　TAD Project

Xiaoning He

5/10/2019

1.1. Preliminary Analysis - cosine similary, euclidean & manhattan distance

rm(list =ls())  
library(quanteda)

## Package version: 1.4.0

## Parallel computing: 2 of 8 threads used.

## See https://quanteda.io for tutorials and examples.

##   
## Attaching package: 'quanteda'

## The following object is masked from 'package:utils':  
##   
## View

library(quanteda.corpora)  
setwd("~/Desktop/GitHub/Movie\_Reviews\_Text\_Analyzation\_Project") # set working directory  
dc<- read.csv('Reviews\_DC.csv')  
mv<- read.csv('Reviews\_Marvel.csv')  
dc$review <- as.character(dc$review)  
mv$review <- as.character(mv$review)  
  
# Combine the reviwes for each into one document,and combine the two documents into one corpus.  
library(tm)

## Loading required package: NLP

##   
## Attaching package: 'tm'

## The following objects are masked from 'package:quanteda':  
##   
## as.DocumentTermMatrix, stopwords

dc\_review <- paste(unlist(dc$review), collapse =" ")  
mv\_review <- paste(unlist(mv$review), collapse =" ")  
  
dc\_corpus = corpus(dc\_review)  
mv\_corpus = corpus(mv\_review)  
review\_corpus <- c(dc\_corpus,mv\_corpus)  
docnames(review\_corpus) <- c('DC','Marvel')  
  
# covert to dfm and compare the cosine similary, euclidean & manhattan distance between the two documents  
  
Movielist\_DC <- read.csv('Movielist\_DC.csv')  
Movielist\_MV <- read.csv('Movielist\_Marvel.csv')  
  
Movielist\_DC\_token <- paste(unlist(Movielist\_DC$name), collapse =" ") %>%   
 as.character() %>% tolower() %>% unique() %>%  
 tokens(remove\_punct = TRUE, remove\_numbers = TRUE) %>%  
 unlist() %>% unique()  
  
Movielist\_MV\_token <- paste(unlist(Movielist\_MV$name), collapse =" ") %>%   
 as.character() %>% tolower() %>%   
 tokens(remove\_punct = TRUE, remove\_numbers = TRUE) %>%  
 unlist() %>% unique()  
Movielist\_MV\_token

## [1] "captain" "america" "the" "first" "avenger"   
## [6] "iron" "man" "incredible" "hulk" "thor"   
## [11] "avengers" "three" "dark" "world" "winter"   
## [16] "soldier" "guardians" "of" "galaxy" "vol"   
## [21] "age" "ultron" "ant-man" "civil" "war"   
## [26] "spider-man" "homecoming" "doctor" "strange" "ragnarok"   
## [31] "black" "panther" "infinity" "and" "wasp"   
## [36] "marvel"

review\_dfm <- dfm(review\_corpus, stem = TRUE, remove\_punct = TRUE, remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token), remove\_numbers = TRUE)  
  
dim(review\_dfm)

## [1] 2 51303

similarity <- textstat\_simil(review\_dfm, margin = "documents", method = 'cosine')  
as.matrix(similarity)

## DC Marvel  
## DC 1.000000 0.950243  
## Marvel 0.950243 1.000000

euclidean <- textstat\_dist(review\_dfm, margin = "documents", method = "euclidean")  
as.matrix(euclidean)

## DC Marvel  
## DC 0.00 21661.05  
## Marvel 21661.05 0.00

manhattan <- textstat\_dist(review\_dfm, margin = "documents", method = "manhattan")  
as.matrix(manhattan)

## DC Marvel  
## DC 0 579873  
## Marvel 579873 0

1.2. Preliminary Analysis - Most feaqurent terms (I used the topfearues in dfm, but this is not the only way to find top terms, should be others as well).

dc\_dfm <- dfm(dc\_corpus, stem = TRUE, remove\_punct = TRUE, remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token), remove\_numbers = TRUE)  
mv\_dfm <- dfm(mv\_corpus, stem = TRUE, remove\_punct = TRUE, remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token), remove\_numbers = TRUE)  
  
dc\_top <- topfeatures(dc\_dfm, n = 300, decreasing = TRUE, scheme = "count")  
mv\_top <- topfeatures(mv\_dfm, n = 300, decreasing = TRUE, scheme = "count")  
  
  
# top terms based on counts for each document  
dc\_top <- names(dc\_top)  
mv\_top <- names(mv\_top)  
  
  
  
  
# shared terms in the top 20 temrs   
shared\_terms <- intersect(dc\_top,mv\_top)  
#shared\_terms   
length(shared\_terms)

