Tema4_Ejercicio

Fran Camacho

2025-01-15

Tema4 - Ejercicio

El fichero "movie-pang02.csv", disponible en la carpeta de Pruebas de Evaluación del Máster, contiene una muestra de 2000 reviews de películas de la página web IMDB utilizada en el artículo de Pang, B. y Lee, L., "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts, Proceedings of ACL 2004".

Dichas reviews están etiquetadas mediante la variable class como positivas ("Pos") o negativas ("Neg"). Utilizando dicho dataset, elabore un modelo de clasificación de reviews en base a su texto siguiendo el procedimiento descrito en el capítulo 4 del texto base en el ejemplo de SMS Spam. En particular, genere las nubes de palabras para las reviews positivas y negativas, y obtenga el modelo asignando los valores 0 y 1 al parámetro laplace de la función naiveBayes(), comparando las matrices de confusión de cada variante del modelo.

Paso 2: Explorar y preparar los datos

Carga de paquetes que son necesarios para diversas funciones.

```
#install.packages("tm")           # text mining
library(tm)
#install.packages("SnowballC")    # stemming
library(SnowballC)
#install.packages("wordcloud")    # wordclouds
library(wordcloud)
#install.packages("RColorBrewer") # color palettes
library("RColorBrewer")

#install.packages("naivebayes")
library(naivebayes)

#install.packages("gmodels")   # cross tables
library(gmodels)
#install.packages("caret")     # data partitioning, confusion matrix
library(caret)
```

Cargar datos del fichero movie-pang02.csv. Examinar su contenido.

```
# import the CSV file
movies_raw <- read.csv(file.path("Chapter04", "Movie_pang02.csv"))
```

Examinamos la estructura y el aspecto del fichero importado:

```
#See the structure
str(movies_raw)
```

```
## 'data.frame':    2000 obs. of  2 variables:
##  $ class: chr  "Pos" "Pos" "Pos" "Pos" ...
##  $ text : chr  " films adapted from comic books have had plenty of success   whether they re about su
```

```
#See some records
head(movies_raw)
```

```
##   class
## 1   Pos
## 2   Pos
## 3   Pos
## 4   Pos
## 5   Pos
## 6   Pos
##
## 1
## 2
## 3
## 4     jaws   is a rare film that grabs your attention before it shows you a single image on screen
## 5
## 6
```

```
#See some records
tail(movies_raw)
```

```
##      class
## 1995   Neg
## 1996   Neg
## 1997   Neg
## 1998   Neg
## 1999   Neg
## 2000   Neg
##
## 1995
## 1996  if anything     stigmata   should be taken as a warning against releasing similarly themed filr
## 1997
## 1998
## 1999
## 2000
```

La columna "class" es de tipo carácter. Ya que se trata en realidad de una variable categórica, es conveniente transformarla en un factor:

```
#better to convert class into a factor
movies_raw$class <- factor(movies_raw$class)
```

Y examinamos el resultado

```r
table(movies_raw$class)
```

```
##
##  Neg  Pos
## 1000 1000
```

Vemos que hay tantas críticas positivas como negativas.

Procesado del texto de las críticas

Procedemos ahora a preparar el fichero para que pueda ser procesado mediante el algoritmo naive bayes: Hay que eliminar mayúsculas y signos de puntuación, números, realizar la lematización ("stemming" en inglés) ...

El primer paso consiste en la creación del objeto corpus con todas las críticas:

```r
#create corpus
movies_corpus <- VCorpus(VectorSource(movies_raw$text))
print(movies_corpus)
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 2000
```

```r
#inspect(movies_corpus) omitted for brevity
```

```r
# examine the movies corpus
# lapply(movies_corpus[1:5], as.character)
# we omit the output for brevity
```

