

# Formato tidy para texto

## HR Analytics: Teoría y Práctica

<http://pablohaya.com/contact>

01/2023

# Instalación

El código ha sido probado en R version 4.0.2 (2020-06-22)

Es preciso instalar los siguientes paquetes:

- `install.packages("tidyverse")`
- `install.packages("tidytext")`
- `install.packages("textdata")`
- `install.packages("here")`
- `install.packages("reshape2")`

# Referencias

Estas transparencias han tomado como referencia el libro de texto de **Text Mining with R** de Julia Sigle y David Robinson:

- <https://www.tidytextmining.com>

El conjunto de datos original ha sido extraído de:

- <https://www.kaggle.com/saqlainrehan/employeesreviews-dataset>

# El formato tidy

El formato *tidy* es una manera de representar datos tabulados bastante cómoda para realizar análisis de datos. Este formato lo describe Hadley Wickham en detalle en Tidy Data, 2014.

En resumen, una tabla en formato '*tidy*' cumple:

- Cada variable es una columna
- Cada observación es una fila (ej. un empleado)
- Cada tipo de observación es una tabla (ej. una tabla para los datos de empleado, y otra para sus comentarios)

P: ¿Cómo codificamos un texto en formato *tidy*?

R: En una tabla donde cada fila almacena un *token*

Un *token* es la unidad mínima con sentido que vamos a analizar.

Normalmente se corresponde con una palabra, aunque podríamos definir *tokens* que agrupen más de una palabra (ej. n-gramas), u otra unidad de análisis como oraciones o párrafos.

El proceso de dividir un texto en *tokens* se llama *tokenización*.

Tomemos una revisión del archivo de datos.

```
text <- c('A company culture that encourages dissent, discourse,  
         'Strong compensation, from benefits, to perks, to base  
         'Decent internal mobility opportunities.',  
         'Employees are proud to work on globally impactful p  
         )  
  
text
```

```
## [1] "A company culture that encourages dissent, discourse,  
## [2] "Strong compensation, from benefits, to perks, to base  
## [3] "Decent internal mobility opportunities."  
## [4] "Employees are proud to work on globally impactful prod
```

Convertimos el vector `text` en un *data frame* donde cada fila es una oración.

```
text_df <- tibble(line = 1:4, text = text)
```

```
text_df
```

```
## # A tibble: 4 x 2
```

```
##   line text
```

```
##   <int> <chr>
```

```
## 1     1 A company culture that encourages dissent, discourages
```

```
## 2     2 Strong compensation, from benefits, to perks, to ba
```

```
## 3     3 Decent internal mobility opportunities.
```

```
## 4     4 Employees are proud to work on globally impactful p
```

La función que realiza la tokenización es `unnest_tokens()`

```
text_df %>%  
  unnest_tokens(word, text)
```

```
## # A tibble: 33 x 2  
##   line word  
##   <int> <chr>  
## 1     1 a  
## 2     1 company  
## 3     1 culture  
## 4     1 that  
## 5     1 encourages  
## 6     1 dissent  
## 7     1 discourse  
## 8     1 transparency  
## 9     1 and  
## 10    1 fairness  
## # ... with 23 more rows
```



`unnest_tokens()` tiene dos argumentos obligatorios.

- 1 Nombre de la columna que se va a crear para almacenar los tokens.
- 2 Nombre de la columna del data frame original que contiene el texto (text en el ejemplo)

Nótese que:

- Cualquier otra columna que tuviera el *data frame* se mantiene. En el ejemplo la columna `line`
- Los signos de puntuación han sido eliminados (ej. `dissent,`)
- Los tokens se convierten por defecto a minúsculas. Para deshabilitar esta opción tendríamos que añadir el parámetro `to_lower = FALSE`)

El parámetro token permite cambiar la unidad de tokenización (word, sentence, ngram...). Por defecto es word.

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

```
## # A tibble: 29 x 2  
##   line ngram  
##   <int> <chr>  
## 1      1 a company  
## 2      1 company culture  
## 3      1 culture that  
## 4      1 that encourages  
## 5      1 encourages dissent  
## 6      1 dissent discourse  
## 7      1 discourse transparency  
## 8      1 transparency and  
## 9      1 and fairness  
## 10     2 strong compensation
```

## Primeros análisis

Vamos a realizar un ejemplo más sofisticado partiendo de un dataset que contiene 10 000 revisiones aleatorias extraídas del conjunto de datos original.

Leemos las revisiones de la columna `summary`

```
reviews <- read_csv(here("data/employee_reviews_10000.csv"))
```

```
head(reviews$summary)
```

```
## [1] "Better for the short term"
```

```
## [2] "Head of Industry"
```

```
## [3] "overall is average on balance, but with great pluses a
```

```
## [4] "Microsoft is a good place to work, the environment is
```

```
## [5] "Good company, very stressful"
```

```
## [6] "Company"
```

y las tokenizamos:

```
tidy_reviews <- reviews %>%  
  select(summary) %>%  
  unnest_tokens(word, summary)
```

```
tidy_reviews
```

```
## # A tibble: 43,693 x 1  
##   word  
##   <chr>  
## 1 better  
## 2 for  
## 3 the  
## 4 short  
## 5 term  
## # ... with 43,688 more rows
```

obteniendo 43693 *tokens*

¿Qué *tokens* se utilizan más frecuentemente?

```
tidy_reviews %>%  
  count(word, sort = TRUE)
```

```
## # A tibble: 4,024 x 2
```

```
##   word      n
```

```
##   <chr>  <int>
```

```
## 1 to      1737
```

```
## 2 great   1718
```

```
## 3 work    1569
```

```
## 4 company 1189
```

```
## 5 place   1090
```

```
## 6 good    1010
```

```
## 7 a        944
```

```
## 8 and      785
```

```
## 9 for      758
```

```
## 10 the     703
```

```
## # ... with 4,014 more rows
```

