clustering

Alejandro Campoy Nieves

25 de enero de 2019

Clustering sobre datos de texto

Vamos a ver cómo podemos aplicar técnicas de clustering a las distintas columnas de texto de nuestro dataset. Para ello vamos a explicar en mayor detalle el proceso seguido en la columna Benefits Review preprocesada y luego mostraremos los resultados con los otros textos de una forma más directa.

Esta es un técnica no supervisada, por lo que no sabemos las clases que clasifica realmente, solo agrupa las instancias en grupos sin definir sin tener nada de información previa acerca de la estructura.

Clustering de Benefits Review

Lo primero que debemos hacer es cargar el dataset preprocesado que hemos creado en esta práctica. Formamos el corpus correspondiente a la columna de beneficios que estamos analizando.

```
# Cargamos los tados
datos_train <- read.table("datos/datos_train_preprocesado.csv",
                          sep=",", comment.char="",quote = "\"", header=TRUE)
# Establecemos la semilla
set.seed(3)
corpus = tm::Corpus(tm::VectorSource(datos_train$benefits_preprocesad))
```

Como ya hemos hecho en alguna ocasión a lo largo de esta práctica, vamos a crear una matriz de términos en la que cada fila es un documento o comentario de beneficios y cada columna es un término o palabra que aparece en los textos.

```
tdm <- tm::DocumentTermMatrix(corpus)
tdm.tfidf <- tm::weightTfIdf(tdm)
tfidf.matrix <- as.matrix(tdm.tfidf)
```

Para poder trabajar con clustering necesitamos trabajar con valores numéricos continuos. Es por ello que hemos utilizado de nuevo TFIDF (esta vez hemos usado una función de más alto nivel, sin entrar en detalles). Necesitamos tener estos datos como matriz para poder continuar con los siguientes pasos.

```
tfidf.matrix[1:20,1:20]
```

Esta es una muestra de la esquina superior izquierda de la matriz para ver el formato explicado anteriormente. Los valores son pesos normalizados de los términos en cada uno de los comentarios. Estos pesos se basan en la frecuencia en la que aparece cada palabra en cada documento.

En una de las técnicas que veremos más adelante, necesitamos las 100 palabras con mayor peso. La forma que se nos ocurrió de hacerlo consistía en hacer la sumatoria de los pesos por columnas, de esta forma teníamos el peso total de cada palabra para el total de documentos, y después ordenarlas en función del valor. El problema reside en que no podemos perder el número de filas que tenemos (la forma de la matriz), cosa que ocurre si hacemos la sumatoria por columnas. Entonces se nos ocurrió lo siguiente:

```
#Hacemos la sumatoria de columnas para averiguar el peso total de cada término.
v <- c()
for(i in 1:length(tfidf.matrix[,1])){
  v[i] <- sum(tfidf.matrix[,i])
}
```

```
    Terms
Docs     agent      alon   congest  dysfunct   failur     heart hypertens      left     manag    mangag
  1  0.6624946 0.4594991 0.4643742 0.6624946 1.126869 0.7208779 0.5634344 0.3946569 0.3946569 0.7249946
  2  0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  3  0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  4  0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  5  0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  6  0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  7  0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  8  0.0000000 0.2827687 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  9  0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  10 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  11 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  12 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  13 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  14 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  15 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  16 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  17 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  18 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  19 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  20 0.0000000 0.0000000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
    Terms
Docs     overt  progress      slow ventricular     along       also  although     birth       con    condom
  1  0.7249946 0.4312171 0.3901476   0.7249946 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  2  0.0000000 0.0000000 0.0000000   0.0000000 0.4213707 0.19757006 0.3558139 0.4153573 0.7249946 0.5999946
  3  0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  4  0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  5  0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  6  0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  7  0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  8  0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.2189624 0.0000000 0.0000000 0.0000000
  9  0.0000000 0.0000000 0.0000000   0.0000000 0.1532257 0.07184366 0.0000000 0.0000000 0.0000000 0.0000000
  10 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  11 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  12 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  13 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  14 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  15 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  16 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  17 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  18 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  19 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
  20 0.0000000 0.0000000 0.0000000   0.0000000 0.0000000 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000
```

Figure 1: Matriz de pesos para cada uno de los términos en cada uno de los documentos.

```
# Averiguamos el orden de índices de los pesos de mayor
# a menor y nos quedamos con los 100 mas grandes.
indices <- order(v,decreasing = TRUE)[1:100]

# Nos creamos una submatriz de la original con los términos mas relevantes.
tfidf.matrix100 <- tfidf.matrix[,indices]
tfidf.matrix100 <- t(tfidf.matrix100)
```

En definitiva, en lugar de ordenar la matriz lo que obtenemos es una lista de los índices en la posición que deberían estar para que estuvieran ordenados sus pesos de mayor a menor. Después solo tenemos que quedarnos con los 100 primeros índices (los 100 términos con mayor peso) y generar una submatriz de la matriz original con las columnas que corresponden a esos índices.

