Proyecto - Minería de Datos

José Manuel Alvarez García y Alvaro Sanabria

Abril 18, 2017

Inicialmente se debe estar posicionado en el directorio del proyecto:

```
setwd("Documents/Proyecto-Mineria-de-Datos/")
```

Análisis exploratorio.

Se trabajará con un conjunto de archivos .csv del dataset ml-latest-small, el cual describe la calificación de 5 estrellas y la actividad de marcado de texto libre de MovieLens. Los archivos que se trabajarán son ratings.csv, movies.csv y links.csv.

El archivo ratings.csv tiene el siguiente conjunto de atributos (columnas):

1. userId: Representa el identificador de un usuario de MovieLens.
2. movieId: Representa el identificador de una película usado por https://movielens.org.
3. rating: Representa la calificación de una película. Esta variable toma valores sobre una escala de 5 estrellas,con incrementos de media estrella (0,5 estrellas - 5,0 estrellas).
4. timestamp: Representa los segundos desde la medianoche Tiempo Universal Coordinado (UTC) del 1 de enero de 1970.

El archivo movies.csv tiene el siguiente conjunto de atributos (columnas):

1. movieId: Representa el identificador de una película usado por https://movielens.org.
2. title: Representa el título y el año de lanzamiento de una película.
3. genres: Representa el/los géneros asociados a una película. Los géneros son una lista separada por pipes y pueden ser: Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western, (no genres listed).

El archivo links.csv tiene el siguiente conjunto de atributos (columnas):

1. movieId: Representa el identificador de una película usado por https://movielens.org.
2. imdbId: Representa el identificador de una película usado por http://www.imdb.com.
3. tmdbId: Representa el identificador de una película usado por https://www.themoviedb.org.

Posteriormente, estas columnas serán procesadas en la tarea de pre-procesamiento.

Pre-Procesamiento.

Una vez que el usuario haya presionado el botón de pre-procesamiento, se realizará el pre-procesamiento de los archivos movies.csv, links.csv y ratings.csv.

Pre-procesamiento de movies.csv

Inicialmente se obtiene el archivo:

```
inFile <- input$fileMovies
if (is.null(inFile)) return(NULL)
```

Se valida si el archivo contiene un header.

```
if (input$includeheadermovies) {

          h <- TRUE

      } else {

          h<-FALSE

      }
```

Se valida el caracter separador del archivo.

```
if (input$test=="Coma") {

          separador <- ","

      } else if(input$test=="Puntoycoma") {

          separador <- ";"

      } else if(input$test=="Tabular") {

          separador <- "\t"

      }
```

Se valida como se expresan las cadenas de caracteres del archivo.

```
if(input$radioquotmovies=="doublemovies"){

          comilla <- "\""

      } else if(input$radioquotmovies=="simplemovies"){

          comilla <- "\'"

      } else if(input$radioquotmovies=="nonemovies"){
```

```
          comilla <- ""

      }
```

Se lee el dataset de peliculas segun las validaciones anteriores.

```
movies_aux <- read.csv(inFile$datapath, header=h, sep=separador,quote = comilla)
```

Se identifican las columnas del dataset leido.

```
 colnames(movies_aux) <- c("ID_pelicula", "titulo_pelicula", "generos" )
```

El titulo de cada pelicula va a ser tratado como un tipo de dato String.

```
movies_aux$titulo_pelicula <- as.character(movies_aux$titulo_pelicula)
```

Eliminando espacio al final del titulo de cada pelicula.

```
movies_aux$titulo_pelicula <- gsub(") ", ")", movies_aux$titulo_pelicula)
```

Se asigna a un nuevo vector “annos” los ultimos 6 caracteres de todas las celdas del titulo de cada pelicula.

```
 annos <- ultimosCaracteres(as.character(movies_aux$titulo_pelicula), 6)
```

Se elimina el ultimo caracter a cada año para eliminar el parentesis.

```
annos <- substring(annos, 2)
```

Se elimina el primer caracter a cada año para eliminar el parentesis.

```
annos <- substring(annos, 1, nchar(annos)-1)
```

