Sistema de recomendación

Ybrahin Martinez

Mayo 2016

Sistema de Recomendación

Se requiere hacer un estudio de los sistemas de recomendación, para esto se nos da la información de las transacciones de un periódico virtual, también nos dan un conjunto de requerimientos pedidos por este ente:

1. Proveer información de las transacciones de las cuales el periódico considera Bot.
2. Modificar el dataset de tal manera que no se lean los identificadores de los artículos como **itemN** sino por su tipo de contenido **contenido/articuloN**
3. Conocer los tipos de usuarios que ingresan a su página.
4. Recomendar un artículo a una nueva persona que ingrese a su página.
5. Conocer las 10 visitas con mayor tiempo de estadía en la página y las 10 visitas con menor tiempo de estadía en la página.
6. Conocer las 10 transacciones con mayor número de apariciones en el dataset.

Instalando los paquetes necesarios

```
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, %in%, write
```

Cargando el dataset del periódico

```
setwd("..")
dataset = read.csv("dat/periodico.csv",
                   colClasses = "character",
                   stringsAsFactors = F)

subjects = c("deportes", "politica", "variedades",
             "internacional", "nacionales", "sucesos",
             "comunidad", "negocios", "opinion")
```

Preprocesamiento

Debemos realizar una preparación de los datos previa a la implementación, la razón es por los requerimientos exigidos por el periódico.

- Transformando las columnas tiempo de entrada y salida de la transacción para luego proceder a calcular la duración de la transacción, se realiza con la resta de ambos valores y lo guardamos en una nueva columna en segundos.

```
dataset$entry = as.POSIXct(dataset$entry, "%Y-%m-%d %H:%M:%S")
dataset$exit = as.POSIXct(dataset$exit, "%Y-%m-%d %H:%M:%S")
dataset$duration = difftime(dataset$exit, dataset$entry, unit = "secs")
```

- Cambiando el nombre de la columna X a TID, solamente para fines informativos.

```
dataset$ID = NULL
colnames(dataset)[1] = "ID"
```

- Elminando caracteristicas que ya no se utilizarán más durante el código.

```
dataset$entry = NULL
dataset$exit = NULL
```

Requrimientos

El primer requerimiento exigido por el periodico es el de transformar el dataset y que las transacciones sigan este formato **/articulo** se asume que para propositos de comprender rapidamente las transacciones.

1. Transformando el dataset

Usando una expresión para captar el patrón de los items en el dataset, cambiamos iterando por cada una de las transacciones:

```
itemsID = seq(1, 81)

items = sprintf("%s/articulo%d",
                subjects[(itemsID-1)%/%9+1], (itemsID-1)%%9+1)

for(it in itemsID)
  dataset$articles = gsub(sprintf("item%d(?=[,}])", it),
                          items[it],
                          dataset$articles,
                          perl = T)

dataset$articles = gsub("\\{|\\}", "", dataset$articles)
```

El segundo requerimiento exigido es el de dar información acerca de las transacciones Bot que se encuentren en el dataset

2. Transacciones Bot

Como se realizó el preprocesamiento anterior ya se tiene una columna llamada duration con la duración de cada una de las transacciones en segundos, con estos datos procedemos a buscar las transacciones Bot. El dice que una transacción no es realizada por un Bot, cuando la persona vé por mas de 20 segundos un artículo, como no podemos asegurar que durante el tiempo de la transacción la persona vio por más de 20

segundos cada artículo, se considera es el caso promedio. Por esta razón, para verificar que una transacción no es un bot esta debe durar al menos 20 segundos por la cantidad de articulos que posee, es decir,

$$x > articles \times 20$$

Se crea una lista de transacciones como vectores y se verifica la desigualdad usada para filtrar las reglas que no son Bots.

```
trans = strsplit(dataset$articles, ",")
bots = dataset$duration <= (lengths(trans)*20)
```

Al tener las posiciones (en formato de arreglo lógico) cuales son las que no cumplen dicha condición procedemos a contar cuantas transaciones son bot e informarlo al periodico.

```
nrow(dataset[bots,])
```

```
## [1] 6599
```

Después de haber informado sobre cuantas transacciones Bot hay, el periodico no indicó que se tenía que hacer con estas transacciones, pero se asumió que había que eliminarlas para que estas no comprometan la calidad del sistema de recomendacion.

```
dataset = dataset[!bots,]
trans = trans[!bots]
```

Antes de seguir con los siguientes requerimientos se necesita crear la matriz de transacciones, en este caso será de tamaño

$$81 \times 81$$

y que cada fila representa una transacción pero ahora en formato de matriz, también en ella solo existen valores 1 u 0 para indicar la presencia o no de un item en la transacción, dicho esto esta estructura la usaremos muy seguido para diferentes propósitos.

```
names(trans) = dataset$ID
trans = as(trans, "transactions")
```

