Minería de Texto

Iván Mendivelso - Universidad Javeriana - Icetex

2023-11-20

Paquetes para esta sesión

```
install.packages("tm")
install.packages("pdftools")
install.packages("stopwords")
install.packages("wordcloud")
install.packages("quanteda")
```

```
library(tidyverse)
library(tm)
library(pdftools)
library(stopwords)
library(wordcloud)
library(quanteda)
```

Preliminares

El análisis de datos textuales, también conocido como minería de texto o *text mining*, permite extraer información valiosa y conocimientos de inmensas cantidades de texto no estructurado. Esta área interdisciplinaria combina técnicas de la lingüística computacional, la estadística, y la inteligencia artificial para procesar, analizar y comprender el lenguaje natural, abriendo puertas a un mundo de posibilidades en diversos campos, desde el marketing hasta la salud. En tiempos recientes, el avance de los modelos de lenguaje de aprendizaje profundo, especialmente los Grandes Modelos de Lenguaje (LLM, por sus siglas en inglés), ha revolucionado aún más este campo. Estos modelos, entrenados en gigantescos corpus de texto, no solo entienden y generan lenguaje natural con una precisión asombrosa, sino que también son capaces de captar matices y contextos complejos, ofreciendo interpretaciones y análisis más ricos y profundos.

Los LLM, como ChatGPT de OpenAI, Bard de Google, entre otros, han ampliado significativamente el alcance y la profundidad del análisis de texto. No se limitan a tareas tradicionales como la clasificación de texto, la extracción de entidades o el análisis de sentimientos; pueden realizar tareas más sofisticadas como la generación de resúmenes, la creación de contenido, la traducción automática, y hasta la programación basada en descripciones en lenguaje natural. Esta capacidad para manejar tareas complejas y diversas ha llevado a que los LLM se integren en una variedad de aplicaciones, desde asistentes virtuales hasta herramientas avanzadas de análisis de datos.

Sin embargo, a pesar de sus capacidades, los LLM presentan desafíos importantes, como la necesidad de grandes recursos computacionales y la gestión de sesgos inherentes a los datos de entrenamiento. A medida que el tiempo avanza y estos modelos se desarrollan, es crucial abordar estos desafíos, garantizando que el análisis de texto siga siendo una herramienta poderosa y ética para descubrir *insights* entre grandes cantidades de texto sin estructurar.

El desarrollo de los LLM es complejo y está por fuera del alcance de esta sesión de clase, puesto que requiere un alto acervo teórico y capacidades computacionales (si se tiene interés en este tema, se sugiere buscar en Google el artículo "*Attention is All You Need*"), por lo que aquí se tratarán las aplicaciones tradicionales del análisis de datos textuales como son la creación de un corpus de texto, la tokenización, el análisis de frecuencias, la visualización de datos y generación de digramas y trigramas.

```
textos <- c("Soy un dios en mi pueblo y en mi valle.",
            "No porque me adoren, sino porque yo lo hago.",
            "Porque me inclino ante quien me regala",
            "Unas granadillas o una sonrisa de su heredad...")
class(textos)
```

```
## [1] "character"
```

```
textos
```

```
## [1] "Soy un dios en mi pueblo y en mi valle."
## [2] "No porque me adoren, sino porque yo lo hago."
## [3] "Porque me inclino ante quien me regala"
## [4] "Unas granadillas o una sonrisa de su heredad..."
```

Conceptos importantes

Corpus

Un corpus de texto es una colección grande y estructurada de textos que se utiliza para el análisis lingüístico y el procesamiento del lenguaje natural (PLN). En R, un corpus puede ser manipulado y analizado utilizando diversos paquetes y funciones especializadas.

Un corpus en R generalmente se crea importando datos de texto desde diversas fuentes como archivos de texto, PDFs, o incluso directamente desde la web. El paquete `tm` (Text Mining) es uno de los más populares para trabajar con corpus de texto.

Un corpus puede ser una colección simple de textos o una estructura más compleja que incluye metadatos sobre cada texto, como autor, fecha, fuente, etc. Esto permite análisis más sofisticados y específicos.

