Sistemas de recomendación y evaluación de modelos

Eric Bellet

18 mayo 2016

Sistemas de recomendación

Introducción

El siguiente sistema de recomendación esta basado en en 131000 transacciones de artículos de un periódico, donde existen 9 artículos por cada contenido, los cuales son:

- Deportes.
- Politica.
- Variedades.
- Internacional.
- Nacionales.
- Sucesos.
- Comunidad.
- Negocios.
- Opinión.

Arules

Se utilizó el paquete arules de R para poder generar reglas de asociación.

Primera parte

Modificar su dataset de tal manera que no se lean los identificadores de los artículos como itemN sino por su tipo de contenido contenido/articuloN. Ejemplo: {item1, item10, item81} es la transacción {deportes/articulo1, politica/articulo1, opinion/articulo9}.

Para modificar el dataset dada estas condiciones se realizó lo siguiente:

- Se cambio el nombre la columna 5 y se creó la columna **articles** y se llenó con vectores númericos que representan los items que observó el usuario.

```
#Cambio el nombre de la columna para que tenga coherencia con el ejemplo
dado.
colnames(periodico)[5] <- "items"
#Creo la columna de los articulos
periodico$articles <- periodico$items

#Se sabe que el portal ofrece 9 tipos de contenidos
```

```
#y nos ofrecen solo información de 9 artículos.

#Obtengo el numero de los articulos.
periodico$articles <- strsplit(gsub("[{}item]","",periodico$articles), ",")
```

- Dado los vectores de cada fila de la columna *articles* se modificó para que tuviera el formato *contenido/articuloN* utilizando la función genArticles.

```
genArticles <- function(articles){
  # Genera la columna articles utilizando los items.
  #
  # Args:
  #   articles: Son arreglos númericos que representan los items (EJ:
{item1,item9,item63} -> 1,9,63)
  #
  # Returns:
  #   Retorna la columna articles.
  articulo <- ""
  for (i in 1:length(articles)) {

    if (as.integer(articles[i]) <= 9 & as.integer(articles[i]) >= 1){
      articulo <- paste(articulo, gsub("
","",paste("deportes/articulo",articles[i])))

    }
    if (as.integer(articles[i]) <= 18 & as.integer(articles[i]) >= 10){
      articulo <- paste(articulo, gsub("
","",paste("politica/articulo",(as.integer(articles[i])-9))))

    }
    if (as.integer(articles[i]) <= 27 & as.integer(articles[i]) >= 19){
      articulo <- paste(articulo, gsub("
","",paste("variedades/articulo",(as.integer(articles[i])-18))))

    }
    if (as.integer(articles[i]) <= 36 & as.integer(articles[i]) >= 28){
      articulo <- paste(articulo, gsub("
","",paste("internacional/articulo",(as.integer(articles[i])-27))))

    }
    if (as.integer(articles[i]) <= 45 & as.integer(articles[i]) >= 37){
      articulo <- paste(articulo, gsub("
","",paste("nacionales/articulo",(as.integer(articles[i])-36))))

    }
    if (as.integer(articles[i]) <= 54 & as.integer(articles[i]) >= 46){
      articulo <- paste(articulo, gsub("
","",paste("sucesos/articulo",(as.integer(articles[i])-45))))

    }
```

```
    if (as.integer(articles[i]) <= 63 & as.integer(articles[i]) >= 55){
      articulo <- paste(articulo, gsub("
","",paste("comunidad/articulo",(as.integer(articles[i])-54))))

    }
    if (as.integer(articles[i]) <= 72 & as.integer(articles[i]) >= 64){
      articulo <- paste(articulo, gsub("
","",paste("negocios/articulo",(as.integer(articles[i])-63))))

    }
    if (as.integer(articles[i]) <= 81 & as.integer(articles[i]) >= 73){
      articulo <- paste(articulo, gsub("
","",paste("opinion/articulo",(as.integer(articles[i])-72))))

    }
  }
  return(articulo)

}


#Modifico el dataset con las condiciones dadas.
periodico$articles <- lapply(periodico$articles, genArticles)
```