2. Recomendar un artículo a un nuevo usuario

Para este nuevo requerimiento es necesario antes es necesario crear las reglas de asociación, se utilizará el algorimo **apriori**, usando la matriz de transacciones creada, dicho algoritmo lo usamos con dos variables importantes **Support*** y **Confidence**, estas variables son necesarias para la creación correcta de reglas ya que proveen unos limites mínimos que deben cumplir las reglas para que puedan ser consideradas. Para nuestra implementación se usa los valores de 0.00004 para **soporte** y 0.6 para **confianza**, el motivo de estos es porque son valores con los cuales se generan una cantidad considerable de reglas.

```
# Generando reglas
rules = apriori(trans,
                parameter = list(sup = 0.00004,
                                 conf = 0.6,
                                 target="rules"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.6    0.1    1 none FALSE            TRUE   4e-05      1     10
##  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: 4
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[81 item(s), 124701 transaction(s)] done [0.07s].
## sorting and recoding items ... [81 item(s)] done [0.01s].
## creating transaction tree ... done [0.35s].
## checking subsets of size 1 2 3 4 5 6 7 done [0.10s].
## writing ... [212 rule(s)] done [0.01s].
## creating S4 object  ... done [0.10s].
```

```
rules = sort(rules, decreasing = T, by = "lift")

summary(rules)
```

```
## set of 212 rules
##
## rule length distribution (lhs + rhs):sizes
##   4   5   6   7
## 102  99  10   1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.000   4.000   5.000   4.575   5.000   7.000
##
## summary of quality measures:
##     support            confidence          lift
##  Min.   :4.010e-05   Min.   :0.6000   Min.   : 3.500
##  1st Qu.:4.010e-05   1st Qu.:0.6250   1st Qu.: 3.772
##  Median :4.010e-05   Median :0.6364   Median : 4.166
##  Mean   :4.513e-05   Mean   :0.6855   Mean   : 5.490
##  3rd Qu.:4.812e-05   3rd Qu.:0.7143   3rd Qu.: 4.672
##  Max.   :8.821e-05   Max.   :1.0000   Max.   :63.818
##
## mining info:
##   data ntransactions support confidence
##  trans        124701   4e-05        0.6
```

Función que recomienda un artículo a un nuevo usuario de acuerdo a los artículos vistos (conjunto de items).

```
predictArticle = function(itemset, rules){
    subsetRules = subset(rules, lhs %ain% itemset & !rhs %in% itemset)
```

```
    if(length(subsetRules) == 0)
        subsetRules = subset(rules, lhs %in% itemset & !rhs %in% itemset)

    return(inspect(subsetRules@rhs[1]))
}

seen = c("opinion/articulo1",
        "nacionales/articulo2",
        "deportes/articulo3",
        "variedades/articulo4",
        "negocios/articulo5")

predictArticle(seen, rules)
```

```
##   items
## 1 {variedades/articulo1}
```

4. Tipos de usuarios que ingresan al portal.

En este requerimiento se pensó en una primera instancia en tratar de agrupar a los usuarios del portal de acuerdo a las reglas creadas, pero luego se implementó usando las transacciones ya que estas proveen mayor información relevante. Sólo se considerará las transacciones únicas porque sería tener información redundante y no es algo que se quiera para analizar.

```
transUnique = unique(trans)
```

Buscamos los clusters usando **clustering jerárquico** sobre una muestra pequeña de las transacciones únicas usando la medida de similaridad **Jaccard**,esta mide la similaridad entre conjuntos, para esto hacemos uso de una función implementada en el paquete **arules**.

```
small = sample(transUnique, 10000)
jaccard = dissimilarity(small)
jaccard[is.na(jaccard)] <- 1
```

Se optó por clustering jerárquico ya que es ideal para clusters de formas rectangulares, como el periódico cree que son 8 tipos de usuarios se hizo pruebas dada esta premisa.

```
hc = hclust(jaccard, method = "ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
cut = cutree(hc, k = 8)
```

```
big = transUnique[!(transUnique %in% small)]
labels = predict(small, big, cut)
```

Ahora podemos ver las proporciones de los clusters.

```r
table(cut)
```

```
## cut
##    1    2    3    4    5    6    7    8
## 1062 5418  721  725  275  321  991  487
```

```r
table(labels)
```

```
## labels
##     1     2     3     4     5     6     7     8
##  4184 21074  3131  3442  1208  1402  4650  2159
```