Práctica PLN Grupo 6

2024-11-22

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

Cargamos las librerías que vamos a utilizar.

```
library(httr)
library(rvest)
library(spacyr)
library(jsonlite)
```

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 caso contrario se devolverá un string vacío y en caso de error a la hora de obtener la definición se devolverá NA.

```
# 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
  # el texto de la condicion) 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)
}
```

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 definciones, 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")

  # Definimos patrones de enfermedades
  patrones <- list(
    dolor = "\\b\\w*algia\\b",
    alteracion = "\\b([H|h]ipo|[H|h]iper)\\w*\\b",
    infeccion = "\\b\\w*itis\\b",
    cancer = "\\b\\w*oma\\b"
  )

  clasificar_enfermedades_nouns <- function(patron, columna) {
    # Vector para almacenar las coincidencias encontradas
    enfermedades <- c()

    # Itera sobre cada elemento de la columna
    for (texto in columna) {
      # Busca todas las coincidencias del patrón en el texto
      coincidencias <- regmatches(texto, gregexpr(patron, texto,
                                                  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, columna) {
  # Filtra los elementos que contienen al menos una palabra que coincida
  # con el patrón
  textos_filtrados <- columna[grep(patron, columna, 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([Hh]ipo\\w*|[Hh]iper\\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 = " "))
  }

  # Clasificamos el vector con los sustantivos según el tipo de enfermedad que
  # corresponda.
  enfermedades_nouns_clasificadas <- lapply(patrones, function(patron) {
    clasificar_enfermedades_nouns(patron, sn_nouns_filtrado)
  })

  # Clasificamos el vector con los sintagmas nominales según el tipo de
  # enfermedad que corresponda.
  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)
}
```

NOTA IMPORTANTE: Al filtrar únicamente los sustantivos, perdemos todos los tokens que spacy considere que no son susantivos, 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.

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 librería kable y kableExtra, construimos una tabla en TeX de forma
# que todo el contenido del dataframe sea legible y no se salga de la página.
kable(resultado, caption = "Dataframe resultado", booktabs = TRUE) %>%
  column_spec(2, width = "2cm") %>%
  column_spec(3, width = "5cm") %>%
  column_spec(4, width = "4cm") %>%
  kable_styling(latex_options = c("hold_position"))
```

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

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

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

cancer4	linfoma	, linfoma	m. Grupo heterogéneo y amplio de síndromes linfoproliferativos tumorales, con expansión clonal de una línea o sublínea linfoide, alterada por mecanismos que inciden en la transformación neoplásica de estas células. Los linfomas tienen una diversidad clínico evolutiva, así como	linfomo

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