Práctica PLN Grupo 6

Integrantes:

- Álvaro Herrero: alvaro.herrero.vidal@alumnos.upm.es
- Eduardo Vicente: eduardo.vicente@alumnos.upm.es
- Mateo Bordoy: mateo.bordoy@alumnos.upm.es
- Laura García: laura.gagonzalez@alumnos.upm.es
- Esteban Moreno: esteban.moreno@alumnos.upm.es

Introducción

En esta práctica vamos a desarrollar un script que va a actuar como un buscador de enfermedades. Dado un texto, el script buscará palabras que terminen en -itis, -oma, -algia o que empiecen por hipo- o hiper-, que se corresponden con varios tipos de enfermedades. Luego, el script buscará las definiciones de esas enfermedades en un diccionario médico.

El resultado final será un dataframe con los siguientes elementos por enfermedad:

- Nombre de la enfermedad.
- Sintagma nominal donde se encontraba la enfermedad.
- Definición de la enfermedad.
- Lema (o forma invariante) de la enfermedad.

Elaboración del script

Librerías utilizadas

Cargamos las librerías que vamos a utilizar.

```
# install.packages(rvest)
library(rvest)
library(spacyr)
```

"rvest" es una librería que permite hacer web scraping en R. La emplearemos para buscar la definición de las enfermedades que filtremos.
"spacyr" es una librería que usaremos para parsear los textos por palabras y sintagmas nominales, así como filtrar las palabras según su tipo.

Función para buscar la definición en el diccionario médico

Definimos la función buscar_definicion_diccionario.

Esta función se encarga de, dado una palabra que se corresponde con una enfermedad, consultar un diccionario de términos médicos y devuelve su definición si se encuentra registrado en el diccionario. En el caso de que esa enfermedad no se encuentre en el diccionario, se devolverá un string vacío.

```
# Función para buscar definiciones en el diccionario médico
buscar_definicion_diccionario <- function(enfermedad) {
  url <- paste0("https://www.cun.es/diccionario-medico/terminos/", enfermedad)

  # Se lee la página web.
  pagina <- read_html(url)

  # Usando XPath se obtiene el elemento del HTML que contiene el texto con
  # la definición.
  html_xpath = '//*[@class="textImageComponent textImageComponent"]'
  elemento <- html_element(pagina, xpath = html_xpath)

  # Lee el texto del elemento.
  definicion <- html_text2(elemento)

  # Si no se encuentra la definición en el diccionario (en ese caso se muestra
  # not_found) se devuelve un string vacío.
  not_found = paste0("ENFERMEDADES Y TRATAMIENTOS\n\nEnfermedades\nPruebas",
                     " diagnósticas\nTratamientos\nCuidados en casa\n",
                     "Chequeos y salud") # Dividimos el texto y lo concatenamos
                                         # para que no se salga del PDF al
                                         # compilar el R Markdown

  if (definicion == not_found) {
    definicion <- ""
  }

  return(definicion)
}
```

Función para filtrar enfermedades de los textos

Definimos la función filtrar_enfermedades, la cual será la función principal del script.

Esta función se encarga de, dado un texto, buscar las palabras con los prefijos y sufijos descritos en la introducción, junto con el sintagman nominal completo en el que se encuetran. Luego, usando la función anterior se buscan las defincciones, también se obtienen los lemas y se devuelve todo en un dataframe.