Limpieza de los textos:

```r
# Process the reviews (we will use some functions from the packate tm)

#To lowercase (content was already in lowercase and without punctuation signs, and but just in case)
movies_corpus_clean <- tm_map(movies_corpus, content_transformer(tolower))

#Check the result (output omitted for brevity)
#as.character(movies_corpus[[1]])
#as.character(movies_corpus_clean[[1]])

#Remove numbers
movies_corpus_clean <- tm_map(movies_corpus_clean, removeNumbers)

#Remove stopwords
# check words and languages with ?stopwords
movies_corpus_clean <- tm_map(movies_corpus_clean, removeWords, stopwords())
#Remove punctuation signs
movies_corpus_clean <- tm_map(movies_corpus_clean, removePunctuation)
```

```r
#Carry out the stemming:
# To apply the wordStem() function to an entire corpus of text documents, the tm package includes
# the stemDocument() transformation.
movies_corpus_clean <- tm_map(movies_corpus_clean, stemDocument)

#Finally eliminate unneeded whitespace produced by previous steps
movies_corpus_clean <- tm_map(movies_corpus_clean, stripWhitespace)

#Check the final result (output omitted for brevity)
#before cleaning
#as.character(movies_corpus[[1]])
#after
#as.character(movies_corpus_clean[[1]])
```

Finalmente, se procede a la **"tokenización"** de los textos de las críticas.

Mediante la función DocumentTermMatrix() del paquete "tm", se crea una estructura llamada "document-term matrix (DTM)", que como su nombre indica es una matriz, cuyas filas consisten en los textos(documentos) y cuyas columnas son las palabras que aparecen en esos documentos.

```r
movies_dtm <- DocumentTermMatrix(movies_corpus_clean)
movies_dtm
```

```
## <<DocumentTermMatrix (documents: 2000, terms: 24951)>>
## Non-/sparse entries: 501761/49400239
## Sparsity           : 99%
## Maximal term length: 53
## Weighting          : term frequency (tf)
```

Ahora hay que crear los conjuntos de entrenamiento y de test. Las críticas vienen ordenadas, primero las 1000 críticas positivas, y después las 1000 críticas negativas. Por tanto hay que crear estos conjuntos de manera aleatoria.

```r
#Set seed to make the process reproducible
set.seed(8)

#partitioning data frame into training and testing sets
train_indices <- createDataPartition(movies_raw$class, times=1, p=.75, list=FALSE)

#create training set
movies_dtm_train <- movies_dtm[train_indices, ]

#create testing set
movies_dtm_test  <- movies_dtm[-train_indices, ]

#create labels sets
movies_train_labels <- movies_raw[train_indices, ]$class
movies_test_labels <- movies_raw[-train_indices, ]$class

#view number of rows in each set
nrow(movies_dtm_train)  # 1500
```

```
## [1] 1500
```

```r
nrow(movies_dtm_test)   # 500
```

```
## [1] 500
```

```r
length(movies_train_labels)  # 1500
```

```
## [1] 1500
```

```r
length(movies_test_labels)   # 500
```

```
## [1] 500
```

Vamos a comprobar ahora que los dos conjuntos tienen la misma proporción de críticas positivas y negativas (si no, el entrenamiento no serviría para nada):

```r
prop.table(table(movies_train_labels))
```

```
## movies_train_labels
## Neg Pos
## 0.5 0.5
```

```r
prop.table(table(movies_test_labels))
```

```
## movies_test_labels
## Neg Pos
## 0.5 0.5
```

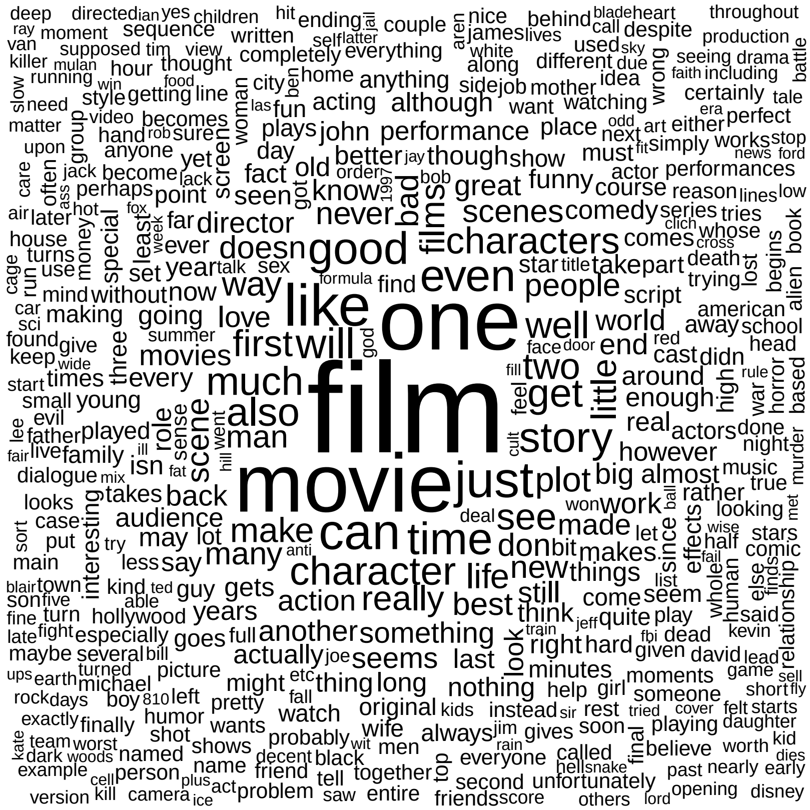
La proporción se mantiene en los dos conjuntos.

