Yelp Dataset Review

John Antony

March 4, 2017

Reading review data from JSON file

```
suppressMessages(library(readr))
suppressMessages(library(dplyr))
reviews_file_path = "/Users/JohnAntony/Desktop/Main/Applications/R/MachineLearning/Yelp/yelp_dataset_challenge_ro
und9/yelp_academic_dataset_review.json"

review_lines <- read_lines(reviews_file_path,n_max = 100000,progress = FALSE)#n_max = 1000000

suppressMessages(library(stringr))
suppressMessages(library(jsonlite))

reviews_combined <- str_c("[", str_c(review_lines, collapse = ", "), "]")

reviews <- fromJSON(reviews_combined) %>%
  flatten() %>%
  tbl_df()

remove(review_lines)
remove(reviews_combined)
reviews['stars_agg'] = 1
```

Reading User data from JSON File

```
user_file_path = "/Users/JohnAntony/Desktop/Main/Applications/R/MachineLearning/Yelp/yelp_dataset_challenge_round
9/yelp_academic_dataset_user.json"

user_lines <- read_lines(user_file_path, n_max = 100000, progress = FALSE)

user_combined <- str_c("[", str_c(user_lines, collapse = ", "), "]")

user <- fromJSON(user_combined) %>%
  flatten() %>%
  tbl_df()

remove(user_lines)
remove(user_combined)
```

Reading Business data from JSON File

```
business_file_path = "/Users/JohnAntony/Desktop/Main/Applications/R/MachineLearning/Yelp/yelp_dataset_challenge_r
ound9/yelp_academic_dataset_business.json"

business_lines <- read_lines(business_file_path, n_max = 100000, progress = FALSE)

business_combined <- str_c("[", str_c(business_lines, collapse = ", "), "]")


business <- fromJSON(business_combined) %>%
  flatten() %>%
  tbl_df()

remove(business_lines)
remove(business_combined)

business['counter'] = 1
#aggregate(business$counter)
```

Finding how many raters provide rating more than once and find correlation b/w stars and length.

Second phase we will find spammers who give only 1 and 5 star rating

```
#Step-1: Filer one time raters
raters_count = aggregate.data.frame(x=reviews$stars_agg,by=list(reviews$user_id),FUN = "sum")
raters_count_filter = dplyr::filter(raters_count,raters_count$x>1)
number_of_raters_g1 = nrow(raters_count_filter)
pct_raters_g1 = number_of_raters_g1/2000000
cat("Number of Customers with more than one review:",number_of_raters_g1)
```

```
## Number of Customers with more than one review: 11802
```

```
cat("\nTo check what percentage of total review are they:",pct_raters_g1)
```

```
##
## To check what percentage of total review are they: 0.005901
```

```
#Getting data for the reviews having more that one rating by a user
review_filtered = reviews %>%
  filter(reviews$user_id %in% raters_count_filter$Group.1)
#Now Find those users who are spamers, who give only 1 and 5 star rating
review_filtered_agg = aggregate.data.frame(x=review_filtered$stars_agg,by=list(review_filtered$user_id,review_fil
tered$stars),FUN="sum")
#Sorting data
review_filtered_agg = review_filtered_agg[order(review_filtered_agg$Group.1,review_filtered_agg$Group.2),]

#Initial filter: Removing users who had given more than two types of rating
review_filtered_agg['counter']=1
review_filtered_l2 = aggregate.data.frame(x=review_filtered_agg$counter,by=list(review_filtered_agg$Group.1),FUN
= "sum")
review_filtered_l2 = dplyr::filter(review_filtered_l2,review_filtered_l2$x<3)
review_join = dplyr::semi_join(review_filtered_agg,review_filtered_l2, by="Group.1")

#Now Filtering by Ratings for only 1 star and 5 star
review_only_1 =  dplyr::filter(review_join,(review_join$Group.2==1))
review_only_5 = dplyr::filter(review_join,(review_join$Group.2==5))
review_only_1_5 = rbind.data.frame(review_only_1,review_only_5)
review_only_1_5 = review_only_1_5[order(review_only_1_5$Group.1),]

review_only_agg = aggregate.data.frame(review_only_1_5$counter,by=list(review_only_1_5$Group.1),FUN = "sum")
review_only_agg = dplyr::filter(review_only_agg,review_only_agg$x>1)

final_filtered_review = dplyr::semi_join(review_join,review_only_agg,"Group.1")
colnames(final_filtered_review)=c("user_id","stars","count","dummy")

bad_reviews_df = final_filtered_review
bad_reviews_df = dplyr::left_join(bad_reviews_df,reviews,by = c("user_id","stars"))
bad_reviews_df['text_length'] = 1
for(x in 1:nrow(bad_reviews_df)){
  #print(nchar(as.character(bad_reviews_df$text[x])))
  bad_reviews_df$text_length[x] = nchar(bad_reviews_df$text[x])
}

#Let's find the correlation between length of rating and stars given
cor(bad_reviews_df$stars,bad_reviews_df$text_length)
```