Dentro de esta función se definen otras funciones auxiliares: clasificar_enfermedades_nouns, que extrae las palabras coincidentes con los patrones proporcionados de los sintagmas nominales y las almacena en un vector, y clasificar_enfermedades, que extrae los sintagmas nominales que contienen al menos una palabra que coincida con los patrones proporcionados.

```
filtrar_enfermedades <- function(txt) {
  spacy_initialize(model = "es_core_news_sm")
```

```r
# Definimos patrones de enfermedades
patrones <- list(
  dolor = "\\b\\w*algia\\b",
  alteracion = "\\b(hipo|hiper)\\w*\\b",
  infeccion = "\\b\\w*itis\\b",
  cancer = "\\b\\w*oma\\b"
)

clasificar_enfermedades_nouns <- function(patron, texto) {
  # Filtra las palabras que coincidan con el patrón proporcionado
  enfermedades <- c()

  # Itera sobre cada palabra del texto proporcionado
  for (palabra in texto) {
    # Extrae las palabras que coincidan con el patrón
    coincidencias <- regmatches(palabra, gregexpr(patron, palabra,
                                                  ignore.case = TRUE))[[1]]
    # Agrega las coincidencias encontradas
    if (length(coincidencias) > 0) {
      enfermedades <- c(enfermedades, coincidencias)
    }
  }

  # Devuelve las coincidencias
  return(enfermedades)
}

clasificar_enfermedades <- function(patron, texto) {
  # Filtra los sintagmas nominales que contienen al menos una palabra que
  # coincida con el patrón proporcionado
  textos_filtrados <- texto[grep(patron, texto, ignore.case = TRUE)]

  # Devuelve los textos completos que cumplen con el criterio
  return(textos_filtrados)
}

# Se separa el texto por sintagmas nominales.
sn <- spacy_extract_nounphrases(txt)$text

# Se filtran los sintagmas nominales que no contengan alguna de las palabras
# que buscamos.
# Inicializamos un vector para almacenar los sintagmas nominales válidos.
sn_filtrado <- c()

# Por cada sintagma nominal, analizamos sus palabras con spaCy
for (text in sn) {
  # Parseamos el texto
  parsed_text <- spacy_parse(text)

  # Buscamos las palabras que coincidan con el patrón y sean sustantivos
  # o nombres propios
  palabras_validas <- subset(
    parsed_text,
```

```
    grepl("\\b(hipo\\w*|hiper\\w*|\\w*itis|\\w*algia|\\w*oma)\\b", token, ignore.case = TRUE)
    & pos %in% c("NOUN", "PROPN")
  )

  # Si hay al menos una palabra válida, añadimos el sintagma nominal al filtro
  if (nrow(palabras_validas) > 0) {
    sn_filtrado <- c(sn_filtrado, text)
  }
}

# Por cada sintagma nominal filtrado, lo parseamos con spacy y nos quedamos
# con aquellas palabras que sean sustantivos o nombres propios (a veces spacy
# interpreta una enfermedad como nombre propio).
sn_nouns_filtrado <- c()
for (text in sn_filtrado) {
  parsed_text <- spacy_parse(text)
  filtrado <- subset(parsed_text, pos %in% c("NOUN", "PROPN"))
  sn_nouns_filtrado <- c(sn_nouns_filtrado, paste(filtrado$token,
                                                  collapse = " "))
}

# Para cada patrón en la lista de expresiones regulares, clasificamos los
# sustantivos y los guardamos en una lista según su tipo
enfermedades_nouns_clasificadas <- lapply(patrones, function(patron) {
  clasificar_enfermedades_nouns(patron, sn_nouns_filtrado)
})

# Para cada patrón en la lista de expresiones regulares, clasificamos los
# sintagmas nominales según el tipo de enfermedad que contengan
enfermedades_clasificadas <- lapply(patrones, function(patron) {
  clasificar_enfermedades(patron, sn_filtrado)
})

# Buscamos las definiciones de las enfermedades encontradas para cada
# enfermedad con la función buscar_definicion_diccionario
definiciones <- list()

for (tipo in names(enfermedades_nouns_clasificadas)) {
  definiciones[[tipo]] <- sapply(enfermedades_nouns_clasificadas[[tipo]],
                                 buscar_definicion_diccionario,
                                 USE.NAMES = FALSE)
}


# Pasamos las listas a vectores
enfermedades_nouns_combinadas <- unlist(enfermedades_nouns_clasificadas)
enfermedades_combinadas <- unlist(enfermedades_clasificadas)
definiciones_combinadas <- unlist(definiciones)

# Hallamos los lemas de las enfermedades
lemmas_enfermedades <- spacy_parse(paste(enfermedades_nouns_combinadas,
                                         collapse = " "))$lemma
```

```
  spacy_finalize()

  # Creamos un dataframe con toda la información obtenida
  df <- data.frame(
    Enfermedad = enfermedades_nouns_combinadas,
    Sintagma_nominal_completo = enfermedades_combinadas,
    Definicion = definiciones_combinadas,
    Lema = lemmas_enfermedades
  )

  return(df)
}
```

Resultado

Podemos probar el script con un texto de ejemplo extraído del JSON con artículos médicos.

```
texto <- readLines("datos/texto 120.txt")
```

```
# Texto de prueba
resultado <- filtrar_enfermedades(texto)
```

```
## successfully initialized (spaCy Version: 3.7.6, language model: es_core_news_sm)
```

```
# Usando la función kable (incluida en knitr) y la librería kableExtra
# construimos una tabla en LaTeX de forma que todo el contenido
# del dataframe sea legible y no se salga de la página.
# install.packages(kableExtra)
library(kableExtra)
kable(resultado, caption = "Dataframe resultado") %>%
  column_spec(2, width = "2cm") %>% # Ancho columna enfermedad
  column_spec(3, width = "5cm") %>% # Ancho columna sintagma_nominal_completo
  column_spec(4, width = "4cm") %>% # Ancho columna definición
  kable_styling(latex_options = c("hold_position")) # Evita que la tabla se desplace a otras partes
```

