Recommendation Engines Quora

ignacio

10/12/2019

## Simple Unary

### user profile

simple\_unary\_user\_profile <- data.frame()  
for(k in 1:ncol(feedback)){  
 j<- vector()  
 for(i in 1:ncol(question\_topics)){  
 j[i]<- sum(question\_topics[,i]\*feedback[,k], na.rm = T)  
 }  
 simple\_unary\_user\_profile<-rbind(simple\_unary\_user\_profile,j)  
  
}  
  
colnames(simple\_unary\_user\_profile)<-colnames(question\_topics)  
rownames(simple\_unary\_user\_profile) <- colnames(feedback)  
simple\_unary\_user\_profile

## Sports Books Leadership Philosophy Society Fiction Security Love  
## User.1 3 -2 -1 0 0 2 -1 -1  
## User.2 -2 2 2 3 -1 -2 0 3  
## User.3 -2 1 1 0 0 -3 -1 -2  
## User.4 0 0 0 0 0 0 0 0  
## VideoGames Superheroes  
## User.1 1 0  
## User.2 0 -1  
## User.3 0 1  
## User.4 0 0

### calculating predictions

unary\_predict\_table <- data.frame()  
  
  
for(i in 1:nrow(simple\_unary\_user\_profile)){  
 for(k in 1:nrow(question\_topics)){  
 unary\_predict\_table[k,i]<- cosine(as.numeric(question\_topics[k,]),as.numeric(simple\_unary\_user\_profile[i,]))  
   
 }  
}  
colnames(unary\_predict\_table) <- rownames(simple\_unary\_user\_profile)  
rownames(unary\_predict\_table) <- rownames(question\_topics)  
unary\_predict\_table

## User.1 User.2 User.3 User.4  
## question1 0.3903600 -0.29814240 -0.2927700 NaN  
## question2 -0.4364358 0.83333333 0.0000000 NaN  
## question3 0.2519763 0.00000000 -0.3779645 NaN  
## question4 -0.3273268 0.66666667 -0.2182179 NaN  
## question5 -0.1259882 0.09622504 0.2519763 NaN  
## question6 0.4629100 0.11785113 -0.3086067 NaN  
## question7 -0.1543033 0.23570226 -0.1543033 NaN  
## question8 -0.2182179 0.33333333 0.1091089 NaN  
## question9 0.4629100 -0.23570226 -0.4629100 NaN  
## question10 -0.3779645 0.09622504 0.0000000 NaN  
## question11 0.0000000 0.09622504 0.1259882 NaN  
## question12 0.5039526 -0.38490018 -0.7559289 NaN  
## question13 -0.2182179 0.58333333 -0.1091089 NaN  
## question14 -0.2182179 0.58333333 0.2182179 NaN  
## question15 0.0000000 0.33333333 -0.6546537 NaN  
## question16 0.7559289 -0.38490018 -0.6299408 NaN  
## question17 -0.4364358 0.83333333 0.0000000 NaN  
## question18 0.1543033 0.35355339 0.0000000 NaN  
## question19 -0.3903600 0.14907120 0.1951800 NaN  
## question20 -0.1091089 0.41666667 0.0000000 NaN

### likes tables

likes\_dislikes <- rbind(sapply(unary\_predict\_table, function(x){sum(x>0)}))  
likes\_dislikes <- rbind(likes\_dislikes,sapply(unary\_predict\_table, function(x){sum(x<0)}))  
likes\_dislikes <- rbind(likes\_dislikes,sapply(unary\_predict\_table, function(x){sum(x==0)}))  
rownames(likes\_dislikes) <- c("likes", "dislikes","neutral")  
likes\_dislikes

## User.1 User.2 User.3 User.4  
## likes 7 15 5 NA  
## dislikes 11 4 10 NA  
## neutral 2 1 5 NA

## Unit Weight

### user profile

CB\_unit\_weight <- data.frame()  
  
  
for(k in 1:nrow(question\_topics)){  
 for(i in 1:ncol(question\_topics)){  
 CB\_unit\_weight[k,i] <- question\_topics[k,i]/sum(question\_topics[k,])   
 }  
}  
  
unit\_weight\_user\_profile <- data.frame()  
for(k in 1:ncol(feedback)){  
 j<- vector()  
 for(i in 1:ncol(CB\_unit\_weight)){  
 j[i]<- sum(CB\_unit\_weight[,i]\*feedback[,k], na.rm = T)  
 }  
 unit\_weight\_user\_profile<-rbind(unit\_weight\_user\_profile,j)  
   