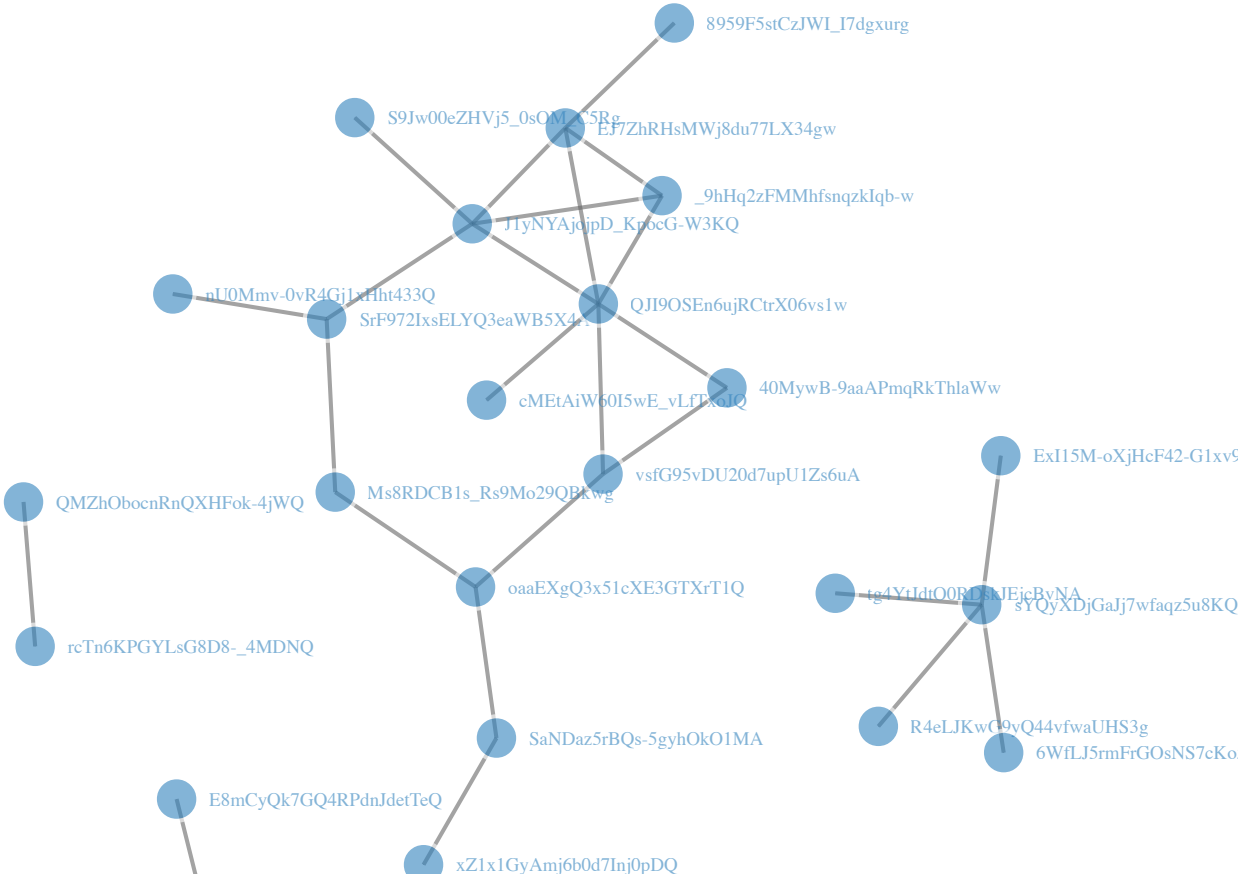
```
## [1] -0.2323213
```

```
remove(review_filtered,review_filtered_agg,raters_count,raters_count_filter,review_join,number_of_raters_g1)
```