- Finalmente se realizan modificaciones en la columna articles para que tengan el formato adecuado y se calcula el tiempo total que estuvó un usuario observando los artículos.

```
#Convierto los espacios en ,
periodico$articles <- gsub(" ",",",periodico$articles)

#Elimino la primer valor del string.
periodico$articles <- substring(periodico$articles, 2)

#Calculo el tiempo totan el segundos que dura el usuario en la pagina.
periodico$tiempototal <- difftime(periodico$exit, periodico$entry, units =
"secs")
```

Este es el resultado de la primera parte.

```
head(periodico[,c(1,5,6,7)])
```

```
##   X                 items
## 1 1  {item1,item9,item63}
## 2 2   {item1,item2,item3}
## 3 3 {item9,item43,item57}
## 4 4 {item2,item14,item72}
## 5 5              {item11}
## 6 6        {item6,item53}
##                                                    articles tiempototal
## 1   deportes/articulo1,deportes/articulo9,comunidad/articulo9    556 secs
```

```
## 2    deportes/articulo1,deportes/articulo2,deportes/articulo3    874 secs
## 3 deportes/articulo9,nacionales/articulo7,comunidad/articulo3    167 secs
## 4    deportes/articulo2,politica/articulo5,negocios/articulo9    309 secs
## 5                                          politica/articulo2   1967 secs
## 6                       deportes/articulo6,sucesos/articulo8    350 secs
```

Generación de matriz de transacciones

Para poder generar las reglas utilizando el paquete **arules** es necesario tener una matriz de transacciones, para crearla se realizó lo siguiente.

Se utilizó la función **llenar**, el cual dado lo un vector númerico que representa los items va llenando con 1 aquellos artículos que el usuario observó.

```
#Generar la matriz de transacciones.
#fila es un row inicializado en 0.
fila <- matrix(data = 0, nrow = 1, ncol = 81)
#------------------FUNCTION llenar-----------------
llenar <- function(periodico,fila){
  # Llena la matriz de transacciones con 1 en caso de que el usuario observo
los articulos.
  #
  # Args:
  #   periodico: Recibe los items.
  #   fila: recibe una fila vacia.
  #
  # Returns:
  #   Retorna la matriz de transacciones llena.
  items <- as.numeric(unlist(strsplit(gsub("[{}item]","",unlist(periodico)),
","")))
  fila[items]=1
  return(fila)
}

#--------------END FUNCTION llenar-----------------

#Lleno la matriz con 1 donde un usuario observo un articulo
matriz <- lapply(periodico$items, llenar,fila)
```

Transformo el resultado obtenido por la función **llenar** en una matriz y se le asigna el nombre correspondiente a cada columna.

```
#Transformo matriz en una matrix.
matriz <- matrix(unlist(matriz), byrow=T, ncol=81)

#Nombro las columnas
colnames(matriz) <-  c(gsub(" ","",paste("deportes/articulo",1:9)),gsub("
","",paste("politica/articulo",1:9)),
                       gsub(" ","",paste("variedades/articulo",1:9)), gsub("
","",paste("internacional/articulo",1:9)),
```

```
                        gsub(" ","",paste("nacionales/articulo",1:9)), gsub("
","",paste("sucesos/articulo",1:9)),
                        gsub(" ","",paste("comunidad/articulo",1:9)), gsub("
","",paste("negocios/articulo",1:9)),
                        gsub(" ","",paste("opinion/articulo",1:9)))
```

Detección de usuarios bots

El periódico tiene sospechas de que existen bots que están ganando dinero al hacer clicks en artículos con promociones. En consecuencia, le piden a usted que realice un análisis exploratorio sobre las transacciones para determinar el número de posibles transacciones bot que tienen en su dataset (ellos aceptan que si una persona ve un artículo más de 20 segundos entonces no es un bot).

