**Movie Recommender Code**

This document contains the details of **main methods used in the system built, followed by the complete code with comments and documentation.**

**Libraries Used :**

* **Data.table**
* **Proxy**
* **Vegan**
* **Matrix**
* **Reshape2**
* **Recommender Lab**
* **Shiny**

**Movie Recommender Engine**

**Two approaches Used,**

1. **Content based**

**Short Abstract :**

Till now, we have built a Movie recommender Engine with R based on MovieLens Data Set. We built the recommender system using two approaches-Content Based Filtering and Collaborative Filtering. The Content based Filtering approach analyses an item user interacted with and provides recommendations of other item with similar features. In our case,the item is movie and features would be genres,actors,directors etc. We have build a content based recommender system based on genres. UBCF based method is explained in next part.

**In Detail :**

The code does the following,

* Data preprocessing, pipe separated genres had to split
* Created matrix with columns representing every unique genre
* Convert into Data Frame for processing user profile matrix in the next step
* ( Temporary Conversion for faster processing , I first convert the ratings into a binary format to keep things simple. ratings of 4 and 5 are mapped to 1, representing likes, and ratings of 3 and below are mapped to -1, representing dislikes. )
* Dcast function is used to build user profile matrix ( change format to wide from long )
* To generate simple user profile matrix, calculated the dot product of the movie genre matrix and the binaryratings matrix. **This user profiles shows the aggregated inclination of each user towards movie genres.**
* Generate two separate matrix, where in the rows,i.e, movies which has no ratings are removed and another vice- versa, not necessary just for easier processing. (Also since there was observed differences in numbers between both, coz of null sets)
* In order to generate user profile similarity, we are used **Jaccard distance** method of finding from vegan library
* **Recommend movies - Simple content based method**

**2. User Based Collaborative Filtering**

**Short Abstract :**

Now that we have user profiles, we are assuming that users like similar items, and retrieve movies that are closest in similarity to a user’s profile, which represents a user’s preference for an item’s feature. Collaborative Filtering System works like how word of mouth works in the real world. For example, if Alice liked movies A,B and C and Bob liked movies A and B, then Bob is more likely to lik C too. It groups users with similar likings and provides recommendations based on that.To implement Collaborative Filtering,we are using the recommenderlab package available in R.

**In Detail :**

* We first create the rating matrix
* Notes :
  + Cosine Similarity Method in UBCF is used. Nearest Neighbours considered 30.
  + The predicted movie ratings of the user will be derived from the 5 nearest neighbors in its neighborhood.
  + When the predicted item ratings are obtained, the top 10 most highly predicted ratings will be returned as the recommendations.
* First converted rating matrix into a recommenderlab sparse matrix.
* Normalize the data and obtain recommendations

**Updates for Top 3 based Updated Recomendation**

**Method Used:**

For this we get the movie ids from the user, those movies which he feels like he wants to see, and we make the change in the ratings of his, to make all the other movie's ratings given by him normalised and standardised. At the same time, we make these 3 with rating 5, giving it the highest priority, followed by which on running UBCF, we get the appropriate results.

**Updates for Evaluation Scheme**

**Recommenderlab package provides a way for us to evaluate our model.**

#method cross validation

#k=5 meaning a 5-fold cross validation.

#given=3 meaning a Given-3 protocol

**#evaluate model**

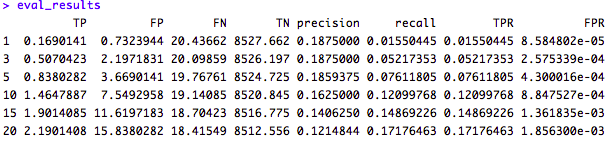
evaluation\_scheme <- evaluationScheme(ratingmat, method="cross-validation", k=5, given=3, goodRating=5)

evaluation\_results <- evaluate(evaluation\_scheme, method="UBCF", n=c(1,3,5,10,15,20))

eval\_results <- getConfusionMatrix(evaluation\_results)[[1]]

print(eval\_results)

