Sistemas de Recomendación

Alejandro Acosta

18 de mayo del 2016

Parte I - Sistemas de recomendación:

En este documento se realizará un sistema de recomendación de artículos para un periódico dado un datasets de datos con transacciones de distintos usuarios. Se verificará además cuáles de estos usuarios son bots, y serán eliminados de las transacciones. Se responderán las dudas del cliente sobre las visitas con mayor y menor tiempo de estadía (esta última sin contar transacciones hechas por bots), las 10 transacciones que más se repiten en el dataset.

Lectura y preprocesamiento de la entrada:

```
##Preprocesamiento de los datos
#Ciclo para preprocesar la data
arts = c()

for (item in items) {
  #Obtener solo caracteres alfanumericos
  x = gsub('[^[:alnum:] ]','' ,toString(item))
  #Obtener los numeros de los items
  x = (as.numeric(unlist(unlist(strsplit(x, "[^0-9]+")))))
  #Obtener temas/articulos
  art = paste(toString(temas[((x[2]-1)/9)+1]),
              toString(articulos[((x[2]-1)%%9)+1])
              , sep="/")
  if(length(x)>2){
    for(i in (3:length(x))){
      art = paste(art,
                  paste(toString(temas[((x[i]-1)/9)+1]),
                        toString(articulos[((x[i]-1)%%9)+1])
                        , sep="/"),
                  sep=",")
    }
  }
  #Crear vector con transacciones
  arts = c(arts, art)
}
```

Encontrar y eliminar Bots de la entrada:

El primer paso antes de analizar la data es eliminar aquellas transacciones hechas por bots. Según el cliente si una transaccion dura a lo sumo 20 segundos la transacción es considerada como un bot. El siguiente código se encarga de identificar y eliminar este tipo de transacciones.

```r
#Obtener cantidad de elementos de articulos
l = length(periodico[,1])

bots = c()
times = c()
index = c()
#Obtener el tiempo de una transaccion
#Verificar que sean mas de 20 segundos
#Obtener el indice de cuando es un bot
#Obtener el tiempo y los indices cuando no lo es
for(i in (1:l)){
  secs = difftime(periodico$exit[i],periodico$entry[i], units = "secs")
  if(secs <= 20){
    bots = c(bots, i)
  }else{
    times = c(times, secs)
    index = c(index, i)
  }
}

#Eliminar los bots de la entrada
arts=arts[-bots]

# Crear archivo con las transacciones
write(arts, file="transacciones")
```

El número de bots de esta entrada es:

```r
#Imprimir la cantidad de bots
length(bots)
```

```
## [1] 2302
```

Recomendacion de artículos:

A continuación se presenta una función capaz de recomendar un artículo nuevo a un usuario dependiendo de los artículos vistos previamente.

```r
##Usando arules para las transacciones
#Leer como transacciones
transacciones = read.transactions("transacciones", format="basket", sep=",")
unlink("transacciones")

#Obtener las reglas
rules <- apriori(transacciones, parameter = list(supp = 0.000022, conf = 0.85, maxlen=5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##        0.85    0.1    1 none FALSE            TRUE 2.2e-05      1      5
```

```
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 2
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[81 item(s), 128998 transaction(s)] done [0.02s].
## sorting and recoding items ... [81 item(s)] done [0.00s].
## creating transaction tree ... done [0.06s].
## checking subsets of size 1 2 3 4 5 done [0.04s].
## writing ... [419 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```r
#Funcion de recomendacion de un articulo
#rules son las reglas
#trans son los articulos leidos
recomendacion = function(rules, trans){
  articulo = sort(subset(rules, subset = lhs %in% trans),decreasing=TRUE,by="confidence")
  articulo = inspect(unique(articulo@rhs))
  l=c()
  for(i in (1:length(articulo[,1]))){
    l=c(l,gsub('[^[:alnum:]/]','' ,toString(articulo[i,])))
  }
  #Se verifica que los articulos de la lista no sea alguno que este en la entrada.
  l=l[!l %in% trans]

  return (l[1])
}

#Prueba de la funcion
#siempre dejar rules
#respetar formato de entrada de los articulos (solo las palabras sin los '{}')
rec = recomendacion(rules, c("nacionales/articulo6", "deportes/articulo4", "deportes/articulo1"))
```

```
##    items
## 1  {deportes/articulo1}
## 2  {deportes/articulo2}
## 3  {deportes/articulo3}
## 4  {deportes/articulo8}
## 5  {politica/articulo9}
## 6  {sucesos/articulo6}
## 7  {deportes/articulo9}
## 8  {variedades/articulo4}
## 9  {variedades/articulo3}
## 10 {deportes/articulo5}
## 11 {deportes/articulo7}
## 12 {deportes/articulo4}
## 13 {variedades/articulo1}
## 14 {deportes/articulo6}
## 15 {politica/articulo5}
```