Es importante decir que la técnica que vamos a utilizar después con estos 100 términos necesita la traspuesta de la matriz que nosotros hemos calculado. Tardamos un tiempo en darnos cuenta de que la y y la x estaban invertidas para la función que usabamos y se debía a este problema.

El siguiente paso sería calcular la matriz de distancias. Este es el proceso que consume más tiempo de todo el desarrollo de clustering por lo que debemos ser pacientes.

```
# Calculamos la distancia de cada término en cada documento
dist.matrix = proxy::dist(tfidf.matrix, method = "cosine")
dist.matrix100 = proxy::dist(tfidf.matrix100, method = "cosine")
```

Hemos visto distintas técnicas en clase de teoría sobre el cálculo de distancias y semejanzas. Como lo que tenemos es un vector de pesos por cada fila (documento), nos pareció muy apropiado calcular la semejanza utilizando el coseno del ángulo que forman dos vectores.

Como resultado, obtenemos una matriz en la que cada valor representa la medida de proximidad del item de la fila con el item de la columna, por lo que hay mucha información.

Ya tenemos los datos sobre los comentarios de los beneficios de los medicamentos listos para ser utilizados en la técnica de clustering. Principalmente hemos creado tres estrategias para generar los clusters como se puede observar a continuación.

```
#-------------------------------------------------------------------------------------------#

### FUNCIONES: QUEREMOS HACER CLUSTERING EN FUNCIÓN DEL NÚMERO DE CLUSTERS QUE DESEEMOS.

#-------------------------------------------------------------------------------------------#

clustering.kmeans <- function(centers=5){
  # Calulamos el cluster
  cluster <- kmeans(tfidf.matrix, centers)

  # Para el de 100 terminos más frecuentes
  cluster100 <- kmeans(tfidf.matrix100, centers)

  # Pintamos la nube de puntos perteneciente a cada uno de los cluster
  # (cada color es un cluster)
  points <- cmdscale(t(dist.matrix), k = 2)
  palette <- colorspace::diverge_hcl(centers) # Creating a color palette

  plot(points, main = 'K-Means clustering', col = as.factor(cluster$cluster),
       mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
       xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')

}

#-------------------------------------------------------------------------------------------#

clustering.hierarchial <- function(centers=5){
  # Calulamos el cluster
  cluster <- hclust(dist.matrix, method = "ward.D2")

  # Para el de 100
  cluster100 <- hclust(dist.matrix100, method = "ward.D2")

  # Pintamos la nube de puntos de cada uno de ellos
  points <- cmdscale(dist.matrix, k = 2)
  palette <- colorspace::diverge_hcl(centers) # Creating a color palette
  previous.par <- par(mfrow=c(1,2), mar = rep(1.5, 4))

  plot(points, main = 'Hierarchical clustering',
       col = as.factor(cutree(cluster, k = centers)),
       mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
       xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')

  # Para el de 100
```

```
  plot(cluster100, cex=0.9, hang=-1)
  rect.hclust(cluster100, centers, border=rainbow(centers))

}

#-------------------------------------------------------------------------------------#

clustering.dbscan <- function(centers=5){
  # Calulamos el cluster
  cluster <- dbscan::hdbscan(dist.matrix, minPts = 10)

  # Pintamos la nube de puntos de cada uno de ellos
  points <- cmdscale(dist.matrix, k = 2)
  palette <- colorspace::diverge_hcl(centers)

  plot(points, main = 'Density-based clustering',
       col = as.factor(cluster$cluster),
       mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
       xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
}
#-------------------------------------------------------------------------------------#
```

Cada función utilizará por defecto 5 grupos en los que clasificar los diferentes términos. Si se desea cambiar este hecho es posible simplemente especificando el número de centros por medio de un argumento al llamar la función. Vamos a comentar el proceso de cada uno de ellos y los datos obtenidos.

```
# Primera técnica
clustering.kmeans()
```