## [1] 237

dc\_unique\_top<- setdiff(dc\_top,mv\_top) # in dc,not mv  
dc\_unique\_top

## [1] "joker" "dc" "reev" "christoph" "bruce"   
## [6] "nolan" "terribl" "burton" "anim" "classic"   
## [11] "kid" "citi" "trilog" "adapt" "wayn"   
## [16] "richard" "beauti" "style" "score" "version"   
## [21] "cut" "portray" "lex" "poor" "idea"   
## [26] "tim" "graphic" "novel" "cheesi" "clark"   
## [31] "minut" "half" "hate" "critic" "stupid"   
## [36] "keaton" "luthor" "snyder" "aw" "loi"   
## [41] "famili" "horribl" "greatest" "true" "dialogu"   
## [46] "rememb" "john" "campi" "silli" "music"   
## [51] "gotham" "base" "truli" "face" "wrong"   
## [56] "wonder" "donner" "read" "write" "name"   
## [61] "shot" "near" "els"

length(dc\_unique\_top)

## [1] 63

mv\_unique\_top <- setdiff(mv\_top,dc\_top) # in mv,not mv  
mv\_unique\_top

## [1] "mcu" "toni" "stark" "cinemat" "downey"   
## [6] "chris" "loki" "jr" "robert" "marvel"   
## [11] "suit" "peter" "battl" "deliv" "next"   
## [16] "wait" "pace" "lead" "previous" "twist"   
## [21] "team" "addit" "rudd" "paul" "forc"   
## [26] "evan" "spiderman" "side" "less" "hemsworth"   
## [31] "care" "date" "pack" "weak" "impress"   
## [36] "heart" "build" "rest" "entri" "studio"   
## [41] "friend" "cap" "humour" "introduc" "throughout"   
## [46] "stand" "steve" "manag" "earth" "banner"   
## [51] "thank" "continu" "opinion" "along" "storylin"   
## [56] "found" "instead" "holland" "let" "norton"   
## [61] "phase" "hand" "predecessor"

length(mv\_unique\_top)

## [1] 63

1.3 Preliminary Analysis - rating score for each

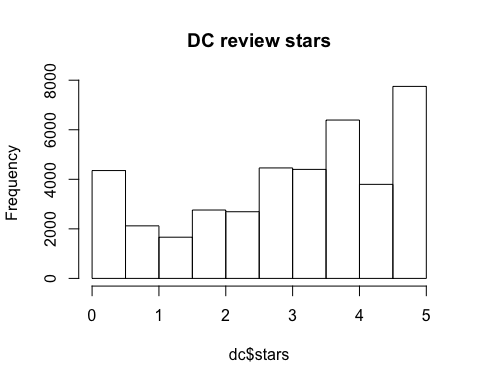
median(dc$stars)

## [1] 3.5

mean(dc$stars)

## [1] 3.17795

hist(dc$stars, main = 'DC review stars')



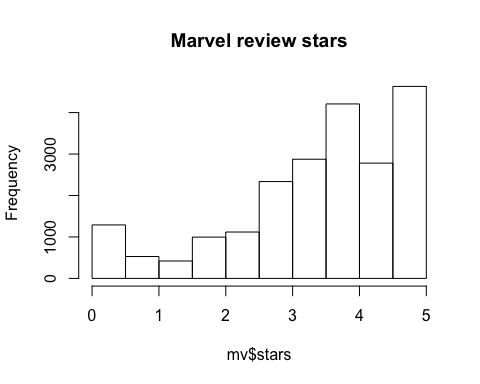
median(mv$stars)

## [1] 4

mean(mv$stars)

## [1] 3.582885

hist(mv$stars, main = 'Marvel review stars')



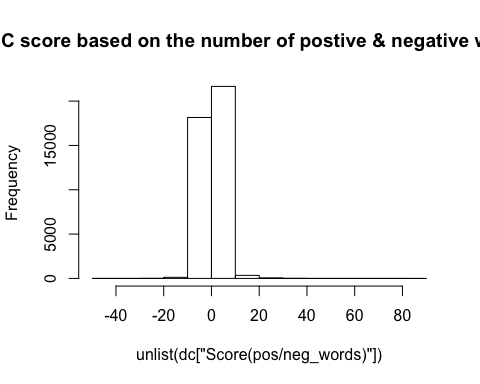
# using the positive words and negatives words doc. given from HW2  
library(stringr)  
pos <- readLines('positive-words.txt')  
neg <- readLines('negative-words.txt')  
  
for (row in 1:nrow(dc)) {  
 text <- tolower(dc[row, 'review'])  
 words <- unlist(str\_split(text, " "))  
 pos.matches <- match(words, pos)  
 neg.matches <- match(words, neg)  
 score <- sum(!is.na(pos.matches))-sum(!is.na(neg.matches))  
 dc[row, 'Score(pos/neg\_words)']<-score  
 }  
median(unlist(dc['Score(pos/neg\_words)']))

## [1] 1

mean(unlist(dc['Score(pos/neg\_words)']))

## [1] 0.9554538

hist(unlist(dc['Score(pos/neg\_words)']), main = 'DC score based on the number of postive & negative words')