Para calcular el número de articulos que observó un usuario se suman las filas de la matriz de transacciones, y luego utilizando el **tiempo total** de un usuario observando los artículos, se calcula cuales son los usuarios **bots** bajo el criterio si una persona ve un artículo 20 segundos o menos entonces es un bot.

```
#El número de posibles transacciones bot que tienen en su dataset
#(ellos aceptan que si una persona ve un artículo más de 20 segundos entonces
no es un bot).
periodico$numItems <- rowSums(matriz)
numerobots <- periodico[periodico$numItems >= periodico$tiempototal/20,]
print(paste("El numero de transacciones bot es:",nrow(numerobots)))
```

```
## [1] "El numero de transacciones bot es: 6599"
```

Finalmente nos interesa generar las reglas de asociación sin los usuarios bots, por lo tanto se eliminan del dataset y de la matriz de transacciones.

```
periodicoSinBots <- periodico[-numerobots$X,]
#Utilizamos la matriz de transacciones sin las transacciones bots.
matriz <- matriz[-numerobots$X,]
#Matriz de transacciones
mm <- matriz
matriz <- as(matriz, "transactions")
```

Segunda parte

Conocer los tipos de usuarios que ingresan a su página (ellos creen que son 8 tipos de usuarios) y tratar de determinar la proporción de cada tipo de usuario.

En esta fase existen 2 enfoques, el primero es ver los tipos de usuarios solamente por el tipo de **contenido** que ven y el segundo por **contenido y artículo**. Mi criterio fue hacerlo por **contenido y artículo** ya que me parece más específico, por ejemplo: supongamos que deportes/articulo1 habla sobre beisbol, deportes/artículo2 habla sobre fútbol, es distinto un tipo de usuario que solo por contenido (deportes) a uno por **contenido/artículo**.

Si agrupamos utilizando el **lhs** de las reglas, obtenemos estos 8 grupos:

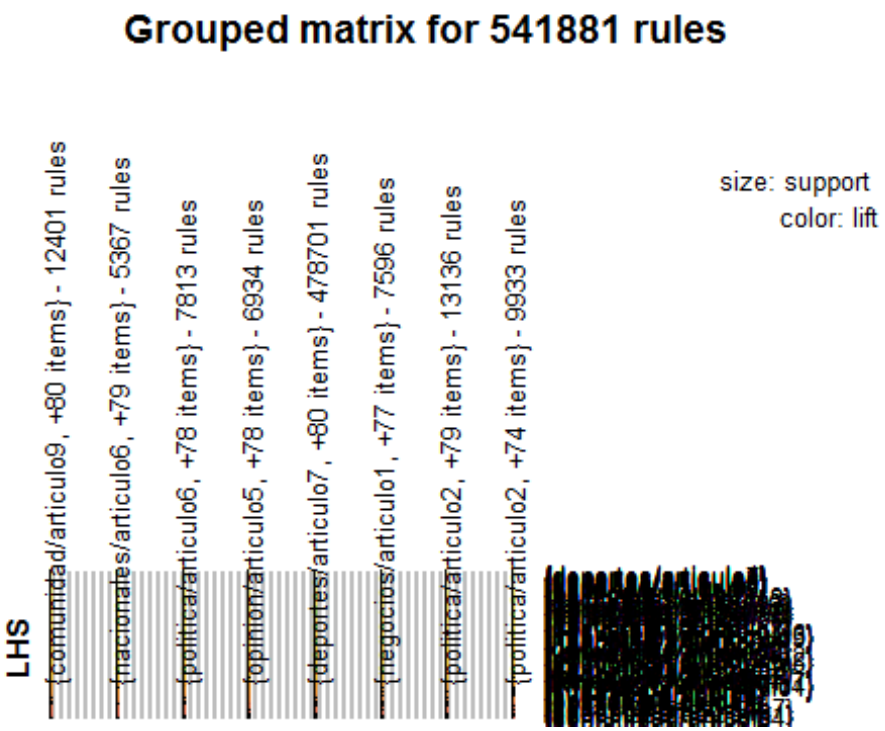
```r
rules <- apriori(matriz,parameter = list(support = 0.000008019181883,
confidence = 1.0))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport      support minlen
##           1    0.1    1 none FALSE            TRUE 8.019182e-06      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 0
```

```
## Warning in apriori(matriz, parameter = list(support = 8.019181883e-06,
confidence = 1)): You chose a very low absolute support count of 0. You might
run out of memory! Increase minimum support.
```

```
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[81 item(s), 124701 transaction(s)] done [0.05s].
## sorting and recoding items ... [81 item(s)] done [0.01s].
## creating transaction tree ... done [0.11s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.28s].
## writing ... [541881 rule(s)] done [0.20s].
## creating S4 object  ... done [0.34s].
```

```r
plot(rules, method = "grouped", control = list(k = 8))
```



