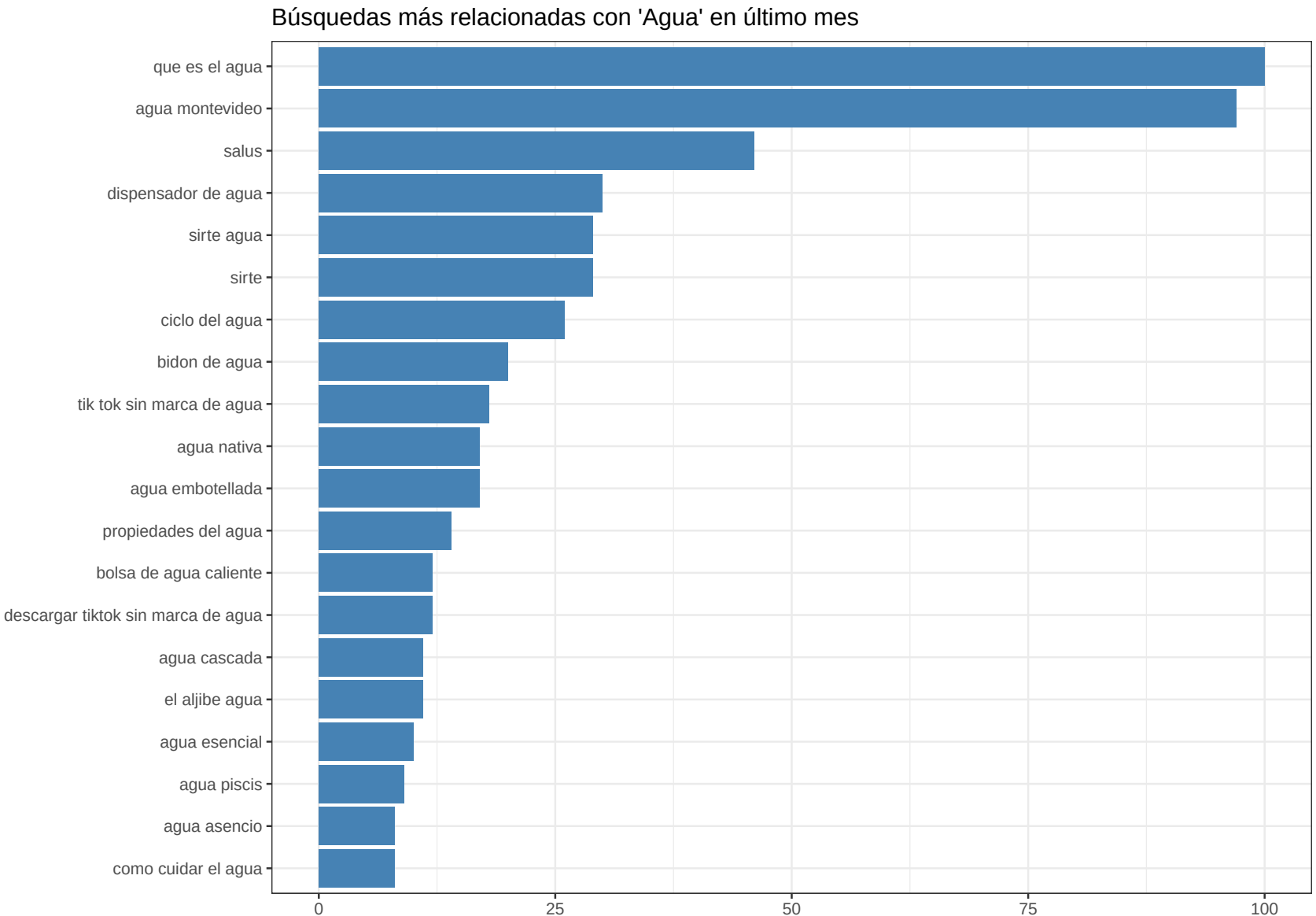


gtrendsR: ejemplo con análisis del tema agua

Búsquedas de 'Agua' en último mes por departamento



gtrendsR: ejemplo con análisis del tema agua



Otros recursos disponibles

- Recursos en línea para el estudio de la conflictividad http://observatoriodeconflictividad.org/
- Paquete ACEP: Analisis Computacional de Eventos de Protesta
- *ACEP es un paquete de funciones en lenguaje R utiles para la deteccion y el analisis de eventos de protesta en corpus de textos periodísticos. Sus funciones son aplicables a cualquier corpus de textos. Ademas de las funciones, ACEP contiene también bases de datos con colecciones de notas sobre protestas y una colección de diccionarios de palabras conflictivas y otros tópicos referidos a diferentes aspectos del análisis de eventos de protesta.*
- Autor: Agustín Nieto (Universidad Nacional de Mar del Plata)