Table 1: Dataframe resultado

	Enfermedad	Sintagma_nominal_completo	Definicion	Lema
cancer1	carcinoma	Manejo Quirúrgico del carcinoma metastásico primario de cabeza y cuello	m. Neoplasia maligna constituida por células epiteliales anáplasicas con capacidad metastásica.	carcinoma
cancer2	carcinoma	, carcinoma	m. Neoplasia maligna constituida por células epiteliales anáplasicas con capacidad metastásica.	carcinoma

cancer3	Carcinoma	Carcinoma Metastásico de Primario Desconocido de Cabeza y Cuello	m. Neoplasia maligna constituida por células epiteliales anáplasicas con capacidad metastásica.	Carcinoma

```
                                                    # del documento
```

NOTA IMPORTANTE: Al filtrar únicamente los sustantivos, perdemos todos los tokens que spacy considere que no son sustantivos, pese a que se correspondan a una enfermedad. Por ejemplo, en este texto, palabras como linfoma o melanoma no son incluidas porque spacy considera que son verbos.

Si no hubiéramos filtrado únicamente los sustantivos, estas palabras también aparecerían en el dataframe:

Figure 1: Análisis de la palabra "melanoma" en el texto.

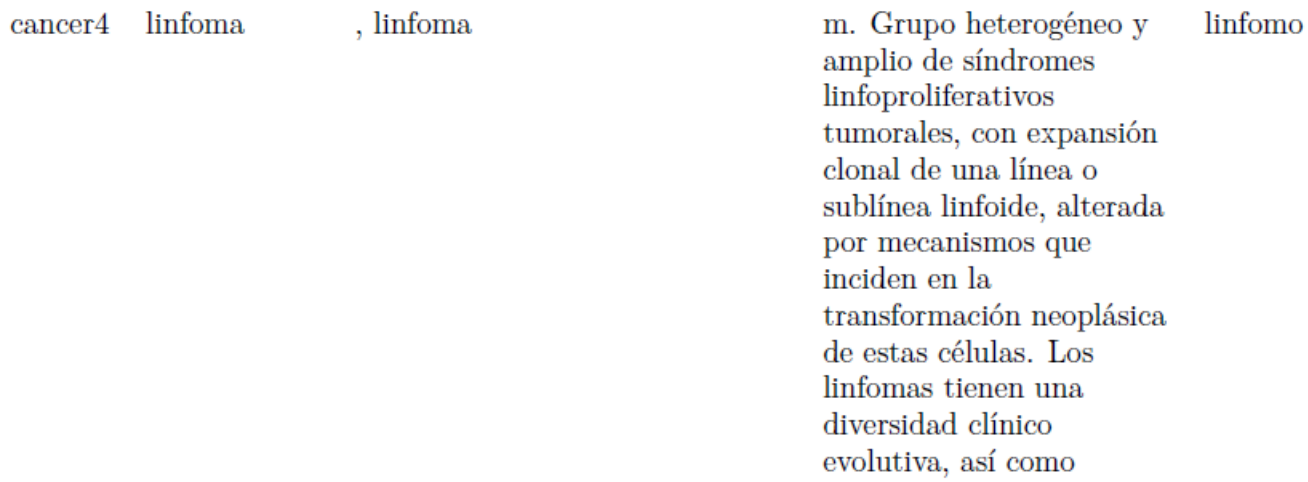

Figure 2: Análisis de la palabra "linfoma" en el texto.

Testeo

Para comprobar que el script funciona correctamente con cualquier texto, hemos extraído algunos de los textos que más palabras con los afijos que nos interesan tienen de “MESINESP_ORIGINAL_TRAINING.json”, los cuáles se encuentran en la carpeta “datos” y los hemos probado con el script “tester.R”, dentro de la carpeta “pruebas”. Hemos exportado los dataframes resultados como archivos CSV a la carpeta “resultados”.