Utilizando **kmedias** usando el algoritmo de **Harting-Wong** se obtiene la siguiente proporción de cada cluster:

```
kmedias <- kmeans(mm, 8,algorithm = "Hartigan-Wong")
periodicoSinBots$cluster <- kmedias$cluster
table(periodicoSinBots$cluster)

##
##     1     2     3     4     5     6     7     8
## 16944 63784  3336  8341  3477  3510 12383 12926
```

Luego probamos con clusterización mediante **propagación por afinidad** que toma como conjunto de datos principal similitudes entre los datos. El objetivo es minimizar los errores al cuadrado, cada similaridad se establece como el inverso del error cuadrado (distancia euclídea). Este es un algoritmo de agrupamiento, (clustering), dado un conjunto de puntos y una medida de similaridad entre ellos, proporciona grupos de puntos similares y además para cada grupo da un ejemplar representativo. La medida de similaridad es la información mutua entre cada par de variables aleatorias, la cual expresa la información que ellas comparten.

El número de grupos o clusters no se determina de antemano, sino que es entregado por el algoritmo de propagación de afinidades. El tamaño de cada cluster determina obviamente el número de factores que aparecen en cada distribución marginal y por lo tanto el número de parámetros a estimar.

Podemos observar que utilizando clusterización mediante la propagación por afinidad detecta **8 clusters** donde los tipos de usuarios son los siguientes: * deportes-articulos1. *

variedades-articulo4. * sucesos-articulo1. * sucesos-articulo6. * negocios-articulo4. * negocios-articulo-7. * opinion-articulo4. * opinion-articulo7.

```
#Calculamos la matriz de similaridad utilizando el inverso del error cuadrado (distancia euclidea).
sim <- crossprod(mm)
sim <- sim / sqrt(sim)
#Corremos la affinity propagation
clust_ap <- apcluster(sim)
show(clust_ap)
```

```
##
## APResult object
##
## Number of samples     =  81
## Number of iterations  =  137
## Input preference      =  7.745967
## Sum of similarities   =  1689.242
## Sum of preferences    =  61.96773
## Net similarity        =  1751.21
## Number of clusters    =  8
##
## Exemplars:
##    deportes/articulo1 variedades/articulo4 sucesos/articulo1
##    sucesos/articulo6 negocios/articulo4 negocios/articulo7
##    opinion/articulo4 opinion/articulo7
## Clusters:
##    Cluster 1, exemplar deportes/articulo1:
##       deportes/articulo1 deportes/articulo2 deportes/articulo3
##       deportes/articulo4 deportes/articulo5 deportes/articulo6
##       deportes/articulo7 deportes/articulo8 deportes/articulo9
##       politica/articulo1 politica/articulo2 politica/articulo3
##       politica/articulo4 politica/articulo5 politica/articulo6
##       politica/articulo7 politica/articulo8 politica/articulo9
##       variedades/articulo2 variedades/articulo3 variedades/articulo5
##       variedades/articulo6 variedades/articulo8 variedades/articulo9
##       internacional/articulo1 internacional/articulo2
##       internacional/articulo3 internacional/articulo4
##       internacional/articulo5 internacional/articulo6
##       internacional/articulo7 internacional/articulo8
##       internacional/articulo9 nacionales/articulo1 nacionales/articulo2
##       nacionales/articulo3 nacionales/articulo4 nacionales/articulo5
##       nacionales/articulo6 nacionales/articulo7 nacionales/articulo8
##       nacionales/articulo9 sucesos/articulo2 sucesos/articulo4
##       sucesos/articulo5 sucesos/articulo8 comunidad/articulo1
##       comunidad/articulo2 comunidad/articulo3 comunidad/articulo4
##       comunidad/articulo5 comunidad/articulo6 comunidad/articulo7
##       comunidad/articulo8 comunidad/articulo9 negocios/articulo1
##       negocios/articulo2 negocios/articulo3 negocios/articulo5
##       negocios/articulo6 negocios/articulo8 negocios/articulo9
```

```
##       opinion/articulo1 opinion/articulo2 opinion/articulo3
##       opinion/articulo5 opinion/articulo6 opinion/articulo8
##       opinion/articulo9
##    Cluster 2, exemplar variedades/articulo4:
##       variedades/articulo1 variedades/articulo4 variedades/articulo7
##    Cluster 3, exemplar sucesos/articulo1:
##       sucesos/articulo1
##    Cluster 4, exemplar sucesos/articulo6:
##       sucesos/articulo3 sucesos/articulo6 sucesos/articulo7
##       sucesos/articulo9
##    Cluster 5, exemplar negocios/articulo4:
##       negocios/articulo4
##    Cluster 6, exemplar negocios/articulo7:
##       negocios/articulo7
##    Cluster 7, exemplar opinion/articulo4:
##       opinion/articulo4
##    Cluster 8, exemplar opinion/articulo7:
##       opinion/articulo7
```

