Introduction to Recommender Systems

Code

Getting Started

- Open a new R Script
- Install (if necessary) and load the *data.table* and *RANN* package

Import data

- Import the the books dataset as a **data.table** from https://github.com/zygmuntz/goodbooks-10k/raw/master/books.csv and assign it the variable *books*
- Import the the ratings dataset as a **data.table** from https://github.com/zygmuntz/goodbooks-10k/raw/master/ratings.csv and assign it the variable *ratings*
- Import the the book to tags dataset as a **data.table** from https://github.com/zygmuntz/goodbooks-10k/raw/master/book_tags.csv and assign it the variable *book_tags*
- Import the tags lookup dataset as a **data.table** from https://github.com/zygmuntz/goodbooks-10k/raw/master/tags.csv and assign it the variable *tags*

Hide

```
library(data.table)
library(RANN)

#Import Book Data
books = fread("https://github.com/zygmuntz/goodbooks-10k/raw/master/books.csv")

#Import Ratings Data
ratings = fread("https://github.com/zygmuntz/goodbooks-10k/raw/master/ratings.csv")

#Import Tags Data
book_tags = fread("https://github.com/zygmuntz/goodbooks-10k/raw/master/book_tags.csv")
tags = fread("https://github.com/zygmuntz/goodbooks-10k/raw/master/tags.csv")
```

Processing the data

- Filter *book_tags* and keep only the top 3 tags (by counts), for each goodreads_book_id

Hide

```
#Get top 3 tags by books
book_tags = book_tags[order(rank(goodreads_book_id),-count)]
book_tags = book_tags[,head(.SD,3),.(goodreads_book_id)]
```

- Run the following code to generate indicator columns for a combination of genre types. Explore the main_tags data.frame.

Hide

```
#Get the main categories from tags for each book

main_tags_labels = c('romance','fiction','young-adult','fantasy','science-fiction',
'children','best','covers','non-fiction', 'history','mystery',
'paranormal','love','horror','historical','gay','sci-fi',
'historical-fiction','nonfiction','series','literature',  'contemporary',
'thriller','women','novels','suspense','classics' ,'graphic-novels',
'historical-romance', 'christian')

main_tags = merge(x=book_tags,y=tags,by="tag_id")
main_tags = main_tags[,.(tags = paste(tag_name,collapse=",")),.(goodreads_book_id)]

for(j in main_tags_labels){
  set(main_tags,j = j,value = grepl(x = main_tags$tags,pattern = j)*1)

  print(j)
}
main_tags[,tags:=NULL]
```

- Add the following columns to the *books* data.table. 1. *primary_author*: The name of the first author of a book. 2. *english*: A binary (0/1) indicator for the letters “en” in a books language_code.
- Remove all other columns except book_id,work_id,goodreads_book_id,primary_author, original_publication_year,english,average_rating,ratings_1,ratings_2, ratings_3,ratings_4,ratings_5 from books

Hide

```
#Get the primary author and determine if the book is printed in english
books = books[,.(book_id,work_id,goodreads_book_id,primary_author=unlist(lapply(strsplit(authors,","),function(X){X[1]})),
                 original_publication_year,english = grepl(pattern = "en",x = language_code,ignore.case = T)*1,
                 average_rating,ratings_1,ratings_2,ratings_3,ratings_4,ratings_5)]
```

- Join *main_tags* data to *books* on *goodreads_book_id*

Hide

```
books = merge(x= books,y=main_tags,by="goodreads_book_id")
```

Exploratory Data Analysis

- Create a new books data.table called *books_wide* by “melting” the genre columns.