La salida anterior es el resultado de la función *inspect* donde se ven algunos de los artículos que el sistema recomienda.

La siguiente línea muestra el artículo recomendado. Como se puede observar el artículo *deportes/articulo1*, fue eliminado de la lista de recomendaciones pues este ya fue visto por el usuario.

```
rec
```

```
## [1] "deportes/articulo2"
```

Visitas con menor tiempo de estadía:

El siguiente código utiliza los tiempos de las transacciones para ver cuales son las transacciones con menor tiempo de estadía.

```
##Top 10 de tiempos
#Se crea un dataframe con los tiempos y el indice de todas las transacciones
pair = data.frame(times, index)

#Visitas con menor tiempo de estadia
#se ordenan las transacciones por tiempo
menor = pair[order(pair$times),]
#se obtienen los 10 primeros indices
id = menor$index[1:10]
#se obtienen los 10 primeros tiempos
segundos = menor$times[1:10]

#Se obtienen las transacciones de los indices obtenidos
transaccion = arts[id[1:10]]
#Se crea un dataframe
menorTiempo = data.frame(id, transaccion, segundos)
#impresion
menorTiempo
```

```
##       id
## 1   1000
## 2   1172
## 3   2144
## 4   4698
## 5   5216
## 6   5765
## 7   6258
## 8  11346
## 9  12413
## 10 13046
##
## 1  deportes/articulo1,deportes/articulo3,deportes/articulo4,deportes/articulo5,deportes/articulo6,pol
## 2                deportes/articulo3,deportes/articulo9,politica/articulo5,internacional/articulo5,na
## 3
## 4                                                                                 deportes/articulo5,d
## 5
## 6                                                                                                    c
## 7                                                                                 deportes/articulo1,d
```

```
## 8                                                                                    deportes/articulo1
## 9                                                      deportes/articulo1,deportes/articulo9,naci
## 10
##    segundos
## 1        21
## 2        21
## 3        21
## 4        21
## 5        21
## 6        21
## 7        21
## 8        21
## 9        21
## 10       21
```

Visitas con mayor tiempo de estadía:

Y este código verifica cuales son las transacciones con el mayor tiempo.

```
#Visitas con mayor tiempo de estadia

#se ordenan las transacciones por tiempo
mayor = pair[rev(order(pair$times)),]
#se obtienen los 10 primeros indices
id = mayor$index[1:10]
#se obtienen los 10 primeros tiempos
segundos = mayor$times[1:10]

#Se obtienen las transacciones de los indices obtenidos
transaccion = arts[id[1:10]]
#Se crea un dataframe
mayorTiempo = data.frame(id, transaccion, segundos)
#impresion
mayorTiempo
```

```
##        id
## 1   93676
## 2    7511
## 3   80516
## 4  122879
## 5  130641
## 6   66995
## 7  111953
## 8   23099
## 9   55628
## 10  48007
##                                                                                           transaccion
## 1                                                            politica/articulo4,politica/articulo
## 2      deportes/articulo3,deportes/articulo5,deportes/articulo7,politica/articulo2,politica/articulo
## 3                                  deportes/articulo4,deportes/articulo9,internacional/articulo
## 4  politica/articulo4,nacionales/articulo6,comunidad/articulo2,comunidad/articulo6,negocios/articulo
## 5                                                                                              <NA
## 6                                                            deportes/articulo3,deportes/articulo
```

```
## 7                      politica/articulo3,nacionales/articulo1,nacionales/articulo7,sucesos/articulo
## 8                                          deportes/articulo6,deportes/articulo7,comunidad/articulo
## 9                                                                deportes/articulo5,sucesos/articulo
## 10                                                               deportes/articulo2,opinion/articulo
##    segundos
## 1     12264
## 2     12234
## 3     11743
## 4     11597
## 5     11508
## 6     11481
## 7     11421
## 8     11419
## 9     11381
## 10    11372
```