**for which we get the output,**

****

**Code Documentation**

**#load all library files**

library(data.table)

library(Matrix)

library(proxy)

**#load the data files as movies and ratings**

movies <- read.csv("~/Works/Rworks/Movielens/Recom/movies.csv", stringsAsFactors=FALSE)

str(movies)

ratings <- read.csv("~/Works/Rworks/Movielens/Recom/ratings.csv", stringsAsFactors=FALSE)

str(ratings)

**# simple content-based recommender engine**

**# data preprocessing - pipe seperated genres - had to be split**

genres <- as.data.frame(movies$genres, stringsAsFactors=FALSE)

**#tstrsplit**

genres2 <- as.data.frame(tstrsplit(genres[,1], '[|]', type.convert=TRUE), stringsAsFactors=FALSE)

colnames(genres2) <- c(1:6)

head(genres2)

**# create a matrix with columns representing every unique genre**

genre\_list <- c("Action", "Adventure", "Animation", "Children", "Comedy", "Crime","Documentary", "Drama", "Fantasy","Film-Noir", "Horror", "Musical", "Mystery","Romance","Sci-Fi", "Thriller", "War", "Western")

genre\_matrix <- matrix(0,8571,18) #empty matrix

genre\_matrix[1,] <- genre\_list #set first row to genre list

colnames(genre\_matrix) <- genre\_list #set column names to genre list

**#iterate through matrix**

for (i in 1:nrow(genres2)) {

for (c in 1:ncol(genres2)) {

genmat\_col = which(genre\_matrix[1,] == genres2[i,c])

genre\_matrix[i+1,genmat\_col] <- 1

}

}

**#convert into dataframe**

genre\_matrix2 <- as.data.frame(genre\_matrix[-1,], stringsAsFactors=FALSE) #remove first row, which was the genre list

for (c in 1:ncol(genre\_matrix2)) {

genre\_matrix2[,c] <- as.integer(genre\_matrix2[,c])

} #convert from characters to integers

**# user profile matrix. This can be easily done with the dcast() function in the reshape2 package**

binaryratings <- ratings

for (i in 1:nrow(binaryratings)){

if (binaryratings[i,3] > 3){

binaryratings[i,3] <- 1

}

else{

binaryratings[i,3] <- -1

}

}

**#long format to a wide format using dcast function**

binaryratings2 <- dcast(binaryratings, movieId~userId, value.var = "rating", na.rm=FALSE)

for (i in 1:ncol(binaryratings2)){

binaryratings2[which(is.na(binaryratings2[,i]) == TRUE),i] <- 0

}

binaryratings2 = binaryratings2[,-1]

**#remove movieIds col. Rows are movieIds, cols are userIds**

**#Remove rows that are not rated from movies dataset - dakalti movies :|**

movieIds <- length(unique(movies$movieId))

print(movieIds)

ratingmovieIds <- length(unique(ratings$movieId))

print(ratingmovieIds)

movies2 <- movies[-which((movieIds %in% ratingmovieIds) == FALSE),]

rownames(movies2) <- NULL

**#Remove rows that are not rated from genre\_matrix2**

genre\_matrix3 <- genre\_matrix2[-which((movieIds %in% ratingmovieIds) == FALSE),]

rownames(genre\_matrix3) <- NULL

**#Calculate dot product for User Profiles**

result = matrix(0,18,706)

for (c in 1:ncol(binaryratings2)){

for (i in 1:ncol(genre\_matrix3)){

result[i,c] <- sum((genre\_matrix3[,i]) \* (binaryratings2[,c]))

}

}

**#Convert to Binary scale**

for (i in 1:nrow(result)){

if (result[i] < 0){

result[i] <- 0

}

else {

result[i] <- 1

}

}

**#user profile generated from the above**

**# Jaccard Distance to measure the similarity between user profiles, and the movie genre matrix**

print("enter the user to be profiled for")

tofinduser = 2

result2 <- result[tofinduser,] #n th user's profile

sim\_mat <- rbind.data.frame(result2, genre\_matrix2)

sim\_mat <- data.frame(lapply(sim\_mat,function(x){as.integer(x)}))

