Basic Recommender System

Cheryl Bowersox

Tuesday, February 14, 2017

** Description of system: This system provides recommendations of top 10 tasty beers for users based collaborative filtering.

Input: Matrix of users and rankings give to each beer across several attributes.

Data Source:

https://data.world/socialmediadata/beeradvocate (https://data.world/socialmediadata/beeradvocate)

Unique Users are defined by variable review_profilename Unique beer (items) are defined by variable beer_beerid For this model I am only using the revew_overall rating to establish the recommendation, and only pulling the first 10K records.

The function ‘manualrec’ was created not using any built in recommendation packages. This funciton takes the data table and a user name who the recommendation is for as input, and then creates a list of 10 recommended beers by finding the cosine similarity, pulling those identified users highest rated beers, and weighting the overall results by a factor of how many similar profiles listed that beer. In the case of a tie, the data is also sorted by most recent reviews. This manual function is inelegant, uses loops to create the final list, and is very resource-intensive to run.

```
# import libraries
library(recommenderlab)
library(dplyr)
library(tidyr)
library(reshape)

#read data
beerdata <- read.csv("~/GitHub/IS643/beerdata.csv")

#shape data to create matrix, profile X beer

beerrates <- beerdata %>%select(beer_beerid, review_profilename, review_overal
l)
df <- head(beerrates, 10000)%>%group_by(beer_beerid,review_profilename)%>%summa
rise(avgrev= mean(review_overall))

dfwide <- spread(df,beer_beerid, avgrev)

dfwide[is.na(dfwide)] <-0

# function to compute simliarity between vectors in this case comparing profil
e for user A with user
#input X matrix the wide matrix, a username, output simliarity with other users

manualrec <- function(df, user){

  #calculate the affinity for this user's vector with all other users
  #get users vector
  #user <- "abrand"
  u <- as.numeric(dfwide%>%filter(review_profilename == user)%>%select(-review_
profilename))
  dfv <- dfwide%>%filter(review_profilename != user)
  dfrank <- data.frame(name = dfv$review_profilename, sim = -10)

  #awful loop use apply(?)
  for (i in 1:nrow(dfrank)){
    v <- as.numeric((dfv%>%select(-review_profilename))[i,])
    dfrank$sim[i] <- crossprod(u, v)/(sqrt(crossprod(u)) * sqrt(crossprod(v)))
  }
  # arrange users, top 5 users that match
  topten <- head(dfrank%>%arrange(desc(sim)),10)

  results <- beerdata%>%
    select(beer_name,review_profilename,review_overall,review_time)%>%
    filter(review_profilename %in% topten[,1])

  results <- merge(x = results, y = topten, by.x = "review_profilename",
                   by.y = "name", all.x = TRUE)
```

```
  #weight by count of recommendations
  results$beerrated <- results$review_overall * results$sim
  results <- results%>%select(beer_name,beerrated, review_time)%>%
    group_by(beer_name)%>%
    summarise(rank = mean(beerrated), count=n(), avgtime = mean(review_time))
  results$wrank <- results$rank * results$count

  #sort by max beerrated score, then by max review time
  recommend <- results%>%arrange(desc(wrank),desc(avgtime))%>%select(beer_name,
wrank)

  return (head(recommend,10))
  }
```

For the first example I used user named “abrand” and found the related list

```
(manualrec(dfwide,"abrand"))
```

```
## Source: local data frame [10 x 2]
##
##                                        beer_name     wrank
## 1                                  Ashland Amber 12.374369
## 2  D.O.R.I.S. The Destroyer Double Imperial Stout 12.374369
## 3                     Oaked Arrogant Bastard Ale 12.374369
## 4                            Stone Ruination IPA 12.020815
## 5                   Sierra Nevada Celebration Ale 11.667262
## 6                        Founders Breakfast Stout 10.253048
## 7                   Weihenstephaner Hefeweissbier  9.545942
## 8                      Stone IPA (India Pale Ale)  9.545942
## 9                         Trappistes Rochefort 10  9.545942
## 10                  Ayinger Celebrator Doppelbock  9.192388
```

A second recommendation example I used the user named “beerguy101”

```
(manualrec(dfwide,"beerguy101"))
```

```
## Source: local data frame [10 x 2]
##
##                                     beer_name    wrank
## 1                            Dark Star Porter 24.07954
## 2                           American Pale Ale 24.04728
## 3                                    Pale Ale 20.11213
## 4                      Sierra Nevada Pale Ale 15.49497
## 5                                 Dead Guy Ale 14.10680
## 6         Casco Bay Winter Ale (Old Port Ale) 13.83207
## 7                    Franziskaner Hefe-Weisse 13.58535
## 8                      Bell's Kalamazoo Stout 12.97371
## 9                          Fat Tire Amber Ale 12.82896
## 10 Sierra Nevada Bigfoot Barleywine Style Ale 12.55689
```

Using the built-in functions in the recommendlab package I get different, but much faster results:

```
#convert to matrix
beerrates <- beerdata %>%select(user = review_profilename, beer = beer_name, ra
te = review_overall)
df <- head(beerrates, 10000)

ratemat <- as(df,"realRatingMatrix")

train <- ratemat[1:150]  # trainng data for model
rec =Recommender(train,method="UBCF",
      param=list(normalize = "Z-score",method="Cosine",nn=5, minRating=1))
```

Predicting top 10 for “abrand” user again, this time using the UBCF method of the Recommender function in the package yields a different, and quicker, result

```
pre_for_abrand <- predict(rec, ratemat["abrand",], n=10)

(as(pre_for_abrand, "list"))
```

```
## [[1]]
##  [1] "'Pooya Porter"
##  [2] "'Sconnie Pale Ale"
##  [3] "'Sconnie Rustic Trail Amber"
##  [4] "'Sconnie Tall Blonde Ale"
##  [5] "'Tis The Seasonator"
##  [6] "1 A.M. Ale"
##  [7] "10 Blocks South"
##  [8] "10 Year Clelebration Ale"
##  [9] "12 Year Anniversary Ale"
## [10] "13th Anniversary Imperial India Pale Ale"
```

Similarly, the prediction for the user “beerguy101” has different outcomes.

```
pre_for_beerguy101 <- predict(rec, ratemat["beerguy101",], n=10)
(as(pre_for_beerguy101, "list"))
```

```
## [[1]]
##  [1] "B.O.R.I.S. The Crusher Oatmeal-Imperial Stout"
##  [2] "Kirkland Signature Amber Ale"
##  [3] "Akron Dopple Alt"
##  [4] "Barrel Aged B.O.R.I.S. Oatmeal Imperial Stout"
##  [5] "Nut Brown Ale"
##  [6] "Buckeye Brown"
##  [7] "Steel Valley Stout"
##  [8] "Twice Bitten"
##  [9] "Verich Gold"
## [10] "Ashland Amber"
```