Se agrega una nueva columna al dataset correspondiente al año de cada pelicula.

```
movies_aux[ , "anno_pelicula"] <- annos
```

Eliminando parentesis en el titulo de cada pelicula.

```
 movies_aux$titulo_pelicula <- gsub("\\(", "", movies_aux$titulo_pelicula)
 movies_aux$titulo_pelicula <- gsub(")", "", movies_aux$titulo_pelicula)
```

Colocando annos como titulo - anno.

```
movies_aux$titulo_pelicula <- sapply(movies_aux$titulo_pelicula, function(string) {

        yearMovie <- NULL
        yearMovie <- ultimosCaracteres(string,4)
        string <- substr(string, 1, nchar(string)-4)
        string <- paste(string,"- ",yearMovie, sep="")

      }  )
```

Eliminando signos en el titulo de cada pelicula.

```
movies_aux$titulo_pelicula <- gsub("\\?", "", movies_aux$titulo_pelicula)
movies_aux$titulo_pelicula <- gsub("\\!", "", movies_aux$titulo_pelicula)
```

Eliminar variable generos.

```
movies_aux$generos <- NULL
```

Pre-procesamiento de links.csv

Inicialmente se obtiene el archivo:

```
inFile <- input$fileLinks
if (is.null(inFile)) return(NULL)
```

Se valida si el archivo contiene un header.

```
if (input$includeheaderlinks) {

          h <- TRUE

     } else {

          h <- FALSE

     }
```

Se valida el caracter separador del archivo.

```
if (input$radioseparatorlinks=="commalinks") {

          separador <- ","

     } else if(input$radioseparatorlinks=="semicolonlinks") {

          separador <- ";"

     } else if(input$radioseparatorlinks=="tablinks") {

          separador <- "\t"

     }
```

Se valida como se expresan las cadenas de caracteres del archivo.

```
if (input$radioquotlinks=="doublelinks") {

          comilla <- "\""

     } else if (input$radioquotlinks=="simplelinks") {
```

```
          comilla <- "\'"

      } else if (input$radioquotlinks=="nonelinks") {

          comilla <- ""

      }
```

Se lee el dataset de links segun las validaciones anteriores.

```
links <- read.csv(inFile$datapath, header=h, sep=separador,quote = comilla)
```

Se elimina el id de imdb.

```
links$imdbId <- NULL
```

Se identifican las columnas del dataset leido.

```
colnames(links) <- c("ID_pelicula", "tmdbId")
```

Se corresponden los links y movies segun el id de cada pelicula.

```
movies_aux <- merge(links,movies_aux,by="ID_pelicula")
```

Pre-procesamiento de ratings.csv

Inicialmente se obtiene el archivo:

```
inFile <- input$fileRatings
if (is.null(inFile)) return(NULL)
```

Se valida si el archivo contiene un header.

```
if (input$includeheaderratings) {

          h <- TRUE

      } else {

          h <- FALSE

      }
```

Se valida el caracter separador del archivo.

```
if (input$radioseparatorratings=="commaratings") {

          separador <- ","
```

```
      } else if (input$radioseparatorratings=="semicolonratings") {

          separador <- ";"

      } else if (input$radioseparatorratings=="tabratings") {

          separador <- "\t"

      }
```

Se valida como se expresan las cadenas de caracteres del archivo.

```
if (input$radioquotratings=="doubleratings") {

          comilla <- "\""

      } else if (input$radioquotratings=="simpleratings") {

          comilla <- "\'"

      } else if (input$radioquotratings=="noneratings") {

          comilla <- ""

      }
```

Se lee el dataset de ratings segun las validaciones anteriores.

```
ratings_aux <- read.csv(inFile$datapath, header=h, sep=separador,quote = comilla)
```

Se elimina la columna timestamp.

```
ratings_aux$timestamp <- NULL
```

Se identifican las columnas del dataset leido.

```
colnames(ratings_aux) <- c("userId", "ID_pelicula", "puntuacion")
```