Hide

```
books_wide = melt(books, id.vars = names(books)[!names(books) %in% main_tags_labels],measure.vars = main_tags_labels,variable.name = "genre",v
alue.name = "in_genre")
books_wide = books_wide[`in_genre`>0]
```

Use the books_wide data set for the following

- Calculate the average book rating by author

Hide

```
ratings_author = books_wide[!duplicated(books_wide[,.(goodreads_book_id)]),.(average_rating = median(average_rating),Books=.N),.(primary_author)]
ratings_author[,average_rating_bayes := (2*4+average_rating*Books)/(2+Books)]
ratings_author = ratings_author[order(-average_rating)]
```

- Calculate the average book rating, and number of published book by author in each genre

Hide

```
ratings_genre = books_wide[,.(average_rating = mean(average_rating),Books = .N),.(primary_author,genre)]
```

- Calculate the three top rated authors in each genre

Hide

```
ratings_genre = ratings_genre[order(rank(genre),-average_rating)]
ratings_author_genre = ratings_genre[,head(.SD,3),.(genre)]
```

- Calculate the best genre of each author

Hide

```
ratings_genre = ratings_genre[order(rank(primary_author),-average_rating)]
ratings_genre_author = ratings_genre[!duplicated(ratings_genre[,.(primary_author)])]
```

Content-Based Filtering

- Create *books_cb* a copy of *books*, and delete the primary_author, goodreads_book_id and work_id column

Hide

```
#Prepare the data
books_cb = books
books_cb$primary_author = NULL
books_cb$goodreads_book_id = NULL
books_cb$work_id = NULL
```

- Normalize all remaining numeric columns with the exception of *book_id*

Hide

```
normalize = function(x,min,width){
  return((x-min)/width)
}
normalize = Vectorize(normalize,vectorize.args = "x")


books_cb[,average_rating := normalize(average_rating,min = min(average_rating,na.rm = T),width = diff(range(average_rating,na.rm = T)))]
books_cb[,ratings_5 := normalize(ratings_5,min = min(ratings_5,na.rm = T),width = diff(range(ratings_5,na.rm = T)))]
books_cb[,ratings_4 := normalize(ratings_4,min = min(ratings_4,na.rm = T),width = diff(range(ratings_4,na.rm = T)))]
books_cb[,ratings_3 := normalize(ratings_3,min = min(ratings_3,na.rm = T),width = diff(range(ratings_3,na.rm = T)))]
books_cb[,ratings_2 := normalize(ratings_2,min = min(ratings_2,na.rm = T),width = diff(range(ratings_2,na.rm = T)))]
books_cb[,ratings_1 := normalize(ratings_1,min = min(ratings_1,na.rm = T),width = diff(range(ratings_1,na.rm = T)))]
books_cb[,original_publication_year := normalize(original_publication_year,min = min(original_publication_year,na.rm = T),width = diff(range(original_publi
cation_year,na.rm = T)))]
```

- Randomly assign 500 unique user_ids from *ratings* into a variable to *test_user*. Assign the remaining to *train_user*

Hide

```
user_ratings_profile = ratings[,.(Reviews = .N,AvgRatings = mean(rating,na.rm = T)),.(user_id)]
test_users = sample(user_ratings_profile$user_id,size = 500,replace = F)
train_users = user_ratings_profile$user_id[!user_ratings_profile$user_id %in% test_users]
```

- Create *user_affinity* a data.table of the high rated books by user_id

Hide

```
user_affinity = ratings
user_affinity = user_affinity[order(rank(user_id),-rating)]
user_affinity = user_affinity[,head(.SD,1),.(user_id)]
```

- For each user in *test_user*, find the top 5 books(book_id) that are most related to their highest rated book based on the **Cosine Similarity** metric