Otros recursos disponibles

- Paquete internetarchive permite scrapear del sitio **Internet Archive**
- Hemeroteca o biblioteca digital archive.org *gestionada por una organización sin ánimo de lucro dedicada a la preservación de archivos, capturas de sitios públicos de la Web, recursos multimedia, etc.*

4. Audio

El paquete audio.whisper permite utilizar en R la herramienta de reconocimiento de voz *"Whisper" Automatic Speech Recognition model* desarrollada por openaAI.

Recuperar texto de audios es una fuente casi inagotable (entrevistas, discursos, conversaciones, podcast, etc.).

audio.whisper

- Tiene diferentes modelos que van desde el menos potente *(tiny)* al más potente *(large)*
- Cuanto mayor es la potencia y precisión del modelo más demora la transcripción
- No todos están disponibles para español
- Los pasos son sencillos y están bien explicados en el repositorio del paquete
- Se combina con la librería `av` para transformar los audios a formato de archivo *.wav de 16 bit*, que es el requerido por `audio.whisper`

audio.whisper + av

Obtengo un audio de interés y lo convierto a **.wav** con el paquete av

```
library(av) # conversor a .wav
library(audio.whisper) # transcpción

# 1. OBTENGO UN ARCHIVO DE AUDIO
# descargo para el ejemplo un audio de la web (podría ser un archivo que ya tengo en mi pc)
download.file("https://medios.presidencia.gub.uy/tav_portal/2018/noticias/AD_103/vazquez-cuidados.mp3", #
              "cuidados.mp3", # nombre del archivo que quedará en mi pc
              mode="wb") # modo web

# 2. CONVERSIÓN (av)
# convierto a .wav
av_audio_convert("cuidados.mp3", # nombre del archivo en mi pc
                 output = "cuidados.wav", # nombre del archivo convertido
                 format = "wav", sample_rate = 16000) # formato
```

audio.whisper

Realizo la transcripción con el modelo *tiny* (el menos potente)

```
# Descargo el modelo
# (podría saltear este paso poniendo la ruta en la función predict())
model <- whisper("tiny") # descargo modelo liviano
# lo corro indicando el idioma (es multilingual)
transcript <- predict(model, newdata = "cuidados.wav", language = "es")
# extraigo el df donde está el texto transcripto
texto_df <- transcript$data # df tiene 4 cols segmento, inicio, fin, texto
# guardo el df
save(texto_df,file="texto_df.RData") #o en el formato que quieras
```

audio.whisper

Construyo un cuadro con `knitr` y `kableExtra` con el texto

```
#olapso la columna text también podría usar un identificador y agrupar
texto_vec <- paste(texto_df$text,collapse="")
tabla1 <- knitr::kable(texto_vec,
          col.names = "Tabaré Vázquez - Sistema de Cuidados", # agrego nombre
          format = "html", table.attr = "style='width:100%;'") %>% #formato
  kableExtra::kable_styling(font_size = 24) %>% # defino tamaño de letra
  kableExtra::kable_classic() # defino el estilo de la tabla
```