Reglas de Asociación

Una vez que se haya realizado el pre-procesamiento se generarán las reglas de asociación. Para ello en primera instancia se corresponden los ratings y movies segun el id de cada pelicula:

```
merged <- merge(global$ratings,global$movies,by="ID_pelicula")
```

Se convierte el dataframe anterior a un conjunto de transacciones.

```
 merged2 <- as( split(as.vector(merged$titulo_pelicula),
 as.vector(merged$userId)),"transactions")
```

Se llama a la funcion apriori con un soporte de 0.005 y confianza de 0.5.

```
rules <- apriori( merged2 , parameter=list(supp=0.005, conf=0.5,
target="rules", minlen=2, maxlen=2, maxtime=60))
```

Se ordenan las reglas por su lift de mayor a menor.

```
rules <- sort(rules, by ="lift")
global$rulesByName <- rules
```

Luego se van a desplegar en la interfaz de usuario las 6 primeras reglas que se generaron.

```
output$inspectRules <- renderPrint({

      # Se coloca un ancho bastante alto
      options(width = 1500);

      # Se imprimen las 6 primeras reglas
      inspect(head(global$rulesByName));

    })
```

Recomendación de películas

Ya que se hizo el pre-procesamiento y se generaron las reglas de asociación ya se puede llevar a cabo la recomendación de las películas. Para ello el usuario debe ingresar el título de una película de su preferencia y luego pulsar sobre el botón de sugerencias. Una vez pulsado el botón, la aplicación internamente ejecuta lo siguiente:

Primero se encuentran todas las subreglas que contengan exactamente el titulo de la película que el usuario seleccionó en el menu.

```
subreglas <- subset(global$rulesByName, subset = lhs %pin% paste("^",
as.character(input$nameMovie), "$", sep=""))
```

Se envia un mensaje al javascript indicando el numero de subreglas encontradas.

```
session$sendCustomMessage(type = 'dataSubreglas', message = as.integer(length(subreglas)))
```

En el caso en que se hayan encontrado subreglas, se va a desplegar en la interfaz de usuario las 6 primeras subreglas.

```
 output$inspectSubRules <- renderPrint({

        # Se coloca un ancho bastante alto
        options(width = 1500);

        # Se imprimen las 6 primeras subreglas
        inspect(head(subreglas));

      })
```

También en el caso que se hayan encontrado subreglas, se va a mostrar en la interfaz de usuario hasta 8 películas como recomendación para el usuario, pudiendo ser menos dependiendo de la cantidad de subreglas generadas.

El proceso de despliegue de una película recomendada consiste en:

1. Mostrar el titulo de la película en pantalla, que es algo como esto:

```
output$summary8 <- renderText({

          consequent8 <- substring(consequent8, 2)
          consequent8 <- substr(consequent8, 1, nchar(consequent8)-1)
          paste0("<h4>",consequent8,"</h4>")

        })
```

2. Mostrar la imagen de la película en pantalla, que es algo como esto:

```
tryCatch({

          # Se busca la pelicula por id en el API de themoviedb
          peliTemp8 <- fromJSON(paste("https://api.themoviedb.org/3/movie/",
```

```
          themoviedbId8, "?api_key=3c9a39604750b33b3e006c0d54a11e55", sep=""))
          src8 <- paste0("https://image.tmdb.org/t/p/w500",peliTemp8$poster_path)

          # Se coloca la imagen obtenida en pantalla
          output$poster8<-renderText({paste0('<img src="',src8,'" width="100%">')})

        },

          # Si ocurre un error o warning al tratar de buscar la imagen, no se muestra
          error = output$poster8<-renderText({paste0('<img src="" width="">')}),
          warning = output$poster8<-renderText({paste0('<img src="" width="">')})

        )
```

En el caso en que no se encontraron subreglas que satisfagan el título de la película ingresado por el usuario, se muestra en la aplicación un mensaje indicando que no han encontrado películas para recomendar.

```
output$summary <- renderText({

        "<h3 style='padding-left:25px;'>No se han encontrado pelÃ?culas para ti.</h3>"

      })
```