Finding Influencer of a Business

```
#Inorder to get the influencer we are picking a business which has highest number of reviews
#Finding the most famous business, which has highest number of reviews
review_by_business = aggregate.data.frame(reviews$stars_agg,by = list(reviews$business_id),FUN = "sum")
review_by_business = review_by_business[order(-review_by_business$x),]
famous_bis = head(review_by_business$Group.1,1)

#Now subset all the reviews for that specific business
review_for_bis = dplyr::filter(reviews,famous_bis==reviews$business_id)
users_list = review_for_bis$user_id
unique_users = unique(users_list)

#Now let's get their friends list
user_friends_list = as.data.frame(unlist(unique_users))
colnames(user_friends_list) <- "user_id"
full_user_info = dplyr::semi_join(user,user_friends_list,by = "user_id")

#Collecting only user and their friends
user_frds = dplyr::select(full_user_info,1,5)

#Filterig user with no friends
user_frds = dplyr::filter(user_frds,user_frds$friends!="None")

#Defining the type of columns
op_df = data.frame(friends = character(),user_id = character())

for(x in user_frds$user_id){
  temp_df = dplyr::filter(user_frds,x==user_frds$user_id)
  frnds = data.frame(friends = lapply(temp_df$friends, as.character), stringsAsFactors=FALSE)
  colnames(frnds)[1] = "friends"
  frnds['user_id'] = x
  op_df = rbind(op_df,frnds)
  #dplyr::bind_rows(op_df,frnds)
  #print(frnds)
}

#Removing Duplicates
op_df = unique(op_df)
op_mat = as.matrix(op_df)

suppressMessages(library(igraph))
```

```
g = graph.data.frame(op_mat,directed=FALSE)
A = get.adjacency(g)
g1 = graph.adjacency(A,weighted=T, mode = "undirected")
g1 <- simplify(g1)
g2 = delete_edges(g1, which(E(g1)$weight < 2))
edge_list2 = get.edgelist(g2)

#Creating a Sample network
suppressMessages(library(networkD3))
networkData = data.frame(edge_list2[,1], edge_list2[,2])
simpleNetwork(networkData,zoom = TRUE)
```



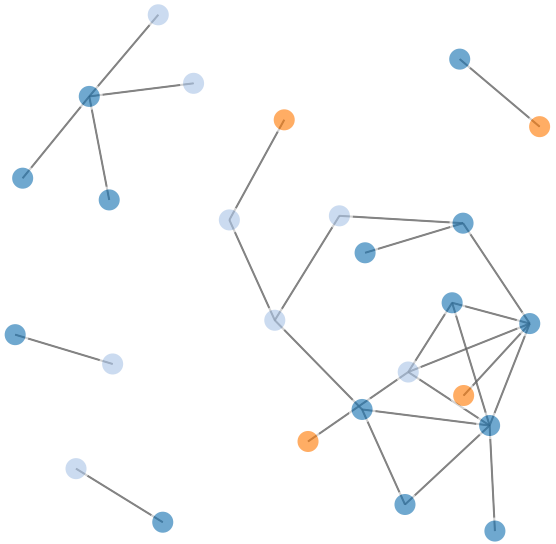
```
#ForceNetwork
links = networkData
names(links) = c("source","target")
links$value = 1

nodes_sample = data.frame(unique(c(as.vector(links$target),as.vector(links$source))))
colnames(nodes_sample) = c("user_id")
nodes_user_id = data.frame(unique(c(nodes_sample$user_id)))
nodes_user_id$unique.c.nodes_sample.user_id.. = nodes_user_id$unique.c.nodes_sample.user_id..-1
nodes_sample['id'] = nodes_user_id$unique.c.nodes_sample.user_id..


#Generating Source column
source_temp = data.frame(networkData$edge_list2...1.)
colnames(source_temp)=c("user_id")
source_temp = dplyr::left_join(source_temp,nodes_sample,by = "user_id")
```

```
## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## factors with different levels, coercing to character vector
```

```
links = data.frame(source_temp$id)
colnames(links) = c("source")

#Generating Target column
target_temp = data.frame(networkData$edge_list2...2.)
colnames(target_temp)=c("user_id")
target_temp = dplyr::left_join(target_temp,nodes_sample,by = "user_id")
```

```
## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## factors with different levels, coercing to character vector
```