Tercera parte

Dado un usuario nuevo que haya ingresado a n artículos (n variable), poder recomendar un artículo n+1 y así aumentar el compromiso del cliente con su portal web. Como usted sabe, para poder calcular las reglas necesita como entrada MinSupport y MinCofianza. Sin embargo, el cliente desconoce cuáles son estos valores en consecuencia es tarea de usted determinar y justficar los mismos de acuerdo a su criterio.

Para recomendar un artículo n + 1 a un usuario se utilizó reglas de asociación. La función implementada es **recomendar** que recibe la matriz de transacciones y el n que representa los artículos ingresados por el usuario. Se generan las reglas con el algoritmo **apriori** con un soporte alto y una confianza baja, en caso de no encontrar un **lhs** en las reglas generadas se va disminuyendo el soporte hasta que se generen reglas, en el caso que no se generen reglas en una cantidad de iteraciones, se recomienda el **rhs** o artículo más popular o que tenga mayor número de apariciones. En el caso que si se generan reglas, se ordenan por confianza de mayor a menor, luego se toman todas las reglas que tengan la misma **confianza máxima** y se ordenan por **soporte** de mayor a menor y finalmente se recomienda el artículo que tenga mayor soporte. En otras palabras se toman las reglas que tengan la confianza máxima y luego la que tenga mayor soporte.

```
recomendar <- function(n, matriz){
  #plot(rules,method="graph",interactive=TRUE,shading=NA)
  #plot(rules, measure=c("support","lift"), shading="confidence");
  #plot(rules, shading="order", control=list(main ="Two-key plot"));


  rules <- apriori(matriz,parameter = list(support = 0.1, confidence = 0.0))

  reglas <- subset(rules, subset = lhs %ain% n )
```

```
  len <- length(reglas)
  div <- 0.1
  cont <- 0
  #En el caso que no se generaron reglas con ese soporte, voy disminuyendo el
soporte
  while (len == 0){
    div <- div /10
    rules <- apriori(matriz,parameter = list(support = div, confidence =
0.0))
    reglas <- subset(rules, subset = lhs %ain% n )
    len <- length(reglas)
    cont <- cont + 1
    #Un criterio de parada ya que puede ser infinito
    if (cont == 7){
      break()
    }
  }
  if (length(reglas)==0){
    trendigtop<-inspect(rules@rhs)
    return(row.names(sort(table(trendigtop),decreasing=TRUE))[1])
  }else{

  confianzaAlta <-sort(reglas, decreasing = TRUE,
                       na.last = NA,by = "confidence",
                       order = FALSE)

  #Obtengo la confianza maxima para luego tomar todos los articulos que
posean esa confianza
  maxConfianza <- max(quality((confianzaAlta))[2])
  #Posiciones que poseen la misma confianza.
  confianzaAlta<- subset(confianzaAlta, subset = confidence == maxConfianza)


  #Ahora ordeno por soporte las que tienen la confianza mas alta
  soportealto <- (sort(confianzaAlta, decreasing = TRUE,
                                  na.last = NA,by = "support",
                                  order = FALSE)[1])
  articulorecomendar <- inspect(soportealto@rhs[1])
  return(articulorecomendar$items[1])
  }
}

n <- c("deportes/articulo6","internacional/articulo9")
articuloARecomendar <- recomendar(n, matriz)

print(paste("El artículo que se recomienda es:", articuloARecomendar))

## [1] "El artículo que se recomienda es: {deportes/articulo1}"
```