}  
  
colnames(unit\_weight\_user\_profile)<-colnames(question\_topics)  
rownames(unit\_weight\_user\_profile) <- colnames(feedback)  
unit\_weight\_user\_profile

## Sports Books Leadership Philosophy Society Fiction  
## User.1 1.0333333 -0.4500000 -0.25 0.25 0.0 0.5333333  
## User.2 -0.5333333 0.5000000 0.55 0.75 -0.2 -0.5333333  
## User.3 -0.6666667 0.3333333 0.25 0.00 0.0 -0.9166667  
## User.4 0.0000000 0.0000000 0.00 0.00 0.0 0.0000000  
## Security Love VideoGames Superheroes  
## User.1 -0.20000000 -0.25 0.3333333 0.00000000  
## User.2 -0.08333333 0.75 0.0000000 -0.20000000  
## User.3 -0.33333333 -0.75 0.0000000 0.08333333  
## User.4 0.00000000 0.00 0.0000000 0.00000000

### predictions

unit\_weight\_predict\_table <- data.frame()  
  
  
for(i in 1:nrow(unit\_weight\_user\_profile)){  
 for(k in 1:nrow(question\_topics)){  
 unit\_weight\_predict\_table[k,i]<- cosine(as.numeric(CB\_unit\_weight[k,]),as.numeric(unit\_weight\_user\_profile[i,]))  
   
 }  
}  
colnames(unit\_weight\_predict\_table) <- rownames(simple\_unary\_user\_profile)  
unit\_weight\_predict\_table

## User.1 User.2 User.3 User.4  
## 1 0.42793450 -0.268372520 -0.38223539 NaN  
## 2 -0.25436333 0.834683430 -0.05698029 NaN  
## 3 0.32867937 0.006299408 -0.36187343 NaN  
## 4 -0.16351928 0.643742776 -0.28490144 NaN  
## 5 -0.04895225 0.113389342 0.16448792 NaN  
## 6 0.65949413 0.100297177 -0.32232919 NaN  
## 7 -0.12847288 0.254600527 -0.32232919 NaN  
## 8 -0.07267524 0.332782283 0.00000000 NaN  
## 9 0.44537266 -0.246885360 -0.44320263 NaN  
## 10 -0.27273394 0.081892302 0.00000000 NaN  
## 11 0.03496589 0.132287566 0.09869275 NaN  
## 12 0.57344059 -0.434659144 -0.75664445 NaN  
## 13 -0.09084405 0.605554645 -0.17094086 NaN  
## 14 -0.04239389 0.589188304 0.19943101 NaN  
## 15 0.12112540 0.289138705 -0.68376346 NaN  
## 16 0.79722229 -0.403162105 -0.62505411 NaN  
## 17 -0.25436333 0.834683430 -0.05698029 NaN  
## 18 0.29977006 0.347182537 0.00000000 NaN  
## 19 -0.29251219 0.165903012 0.10192944 NaN  
## 20 0.04845016 0.398247650 -0.02849014 NaN

### likes tables

likes\_dislikes <- rbind(sapply(unit\_weight\_predict\_table, function(x){sum(x>0)}))  
likes\_dislikes <- rbind(likes\_dislikes,sapply(unit\_weight\_predict\_table, function(x){sum(x<0)}))  
likes\_dislikes <- rbind(likes\_dislikes,sapply(unit\_weight\_predict\_table, function(x){sum(x==0)}))  
rownames(likes\_dislikes) <- c("likes", "dislikes","neutral")  
likes\_dislikes

