Análisis de Sentimientos en las películas de Spielberg

Adquisición de Datos

Primero tenemos que cargar las librerías necesarias.

```
library(NLP)
library(tm)
library(RColorBrewer)
library(wordcloud)
library(SnowballC)
library(reshape)
library(stringi)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following objects are masked from 'package:reshape':
##
##     colsplit, melt, recast
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##     annotate
```

```
library(aplpack)
```

```
## Loading required package: tcltk

## Warning in fun(libname, pkgname): couldn't connect to display ":0"
```

```
library(ggimage)
```

Ahora podemos utilizar la función `Corpus` del paquete `tm`, que genera automáticamente un corpus a partir de un directorio con un archivo por documento.

```
camino <- file.path('datos/peliculas')
docs   <- Corpus(DirSource(camino), readerControl = list(language = 'spa'))
```

Preprocesado

Una vez hemos creado el corpus, vamos a limpiarlo eliminando números, símbolos, espacios y stopwords. Para ello usamos la función `tm_map`.

```
corpus_clean <- tm_map(docs, removePunctuation, preserve_intra_word_dashes = TRUE)
corpus_clean <- tm_map(corpus_clean, content_transformer(tolower))
corpus_clean <- tm_map(corpus_clean, removeWords, stopwords('es'))
corpus_clean <- tm_map(corpus_clean, content_transformer(stripWhitespace))
corpus_clean <- tm_map(corpus_clean, removeNumbers)
```

Finalmente lematizamos las palabras usando el lematizador de la librería de `tm` (pasamos, por ejemplo de 'reconocer' o 'reconocido' a 'reconoc').

```
corpus_clean <- tm_map(corpus_clean, stemDocument, language = 'spanish')
```

Vectorizado

Una vez hemos limpiado los textos, tenemos que convertir las listas de palabras que representan cada película a vectores numéricos para poder analizarlas. Vamos a usar la función `DocumentTermMatrix` que tiene como columnas cada una de las palabras y como filas cada documento (película). De esta forma, en la fila i y columna j aparecerá el número de veces que la palabra j está en el documento i.

```
dtm <- DocumentTermMatrix(corpus_clean, control =  list(stopwords=FALSE, wordLengths=c(0, Inf)))
```

Análisis de Sentimiento

Ahora vamos a generar dos nuevas `DocumentTermMatrix`, una para las palabras positivas y otra para las negativas.

```
camino        <- file.path('datos/diccionarios/positive_words_es.txt')
listapositiva <- read.csv(camino,header=F,quote="",fileEncoding="Latin1")
listapositiva <- as.matrix(listapositiva)
listapositiva <- as.vector(listapositiva)
listapositiva <- wordStem(listapositiva, language='spanish')

ppositivas.tdm<- DocumentTermMatrix(corpus_clean, control = list(tokenize    = 'word',
                                                                 dictionary  = listapositiva,
                                                                 stopwords   = FALSE,
                                                                 wordLengths = c(0, Inf),
                                                                 stemming    = 'spanish'))
ppositivas.tdm
```

```
## <<DocumentTermMatrix (documents: 11, terms: 1199)>>
## Non-/sparse entries: 3899/9290
## Sparsity           : 70%
## Maximal term length: 17
## Weighting          : term frequency (tf)
```

```
camino        <- file.path('datos/diccionarios/negative_words_es.txt')
listanegativa <- read.csv(camino,header=F,quote="",fileEncoding="Latin1")
listanegativa <- as.matrix(listanegativa)
listanegativa <- as.vector(listanegativa)
listanegativa <- wordStem(listanegativa, language='spanish')

pnegativas.tdm<- DocumentTermMatrix(corpus_clean, control = list(tokenize    = 'word',
                                                                 dictionary  = listanegativa,
                                                                 stopwords   = FALSE,
                                                                 wordLengths = c(0, Inf),
                                                                 stemming    = 'spanish'))
pnegativas.tdm
```

```
## <<DocumentTermMatrix (documents: 11, terms: 2100)>>
## Non-/sparse entries: 4132/18968
## Sparsity           : 82%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

Podemos ahora resumir toda esta información en una única tabla con el conteo de palabras positivas, negativas y neutras de cada una de las películas.