En algunos análisis es preciso limpiar este listado de palabras frecuentemente usadas (*stop words*). tidyverse incluye un listado proveniente de tres lexicones distintos.

```
data(stop_words)
```

```
stop_words
```

```
## # A tibble: 1,149 x 2
##   word      lexicon
##   <chr>    <chr>
## 1 a        SMART
## 2 a's      SMART
## 3 able     SMART
## 4 about    SMART
## 5 above    SMART
## 6 according SMART
## # ... with 1,143 more rows
```

Eliminar estas palabras de nuestros datos en formato *tidy* es muy sencillo.

```
tidy_reviews <- tidy_reviews %>%  
  anti_join(stop_words)
```

```
tidy_reviews
```

```
## Joining, by = "word"
```

```
## # A tibble: 21,633 x 1
```

```
##   word
```

```
##   <chr>
```

```
## 1 short
```

```
## 2 term
```

```
## 3 head
```

```
## 4 industry
```

```
## 5 average
```

```
## 6 balance
```

```
## # ... with 21,627 more rows
```

Se han reducido a 21633 *tokens*, y han desaparecido las stop words de los *tokens* que aparecen más frecuentemente

```
tidy_reviews %>%  
  count(word, sort = TRUE)
```

```
## # A tibble: 3,555 x 2
```

```
##   word      n
```

```
##   <chr>    <int>
```

```
## 1 company  1189
```

```
## 2 manager   495
```

```
## 3 amazon    382
```

```
## 4 experience 356
```

```
## 5 engineer   326
```

```
## 6 job        313
```

```
## 7 associate  312
```

```
## 8 software   273
```

```
## 9 people     245
```

```
## 10 apple     239
```



Es posible emplear uno solo de los lexicones ya que emplear los tres puede eliminar demasiadas palabras que dependiendo del tipo de texto puede ser útiles.

El siguiente código elimina las palabras frecuentes (stopwords) del lexicon snowball

```
data(stop_words)

stop_words_snowball <- stop_words %>%
  filter(lexicon == "snowball")

tidy_reviews <- reviews %>%
  select(summary) %>%
  unnest_tokens(word, summary) %>%
  anti_join(stop_words_snowball)

tidy_reviews
```

```
## Joining, by = "word"

## # A tibble: 30,992 x 1
##   word
##   <chr>
## 1 better
## 2 short
## 3 term
## 4 head
## 5 industry
## 6 overall
## 7 average
## 8 balance
## 9 great
## 10 pluses
## # ... with 30,982 more rows
```

Este nuevo filtrado vemos que es menos *agresivo* (ej. vuelve a aparecer great):

```
tidy_reviews %>%  
  count(word, sort = TRUE)
```

```
## # A tibble: 3,887 x 2
```

```
##   word      n
```

```
##   <chr>    <int>
```

```
## 1 great    1718
```

```
## 2 work     1569
```

```
## 3 company  1189
```

```
## 4 place    1090
```

```
## 5 good     1010
```

```
## 6 manager   495
```

```
## 7 amazon    382
```

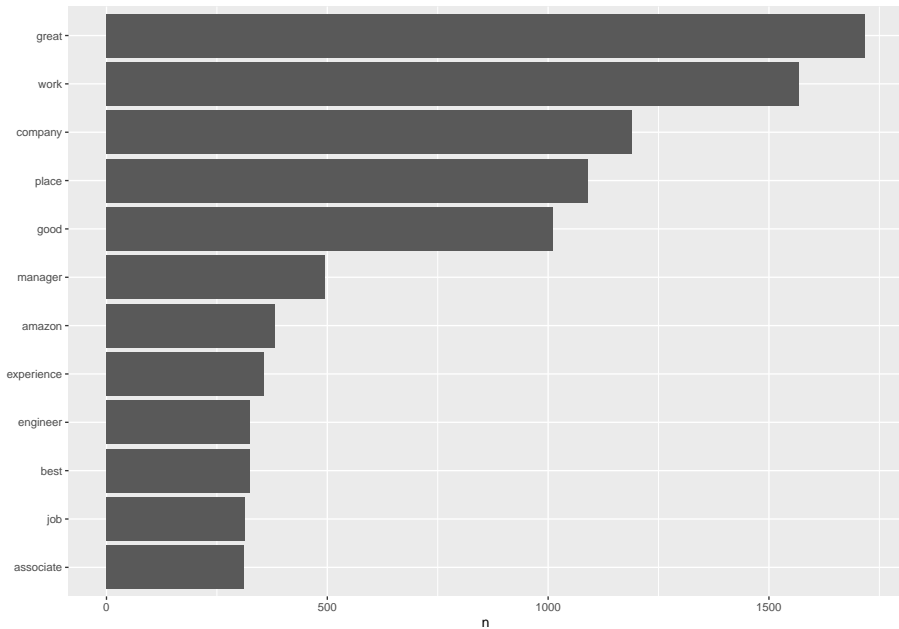
```
## 8 experience 356
```

```
## 9 best      326
```

```
## 10 engineer  326
```

Para terminar como estos primeros análisis básicos visualizamos la frecuencia de palabra en una gráfica:

```
tidy_reviews %>%  
  count(word, sort = TRUE) %>%  
  filter(n > 10) %>%  
  mutate(word = reorder(word, n)) %>%  
  ggplot(aes(n, word)) +  
    geom_col() +  
    labs(y = NULL)
```



# A practicar

**Ejercicio:** Crear un archivo con una lista de palabras prohibidas propia (ej. 10). Emplear este archivo para realizar el filtrado.