## User.1 User.2 User.3 User.4  
## likes 10 16 4 NA  
## dislikes 10 4 13 NA  
## neutral 0 0 3 NA

## IDF

### User Profile

df <- sapply(question\_topics,sum)  
df

## Sports Books Leadership Philosophy Society Fiction   
## 4 6 10 11 6 6   
## Security Love VideoGames Superheroes   
## 7 6 7 5

ldf <- log10(nrow(question\_topics)/df)  
ldf

## Sports Books Leadership Philosophy Society Fiction   
## 0.6989700 0.5228787 0.3010300 0.2596373 0.5228787 0.5228787   
## Security Love VideoGames Superheroes   
## 0.4559320 0.5228787 0.4559320 0.6020600

idf\_user\_profile <- data.frame()  
  
  
for(i in 1:nrow(unit\_weight\_user\_profile)){  
 idf\_user\_profile <- rbind(idf\_user\_profile,ldf\*unit\_weight\_user\_profile[i,])  
}  
idf\_user\_profile

## Sports Books Leadership Philosophy Society Fiction  
## User.1 0.722269 -0.2352954 -0.0752575 0.06490933 0.0000000 0.2788687  
## User.2 -0.372784 0.2614394 0.1655665 0.19472798 -0.1045757 -0.2788687  
## User.3 -0.465980 0.1742929 0.0752575 0.00000000 0.0000000 -0.4793055  
## User.4 0.000000 0.0000000 0.0000000 0.00000000 0.0000000 0.0000000  
## Security Love VideoGames Superheroes  
## User.1 -0.09118639 -0.1307197 0.1519773 0.00000000  
## User.2 -0.03799433 0.3921591 0.0000000 -0.12041200  
## User.3 -0.15197732 -0.3921591 0.0000000 0.05017167  
## User.4 0.00000000 0.0000000 0.0000000 0.00000000

### predictions

### predictions

idf\_predict\_table <- data.frame()  
  
  
for(i in 1:nrow(unit\_weight\_user\_profile)){  
 for(k in 1:nrow(question\_topics)){  
 idf\_predict\_table[k,i]<- cosine(as.numeric(question\_topics[k,]),as.numeric(idf\_user\_profile[i,]))  
   
 }  
}  
colnames(idf\_predict\_table) <- rownames(simple\_unary\_user\_profile)  
idf\_predict\_table

## User.1 User.2 User.3 User.4  
## 1 0.49030911 -0.43636286 -0.45052633 NaN  
## 2 -0.22283194 0.69563292 -0.08761597 NaN  
## 3 0.23502693 -0.14950893 -0.34003155 NaN  
## 4 -0.13750986 0.49019116 -0.28807004 NaN  
## 5 -0.05696118 0.11172769 0.15924090 NaN  
## 6 0.65911062 -0.17276656 -0.40487382 NaN  
## 7 -0.10945262 0.26367435 -0.29714095 NaN  
## 8 -0.06011517 0.13851566 -0.01631067 NaN  
## 9 0.36075074 -0.27058440 -0.41645190 NaN  
## 10 -0.22320225 0.09417315 0.01583125 NaN  
## 11 0.05245020 0.04831938 0.05338959 NaN  
## 12 0.62209640 -0.54636674 -0.77842623 NaN  
## 13 -0.08352149 0.44450958 -0.19469814 NaN  
## 14 -0.05545663 0.42657220 0.15331891 NaN  
## 15 0.07215623 0.18526376 -0.62878274 NaN  
## 16 0.78833747 -0.51626607 -0.67060964 NaN  
## 17 -0.22283194 0.69563292 -0.08761597 NaN  
## 18 0.18160090 0.18894326 0.00000000 NaN  
## 19 -0.21274508 0.10065605 0.08118850 NaN  
## 20 0.02986545 0.22113045 -0.04713516 NaN

### likes tables

likes\_dislikes <- rbind(sapply(idf\_predict\_table, function(x){sum(x>0)}))  
likes\_dislikes <- rbind(likes\_dislikes,sapply(idf\_predict\_table, function(x){sum(x<0)}))  
likes\_dislikes <- rbind(likes\_dislikes,sapply(idf\_predict\_table, function(x){sum(x==0)}))  
rownames(likes\_dislikes) <- c("likes", "dislikes","neutral")  
likes\_dislikes