**#convert data to type integer**

**#Calculate Jaccard distance between user profile and all movies**

library(vegan)

sim\_results <- vegdist(sim\_mat, method="jaccard")

sim\_results <- as.data.frame(as.matrix(sim\_results[1:8552]))

**# how to arrive at number --- to find ? !!! @senthiran**

rows <- which(sim\_results == min(sim\_results))

**#Recommended movies**

movies[rows,2]

**############################**

**# user based collaborative filtering**

library(reshape2)

**#change these inputs manually to get appropriate outputs**

checkvar <- TRUE

useridno <- 1;

towatch1 <- 4;

towatch2 <- 50;

towatch3 <- 100;

**if(checkvar == TRUE){**

**#assign already existing ratings to the medium rating of 3**

ratings$rating <- ifelse(ratings$userId == useridno, 3, ratings$rating)

da <- subset(ratings, subset = userId == useridno)

for(towatch in c(towatch1,towatch2,towatch3)){

if( towatch %in% da$movieId){

print("to watch"+towatch)

ratings$rating <- ifelse(ratings$userId == useridno & ratings$movieId == towatch, 5, ratings$rating)

}

else{

vec <- c(useridno,towatch,5)

ratings <- rbind(ratings,vec)

}

}

}

**#Create ratings matrix. Rows = userId, Columns = movieId**

ratingmat <- dcast(ratings, userId~movieId, value.var = "rating", na.rm=FALSE)

ratingmat <- as.matrix(ratingmat[,-1]) #remove userIds

**#Creation of the Recommender Model**

**#Method: UBCF**

**#Similarity Calculation Method: Cosine Similarity**

**#Nearest Neighbors: 30**

**library(recommenderlab)**

**#Convert rating matrix into a recommenderlab sparse matrix**

ratingmat <- as(ratingmat, "realRatingMatrix")

**#Normalize the data**

**ratingmat\_norm <- normalize(ratingmat)**

**#Create Recommender Model. "UBCF" stands for User-Based Collaborative Filtering**

recommender\_model <- Recommender(ratingmat\_norm, method = "UBCF", param=list(method="Cosine",nn=30))

recom <- predict(recommender\_model, ratingmat[useridno], n=10) #Obtain top 10 recommendations for 1st user in dataset

recom\_list <- as(recom, "list") #convert recommenderlab object to readable list

#Obtain recommendations

recom\_result <- matrix(0,10)

for (i in c(1:10)){

recom\_result[i] <- movies[as.integer(recom\_list[[1]][i]),2]

}

print(recom\_result)

**#evaluate model**

evaluation\_scheme <- evaluationScheme(ratingmat, method="cross-validation", k=5, given=3, goodRating=5) #k=5 meaning a 5-fold cross validation. given=3 meaning a Given-3 protocol

evaluation\_results <- evaluate(evaluation\_scheme, method="UBCF", n=c(1,3,5,10,15,20))

eval\_results <- getConfusionMatrix(evaluation\_results)[[1]]

print(eval\_results)

**References**

[**https://cran.r-project.org/web/packages/vegan/vegan.pdf**](https://cran.r-project.org/web/packages/vegan/vegan.pdf)

[**https://cran.r-project.org/web/packages/recommenderlab/index.html**](https://cran.r-project.org/web/packages/recommenderlab/index.html)

[**http://tutorials.iq.harvard.edu/R/Rintro/Rintro.html**](http://tutorials.iq.harvard.edu/R/Rintro/Rintro.html)

[**https://nelsonmanohar.wordpress.com/2015/04/17/a-recommendation-system-in-r-applied-with-respect-to-the-movielens-database/**](https://nelsonmanohar.wordpress.com/2015/04/17/a-recommendation-system-in-r-applied-with-respect-to-the-movielens-database/)