```
positivas <- apply(ppositivas.tdm,1,sum)
negativas <- apply(pnegativas.tdm,1,sum)
num_total <- apply(dtm,1,sum)

total.df  <- data.frame(positive = positivas,
                        negative = negativas,
                        neutral  = num_total - positivas - negativas)
total.df
```

```
##                                     positive negative neutral
## Always.txt                              1140     1084    5023
## Atrapame si puedes.txt                  1403      860    4818
## el mundo perdido Jurassic Park.txt      1449     1212    5841
## En busca del arca perdida.txt           2337     1613    8722
## E.T., el extraterrestre.txt             1937     1339    7799
## Indiana Jones y la Ultima Cruzada.txt   1615      980    6001
## La lista de Schindler.txt               1714     1442    6760
## Munich.txt                              1563     1576    6488
## Parque Jurasico.txt                     1693     1112    6628
## Salvar al soldado Ryan.txt              1662     1557    6569
## Tiburon.txt                             1696     1393    7008
```

Para poder comparar las películas entre sí, vamos a normalizar los valores, dividiendo entre el total de palabras del documento. Además vamos a generar un score, definido como la diferencia entre el porcentaje de palabras positivas y negativas (así, películas con scores positivos tienen más palabras positivas que negativas y viceversa).

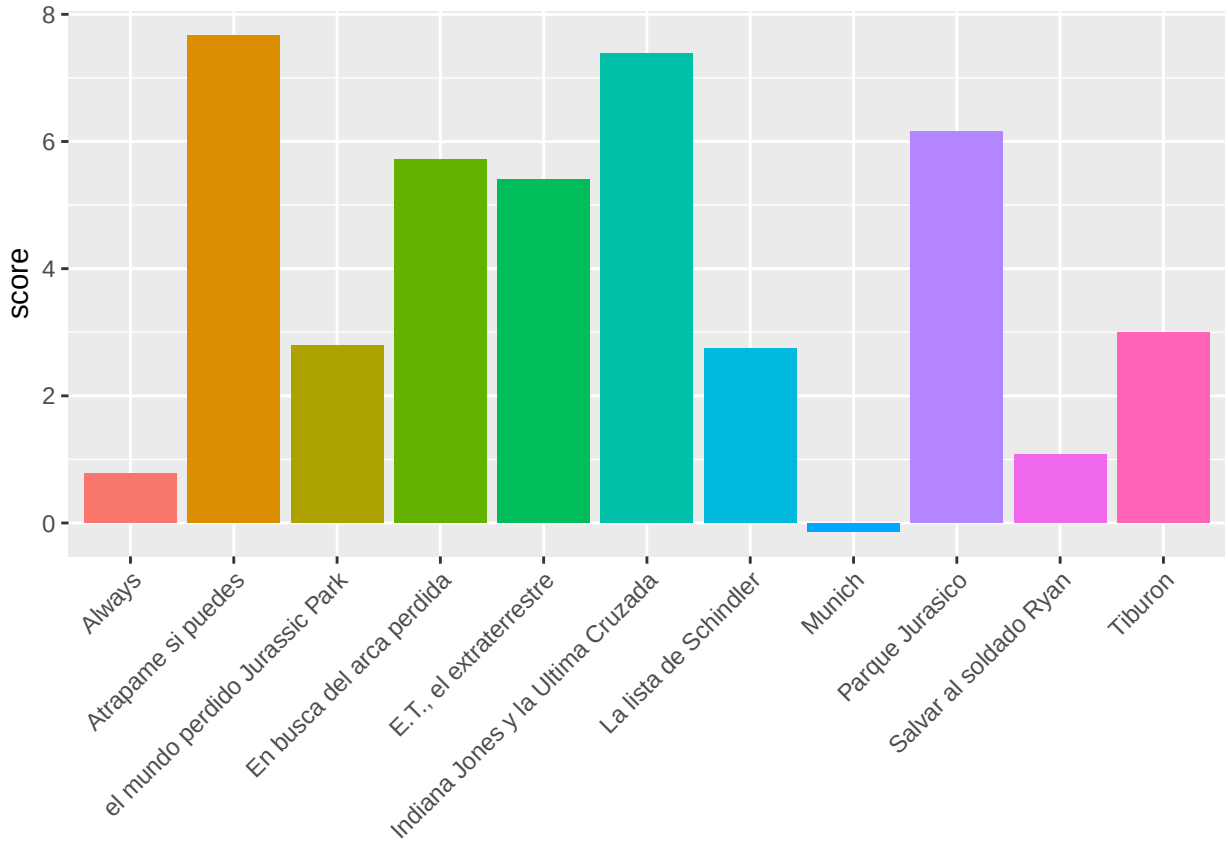
```
scores <- as.data.frame(apply(total.df, 2, function(x)(100 * x / num_total)))
scores$score <- scores$positive - scores$negative
scores$movie <- substr(rownames(scores),1,nchar(rownames(scores))-4)
scores
```