audio.whisper

Tabaré Vázquez - Sistema de Cuidados

Con respecto al sistema nacional de cuidados, dijimos en aquel momento, se implementará este sistema priorizando y aquí definimos tres poblaciones que queríamos atender. La primera infancia, las personas con discapacidad y adultos mayores en situación de dependencia. Lo dijimos en junio, de 2014. Hoy, ¿qué tenemos? Se implementó un proyecto de ley para crear un sistema nacional de cuidados. Y en esa ley aprobada se creó una Junta Nacional de cuidados, una secretaria nacional de cuidados y un comité consultivo de cuidados, la ley 19.353, que constituye del alma institucional de este sistema que pretendemos y estamos seguro, cualquiera de hacer al próximo Gobierno va a continuar adelante porque una acción de este tipo. Por la importancia humana que tiene sin duda creemos y lo creemos sincera, pero modestamente debe constituirse en una política de Estado. Y atendimos a la primera infancia, al día hoy, hay tres, tres, mil, doscientos y en cuenta niños, onineas, de cero a tres años, que están siendo cuidados por personal capacitado especialmente para hacer esta tarea. En personas de situación de dependencia, 4,688 personas cuentan ya con un asistente personal. En algunos casos pagados por todos ustedes, por el Estado de los casos cuando la capacidad económica del hogar permite pagar una parte de otra parte de la pagada del Estado, o simplemente pagar las familias, pero los cuidadores son especializados, especialmente para realizar esta tarea. En este tipo, en tela existencia en casa es decir personas que están conectadas con un sistema central de respuesta hay 832 personas activas en este momento. Para formar a las personas hubo 40 cursos que aún se encuentra en marcha porque la capacitación es permanente. 1773 personas completaron el curso de atención a la dependencia. Más de 3.000 se formaron para la atención a la primera infancia. Y el portal de cuidados que ustedes pueden visitar, todos los datos que estamos pueden ser corroborados y aquí hasta de terminar hablar. El portal de cuidados tiene en atención lo han consultado más de 48.000 740 personas desde el año 2016.

audio.whisper + scraping

- La utilidad de la transcripción es mayor cuanto más audios tengamos
- Transcribir una entrevista puede ser divertido, 10 es agotador, más de 10 hay contratar a alguien y en general no tenemos recursos
- La potencia se acrecienta combinando herramientas
- Un buen ejemplo es realizar scrapeo de audios de la web con `rvest`

audio.whisper + scraping + rgtp3

También podemos usar el paquete `rgtp3` que permite conectar R con la herramienta de openAI *chatGPT3*

La API es de pago, pero para un ejercicio básico alcanza con lo que te permite utilizar gratis

- Acá pueden descargar un ejemplo con:

1. Descarga de audios `rvest`
2. Transcripción con `audio.whisper`
3. Resumen e identificación de tema principal con `rgpt3`

- *Acá* hay otro ejemplo de uso de `rgtp3` (no de audio) donde pueden ver los pasos para conectar con la API

5. YouTube

El paquete `youtubecaption` permite descargar los subtítulos de los videos de YouTube

Trabaja sobre la librería de Python `youtube-transcript-api`

Es necesario conectar R y Pyhton, lo que puede realizarse con librería `reticulate` que permite la instalación de *miniconda* o la interfaz *Anaconda* para gestionar los paquetes (ver archivo *instalaciones* del curso)

5. youtubecaption

- Es posible recuperar texto de todos los videos que cuentan con subtítulos (incoprorados o generados automáticamente)
- Si los subtítulos son automáticos la fidelidad generalmente depende de la claridad del audio
- `youtubecaption` recupera la transcripción de forma tabulada y ordenada para cada secuencia del video, por lo que luego es necesario agrupar por el identificador y recuperar la metadata original (fecha, resumen, canal, visualizciones, etc.)

5. youtubecaption

Hay tantas alternativas como videos de YouTube existan: discursos, conferencias, entrevistas, canciones, películas, programas de tv. . .

```
# hadley wickham
url <- "https://www.youtube.com/watch?v=cpbtcsGE0OA"
caption <- get_caption(url)

# suarez
url2 <- "https://www.youtube.com/watch?v=KsE8a9NOtnU"
caption2 <- get_caption(url2, language = "es")

# agarrate catalina
url3 <- "https://www.youtube.com/watch?v=LApsPiejZLI"
caption3 <- get_caption(url3, language = "es")
```

5. youtubecaption

También `youtubecaption` se potencia con la combinación de herramientas

- Acá hay un ejemplo de uso con videos del presidente Lacalle Pou:

1. Scraping con Apify
2. Descarga con `youtubecaption`
3. Análisis con `quanteda` y `udpipe`
4. Visualización con `ggplot2`

6. APIs de redes sociales

¿Qué son las API?

API significa *Application Programming Interfaces* o *interfaz de programación de aplicaciones* (en español), son un conjunto de protocolos usados para desarrollar aplicaciones y sirven para definir la comunicación entre dos aplicaciones de software a través de un conjunto de reglas (ver más).

Las aplicaciones de redes sociales, por ejemplo, asignan algunos permisos de acceso a desarrolladores para interactuar con las mismas. El nivel de acceso varía de una a otra red.