Con el corpus listo, se pueden realizar diversos análisis como la frecuencia de palabras, la co-ocurrencia de términos, el análisis de sentimientos, la clasificación de textos, entre otros. Estos análisis ayudan a extraer *insights* y patrones de los datos de texto.

```
vector_fuente <- VectorSource(textos)

corpus <- Corpus(vector_fuente)

class(corpus)
```

```
## [1] "SimpleCorpus" "Corpus"
```

```
inspect(corpus)
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
```

```
## Content:  documents: 4
##
## [1] Soy un dios en mi pueblo y en mi valle.
## [2] No porque me adoren, sino porque yo lo hago.
## [3] Porque me inclino ante quien me regala
## [4] Unas granadillas o una sonrisa de su heredad...
```

Term-Document Matrix

La *Term-Document Matrix* es una matriz donde cada fila representa un término único (generalmente una palabra) y cada columna representa un documento del corpus. Los elementos de la matriz indican la frecuencia de cada término en cada documento.

```
tdm <- TermDocumentMatrix(corpus)
tdm_matrix <- as.matrix(tdm)
tdm_matrix %>% head()
```

```
##          Docs
## Terms     1 2 3 4
##   dios    1 0 0 0
##   pueblo  1 0 0 0
##   soy     1 0 0 0
##   valle.  1 0 0 0
##   adoren, 0 1 0 0
##   hago.   0 1 0 0
```

Tokenización

La tokenización es el proceso de dividir un texto o una cadena de caracteres en unidades más pequeñas llamadas "tokens". Un token, en este contexto, suele ser una palabra, aunque también puede ser un símbolo, un número, o cualquier otro elemento significativo en el texto.

La tokenización es a menudo el primer paso en el análisis de texto y PLN. Al dividir el texto en tokens, se facilita la realización de tareas como el conteo de frecuencia de palabras, la identificación de patrones de lenguaje, el análisis de sentimientos, la clasificación de textos, entre otros.

Este proceso ayuda a estandarizar el texto para el análisis posterior. Por ejemplo, al dividir el texto en palabras individuales, se pueden aplicar operaciones como convertir todo el texto a minúsculas, eliminar la puntuación, y realizar la lematización o el *stemming* (reducir las palabras a su raíz o forma base).

La tokenización es crucial para manejar la diversidad en el uso del lenguaje. Diferentes idiomas y contextos (como el lenguaje coloquial, técnico, o literario) pueden requerir diferentes enfoques de tokenización.

```
matriz <- as.matrix(tdm)

palabras_frecuencia <- rowSums(matriz)

palabras_frecuencia <- sort(palabras_frecuencia, decreasing = TRUE)

palabras_frecuencia
```

```
##      porque        dios      pueblo         soy      valle.     adoren,
##           3           1           1           1           1           1
```

```
##       hago.        sino        ante     inclino       quien      regala
##           1           1           1           1           1           1
## granadillas  heredad...     sonrisa        una        unas
##           1           1           1           1           1
```

Preprocesamiento

El preprocesamiento de un corpus de texto es un conjunto de pasos diseñados para limpiar y estandarizar los datos de texto antes de realizar análisis más complejos. Los pasos que normalmente se siguen para realizar el preprocesamiento de un corpus de texto son:

- Convertir a minúsculas: Hacer que todo el texto sea minúscula para que el análisis no sea sensible a las mayúsculas.

```
corpus <- tm_map(corpus, content_transformer(tolower))
```

- Eliminar puntuación: Quitar los signos de puntuación, ya que generalmente no son necesarios para muchos tipos de análisis.

```
corpus <- tm_map(corpus, removePunctuation)
```

- Eliminar números: Opcionalmente es posible quitar los números si no son relevantes para el análisis.

```
corpus <- tm_map(corpus, removeNumbers)
```

- Eliminar espacios en blanco extra: Limpiar espacios extra, como los espacios al principio y al final del texto, o múltiples espacios seguidos.

```
corpus <- tm_map(corpus, stripWhitespace)
```

- Eliminar *stopwords*: Quitar palabras comunes que generalmente no aportan mucho significado al texto, como “el”, “la”, “y”, etc.

```
corpus <- tm_map(corpus, removeWords, stopwords("spanish"))
```

Las funciones mostradas como ejemplos en los anteriores *script* pertenecen a la biblioteca para análisis de texto `tm`.

Biblioteca `tm`

El paquete `tm`, que significa “Text Mining”, es uno de los paquetes más prominentes y utilizados en R para el análisis de texto y la minería de datos textuales. Diseñado para facilitar el manejo, procesamiento y análisis de grandes conjuntos de datos textuales, `tm` proporciona un marco de trabajo integral y flexible para todo tipo de análisis de texto en R.