```
##                                       positive negative  neutral
## Always.txt                            15.73065 14.95791 69.31144
## Atrapame si puedes.txt                19.81359 12.14518 68.04124
## el mundo perdido Jurassic Park.txt    17.04305 14.25547 68.70148
## En busca del arca perdida.txt         18.44223 12.72885 68.82891
## E.T., el extraterrestre.txt           17.48984 12.09029 70.41986
## Indiana Jones y la Ultima Cruzada.txt 18.78781 11.40065 69.81154
## La lista de Schindler.txt             17.28520 14.54215 68.17265
## Munich.txt                            16.23559 16.37062 67.39379
## Parque Jurasico.txt                   17.94763 11.78840 70.26397
## Salvar al soldado Ryan.txt            16.97998 15.90723 67.11279
## Tiburon.txt                           16.79707 13.79618 69.40675
##                                           score
## Always.txt                            0.7727335
## Atrapame si puedes.txt                7.6684084
## el mundo perdido Jurassic Park.txt    2.7875794
## En busca del arca perdida.txt         5.7133838
## E.T., el extraterrestre.txt           5.3995485
## Indiana Jones y la Ultima Cruzada.txt 7.3871568
## La lista de Schindler.txt             2.7430415
## Munich.txt                           -0.1350369
## Parque Jurasico.txt                   6.1592282
## Salvar al soldado Ryan.txt            1.0727421
## Tiburon.txt                           3.0008914
##                                                                movie
## Always.txt                                                    Always
## Atrapame si puedes.txt                                Atrapame si puedes
## el mundo perdido Jurassic Park.txt       el mundo perdido Jurassic Park
## En busca del arca perdida.txt                 En busca del arca perdida
## E.T., el extraterrestre.txt                     E.T., el extraterrestre
## Indiana Jones y la Ultima Cruzada.txt Indiana Jones y la Ultima Cruzada
## La lista de Schindler.txt                        La lista de Schindler
## Munich.txt                                                      Munich
## Parque Jurasico.txt                                      Parque Jurasico
## Salvar al soldado Ryan.txt                      Salvar al soldado Ryan
## Tiburon.txt                                                    Tiburon
```

Análisis

Una vez hemos limpiado los datos y generado una puntuación de sentimiento por película, vamos a analizar los datos resultantes. En primer lugar vamos a ver las puntuaciones que hemos obtenido por película.