Top 10 Transacciones:

Este código se encarga de obtener los las 10 transacciones más comunes del set de datos. Estos son:

```
#Se usa table para hacer un conteo de los distintos tipos de transacciones
#Y se ordena
top10 = sort(table(arts), decreasing = T)[1:10]
# Se ingresa el resultado a un dataset
top10 = data.frame(top10)
#Impresion
top10
```

```
##                     top10
## deportes/articulo1   1050
## deportes/articulo4   1045
## deportes/articulo7   1043
## deportes/articulo3    924
## deportes/articulo9    904
## deportes/articulo6    894
## deportes/articulo8    868
## deportes/articulo5    850
## deportes/articulo2    828
## politica/articulo5    440
```

Parte II - Curvas ROC

En esta segunda parte Se creará un función para generar una curva ROC dada una entrada compuesta de las siguientes variables:

- scores: son los scores de cada una de las instancias.
- real: son las verdaderas clases de las instancias.
- target: es la clase que se considera positiva.

La función generate_ROC toma esta entrada y produce una gráfica donde muestra la curva.

```
generate_ROC = function(scores, real, target){
  # Generar curva
  #Se ordenan las clases segun los scores
  real = real[order(scores, decreasing = T)]
  #Se ordenan los scores
  scores = sort(scores, decreasing = T)
  #Se usa table para conocer el numero de veces que la clase target está en real
  t = table(real)
  #Se obtienen la cantidad de instancias
  l = length(real)
  #Numero de instancias Positivas
  P = t[[target]]
  #Numero de instancias Negativas
  N = l - P
  #Contador de Falsas Positivas
  FP = 0
  #Contador de Verdaderas Positivas
  TP = 0
  #Lista de coordenadas x
  Rx = c()
  #Lista de coordenadas y
  Ry = c()
  #Inicializacion de score previo a -Inf
  fPrev = -Inf
  #contadores
  i = 1
  j = 1
  #lista de labels para impresion de la curva
  labs = c()

  while(i<=length(real)){
    #Se verifica que el nuevo score no sea igual al anterior
    if(scores[i] != fPrev){
     #Se agregan los puntos x,y
     Rx[j] <- FP/N
     Ry[j] <- TP/P
     #Se añade el label
     labs[j] = fPrev
     #Se actualiza fPrev
     fPrev = scores[i]
     j = j + 1
    }
    if(real[i]==target){
      #Si la clase actual es igual a la clase target se suma 1 a TP
      TP = TP + 1
    }else{
      #De lo contrario se suma 1 a FP
      FP = FP + 1
    }
    i = i + 1
  }

  #punto (1,1)
```

```
  Rx[j] <- FP/N
  Ry[j] <- TP/P
  labs[j] = 1.0

  #Impresion de la curva
  plot(Rx, Ry, main = "ROC Curve", xlim = c(-0.005, 1.005), ylim = c(-0.05, 1.05), xlab = "FP-RATE", yla
  lines(Rx, Ry, lty=5)
  lines(c(-2,2),c(-2,2), lty=5, col=16)
  text(Rx+0.025, Ry+0.025, labels = labs)
}

#Entrada de ejemplo
real = c(2, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1)
scores = c(0.9, 0.8, 0.7, 0.6, 0.55, 0.54, 0.53, 0.52, 0.5, 0.5, 0.5, 0.5, 0.38, 0.37, 0.36, 0.35, 0.34
target = 2
#Llamada a la funcion
generate_ROC(scores, real, target)
```