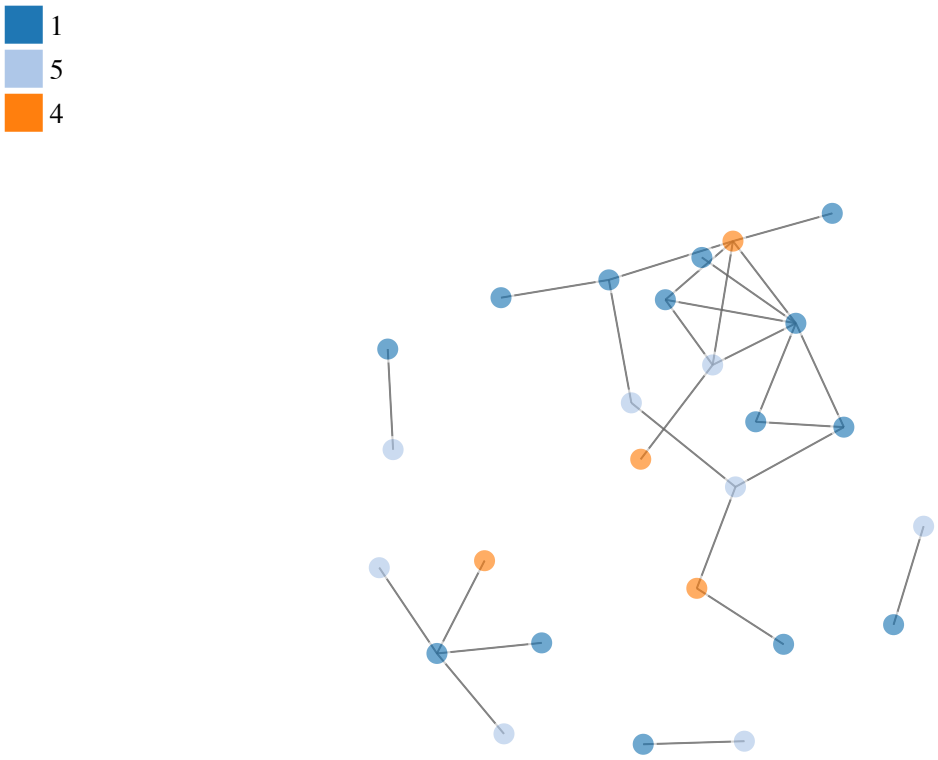
```
links['target'] = target_temp$id
links['value'] = 1

#nodes_user_id['unique_id'] = data.frame(unique(c(as.vector(links$target),as.vector(links$source))))
#Creating Nodes
nodes = data.frame(nodes_sample$id)
names(nodes) = "name"
nodes$group = ceiling(3*runif(length(nodes$name)))

forceNetwork(Links = links, Nodes = nodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 0.8)
```



```
g3 = graph.adjacency(A,mode="undirected",weighted=TRUE,diag=FALSE)
cent = evcent(g2,scale=FALSE)
cent = evcent(g2)$vector
cent_df = data.frame(cent)
cent_df['user_id'] = as.factor(row.names.data.frame(cent_df))

nodes_sample = dplyr::left_join(nodes_sample,cent_df,by = "user_id")
```

```
## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## factors with different levels, coercing to character vector
```

```
nodes_sample['group'] = 1

for(x in 1:nrow(nodes_sample)){
  if(nodes_sample$cent[x]>=0.7){
    nodes_sample$group[x]=4
  }
  else if(nodes_sample$cent[x]>0.1 & nodes_sample$cent[x]<0.7){
    nodes_sample$group[x]=5
  }
}


nodes1 = data.frame(nodes_sample$id)
nodes1['group'] = nodes_sample$group
colnames(nodes1) = c('name','group')

forceNetwork(Links = links, Nodes = nodes1, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 0.8,fontSize = 20,legend = TRUE)
```




```
#Dark Orange color has high centrality
remove(g,g1,g2,g3)
remove(cent_df,edge_list2,op_df,op_mat,review_by_business,review_for_bis,A,cent)
```