```
ggplot(scores, aes(x = movie, y = score, fill = movie)) +
  geom_col() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "none",
        axis.title.x = element_blank())
```

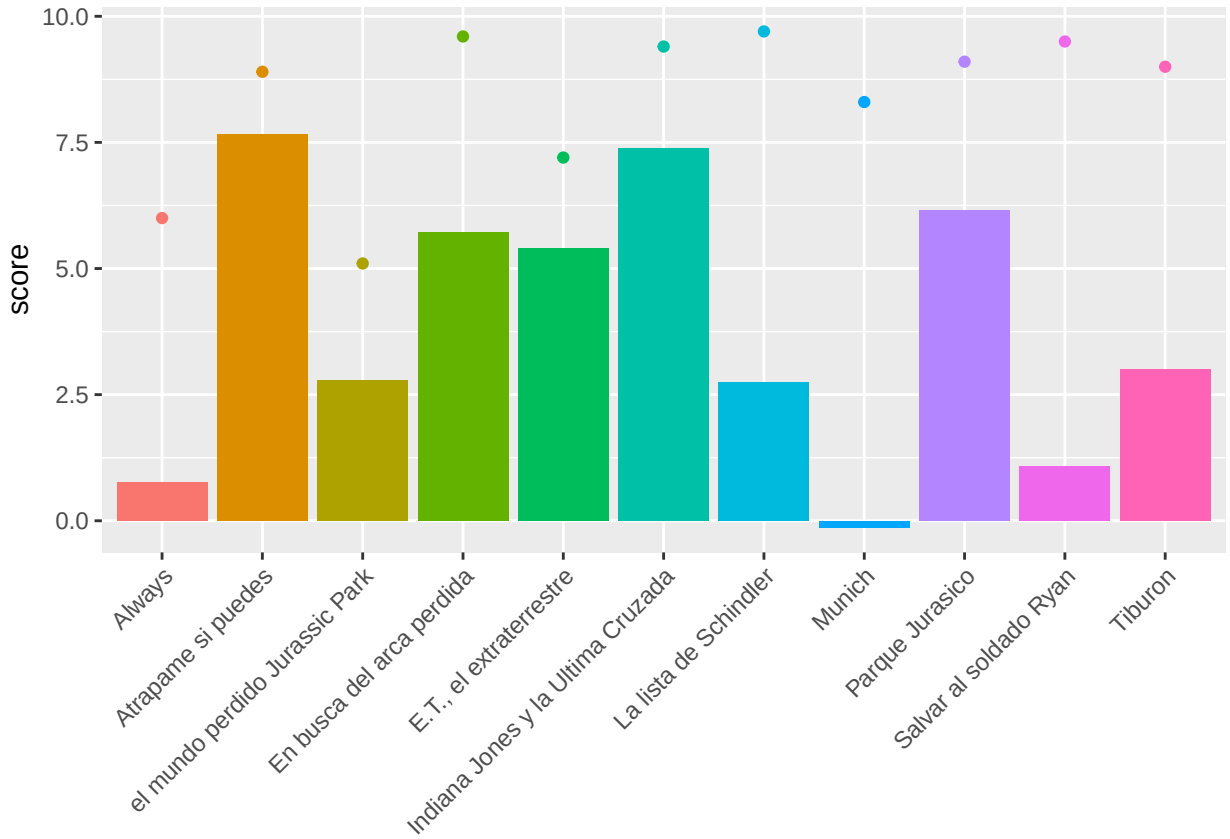


Para tener algo de contexto, al gráfico anterior le vamos a añadir las puntuaciones de las películas en la página web Rotten Tomatoes.

```
rotten_tomatoes <- data.frame('movie' = c('Always', 'Atrapame si puedes', 'el mundo perdido Jurassic Pa
                                          'E.T., el extraterrestre', 'Indiana Jones y la Ultima Cruzada
                                          'Munich', 'Parque Jurasico', 'Salvar al soldado Ryan', 'Tibur
                              'rt_score' = c(6.0, 8.9, 5.1, 9.6, 7.2, 9.4, 9.7, 8.3, 9.1, 9.5, 9.0))

scores <- merge(scores, rotten_tomatoes, 'movie')
```

```
ggplot(scores, aes(x = movie, y = score, fill = movie)) +
  geom_col() +
  geom_point(aes(y = rt_score, color = movie)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "none",
        axis.title.x = element_blank())
```



Vemos que hay cierta relación entre las puntuaciones de Rotten Tomatoes y nuestra puntuación de sentimiento, pero en algunos casos como La lista de Schindler, Munich o E.T. las diferencias son grandes. Esto puede deberse a que las palabras que se emplean en las críticas sean tradicionalmente negativas o positivas (como guerra o conflicto en las primeras), aunque no lo sean en el contexto. Vamos a ver el coeficiente de correlación entre ambas puntuaciones.