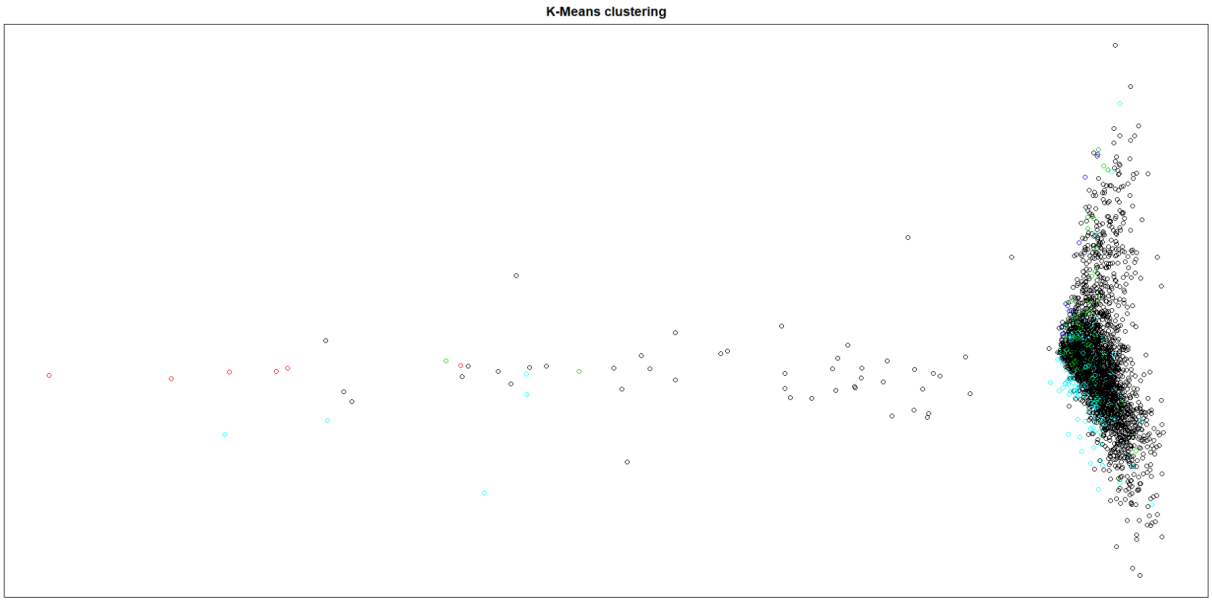

Figure 2: Agrupamiento particional por k-means.

Estuvimos debatiendo el significado de lo obtenido con esta técnica. Llegamos a la conclusión de que los métodos de clustering particional no son muy adecuados para el texto de nuestro dataset. Vemos que se genera una nube de puntos muy agruapada en la parte derecha y un porcentaje muy bajo de los items comienzan a esparcirse por la izquierda. Estás técnicas de agrupamiento particional necesitan que las nubes de puntos se encuentren disjuntas entre sí para poder situar los centroides correctamente. Casi todo el conjunto pertenece al grupo de color negro, se puede ver también como ha sacado un centroide de color azul cian en la parte izquierda inferior de la nube de puntos grande.

En esta ocasión no hemos realizado validación por silueta de los grupos obtenidos, a simple vista se puede ver que los grupos no son muy buenos y que no están claramente diferenciados.

Entonces, decidimos probar con un agrupamiento no particional, como es el jerárquico.

```
#Segunda técnica
clustering.hierarchial()
```

En este caso, vemos que teniendo la misma nube de puntos, los resultados son más razonables. Suponemos que se debe a su criterio de anidación de términos, en los que a partir de la matriz de distancias va agrupando los términos cada vez en grupos más grandes. Vemos que hay una mejor cohesión en los grupos obtenidos, sobretodo en el grupo rojo y negro.

Como esta es la mejor técnica que sacamos, quisimos mostrar más información del proceso. Obtuvimos los 100 términos más frecuentes como se mencionó anteriormente y obtuvimos un dendograma como los vistos en clase de teoría. Podemos ver que los grupos alcanzados tienen coherencia. Por ejemplo, en el primero agrupa sangre, presión, más bajo, alto, nivel...

Dependiendo del número de cluster que fijemos, especifica los grupos en niveles más altos o bajos de las llaves que vemos en la imagen. Por lo que podemos decir que siempre sigue la misma técnica de agrupamiento y lo que cambia es el momento en el que separa los cluster cuando especificamos el número de grupos que queremos tener, lo cual nos pareció curioso destacar.