Recommendation

```
#Filtering for Vegas
biz_filter_df = dplyr::filter(business,business$city=="Las Vegas")
review_las = dplyr::semi_join(reviews,biz_filter_df,by = "business_id")

#Creating  a matrix of users and business with thier ratings
user_ratings = dplyr::select(review_las,2,3,4)
user_ratings = dplyr::sample_n(user_ratings,500,replace=TRUE)
suppressMessages(library(Matrix))
unique_user_id = data.frame(unique((user_ratings$user_id)))
unique_user_id = dplyr::sample_n(unique_user_id,40,replace=TRUE)
unique_business_id = data.frame(unique((user_ratings$business_id)))
unique_business_id = dplyr::sample_n(unique_business_id,40,replace=TRUE)
user_biz_df = expand.grid(unique_user_id$unique..user_ratings.user_id..,unique_business_id$unique..user_ratings.b
usiness_id..)
colnames(user_biz_df) = c("user_id","business_id")

#This is a DF with cartesian product of biz and user with thier ratings
user_biz_df = dplyr::left_join(user_biz_df,user_ratings,by =c("user_id","business_id"))
```

```
## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## character vector and factor, coercing into character vector

## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## character vector and factor, coercing into character vector
```

```
user_biz_df = dplyr::slice(user_biz_df,1:1600)
user_biz_df['user_rnd'] = sample(1:1600)
user_biz_df['biz_rnd'] = sample(1:1600)
user_biz_matrix = matrix(user_biz_df$stars,user_biz_df$user_rnd,user_biz_df$biz_rnd)
```

```
## Warning in matrix(user_biz_df$stars, user_biz_df$user_rnd, user_biz_df
## $biz_rnd): data length [1600] is not a sub-multiple or multiple of the
## number of rows [62]
```

```
#Function to replce NA's
filling_nas = function(x){
  m = mean(x,na.rm =TRUE)
  if(is.na(m)){
    m=1
  }
  else if(m>=5){
    m=2.5
  }
  else if(m<4){
    m=2
  }
  x[is.na(x)] = m
  x
}

final_matrix =  apply(user_biz_matrix, 2, filling_nas)
K=2
#Applying ALS
user_matrix = matrix(rnorm(K*user_biz_df$user_rnd),user_biz_df$user_rnd,K)
```

```
## Warning in matrix(rnorm(K * user_biz_df$user_rnd), user_biz_df$user_rnd, :
## data length [1600] is not a sub-multiple or multiple of the number of rows
## [62]
```

```
business_matrix = matrix(rnorm(K*user_biz_df$biz_rnd),user_biz_df$biz_rnd,K)
```

```
## Warning in matrix(rnorm(K * user_biz_df$biz_rnd), user_biz_df$biz_rnd, K):
## data length [1600] is not a sub-multiple or multiple of the number of rows
## [1054]
```

```
suppressMessages(library(ALS))
res = als(CList=list(user_matrix),S=business_matrix,PsiList=list(final_matrix))
```

```
## Initial RSS 216034.2
## Iteration (opt. S): 1, RSS: 70760.14, RD: 0.6724586
## Iteration (opt. C): 2, RSS: 60.64461, RD: 0.999143
## Iteration (opt. S): 3, RSS: 40.22554, RD: 0.3367004
## Iteration (opt. C): 4, RSS: 40.22514, RD: 1.007222e-05
## Initial RSS / Final RSS = 216034.2 / 40.22514 = 5370.626
```

```
#Results
users_res = t(res$CList[[1]])
movies_res = t(res$S)
ratings = t(users_res) %*% movies_res
err = (ratings-final_matrix)^2

#Check
print(mean(res$resid[[1]]^2))
```

```
## [1] 0.0006155527
```

```
#Weight for each user
print(users_res[,1])
```

```
## [1] 6.248995 0.000000
```

```
#Weight for each biz
print(movies_res[,1])
```

```
## [1] 0.1602042 0.0000000
```

```
#Find new user's weights on attributes
u_new = as.matrix(rowMeans(users_res))
print(u_new)
```

```
##          [,1]
## [1,] 6.242022
## [2,] 0.000000
```

```
#Find predicted ratings for all M movies for the new user
pred_ratings = t(movies_res) %*% u_new
sol = sort(pred_ratings,decreasing=TRUE,index.return=TRUE)
print("List of Business recommended based on users past rating")
```

```
## [1] "List of Business recommended based on users past rating"
```

```
print(head(sol$ix))
```

```
## [1]   71  871  200 1000   46  846
```

```
remove(user)
remove(reviews)
remove(business)
```