```
print(paste0('Correlación entre scores: ', cor(scores$score, scores$rt_score)))
```

```
## [1] "Correlación entre scores: 0.321656729295967"
```

```
print(paste0('Correlación entre positivas y RT score: ', cor(scores$positive, scores$rt_score)))
```

```
## [1] "Correlación entre positivas y RT score: 0.462023639156O3"
```

```
print(paste0('Correlación entre negativas y RT score: ', cor(scores$negative, scores$rt_score)))
```

```
## [1] "Correlación entre negativas y RT score: -0.194508133982665"
```

Parece que nuestra hipótesis es correcta, ya que vemos tres cosas:

1. Existe una correlación débil (0.32) entre nuestra puntuación y la de Rotten Tomatoes.
2. Hay una correlación más fuerte (0.46) entre el porcentaje de palabras positivas y la puntuación de Rotten Tomatoes.

3. La correlación entre palabras negativas y puntuación de Rotten Tomatoes es muy débil (-0.19), lo que refuerza la hipótesis de que estamos tratando como negativas palabras que no lo son en este contexto.

Para entender mejor qué palabras están marcando la diferencia entre positivas y negativas vamos a generar nubes de palabras. Primero, una nube con todas las palabras del corpus.

```
m <- as.matrix(TermDocumentMatrix(corpus_clean))
m <- sort(rowSums(m), decreasing = TRUE)
colorPalette <- brewer.pal(8, "Paired")
wordcloud(names(m), m, min.freq = 200, colors=colorPalette)
```

Lo primero que nos llama la atención es que la palabra más repetída es 'peledcul', que es claramente un typo al escribir 'película'. Lo siguiente que vemos es que hay muchas palabras, como 'Spielberg' o 'historia' que deberíamos considerar stopwords en nuestro contexto.

Vamos a ver ahora la nube de palabras positivas.

```
m <- as.matrix(TermDocumentMatrix(corpus_clean, control = list(tokenize    = 'word',
                                                               dictionary  = listapositiva,
                                                               stopwords   = FALSE,
                                                               wordLengths = c(0, Inf),
                                                               stemming    = 'spanish')))
m <- sort(rowSums(m), decreasing = TRUE)
colorPalette <- brewer.pal(8, "Paired")
wordcloud(names(m), m, min.freq = 100, colors=colorPalette)
```

Parece que en general todas tienen bastante sentido. Veamos las negativas.