`tm` permite importar datos de texto desde varias fuentes, como archivos de texto plano, archivos JSON, XML, y otros formatos. Esto hace que sea fácil trabajar con una amplia gama de conjuntos de datos textuales.

Ejemplo

```
ruta <- 'C:/Users/Ivan Mendivelso/OneDrive - OVERLAP INTERNACIONAL S.A/Documentos/GitHub/Cursos_Overlap/
datos <- read.csv2(ruta)
glimpse(datos)
```

```
## Rows: 1,309
## Columns: 12
## $ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,~
## $ Survived    <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1~
## $ Pclass      <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3, 2, 3, 3~
## $ Name        <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradley (Fl~
## $ Sex         <chr> "male", "female", "female", "female", "male", "male", "mal~
## $ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39, 14, ~
## $ SibSp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0, 1, 0~
## $ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0, 0, 0~
## $ Ticket      <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113803", "37~
## $ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51.8625,~
## $ Cabin       <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA, NA, "G6", "C~
## $ Embarked    <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", "S", "S"~
```

```
nombres <- datos$Name
vector_fuente <- VectorSource(nombres)
corpus_nombres <- Corpus(vector_fuente)
```

```
inspect(corpus_nombres)
```

```
corpus_nombres <- tm_map(corpus_nombres, content_transformer(tolower))

corpus_nombres <- tm_map(corpus_nombres, removePunctuation)

corpus_nombres <- tm_map(corpus_nombres, removeNumbers)

tdm <- TermDocumentMatrix(corpus_nombres)
m <- as.matrix(tdm)
word_freqs <- sort(rowSums(m), decreasing = TRUE)
head(word_freqs)
```

```
##    miss     mrs william    john  master   henry
##     260     201      87      72      61      49
```

Stopwords

```
stopwords_en <- stopwords("en")
```

```
corpus_nombres <- tm_map(corpus_nombres, removeWords, stopwords_en)

tdm <- TermDocumentMatrix(corpus_nombres)

m <- as.matrix(tdm)
```

```r
word_freqs <- sort(rowSums(m), decreasing = TRUE)

head(word_freqs)
```

```
##    miss     mrs william    john  master   henry
##     260     201      87      72      61      49
```

Visualización de Datos

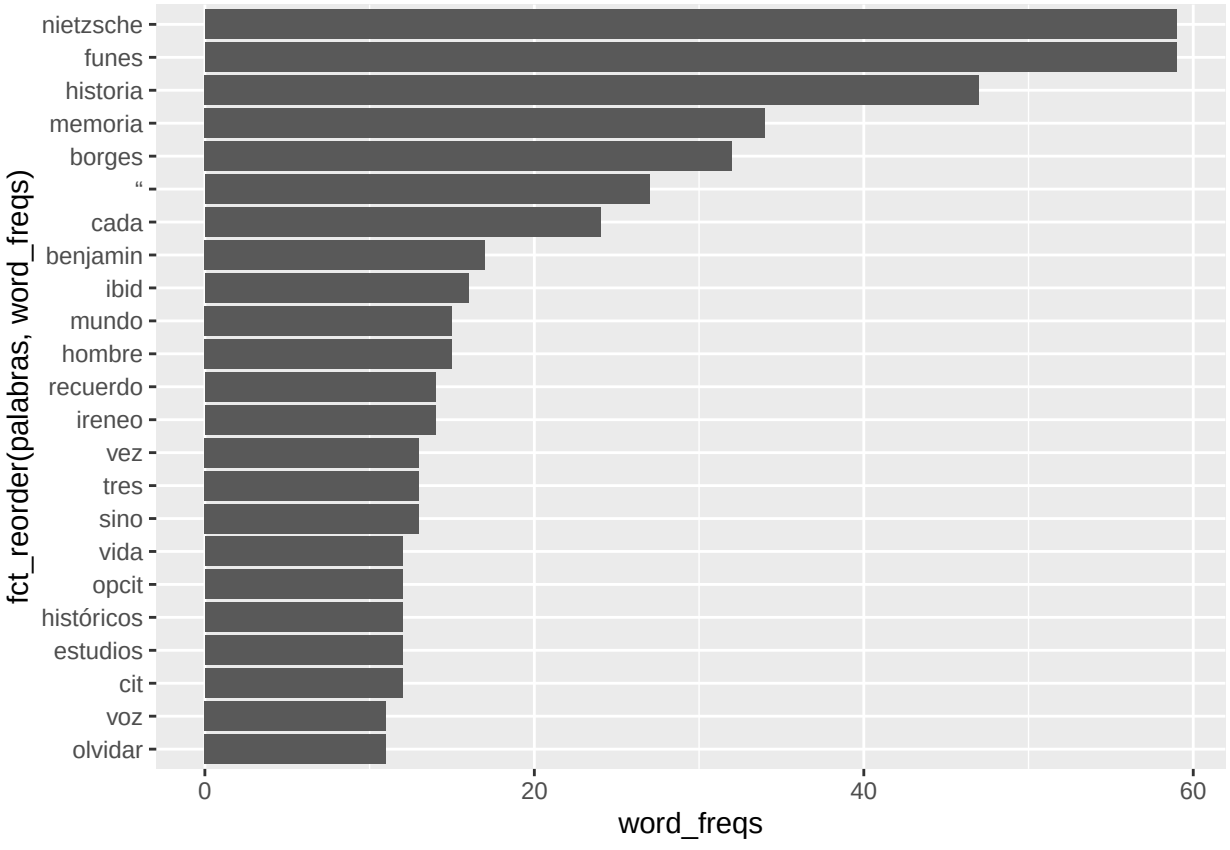
R ofrece potentes herramientas de visualización como `ggplot2` y `wordcloud` que se pueden usar para representar los resultados del análisis de texto, facilitando la interpretación y presentación de los datos.