Dificultades

En general la práctica se ha desarrollado sin dificultades reseñables, ya que la mayoría de los aspectos que se incluyen en la práctica se han visto en clase. Lo único que nos ha supuesto más trabajo son los siguientes aspectos:

- **Función para obtener definiciones:** Este aspecto de la práctica no ha sido complicado en sí, pero si que ha requerido un poco de investigación porque en un principio no sabíamos cómo extraer la definición a partir de la página web proporcionada. Sin embargo, una vez encontrada la librería necesaria para poder extraer la definición de la página web, el desarrollo de esta parte de la práctica no fue difícil.
- **Representación del resultado en el R Markdown:** Cuando intentabamos representar directamente el dataframe resultado en este R Markdown, se representaba de una manera extraña debido a la gran cantidad de texto que contiene la columna de definiciones, ya que la definición se salía de la página y en ocasiones ni siquiera se mostraban todas las definiciones, con lo cual tuvimos que buscar una solución a este problema. Fue entonces cuando vimos que es posible crear tablas de LaTeX en el documento usando una función de knitr. Sin embargo, en ocasiones el texto de las columnas se superponía, haciendo la tabla ilegible, así que tuvimos que usar una librería adicional que permite establecer un formato personalizado a las tablas.

Trabajo grupal

El desarrollo del proyecto del buscador de enfermedades se llevó a cabo en equipo trabajando de manera colaborativa para un desempeño más efectivo. Además, se aprovecharon las horas de trabajo en clase proporcionadas por el profesor para resolver las posibles dudas que fueron surgiendo y colaborar de manera conjunta en la resolución de ciertos apartados del proyecto, favoreciendo la comprensión grupal con los avances que se iban realizando.

A continuación, se detalla la división del trabajo y el rol desempeñado por cada integrante del equipo:

Responsable: Álvaro Herrero y Eduardo Vicente (50% - 50%)
Se enfocaron en desarrollar las expresiones regulares necesarias para identificar las enfermedades en los textos, basándose en los afijos médicos predefinidos y las funciones que se encargan de clasificar las enfermedades según su tipo. Asimismo, elaboraron el fragmento de script que se encarga de escoger las palabras deseadas de los sintagmas nominales fitrados, asegurándose de que las palabras encontradas fueran sustantivos, descartando aquellos casos en los que los afijos estuvieran presentes en adjetivos o verbos.

Responsable: Esteban Moreno
Fue responsable de implementar el análisis sintáctico del texto para extraer el sintagma nominal completo que describiera cada enfermedad (por ejemplo, “fascitis necrosante”). También se encargó de la lematización de las enfermedades identificadas.

Responsable: Laura García y Mateo Bordoy (50% - 50%)
Se encargaron del módulo opcional que buscaba la definición de cada enfermedad en un diccionario médico en línea, enriqueciendo así la información proporcionada por el sistema.

Responsable: Laura García y Mateo Bordoy (50% - 50%)
Fueron los responsables de redactar la memoria del proyecto, incluyendo la descripción del sistema, la metodología empleada, los resultados obtenidos y las conclusiones, así como propuestas para mejoras futuras.

Conclusiones y mejoras futuras

Con esta práctica hemos aprendido a elaborar un código que puede filtrar correctamente palabras según un conjunto de expresiones regulares y a extraer su definición de una página web. Sin embargo, la práctica no está optimizada al completo y por tanto, podemos resaltar alguna propuesta de mejora futura para mejorar la eficiencia del código. Por ejemplo, podemos ver que el script usa el parseador de spacy en dos ocasiones: para filtrar los sintagmas nominales que tienen palabras que coincidan con los patrones proporcionados y para extraer los sustantivos/nombres propios de esos sintagmas nominales. Una posible mejora en eficiencia sería hacer esas dos operaciones con un solo parseado de los textos, filtrando los sintagmas nominales que cumplan con los requisitos dados y al mismo tiempo extrayendo los sustantivos y nombres propios que busquemos de esos sintagmas.

Bibliografía

[1] Artículos médicos en español
Original Train Set – MESINESP: Medical Semantic Indexing in Spanish. (s. f.). https://temu.bsc.es/mesinesp/index.php/download/original-train-set/

[2] Definiciones médicas
Qué es abdominocentesis | Diccionario médico | Clínica U. Navarra. (s. f.). https://www.cun.es. https://www.cun.es/diccionario-medico/terminos/abdominocentesis

[3] Librería de web scraping (rvest)
Easily harvest (Scrape) web pages. (s. f.). https://rvest.tidyverse.org/

[4] Tablas de LaTeX en R
RPUBS - CREACIÓN DE TABLAS EN R | CREATION OF TABLES IN R. (s. f.). https://rpubs.com/JeisonAlarcon/Tables-R