```
m <- as.matrix(TermDocumentMatrix(corpus_clean, control = list(tokenize    = 'word',
                                                               dictionary  = listanegativa,
                                                               stopwords   = FALSE,
                                                               wordLengths = c(0, Inf),
                                                               stemming    = 'spanish')))
m <- sort(rowSums(m), decreasing = TRUE)
colorPalette <- brewer.pal(8, "Paired")
wordcloud(names(m), m, min.freq = 100, colors=colorPalette)
```

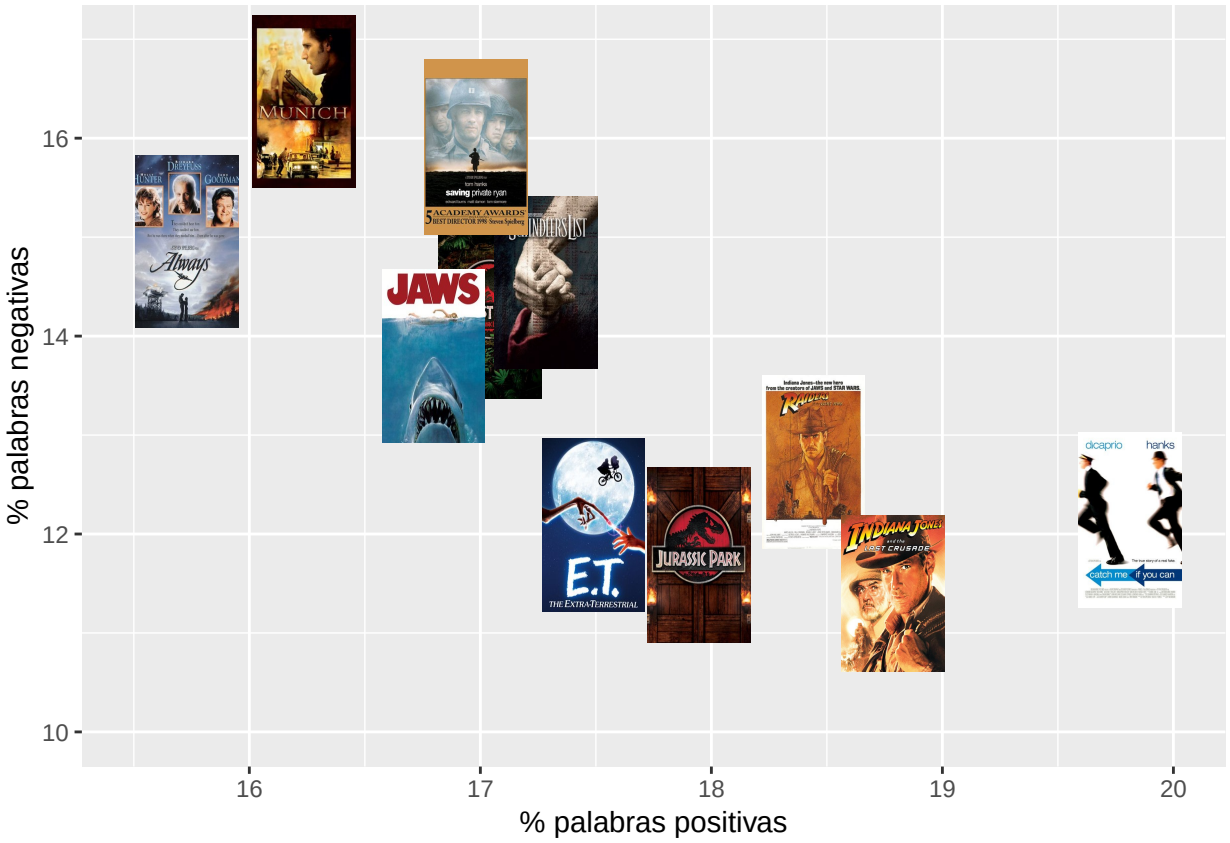
Aquí vemos alguna inconsistencia que puede explicar la baja correlación, como 'terror' o 'soldado'.

Vamos a terminar haciendo un plot de Chernoff Faces, para ver cómo de contentos están los personajes después de ver cada película (y por los memes, claro).

```
faces(scores[5:5] , labels = scores$movie, cex=0.9, print.info = FALSE)
```

Ya estamos! Siempre está bien una visualización chula, así que vamos a intentar una con las carátulas de las películas según su porcentaje de palabras positivas y negativas.

```
scores$image <- paste0('datos/imagenes/', scores$movie, '.jpg')
ggplot(scores, aes(x=positive, y=negative, image=image)) +
  geom_image(asp=1.7, size=0.09) + ylim(c(10, 17)) + xlim(15.5, 20) +
  xlab('% palabras positivas') + ylab('% palabras negativas')
```


% palabras negativas
16
14
12
10
% palabras positivas
16
17
18
19
20
MUNICH
saving private ryan
5 ACADEMY AWARDS
BEST DIRECTOR 1998 Steven Spielberg
SCHINDLER'S LIST
JAWS
Always
E.T.
THE EXTRA-TERRESTRIAL
JURASSIC PARK
INDIANA JONES
THE LAST CRUSADE
dicaprio
hanks
catch me if you can