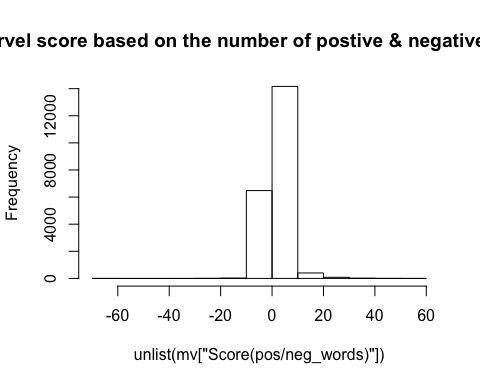
for (row in 1:nrow(mv)) {  
 text <- tolower(mv[row, 'review'])  
 words <- unlist(str\_split(text, " "))  
 pos.matches <- match(words, pos)  
 neg.matches <- match(words, neg)  
 score <- sum(!is.na(pos.matches))-sum(!is.na(neg.matches))  
 mv[row, 'Score(pos/neg\_words)']<-score  
}  
median(unlist(mv['Score(pos/neg\_words)']))

## [1] 1

mean(unlist(mv['Score(pos/neg\_words)']))

## [1] 1.93727

hist(unlist(mv['Score(pos/neg\_words)']), main = 'Marvel score based on the number of postive & negative words')

 Modeling

dc$Label <- "DC"  
mv$Label <- "Marvel"  
  
all\_movies<- rbind(dc, mv)  
head(all\_movies)

## name  
## 1 Superman and the Mole-Men  
## 2 Superman and the Mole-Men  
## 3 Superman and the Mole-Men  
## 4 Superman and the Mole-Men  
## 5 Superman and the Mole-Men  
## 6 Superman and the Mole-Men  
## review  
## 1 An average b-movie of the early 50s, preceding the TV series "Adventures of Superman".  
## 2 About what you would expect. Not that much about Superman but the mole men were pretty weird.  
## 3 A charming little tale where the villains are prejudice and paranoia and Superman's real powers are tolerance and compassion.  
## 4 Where to even begin? For starters, this is more of a B-grade science-fiction picture that happens to feature Superman than a "Superman" movie outright. Whatever the intentions of the producers, it certainly is a product of its time. By that, and given its low budget, I mean that it's kind of what you'd expect from a sci-fi film in the 1950's: shoddy production values, questionable acting, and overt message-making. Still despite all of this there is a certain B-movie charm, and of course George Reeves has a great screen presence as the Man of Steel (not so much Clark Kent, who is played too similarly). Other than Clark Kent/Superman and Lois Lane, though, there isn't much else here that ties it to the Action Comics source material. Ergo, no Daily Planet, no Metropolis, etc. But I didn't really mind. As long as you do away with any expectations of what a Superman movie "should" be, this film can be a lot of fun. And, at 58 minutes, it never wears out its welcome. Considering the time in which this film was made, with liberal Hollywood under attack by fear-mongering by the likes of Joseph McCarthy and racial tensions coming to a boil, the message it conveys is actually quite radical (again, for its time). It basically says that as beings who inhabit this planet, we should all just get along regardless of who we are. There are also other things you could read into it, like anti-oil drilling and gun control, but those are secondary concerns. Did I like it? Well, yes and no. It isn't my idea of what a comic movie should be, but taken as a cheesy sci-movie, it has its charms. I wouldn't bend over backwards to see this if you haven't already, but fans of George Reeves of Superman would be remiss for not checking it out.  
## 5 terrible. superman is only in it for like 2 mins.  
## 6 I own this as a bonus feature on the DVD and on the Blu-Ray of:\* Superman (1978)and i also own it as a two part episode in the first season of the "Adventures Of Superman" TV series.  
## stars date Score(pos/neg\_words) Label  
## 1 2.0 April 6, 2019 0 DC  
## 2 2.0 May 7, 2017 1 DC  
## 3 3.0 June 15, 2015 -2 DC  
## 4 3.0 June 8, 2015 0 DC  
## 5 0.5 September 29, 2014 1 DC  
## 6 3.0 June 18, 2014 1 DC

# Randomize the order of the data frame  
all\_movies <- all\_movies[sample(1:nrow(all\_movies)), ]

2.1 Native Based (uniform and with smooting)

library(readtext)   
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

set.seed(2019)  
prop\_train <- 0.7  
index <- 1:nrow(all\_movies)  
train\_index <- sample(index, ceiling(prop\_train\*length(index)), replace = FALSE)  
test\_index <- index[train\_index]  
train\_set <- all\_movies[train\_index,]  
test\_set <- all\_movies[test\_index,]  
train\_dfm <- dfm(train\_set$review, stem = TRUE, remove\_punct = TRUE, remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token),remove\_numbers = TRUE)  
test\_dfm <- dfm(test\_set$review, stem = TRUE, remove\_punct = TRUE, remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token),remove\_numbers = TRUE)  
test\_dfm <- dfm\_match(test\_dfm, features = featnames(train\_dfm))  
  
# uniform and with smooting  
nb\_model <- textmodel\_nb(train\_dfm, train\_set$Label, smooth = 1, prior = "uniform")  
predicted\_label\_uniform <- predict(nb\_model, newdata = test\_dfm)  
baseline\_acc <- max(prop.table(table(test\_set$Label)))  
matrix <- table(test\_set$Label, predicted\_label\_uniform)  
matrix