## User.1 User.2 User.3 User.4  
## likes 10 14 5 NA  
## dislikes 10 6 14 NA  
## neutral 0 0 1 NA

## Switched Hybrid

answer\_score <- apply(answers,1,function(x){sum(x, na.rm = T)})  
answers$standardized\_answers\_score <- (answer\_score-mean(answer\_score))/sd(answer\_score)  
  
  
idf\_average <- apply(idf\_predict\_table[,1:3],1, mean)  
question\_score <- 0.9\*idf\_average+0.1\*answers$standardized\_answers\_score  
questions\_ratings <- cbind(question\_score)  
rownames(questions\_ratings) <- rownames(question\_topics)  
sorted\_top5<- as.data.frame(sort(questions\_ratings[,1], decreasing = T)[1:5])  
colnames(sorted\_top5)<-"Top 5 Questions"  
sorted\_top5

## Top 5 Questions  
## question17 0.31318883  
## question19 0.28717983  
## question2 0.18802105  
## question20 0.17644433  
## question14 0.09804035

print("all question scores")

## [1] "all question scores"

questions\_ratings

## question\_score  
## question1 -0.12885569  
## question2 0.18802105  
## question3 -0.13564406  
## question4 -0.03990662  
## question5 0.01150000  
## question6 0.04749829  
## question7 -0.10216576  
## question8 -0.05383861  
## question9 -0.15717566  
## question10 -0.09324935  
## question11 -0.01304225  
## question12 -0.27009897  
## question13 0.04659310  
## question14 0.09804035  
## question15 -0.17069882  
## question16 -0.09321036  
## question17 0.31318883  
## question18 0.05187325  
## question19 0.28717983  
## question20 0.17644433

## Hybrid Challenge

we created a table that identifies if the user upvoted, downvoted, or answered the question. if the user didn’t do any, we considered him as not to have read the question. this is only assumed to create a recommendation system for current users.

read\_table <- data.frame()  
for(i in 1:ncol(feedback)){  
 for(k in 1:nrow(feedback)){  
 if(!is.na(feedback[k,i]) | !is.na(answers[k,i]) ){  
 read\_table[k,i] <- 1  
 }  
 else{  
 read\_table[k,i]<- 0  
 }  
 }  
}  
rownames(read\_table) <- rownames(question\_topics)  
colnames(read\_table) <- colnames(idf\_predict\_table)  
read\_table

## User.1 User.2 User.3 User.4  
## question1 1 1 0 0  
## question2 1 1 1 0  
## question3 0 0 0 0  
## question4 0 1 0 0  
## question5 0 1 1 0  
## question6 1 0 0 0  
## question7 0 0 1 0  
## question8 0 1 1 0  
## question9 0 0 0 0  
## question10 0 0 0 0  
## question11 0 0 0 0  
## question12 0 1 1 0  
## question13 0 1 1 0  
## question14 0 0 0 0  
## question15 0 0 1 0  
## question16 1 0 1 0  
## question17 0 1 1 0  
## question18 0 0 0 0  
## question19 1 1 1 0  
## question20 0 1 1 0

for the sake of creating an app and adding more features, we created a random word count for each question. moreover, we also created random dates for those questions, as an extra feature for our app.

wordcount <- c(138,244,32,24,423,34,234,235,133,803,578,54,843,564,83,456,668,126,868,38)  
x<-Sys.Date()  
dates <- c(x-455,x-1,x-1,x-322,x-1,x-157,x-230,x-100,x-455, x-1, x-1, x-56, x-1,x,x-1, x-7,x-1,x-1,x-23, x-1)  
  
wordcount\_date <- data.frame(wordcount,dates)  
colnames(wordcount\_date) <- c("word count", "date posted")  
rownames(wordcount\_date) <- rownames(question\_topics)  
wordcount\_date