Cuarta parte

Conocer las 10 visitas con mayor tiempo de estadía en la página y las 10 visitas con menor tiempo de estadía en la página. * Las 10 visitas con mayor tiempo de estadía en la página:

```
timemayor10 <- periodicoSinBots[order(periodicoSinBots$tiempototal,decreasing
= T),][1:10,c(1,7)]
print("10 visitas con mayor tiempo de estadía en la página:")
```

```
## [1] "10 visitas con mayor tiempo de estadía en la página:"
```

```
print(timemayor10)
```

```
##             X tiempototal
## 93676   93676  12264 secs
## 7511     7511  12234 secs
## 80516   80516  11743 secs
## 122879 122879  11597 secs
## 130641 130641  11508 secs
## 66995   66995  11481 secs
## 111953 111953  11421 secs
## 23099   23099  11419 secs
## 55628   55628  11381 secs
## 48007   48007  11372 secs
```

- Las 10 visitas con menor tiempo de estadía en la página:

```
timemenor10 <-
periodicoSinBots[order(periodicoSinBots$tiempototal,decreasing =
F),][1:10,c(1,7)]
print("10 visitas con menor tiempo de estadía en la página:")
```

```
## [1] "10 visitas con menor tiempo de estadía en la página:"
```

```
print(timemenor10)
```

```
##           X tiempototal
## 2144   2144     21 secs
## 14600 14600     21 secs
## 26914 26914     21 secs
## 30571 30571     21 secs
## 34895 34895     21 secs
## 39574 39574     21 secs
## 50391 50391     21 secs
## 60586 60586     21 secs
## 63057 63057     21 secs
## 77010 77010     21 secs
```

Quinta parte

Conocer las 10 transacciones con mayor número de apariciones en el dataset.

- Los 10 articulos con mayor numero de apariciones.

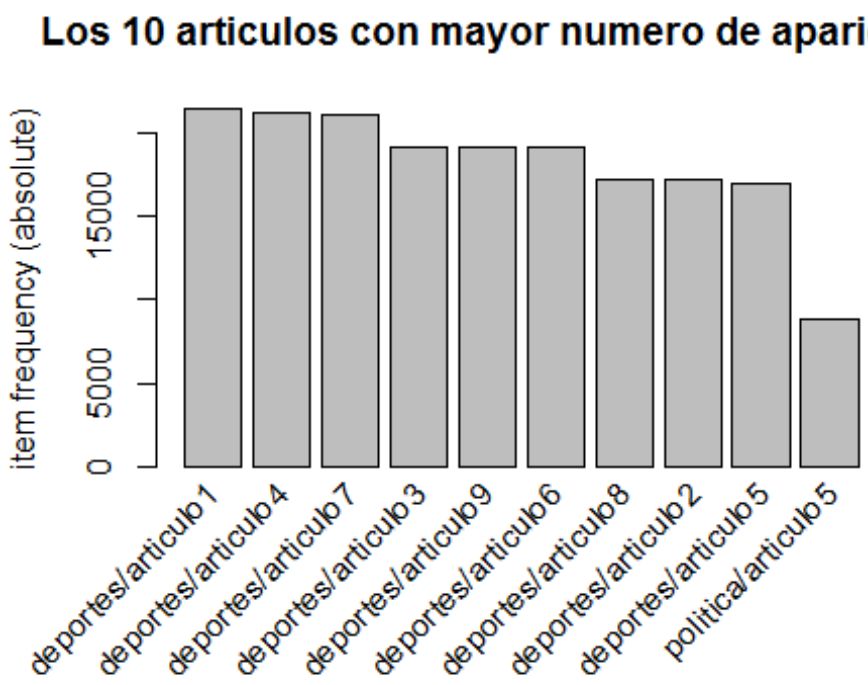
```
top10 <- sort(itemFrequency(matriz, type = "absolute"),decreasing = T)[1:10]
print("Los 10 articulos con mayor numero de apariciones son:")

## [1] "Los 10 articulos con mayor numero de apariciones son:"

print(top10)

## deportes/articulo1 deportes/articulo4 deportes/articulo7
##              21379              21214              21066
## deportes/articulo3 deportes/articulo9 deportes/articulo6
##              19156              19065              19060
## deportes/articulo8 deportes/articulo2 deportes/articulo5
##              17232              17172              16945
## politica/articulo5
##               8834

itemFrequencyPlot(matriz,topN=10,type="absolute", main ="Los 10 articulos con
mayor numero de apariciones.")
```




```
#TRANSACCIONES:
```

- Las 10 transacciones con mayor número de apariciones en el dataset.