Ejercicio con PDF

Seleccione un texto en formato PDF para aplicar los conceptos vistos hasta el momento. Use la función `pdf_text()` del paquete `pdftools` para leer el documento. Luego realice los correspondientes procedimientos para aplicar los métodos vistos.

```
##   que   del   los   las   una funes
##   235    83    79    64    61    59
```

```
##     funes nietzsche  historia   memoria    borges         "
##        59        59        47        34        32        27
```



Wordclouds


nietzsche
borges
historia
ibid
recuerdo
shakespeare
lyotard
memorioso"
mimado
primo
pensar
mundo
tesis
percepción
barcelona
escribe
incapaz
embargo
lugar
saber
naturalis
anestesia
diferendo
debe
catorce
pueblo
temor
pobre
refiere
bernardo
negar
hombre
números
"funes
condenado
siguiente
anticuario
individuo
todas
febrero
golpe
señala
hora
individuos
noche
cigarrillo
mismo
abarrotado
propia
disparest
toda
cambio
vana
rostro
abstraer
catre
posibilidad
pasado
lleva
cuento
aparece
futuro
olvido
acción
aquello
emecé
padre
bien
holgazán
difícil
buenos
ojos
duda
después
considerac
prodigiosa
cielo
horas
epígrafe
manos
condiciones
modo
perfil
san
mil
clases
veces
diversos
ireneo
cosas
progreso
ahora
dolor
saberes
advierte
vida
conocimiento
así
dormir
tierra
éste
río
palabras
ser
francisco
años
enorme
cuarto
visto
sueños
capacidad
históricos
nombre
símbolo
opcit
sólo
realidad
tal
genérico
fondo
vez
evidencia
única
dijo
profusión
voz
sola
primer
perro
idea
tan
residual
muertos
felicidad
encarna
formas
escrito
tarde
detalles
diecinueve
podía
sé
sentir
latín
ruina
tomo
recuerdos
exceso
inconvenientes
sino
sublime
punto
ejemplo
antiguo
ciudad
hace
forma
arte
soergel
hecho
manera
trata
dos
cit
encuentra
siempre
volumen
obra
ver
analogía
diálogo
"
crítica
inmovilidad
casas
alguna
tiempo
día
sabía
cara
cierto
nadie
completas
siglo
jardines
generalizar
obras
modernidad
estudios
dar
bentos
casos
joven
casi
caballo
conjunto
aires
necesidad
muerto
madrid
frente
fray
creo
moderno
abrumadora
hacia
tres
volvía
tamaños
menos
sentido
condición
filosofía
juzga
libros
memorioso
presente
patio
costaba
utilidad
sentimiento
año
facultad
solo
espectador
monumental
olvidar
experiencia
hombres
diferencias
benjamin
cada