## predicted\_label\_uniform  
## DC Marvel  
## DC 23688 4709  
## Marvel 2254 12449

nb\_acc <- sum(diag(matrix))/sum(matrix) # (TP + TN) / (TP + FP + TN + FN)  
nb\_recall <- matrix[2,2]/sum(matrix[2,]) #TP / (TP + FN)  
nb\_precision <- matrix[2,2]/sum(matrix[,2]) # TP / (TP + FP)  
nb\_f1 <- 2\*(nb\_recall\*nb\_precision)/(nb\_recall + nb\_precision)  
cat(  
 " Baseline Accuracy: ", baseline\_acc, "\n",  
 "Accuracy:", nb\_acc, "\n",  
 "Recall:", nb\_recall, "\n",  
 "Precision:", nb\_precision, "\n",  
 "F1-score:", nb\_f1)

## Baseline Accuracy: 0.6588631   
## Accuracy: 0.8384455   
## Recall: 0.846698   
## Precision: 0.7255508   
## F1-score: 0.781457

2.2 Native Based (uniform and without smooting)

nb\_model <- textmodel\_nb(train\_dfm, train\_set$Label, smooth = 0, prior = "uniform")  
predicted\_label\_uniform <- predict(nb\_model, newdata = test\_dfm)  
baseline\_acc <- max(prop.table(table(test\_set$Label)))  
matrix <- table(test\_set$Label, predicted\_label\_uniform)  
matrix

## predicted\_label\_uniform  
## DC Marvel  
## DC 23228 5169  
## Marvel 1609 13094

nb\_acc <- sum(diag(matrix))/sum(matrix) # (TP + TN) / (TP + FP + TN + FN)  
nb\_recall <- matrix[2,2]/sum(matrix[2,]) #TP / (TP + FN)  
nb\_precision <- matrix[2,2]/sum(matrix[,2]) # TP / (TP + FP)  
nb\_f1 <- 2\*(nb\_recall\*nb\_precision)/(nb\_recall + nb\_precision)  
cat(  
 " Baseline Accuracy: ", baseline\_acc, "\n",  
 "Accuracy:", nb\_acc, "\n",  
 "Recall:", nb\_recall, "\n",  
 "Precision:", nb\_precision, "\n",  
 "F1-score:", nb\_f1)

## Baseline Accuracy: 0.6588631   
## Accuracy: 0.8427378   
## Recall: 0.8905666   
## Precision: 0.7169687   
## F1-score: 0.7943942

2.3 Native Based (docfreq and with smooting) # docfreq prior takes into account of the proportions of the document in training set, we have much more DC than Marvel.

nb\_model <- textmodel\_nb(train\_dfm, train\_set$Label, smooth = 1, prior = "docfreq")  
predicted\_label\_uniform <- predict(nb\_model, newdata = test\_dfm)  
baseline\_acc <- max(prop.table(table(test\_set$Label)))  
matrix <- table(test\_set$Label, predicted\_label\_uniform)  
matrix

## predicted\_label\_uniform  
## DC Marvel  
## DC 26461 1936  
## Marvel 4233 10470

nb\_acc <- sum(diag(matrix))/sum(matrix) # (TP + TN) / (TP + FP + TN + FN)  
nb\_recall <- matrix[2,2]/sum(matrix[2,]) #TP / (TP + FN)  
nb\_precision <- matrix[2,2]/sum(matrix[,2]) # TP / (TP + FP)  
nb\_f1 <- 2\*(nb\_recall\*nb\_precision)/(nb\_recall + nb\_precision)  
cat(  
 " Baseline Accuracy: ", baseline\_acc, "\n",  
 "Accuracy:", nb\_acc, "\n",  
 "Recall:", nb\_recall, "\n",  
 "Precision:", nb\_precision, "\n",  
 "F1-score:", nb\_f1)

## Baseline Accuracy: 0.6588631   
## Accuracy: 0.8568677   
## Recall: 0.7120996   
## Precision: 0.8439465   
## F1-score: 0.7724372

2.1.4 Native Based (docfreq and without smooting) # docfreq prior takes into account of the proportions of the document in training set

nb\_model <- textmodel\_nb(train\_dfm, train\_set$Label, smooth = 0, prior = "docfreq")  
predicted\_label\_uniform <- predict(nb\_model, newdata = test\_dfm)  
baseline\_acc <- max(prop.table(table(test\_set$Label)))  
matrix <- table(test\_set$Label, predicted\_label\_uniform)  
matrix

## predicted\_label\_uniform  
## DC Marvel  
## DC 26102 2295  
## Marvel 3461 11242

nb\_acc <- sum(diag(matrix))/sum(matrix) # (TP + TN) / (TP + FP + TN + FN)  
nb\_recall <- matrix[2,2]/sum(matrix[2,]) #TP / (TP + FN)  
nb\_precision <- matrix[2,2]/sum(matrix[,2]) # TP / (TP + FP)  
nb\_f1 <- 2\*(nb\_recall\*nb\_precision)/(nb\_recall + nb\_precision)  
cat(  
 " Baseline Accuracy: ", baseline\_acc, "\n",  
 "Accuracy:", nb\_acc, "\n",  
 "Recall:", nb\_recall, "\n",  
 "Precision:", nb\_precision, "\n",  
 "F1-score:", nb\_f1)