Hide

```
books_list = unique(user_affinity[user_id %in% test_users]$book_id)

recommendation_list_cb=list()

for(i in 1:length(books_list)){
  query = as.matrix(books_cb[book_id == books_list[i],.SD,.SDcols = names(books_cb)[!names(books_cb)=="book_id"]])
  data = as.matrix(books_cb[book_id != books_list[i],.SD,.SDcols = names(books_cb)[!names(books_cb)=="book_id"]])
  recommendation_list_cb[[i]]=apply(data,MARGIN = 1,
        function(X,Y){(Y%*%matrix(X))/(sqrt(sum(X^2))*sqrt(sum(Y^2)))}
        ,Y=query)
  print(i)
}

recommendation_list_cb_dt = data.table(t(do.call("rbind",recommendation_list_cb)))
recommendation_list_cb_dt[ , (names(recommendation_list_cb_dt)) := lapply(.SD, frank), .SDcols = names(recommendation_list_cb_dt)]
recommendation_list_cb_dt=apply(recommendation_list_cb_dt,2,function(X){which(X<=5)})
recommendation_list_cb_dt=data.table(t(recommendation_list_cb_dt))
names(recommendation_list_cb_dt) = paste("Item",1:ncol(recommendation_list_cb_dt),sep = "-")
recommendation_list_cb_dt = data.table(book_id = books_list,recommendation_list_cb_dt)
test_user_recommendations = merge(x= user_affinity[user_id %in% test_users,.(user_id,book_id)],y = recommendation_list_cb_dt,by="book_id")
```

- Generate recommendation for the test_users!

Hide

```
books_fresh = fread("https://github.com/zygmuntz/goodbooks-10k/raw/master/books.csv")


for(i in 1:nrow(head(test_user_recommendations,n = 100))){
  original = books_fresh[book_id %in% as.vector(test_user_recommendations[i,book_id])]$original_title
  recommend = books_fresh[book_id %in% as.vector(test_user_recommendations[i,.(`Item-1`,`Item-2`,`Item-3`,`Item-4`,`Item-5`)])]$original_title
  prompt2 = paste0("Based on your preferences we recommend: ", paste(recommend,collapse = ","))

  print(prompt2)
  print("")

}
```

Collaborative Filtering

- Create a user-item matrix ratings matrix using the *ratings* dataset called *user_item_mat*

Hide

```
user_item_mat = dcast(ratings, user_id ~ book_id, value.var = "rating")
user_item_mat = user_item_mat[,1:501]
```

- Create a subset of *user_item_mat*, for user_id's only in *test_users*

Hide

```
user_item_mat_test = user_item_mat[user_id %in% test_users]
user_item_mat_test_pred = user_item_mat_test
```

- Remove rows correspondig to *test_users* from *user_item_mat_test*

Hide

```
user_item_mat = user_item_mat[user_id %in% train_users]
```

- In *user_item_mat* and *user_item_mat_test*, replace all NA values with 0

Hide

```
for (i in names(user_item_mat)){
  user_item_mat[is.na(get(i)), (i):=0]
  print(i)
    }
for (i in names(user_item_mat_test_pred)){
  user_item_mat_test_pred[is.na(get(i)), (i):=0]
  print(i)
}
```

- For the first 10 rows (user) in *user_item_mat_test*:
- 1. Use the **Euclidean Distance** metric to find the 5 most similar users in *user_item_mat*.
- 2. Find the highest rated book_id's for the 5 most similar users

Hide

```
recommendation_list_cf=list()
for(i in 1:nrow(user_item_mat_test[1:10])){
  recommendation_list_cf[[i]] = user_affinity[user_id %in% user_item_mat[as.vector(nn2(data = user_item_mat[,-1],query = user_item_mat_test[i,-1],k = 5
,treetype = "kd",eps = .4)$nn.idx)]$user_id]$book_id
  print(i)
}

recommendation_list_cf_dt = data.table(do.call("rbind",recommendation_list_cf))
names(recommendation_list_cf_dt) = paste("Item",1:ncol(recommendation_list_cf_dt),sep = "-")
recommendation_list_cf_dt = data.table(user_id=user_item_mat_test[1:10]$user_id,recommendation_list_cf_dt)
```

- Generate recommendation for the 10 test_users!

Hide

```
for(i in 1:nrow(recommendation_list_cf_dt)){
  recommend = books_fresh[book_id %in% as.vector(recommendation_list_cf_dt[i,.(`Item-1`,`Item-2`,`Item-3`,`Item-4`,`Item-5`)])]$original_title
  prompt2 = paste0("Based on your preferences we recommend: ", paste(recommend,collapse = ","))

  print(prompt2)
  print("")

}
```