## word count date posted  
## question1 138 2018-07-15  
## question2 244 2019-10-12  
## question3 32 2019-10-12  
## question4 24 2018-11-25  
## question5 423 2019-10-12  
## question6 34 2019-05-09  
## question7 234 2019-02-25  
## question8 235 2019-07-05  
## question9 133 2018-07-15  
## question10 803 2019-10-12  
## question11 578 2019-10-12  
## question12 54 2019-08-18  
## question13 843 2019-10-12  
## question14 564 2019-10-13  
## question15 83 2019-10-12  
## question16 456 2019-10-06  
## question17 668 2019-10-12  
## question18 126 2019-10-12  
## question19 868 2019-09-20  
## question20 38 2019-10-12

in our app, we created a new feature for bathroom reads. this feature includes questions for users where their predicted score is bigger than 0, the wordcount of each question is less than 300 (quick read) and to make sure that the user hasn’t seen it before (based on previous assumption).

bathroom\_pred <- vector()  
bathroom\_pred\_list <- data.frame("dummy" = c(1:20))  
predict\_table\_all\_users <- idf\_predict\_table  
predict\_table\_all\_users$User.4<- idf\_average  
  
  
for(i in 1:ncol(predict\_table\_all\_users)){  
 for(k in 1:nrow(read\_table)){  
 if(read\_table[k,i]==0 & wordcount[k]<150 & predict\_table\_all\_users[k,i]>0){  
 bathroom\_pred[k]=predict\_table\_all\_users[k,i]  
 }  
 else{bathroom\_pred[k]=0}  
 }  
 bathroom\_pred\_list <- cbind(bathroom\_pred\_list,bathroom\_pred)  
  
}  
bathroom\_pred\_list$dummy <- NULL  
colnames(bathroom\_pred\_list)<- colnames(predict\_table\_all\_users)  
rownames(bathroom\_pred\_list)<-rownames(question\_topics)  
bathroom\_pred\_list

## User.1 User.2 User.3 User.4  
## question1 0.00000000 0.0000000 0 0.00000000  
## question2 0.00000000 0.0000000 0 0.00000000  
## question3 0.23502693 0.0000000 0 0.00000000  
## question4 0.00000000 0.0000000 0 0.02153709  
## question5 0.00000000 0.0000000 0 0.00000000  
## question6 0.00000000 0.0000000 0 0.02715675  
## question7 0.00000000 0.0000000 0 0.00000000  
## question8 0.00000000 0.0000000 0 0.00000000  
## question9 0.36075074 0.0000000 0 0.00000000  
## question10 0.00000000 0.0000000 0 0.00000000  
## question11 0.00000000 0.0000000 0 0.00000000  
## question12 0.62209640 0.0000000 0 0.00000000  
## question13 0.00000000 0.0000000 0 0.00000000  
## question14 0.00000000 0.0000000 0 0.00000000  
## question15 0.07215623 0.1852638 0 0.00000000  
## question16 0.00000000 0.0000000 0 0.00000000  
## question17 0.00000000 0.0000000 0 0.00000000  
## question18 0.18160090 0.1889433 0 0.12351472  
## question19 0.00000000 0.0000000 0 0.00000000  
## question20 0.02986545 0.0000000 0 0.06795358

other than the bathroom feature, we created a commute feature. the commute feature represents questions that usually take more time and therefore makes commute entertaining.

commute\_pred <- vector()  
commute\_pred\_list <- data.frame("dummy" = c(1:20))  
predict\_table\_all\_users <- idf\_predict\_table  
predict\_table\_all\_users$User.4<- idf\_average  
  
  
for(i in 1:ncol(predict\_table\_all\_users)){  
 for(k in 1:nrow(read\_table)){  
 if(read\_table[k,i]==0 & wordcount[k]>150 & predict\_table\_all\_users[k,i]>0){  
 commute\_pred[k]=predict\_table\_all\_users[k,i]  
 }  
 else{commute\_pred[k]=0}  
 }  
 commute\_pred\_list <- cbind(commute\_pred\_list,commute\_pred)  
  
}  
commute\_pred\_list$dummy <- NULL  
colnames(commute\_pred\_list)<- colnames(predict\_table\_all\_users)  
rownames(commute\_pred\_list)<-rownames(question\_topics)  
commute\_pred\_list