## Baseline Accuracy: 0.6588631   
## Accuracy: 0.8664501   
## Recall: 0.7646059   
## Precision: 0.8304647   
## F1-score: 0.7961756

3.1 SVM

#yelp\_svm <- yelp[1:1000,]  
  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

##   
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':  
##   
## annotate

library(dplyr)  
  
movies\_svm <- all\_movies[sample(1:5000),]  
svm\_dfm <- dfm(movies\_svm$review, stem = TRUE, remove\_punct = TRUE, remove\_numbers = TRUE, remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token)) %>% convert("matrix")  
set.seed(2019)  
baseline\_acc <- max(prop.table(table(movies\_svm $Label)))  
baseline\_acc

## [1] 0.6642

ids\_train <- createDataPartition(1:nrow(svm\_dfm), p = 0.8, list = FALSE, times = 1)  
train\_x <- svm\_dfm[ids\_train,] %>% as.data.frame() # train set data  
train\_y <- movies\_svm$Label[ids\_train] %>% as.factor() # train set labels  
test\_x <- svm\_dfm[-ids\_train, ] %>% as.data.frame() # test set data  
test\_y <- movies\_svm$Label[-ids\_train] %>% as.factor() # test set labels  
trctrl <-trainControl(method="cv", number=3)  
svm\_mod\_linear <- train(x = train\_x,  
 y = train\_y,  
 method = "svmLinear",  
 trControl = trctrl)

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

svm\_linear\_pred <- predict(svm\_mod\_linear, newdata = test\_x)  
svm\_linear\_cmat <- confusionMatrix(svm\_linear\_pred, test\_y)  
   
length(train\_x)

## [1] 12137

length(train\_y)

## [1] 4000

svm\_mod\_radial <- train(x = train\_x,  
 y = train\_y,  
 method = "svmRadial",  
 trControl = trctrl)

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

svm\_radial\_pred <- predict(svm\_mod\_radial, newdata = test\_x)  
svm\_radial\_cmat <- confusionMatrix(svm\_radial\_pred, test\_y)  
  
cat(  
 "SVM-Linear Accuracy:", svm\_linear\_cmat$overall[["Accuracy"]], "\n",  
 "SVM-Radial Accuracy:", svm\_radial\_cmat$overall[["Accuracy"]])

## SVM-Linear Accuracy: 0.778   
## SVM-Radial Accuracy: 0.731

svm\_radial\_cmat$byClass

## Sensitivity Specificity Pos Pred Value   
## 0.9984756 0.2209302 0.7096425   
## Neg Pred Value Precision Recall   
## 0.9870130 0.7096425 0.9984756   
## F1 Prevalence Detection Rate   
## 0.8296390 0.6560000 0.6550000   
## Detection Prevalence Balanced Accuracy   
## 0.9230000 0.6097029

1. Random Forest

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

movies\_rf <- all\_movies[sample(1:1000),]  
  
  
prop\_train <- 0.8  
index <- 1:nrow(movies\_rf)  
train\_index <- sample(index, ceiling(prop\_train\*length(index)), replace = FALSE)  
test\_index <- index[train\_index]  
train\_set <- movies\_rf[train\_index,]  
test\_set <- movies\_rf[test\_index,]  
train\_dfm <- dfm(train\_set$review, stem = TRUE, remove\_punct = TRUE, remove\_numbers = TRUE,remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token))  
test\_dfm <- dfm(test\_set$review, stem = TRUE, remove\_punct = TRUE, remove\_numbers = TRUE,remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token))  
test\_dfm <- dfm\_match(test\_dfm, features = featnames(train\_dfm))  
train\_dfm <- train\_dfm %>% convert("matrix")  
test\_dfm <- test\_dfm%>% convert("matrix")  
train\_set$Label <- as.factor(train\_set$Label)  
rf\_model<- randomForest(train\_dfm,train\_set$Label , importance=TRUE)  
rf\_model

##   
## Call:  
## randomForest(x = train\_dfm, y = train\_set$Label, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 66  
##   
## OOB estimate of error rate: 26.62%  
## Confusion matrix:  
## DC Marvel class.error  
## DC 506 27 0.05065666  
## Marvel 186 81 0.69662921

importance <- importance(rf\_model)  
import\_df<- as.data.frame(importance)  
import\_df[order(-import\_df$MeanDecreaseGini),][1:30,]