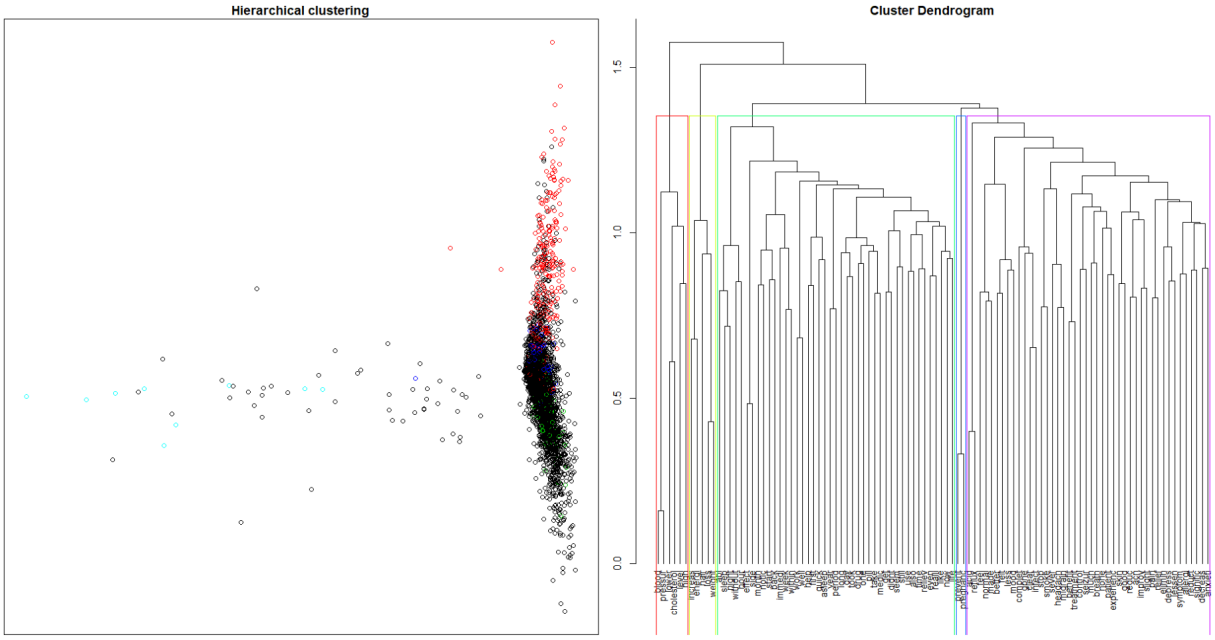

Figure 3: Agrupamiento jerárquico y muestra de anidamiento con los 100 términos más frecuentes.

```
# Tercera técnica
clustering.dbscan()
```

En el agrupamiento por densidad tenemos el mismo problema que con k-medias. Los agrupamientos particionales no funcionan bien cuando los puntos en el espacio se encuentran tan unidos de forma uniforme y no disjunta.

Intentamos también realizar un word cloud en función de estos clústers, aunque de momento no hemos tenido éxito en ese aspecto.

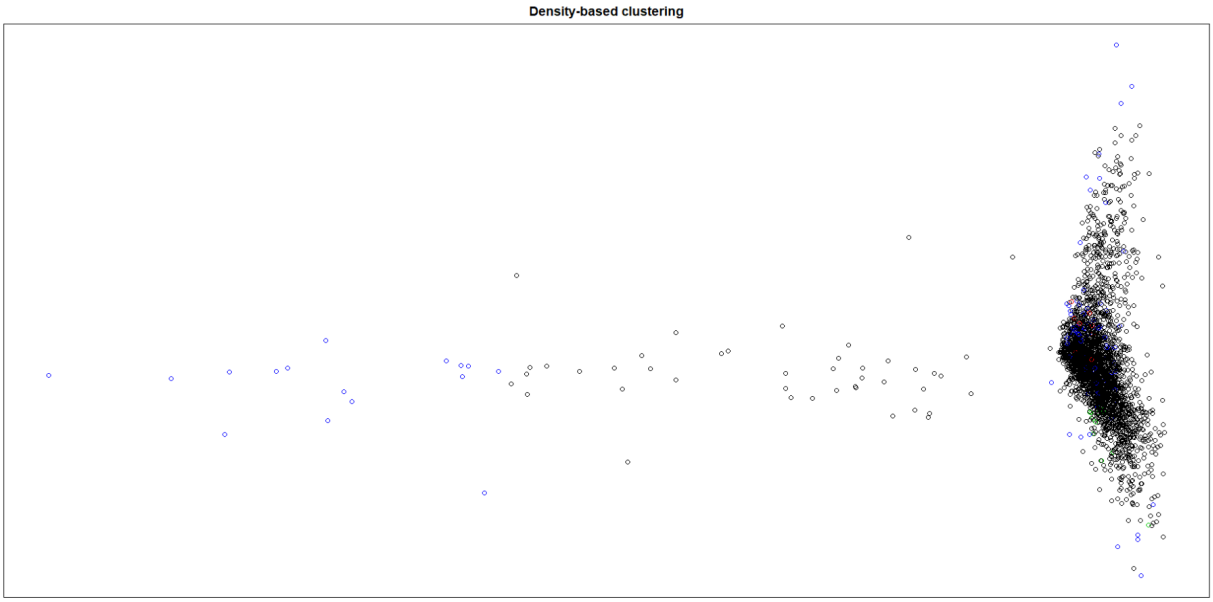

Figure 4: Agrupamiento particional por densidad.