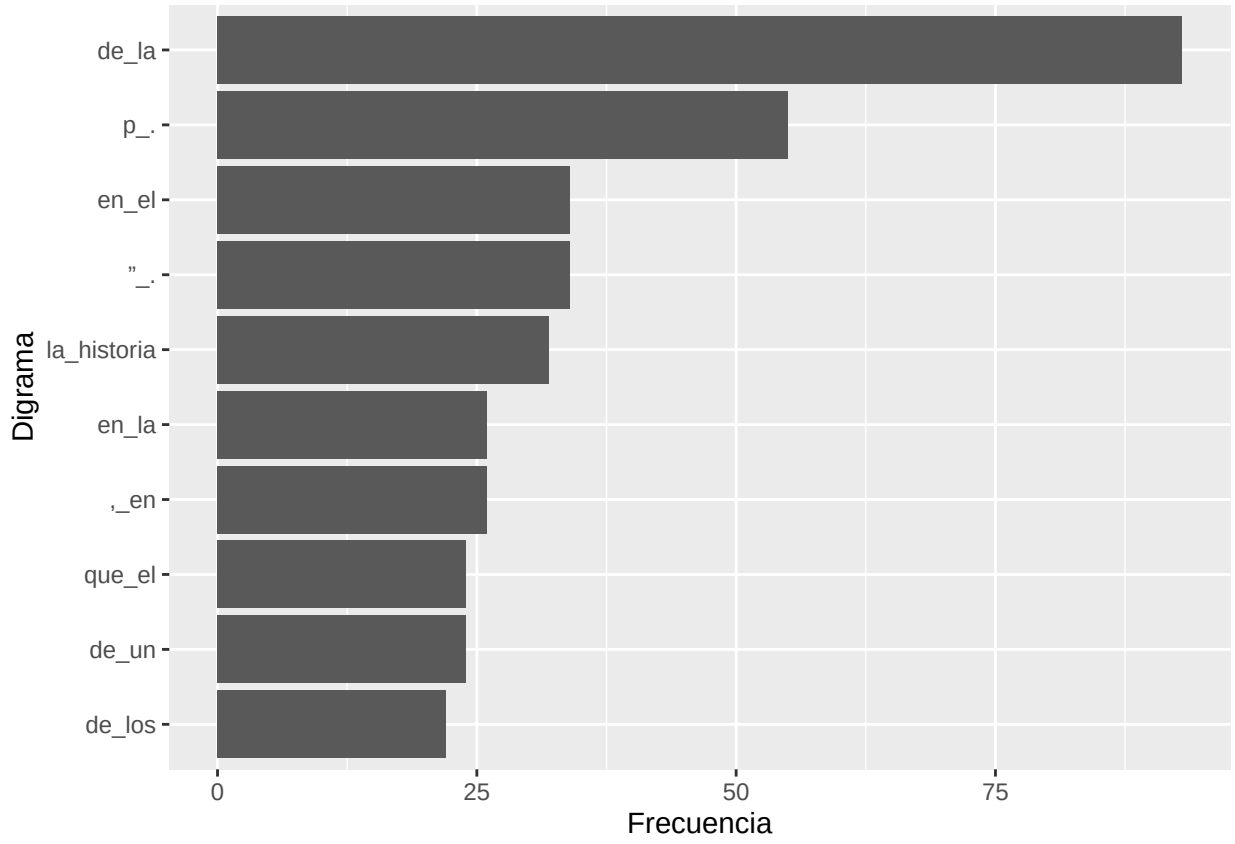
Digramas y trigramas

Los digramas y trigramas son conceptos fundamentales en el procesamiento del lenguaje natural y el análisis de texto. Forman parte de los n-gramas, que son secuencias contiguas de n elementos (o unidades) extraídos de un texto.

```
texto <- sapply(corpus_texto, as.character)

corpus_quanteda <- corpus(texto)

tokens_quanteda <- tokens(corpus_quanteda, what = "word")

tokens_digramas <- tokens_ngrams(tokens_quanteda, n = 2)
tokens_trigrams <- tokens_ngrams(tokens_quanteda, n = 3)

dfm_digramas <- dfm(tokens_digramas)
dfm_trigrams <- dfm(tokens_trigrams)
```

```
top_digramas <- topfeatures(dfm_digramas, 10)

data_digramas <- data.frame(ngram = names(top_digramas), frequency = top_digramas)

ggplot(data_digramas, aes(x = reorder(ngram, frequency), y = frequency)) +
    geom_bar(stat = "identity") +
```

```
    coord_flip() +
    xlab("Digrama") +
    ylab("Frecuencia")
```



Elimine las *stopwords* y realice de nuevo los digramas y trigramas.