## DC Marvel MeanDecreaseAccuracy  
## mcu 29.0635182 21.0180218 27.5105662  
## robert 15.4622201 10.7776676 14.8942611  
## downey 12.2866706 8.3786462 11.5154222  
## funni 10.5621580 3.6467797 9.6041503  
## univers 16.5498838 3.0608233 14.5719110  
## jr 11.4208848 6.3044741 10.3906521  
## toni 11.5872753 3.4484565 9.9306695  
## blockbust 13.0414213 6.1874305 12.2338800  
## best 1.8200971 -0.1048438 1.4971769  
## start 10.1096684 -0.1135969 8.4974250  
## 2nd 9.9452901 0.4088955 7.1209979  
## promis 6.5873092 8.2975884 8.6552019  
## film 0.8893770 -0.1776787 0.6423440  
## stark 9.3422382 1.8120827 7.8775419  
## masterpiec 0.1807879 8.0912562 4.8277862  
## good 3.1098455 -1.4022977 2.1977198  
## second 8.6815264 -2.3916178 5.0942793  
## marvel 8.7193101 1.7124214 7.0117848  
## one 3.1583807 -2.1417825 2.2914987  
## action 2.7277861 0.4539824 2.5056133  
## better 1.6434285 -2.3744290 -0.2649316  
## bucki 8.3210945 7.4096381 8.8658164  
## yasssssssssssssssssssssssssss 0.0000000 0.0000000 0.0000000  
## humor 7.6905295 0.1412295 5.9495246  
## hurt 5.7891647 0.6214035 4.1747482  
## likeabl 7.2487619 4.3953611 6.8701213  
## holland 8.5326815 4.4857102 8.0131181  
## awesom 5.5716104 0.1714180 4.1965010  
## crazi 8.5555059 3.6422970 7.2594398  
## superhero 2.4211895 -2.7706054 0.3663790  
## MeanDecreaseGini  
## mcu 11.2506581  
## robert 3.6420293  
## downey 2.6329089  
## funni 2.4400922  
## univers 2.1267376  
## jr 2.0802043  
## toni 1.8243837  
## blockbust 1.6172351  
## best 1.4882951  
## start 1.4604186  
## 2nd 1.4221922  
## promis 1.4089872  
## film 1.3794697  
## stark 1.2996276  
## masterpiec 1.1877082  
## good 1.1830800  
## second 1.1699943  
## marvel 1.1575385  
## one 1.1376255  
## action 1.1329309  
## better 1.1049118  
## bucki 1.1047540  
## yasssssssssssssssssssssssssss 1.0566533  
## humor 1.0495880  
## hurt 1.0464935  
## likeabl 1.0347672  
## holland 1.0271858  
## awesom 0.9721958  
## crazi 0.9458343  
## superhero 0.9408893

predTest <- predict(rf\_model, test\_dfm , type = "class")  
rf\_cmat <- table(predTest, test\_set$Label)   
rf\_cmat

##   
## predTest DC Marvel  
## DC 533 49  
## Marvel 0 218

rf\_acc <- sum(diag(rf\_cmat))/sum(rf\_cmat) # accuracy = (TP + TN) / (TP + FP + TN + FN)  
rf\_recall <- rf\_cmat[2,2]/sum(rf\_cmat[2,]) # recall = TP / (TP + FN)  
rf\_precision <- rf\_cmat[2,2]/sum(rf\_cmat[,2]) # precision = TP / (TP + FP)  
rf\_f1 <- 2\*(rf\_recall\*rf\_precision )/(rf\_recall + rf\_precision)  
  
baseline\_acc <- max(prop.table(table(test\_set$Label)))  
  
cat(  
 "Baseline Accuracy: ", baseline\_acc, "\n",  
 "Accuracy:", rf\_acc, "\n",  
 "Recall:", rf\_recall, "\n",  
 "Precision:", rf\_precision, "\n",  
 "F1-score:", rf\_f1  
)

## Baseline Accuracy: 0.66625   
## Accuracy: 0.93875   
## Recall: 1   
## Precision: 0.8164794   
## F1-score: 0.8989691

4.2 To see if only using the positive or negative review, can better or worse distinguish the two kinds of movie. Positive/negative review are defined by above or below the medium score.

#setting pos/nega, and seperating them for both movie kinds  
all\_median <- median(all\_movies$stars)  
all\_movies['pos/neg'] <- ifelse(all\_movies$stars>all\_median , 'positive',  
 ifelse(all\_movies$stars<=all\_median , 'negative','n/a'))  
prop.table(table(all\_movies['pos/neg']))

##   
## negative positive   
## 0.5199039 0.4800961

mv\_ <- all\_movies[all\_movies$Label == 'Marvel',]  
prop.table(table(mv\_['pos/neg']))

##   
## negative positive   
## 0.4514302 0.5485698

dc\_ <- all\_movies[all\_movies$Label == 'DC',]  
prop.table(table(dc\_['pos/neg']))

##   
## negative positive   
## 0.5558252 0.4441748

pos\_movies <-all\_movies[all\_movies['pos/neg'] == 'positive',]  
neg\_movies <-all\_movies[all\_movies['pos/neg'] == 'negative',]  
  