## User.1 User.2 User.3 User.4  
## question1 0.0000000 0.00000000 0.00000000 0.00000000  
## question2 0.0000000 0.00000000 0.00000000 0.12839500  
## question3 0.0000000 0.00000000 0.00000000 0.00000000  
## question4 0.0000000 0.00000000 0.00000000 0.00000000  
## question5 0.0000000 0.00000000 0.00000000 0.07133580  
## question6 0.0000000 0.00000000 0.00000000 0.00000000  
## question7 0.0000000 0.26367435 0.00000000 0.00000000  
## question8 0.0000000 0.00000000 0.00000000 0.02069661  
## question9 0.0000000 0.00000000 0.00000000 0.00000000  
## question10 0.0000000 0.09417315 0.01583125 0.00000000  
## question11 0.0524502 0.04831938 0.05338959 0.05138639  
## question12 0.0000000 0.00000000 0.00000000 0.00000000  
## question13 0.0000000 0.00000000 0.00000000 0.05542998  
## question14 0.0000000 0.42657220 0.15331891 0.17481149  
## question15 0.0000000 0.00000000 0.00000000 0.00000000  
## question16 0.0000000 0.00000000 0.00000000 0.00000000  
## question17 0.0000000 0.00000000 0.00000000 0.12839500  
## question18 0.0000000 0.00000000 0.00000000 0.00000000  
## question19 0.0000000 0.00000000 0.00000000 0.00000000  
## question20 0.0000000 0.00000000 0.00000000 0.00000000

and as for our last feature, we created a breakfast feature that includes trending topics from the last 48 hours, taking into considerstion the date column previously created.

breakfast\_pred <- vector()  
breakfast\_pred\_list <- data.frame("dummy" = c(1:20))  
predict\_table\_all\_users <- idf\_predict\_table  
predict\_table\_all\_users$User.4<- idf\_average  
  
  
for(i in 1:ncol(predict\_table\_all\_users)){  
 for(k in 1:nrow(read\_table)){  
 if(read\_table[k,i]==0 & dates[k]>(Sys.Date()-2) & predict\_table\_all\_users[k,i]>0){  
 breakfast\_pred[k]=predict\_table\_all\_users[k,i]  
 }  
 else{breakfast\_pred[k]=0}  
 }  
 breakfast\_pred\_list <- cbind(breakfast\_pred\_list,breakfast\_pred)  
  
}  
breakfast\_pred\_list$dummy <- NULL  
colnames(breakfast\_pred\_list)<- colnames(predict\_table\_all\_users)  
rownames(breakfast\_pred\_list)<-rownames(question\_topics)  
breakfast\_pred\_list

## User.1 User.2 User.3 User.4  
## question1 0.00000000 0.00000000 0.00000000 0.00000000  
## question2 0.00000000 0.00000000 0.00000000 0.12839500  
## question3 0.23502693 0.00000000 0.00000000 0.00000000  
## question4 0.00000000 0.00000000 0.00000000 0.00000000  
## question5 0.00000000 0.00000000 0.00000000 0.07133580  
## question6 0.00000000 0.00000000 0.00000000 0.00000000  
## question7 0.00000000 0.00000000 0.00000000 0.00000000  
## question8 0.00000000 0.00000000 0.00000000 0.00000000  
## question9 0.00000000 0.00000000 0.00000000 0.00000000  
## question10 0.00000000 0.09417315 0.01583125 0.00000000  
## question11 0.05245020 0.04831938 0.05338959 0.05138639  
## question12 0.00000000 0.00000000 0.00000000 0.00000000  
## question13 0.00000000 0.00000000 0.00000000 0.05542998  
## question14 0.00000000 0.42657220 0.15331891 0.17481149  
## question15 0.07215623 0.18526376 0.00000000 0.00000000  
## question16 0.00000000 0.00000000 0.00000000 0.00000000  
## question17 0.00000000 0.00000000 0.00000000 0.12839500  
## question18 0.18160090 0.18894326 0.00000000 0.12351472  
## question19 0.00000000 0.00000000 0.00000000 0.00000000  
## question20 0.02986545 0.00000000 0.00000000 0.06795358