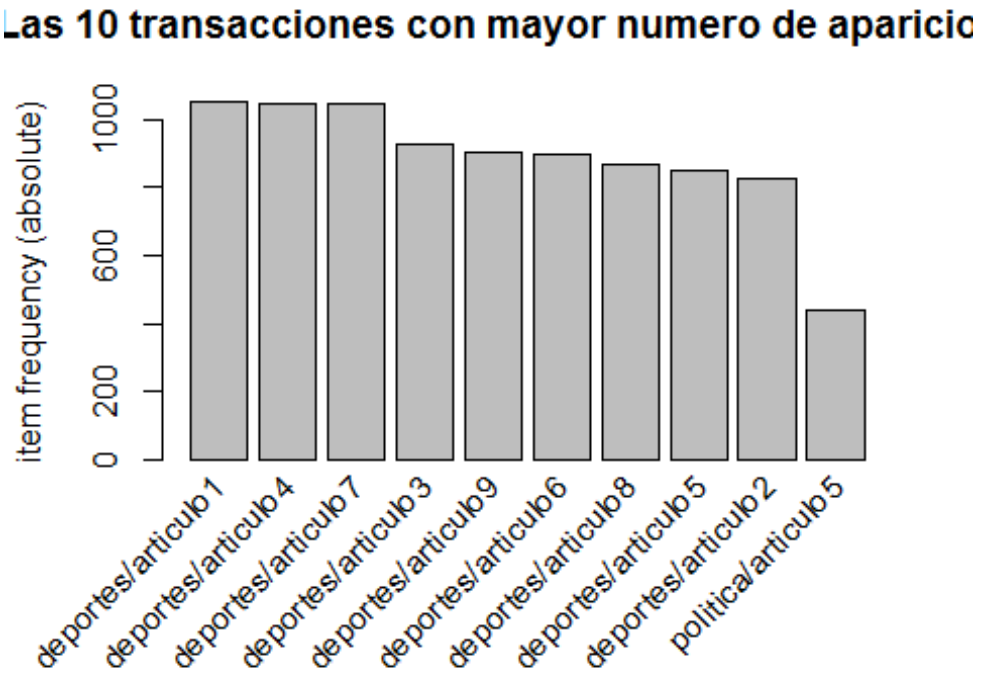
```
MatrizSinBots = split(periodicoSinBots$articles,periodicoSinBots$X)
MatrizSinBots = as(MatrizSinBots,"transactions")
top10transacciones <- sort(itemFrequency(MatrizSinBots, type =
"absolute"),decreasing = T)[1:10]
print("Las 10 transacciones con mayor numero de apariciones son:")
```

```
## [1] "Las 10 transacciones con mayor numero de apariciones son:"

print(top10transacciones)

## deportes/articulo1 deportes/articulo4 deportes/articulo7
##               1050               1045               1043
## deportes/articulo3 deportes/articulo9 deportes/articulo6
##                924                904                894
## deportes/articulo8 deportes/articulo5 deportes/articulo2
##                868                850                828
## politica/articulo5
##                440

itemFrequencyPlot(MatrizSinBots,topN=10,type="absolute", main = "Las 10
transacciones con mayor numero de apariciones.")
```



Evaluación de modelos

Las curvas ROC(Receiver Operating Characteristics) son grá???cos usados como técnica de visualización, organización y selección de clasi???cadores basados en su rendimiento.Los parámetros de gra???cador son:

1. Los scores por instancia (no necesariamente ordenados).
2. La verdadera clase de las instancias.
3. La clase target. En el caso de que nclass > 2 entonces haga un enfoque 1 vs all.

```
generate_ROC <- function(scores, real, target){
  # Genera una curva de ROC.
  #
  # Args:
  #   scores: Los scores por instancia (no necesariamente ordenados).
  #   real: La verdadera clase de las instancias.
  #   target:  La clase target. En el caso de que nclass > 2 entonces haga un
enfoque 1 vs all.
  #
  # Returns:
  #   Genera la curva ROC.


  #En caso que hayan 2 clases nada mas.
  if (length(unique(real)) <= 2){
    df <- data.frame(scores,real)
    df <- df[order(df$scores, decreasing = TRUE),]
    graficador(df$scores, df$real, target)

  }else{
    #En caso que hayan mas de 2 clases.

   clases <- unique(real)
   clases <- clases[!clases %in% target]

  #-------------1 vs all--------------------
    for (i in 1:length(clases)) {
       #Se toma el valor positivo como uno solo, y los demas como negativos.
       df <- data.frame(scores,real)
       class1 <- df[df$real == target,]
       class2 <- df[df$real == clases[i],]
       df <- merge(x = class1, y = class2, all = TRUE)
       df <- df[order(df$scores, decreasing = TRUE),]
       graficador(df$scores, df$real, target)
    }

  }#endif

}#endfunction
```

La función graficador va graficando los puntos y uniendo las líneas.

```
graficador <- function(scores, real, target){
  # Genera una curva de ROC.
  #
  # Args:
  #   scores: Los scores por instancia (no necesariamente ordenados).
  #   real: La verdadera clase de las instancias.
  #   target:  La clase target. En el caso de que nclass > 2 entonces haga un
enfoque 1 vs all.
```

```r
  #
  # Returns:
  #   Genera la curva ROC.
  divy <- 1/length(which(real==target))
  divx <- 1/(length(real)-length(which(real==target)))
  contx <- 0
  conty <- 0

  negativaclase <- unique(real)
  negativaclase <- negativaclase[!negativaclase %in% target]


  plot(x=NULL,y=NULL,xlim=c(0, 1), ylim=c(0, 1), xlab="False positive rate",
ylab="True positive rate",main=paste("Clase target:",target,".","Clase
negativa:", negativaclase[1],"."))
  lines(x = c(0,1), y = c(0,1), col = "blue")
  puntosx <<- c(contx)
  puntosy <<- c(conty)
  id <- order(puntosx)


  i <- 1
  while (i != (length(scores)+1)){
    puntosx <<- c(puntosx, contx)
    puntosy <<- c(puntosy, conty)
    #Existen varios elementos con el mismo score??
    samescore <- length(which(scores==scores[i]))
    if (samescore > 1){
      contador <- 0
      contxORIGEN <- contx
      contyORIGEN <- conty
      points(contx, conty, col = "red")
      while (contador != samescore ) {
        #Si es target
        if (real[i] == target){

          #points(contx, conty, col = "red")

          conty <- conty + divy


        }else{
          #Si es negativo
          #points(contx, conty, col = "red")

          contx <- contx + divx
        }
        #capaz hay que restar
        i <- i + 1
```

```
        contador <- contador + 1

      }#endwhile

      lines(x = c(contxORIGEN, contx) , y = c(contyORIGEN,conty), col =
"green")
    }else{



    #Si es target
    if (real[i] == target){

      points(contx, conty, col = "red")

      lines(x = c(contx, contx), y = c(conty, conty + divy) , col = "green")

      conty <- conty + divy
      i <- i + 1

    }else{
    #Si es negativo
      points(contx, conty, col = "red")

      lines(x = c(contx, contx + divx) , y = c(conty,conty), col = "green")

      contx <- contx + divx
      i <- i + 1
    }

    }

  }#endfor
  #Grafico el ultimo punto.
  puntosx <<- c(puntosx,1)
  puntosy <<- c(puntosy,1)
  points(1, 1, col = "red")
  lines(x = c(contx, 1), y = c(conty, 1) , col = "green")

  #???legend("bottomright", title = paste("ROC area:",auc(puntosx, puntosy)))
}#endgraficador
```

En la carpeta **shiny** se encuentra la aplicación hecha en shiny de esta parte.