  
# using only positives reviews in Random forest  
  
movies\_rf <- pos\_movies[sample(1:1000),]  
prop\_train <- 0.8  
index <- 1:nrow(movies\_rf)  
train\_index <- sample(index, ceiling(prop\_train\*length(index)), replace = FALSE)  
test\_index <- index[train\_index]  
train\_set <- movies\_rf[train\_index,]  
test\_set <- movies\_rf[test\_index,]  
train\_dfm <- dfm(train\_set$review, stem = TRUE, remove\_punct = TRUE, remove\_numbers = TRUE,remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token))  
test\_dfm <- dfm(test\_set$review, stem = TRUE, remove\_punct = TRUE, remove\_numbers = TRUE,remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token))  
test\_dfm <- dfm\_match(test\_dfm, features = featnames(train\_dfm))  
train\_dfm <- train\_dfm %>% convert("matrix")  
test\_dfm <- test\_dfm%>% convert("matrix")  
train\_set$Label <- as.factor(train\_set$Label)  
rf\_model<- randomForest(train\_dfm,train\_set$Label , importance=TRUE)  
rf\_model

##   
## Call:  
## randomForest(x = train\_dfm, y = train\_set$Label, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 65  
##   
## OOB estimate of error rate: 27.88%  
## Confusion matrix:  
## DC Marvel class.error  
## DC 441 39 0.08125  
## Marvel 184 136 0.57500

importance <- importance(rf\_model)  
import\_df<- as.data.frame(importance)  
import\_df[order(-import\_df$MeanDecreaseGini),][1:30,]

## DC Marvel MeanDecreaseAccuracy MeanDecreaseGini  
## mcu 28.9364273 17.973020 26.9010607 12.326824  
## univers 19.9348428 4.852438 16.9942220 3.254413  
## robert 13.6444787 7.185223 12.3980777 2.821839  
## film 0.9180799 4.741798 4.4648240 2.160481  
## best 7.8829818 -2.878854 5.5214850 2.131198  
## marvel 11.9122915 1.249142 8.9776963 1.972983  
## downey 11.1353867 5.760630 10.3223369 1.953260  
## charact 12.6159564 -7.415956 9.9040187 1.734362  
## superhero 8.1123661 -3.058920 4.8418959 1.725687  
## predict 11.4389977 5.359071 9.9217386 1.679607  
## joker 5.5560164 10.487576 9.0321682 1.669459  
## villain 14.1213300 -6.112685 10.7571879 1.644157  
## dc 6.8293534 7.837236 8.6065933 1.595433  
## second 10.0704655 1.966168 9.1973576 1.465933  
## one 7.8535381 -4.218171 6.2396116 1.465267  
## far 10.6920782 -2.079919 7.9588963 1.459469  
## trilog 6.1355030 8.361495 8.7740633 1.375804  
## good 2.0115659 -2.237442 -0.2089279 1.344491  
## jr 10.0105168 4.341880 8.6298106 1.344154  
## great -2.6637582 1.416891 -1.0552151 1.328119  
## cumberbatch 8.6532594 6.666563 8.7354722 1.302792  
## cast 10.2494309 -4.032283 7.6659521 1.255261  
## studio 8.9581050 4.482580 8.2119139 1.241325  
## start 7.5039640 -1.823434 5.1655038 1.226372  
## effect 12.7725563 -7.261391 6.6983707 1.199396  
## adapt 1.2252326 6.649910 4.6377168 1.189738  
## rudd 9.7307506 2.660515 8.0041703 1.159687  
## action 4.0277657 -4.968480 -0.2217326 1.148612  
## holland 8.0445865 2.804434 6.4459725 1.139986  
## anim 5.0068237 7.811433 7.6201630 1.135642

predTest <- predict(rf\_model, test\_dfm , type = "class")  
rf\_cmat <- table(predTest, test\_set$Label)   
rf\_cmat

##   
## predTest DC Marvel  
## DC 479 48  
## Marvel 1 272

rf\_acc <- sum(diag(rf\_cmat))/sum(rf\_cmat) # accuracy = (TP + TN) / (TP + FP + TN + FN)  
rf\_recall <- rf\_cmat[2,2]/sum(rf\_cmat[2,]) # recall = TP / (TP + FN)  
rf\_precision <- rf\_cmat[2,2]/sum(rf\_cmat[,2]) # precision = TP / (TP + FP)  
rf\_f1 <- 2\*(rf\_recall\*rf\_precision )/(rf\_recall + rf\_precision)  
  
baseline\_acc <- max(prop.table(table(test\_set$Label)))  
  
cat(  
 "Baseline Accuracy: ", baseline\_acc, "\n",  
 "Accuracy:", rf\_acc, "\n",  
 "Recall:", rf\_recall, "\n",  
 "Precision:", rf\_precision, "\n",  
 "F1-score:", rf\_f1  
)