Visualización mediante nubes de palabras (wordclouds)

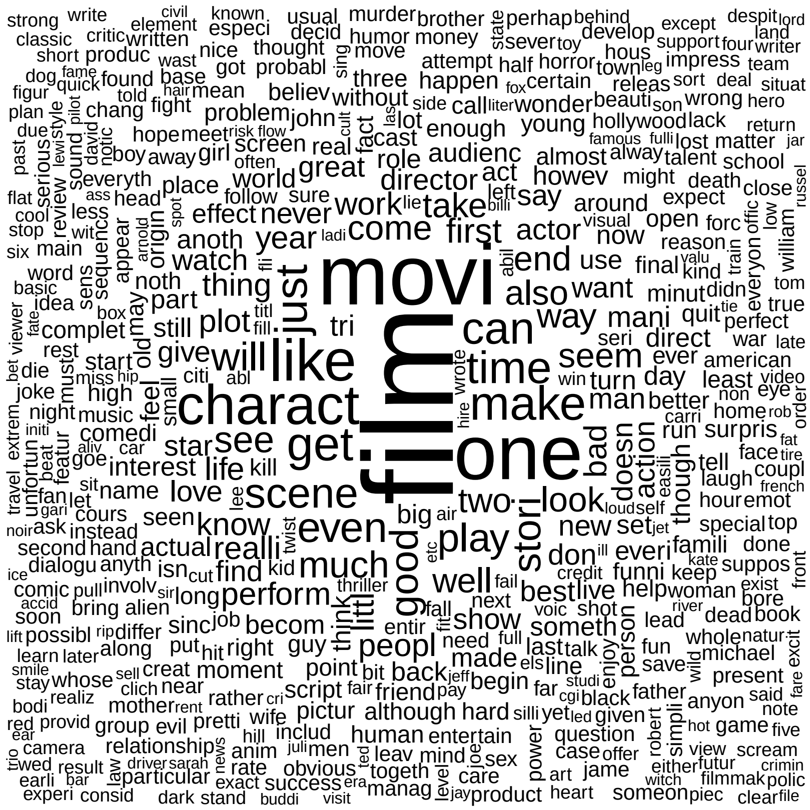
Se puede obtener una nube con las palabras de todas las críticas, y también una nube para las críticas positivas y otra para las negativas.

Total de críticas:

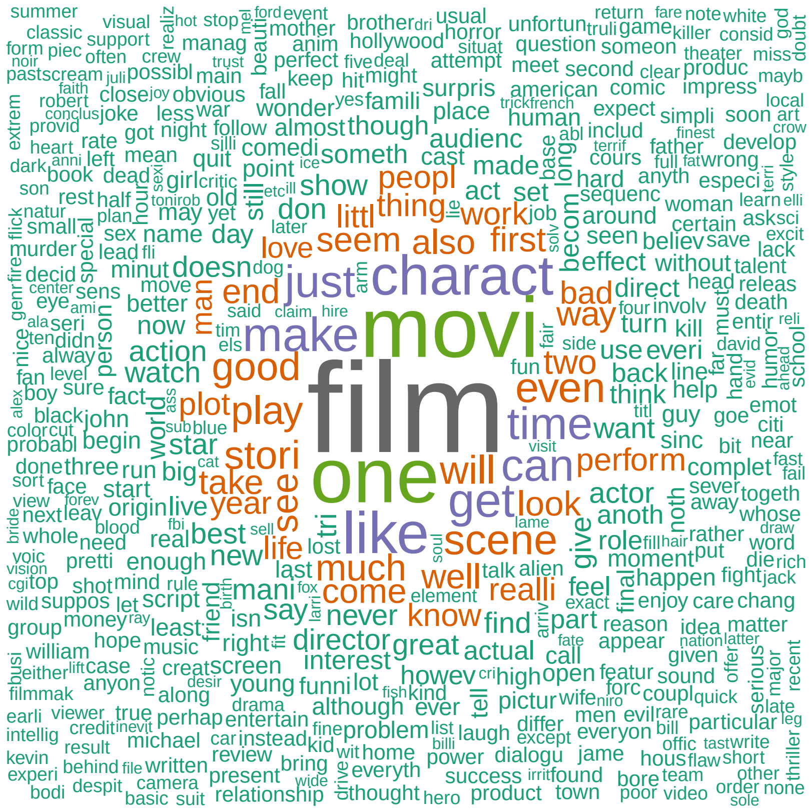
```r
wordcloud(movies_raw$tex, min.freq = 50, random.order = FALSE)
```




```r
#Using the corpus:
wordcloud(movies_corpus_clean, min.freq = 50, random.order = FALSE)
```



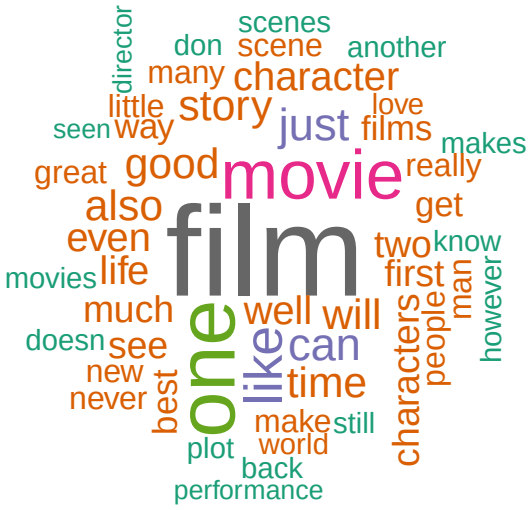

```r
#Using the corpus:
wordcloud(movies_corpus_clean, min.freq = 50, random.order = FALSE, colors=brewer.pal(8,"Dark2"))
```



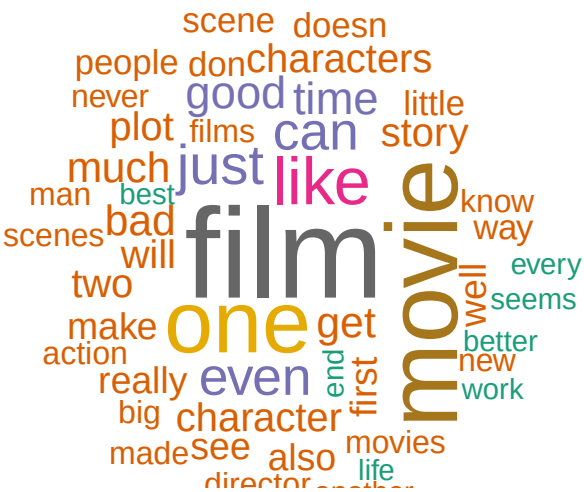
Críticas positivas y críticas negativas por separado:

```
pos <- subset(movies_raw, class == "Pos")
neg <- subset(movies_raw, class == "Neg")

wordcloud(pos$text, max.words = 50, random.order = FALSE, colors=brewer.pal(8,"Dark2"))
```




```
wordcloud(neg$text, max.words = 50, random.order = FALSE, colors=brewer.pal(8,"Dark2"))
```




```
#wordcloud(pos$text, max.words = 50, scale = c(3, 0.5), random.order = FALSE)
#wordcloud(neg$text, max.words = 50, scale = c(3, 0.5), random.order = FALSE)
```

La única diferencia que aprecio entre las dos nubes, es que en la nube negativa se encuentra la palabra “bad” (y “never”). Palabras positivas como “good”, like”, “well”, aparecen en las 2 agrupaciones.