## Baseline Accuracy: 0.6   
## Accuracy: 0.93875   
## Recall: 0.996337   
## Precision: 0.85   
## F1-score: 0.9173693

# using only negative reviews in Random forest  
  
movies\_rf <- neg\_movies[sample(1:1000),]  
prop\_train <- 0.8  
index <- 1:nrow(movies\_rf)  
train\_index <- sample(index, ceiling(prop\_train\*length(index)), replace = FALSE)  
test\_index <- index[train\_index]  
train\_set <- movies\_rf[train\_index,]  
test\_set <- movies\_rf[test\_index,]  
train\_dfm <- dfm(train\_set$review, stem = TRUE, remove\_punct = TRUE, remove\_numbers = TRUE,remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token))  
test\_dfm <- dfm(test\_set$review, stem = TRUE, remove\_punct = TRUE, remove\_numbers = TRUE,remove = c(stopwords("english"), Movielist\_DC\_token, Movielist\_MV\_token))  
test\_dfm <- dfm\_match(test\_dfm, features = featnames(train\_dfm))  
train\_dfm <- train\_dfm %>% convert("matrix")  
test\_dfm <- test\_dfm%>% convert("matrix")  
train\_set$Label <- as.factor(train\_set$Label)  
rf\_model<- randomForest(train\_dfm,train\_set$Label , importance=TRUE)  
rf\_model

##   
## Call:  
## randomForest(x = train\_dfm, y = train\_set$Label, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 67  
##   
## OOB estimate of error rate: 23.38%  
## Confusion matrix:  
## DC Marvel class.error  
## DC 561 18 0.03108808  
## Marvel 169 52 0.76470588

importance <- importance(rf\_model)  
import\_df<- as.data.frame(importance)  
import\_df[order(-import\_df$MeanDecreaseGini),][1:30,]

## DC Marvel MeanDecreaseAccuracy MeanDecreaseGini  
## mcu 25.8759699 20.7344003 25.1759296 9.1201271  
## overr 6.9902618 11.5749067 10.6181234 3.0202962  
## stark 14.1405822 9.0017205 13.2558234 2.9582030  
## sequel 16.0953145 5.9966639 14.1560950 2.4487194  
## toni 12.2135534 6.4273773 11.4656638 2.1204417  
## amaz 11.6214244 5.9335423 10.4889609 1.9783348  
## hype 6.6529255 7.8886376 8.5282303 1.7814915  
## norton 11.5377126 10.3572171 11.8315997 1.7595783  
## bore 9.1247007 0.1108950 7.4153176 1.4762834  
## promis 8.1640661 0.5688166 6.4045174 1.3390305  
## great 4.0901378 0.6510427 4.0515287 1.2594230  
## overhyp 5.7441719 2.4085090 5.1901486 1.2351436  
## dork 0.0000000 0.0000000 0.0000000 1.2239407  
## holland 8.5197771 4.5630168 7.7899136 1.1935279  
## good -0.5003307 2.0633876 0.5453843 1.1605512  
## loki 9.6957563 2.7413528 8.5840390 1.1270634  
## place 11.3691593 -1.4424555 8.6853981 1.0916280  
## okay 2.5804773 -0.3003699 1.8376744 1.0368472  
## decent 6.6331208 -0.2050076 5.1298669 1.0080460  
## shane 0.0000000 0.0000000 0.0000000 1.0001715  
## hurt 8.6519176 -5.2684475 4.6544017 0.9599620  
## chapter 7.1285657 3.4432870 6.5004291 0.9542588  
## els 9.6522787 -4.3123135 6.6328543 0.9493338  
## new 8.6426439 -3.2462390 6.1063240 0.9459348  
## phase 7.2771591 5.1629903 7.2471985 0.9282519  
## seri 4.0983069 6.6767110 6.6098620 0.9154337  
## best 4.4706723 -1.1860380 3.5387710 0.9070026  
## film 2.8309955 0.8551990 2.9317406 0.9027094  
## generic 6.1870136 3.1916934 5.7743867 0.8905197  
## plot 6.2316693 -2.8786798 4.0609697 0.8832892

predTest <- predict(rf\_model, test\_dfm , type = "class")  
rf\_cmat <- table(predTest, test\_set$Label)   
rf\_cmat

##   
## predTest DC Marvel  
## DC 578 41  
## Marvel 1 180

rf\_acc <- sum(diag(rf\_cmat))/sum(rf\_cmat) # accuracy = (TP + TN) / (TP + FP + TN + FN)  
rf\_recall <- rf\_cmat[2,2]/sum(rf\_cmat[2,]) # recall = TP / (TP + FN)  
rf\_precision <- rf\_cmat[2,2]/sum(rf\_cmat[,2]) # precision = TP / (TP + FP)  
rf\_f1 <- 2\*(rf\_recall\*rf\_precision )/(rf\_recall + rf\_precision)  
  
baseline\_acc <- max(prop.table(table(test\_set$Label)))  
  
cat(  
 "Baseline Accuracy: ", baseline\_acc, "\n",  
 "Accuracy:", rf\_acc, "\n",  
 "Recall:", rf\_recall, "\n",  
 "Precision:", rf\_precision, "\n",  
 "F1-score:", rf\_f1  
)

## Baseline Accuracy: 0.72375   
## Accuracy: 0.9475   
## Recall: 0.9944751   
## Precision: 0.8144796   
## F1-score: 0.8955224