> setwd("~/Desktop/final project")

> ##################### setup environment ######################

>

> # setup libraries

> if (!require(lattice)) {install.packages("lattice"); library(lattice)}

Loading required package: lattice

> if (!require(NLP)) {install.packages("NLP"); library(NLP)}

Loading required package: NLP

Warning message:

package 'NLP' was built under R version 3.3.2

> if (!require(topicmodels)) {install.packages("topicmodels"); library(topicmodels)}

Loading required package: topicmodels

Warning message:

package 'topicmodels' was built under R version 3.3.2

> if (!require(tm)) {install.packages("tm"); library(tm)}

Loading required package: tm

Warning message:

package 'tm' was built under R version 3.3.2

> if (!require(slam)) {install.packages("slam"); library(slam)}

Loading required package: slam

Warning message:

package 'slam' was built under R version 3.3.2

>

> ##################### input the data ######################

>

> ## read in the data

>

> library(readr)

Warning message:

package 'readr' was built under R version 3.3.2

> yelp.raw = read.csv("yelp\_academic\_dataset\_review\_train.csv", header = TRUE)

> dim(yelp.raw)

[1] 116474 11

> set.seed(123)

> rs = sample(1:116474, size = 116474/4, replace = F)

> yelp = yelp.raw[-rs, ]

> test = yelp.raw[rs, ]

>

> ## make modifications to the dataset

>

> # copy of the data

> #yelp = read.csv("data.csv")

>

> # change data formats

> yelp$business\_id=as.factor(yelp$business\_id) # change from strings to factors -- easier to find common businesses

> yelp$user\_id=as.factor(yelp$user\_id) # change from strings to factors

>

> ## transform customer reviews into a tm structure that can be used for topic modeling

>

> # first turn the reviews which are strings into words (or tokens)

> creviews = VCorpus(VectorSource(yelp$text))

> meta(creviews,"business\_id")=yelp$business\_id # save the business\_id

> meta(creviews,"user\_id")=yelp$user\_id # save the user\_id

> writeLines(as.character(creviews[1])) # print specific review

list(list(content = "I almost got sick on all of the chocolate coffee beans in this place, they are so good. I sat outside and had a great conversation about the direction music is taking in this day in age. If you don't follow a formula it seems that you can't get a record contract, even in the indie scene. As you can tell, I thoroughly enjoyed the crowd here. \n\nInside there is a bar opposite the coffee line with Creative Loafing and other local newspapers to read, as well as chocolate coffee beans in little quarter machines. Don't mix these with a strong cup of coffee or you will be jumping off the walls.",

meta = list(author = character(0), datetimestamp = list(sec = 22.4144179821014, min = 5, hour = 6, mday = 4, mon = 4, year = 117, wday = 4, yday = 123, isdst = 0), description = character(0), heading = character(0), id = "1", language = "en", origin = character(0))))

list()

list(business\_id = 613, user\_id = 44868)

>

> # second let's modify the reviews (removal of certain words)

> creviews = tm\_map(creviews,stripWhitespace) # remove extra whitespace

> creviews = tm\_map(creviews,content\_transformer(tolower)) # convert all to lower case

> creviews = tm\_map(creviews,removeWords,stopwords("english")) # remove common english words

> creviews = tm\_map(creviews,removeNumbers) # remove numbers

> creviews = tm\_map(creviews,removePunctuation) # remove symbols

> writeLines(as.character(creviews[1])) # print specific review after transforms

list(list(content = " almost got sick chocolate coffee beans place good sat outside great conversation direction music taking day age follow formula seems get record contract even indie scene can tell thoroughly enjoyed crowd inside bar opposite coffee line creative loafing local newspapers read well chocolate coffee beans little quarter machines mix strong cup coffee will jumping walls", meta = list(author = character(0), datetimestamp = list(

sec = 22.4144179821014, min = 5, hour = 6, mday = 4, mon = 4, year = 117, wday = 4, yday = 123, isdst = 0), description = character(0), heading = character(0), id = "1", language = "en", origin = character(0))))

list()

list(business\_id = 613, user\_id = 44868)

>

> # third let's create a term-document matrix (e.g., count # of times word used)

> tmcreviews = DocumentTermMatrix(creviews)

> dim(tmcreviews) # notice that we have one row for each user review, and one column for each word

[1] 87356 69809

> inspect(tmcreviews[1:10,c("amazing","happy","indian","food")]) # print out word counts for the first 10 users for four specific words

<<DocumentTermMatrix (documents: 10, terms: 4)>>

Non-/sparse entries: 3/37

Sparsity : 92%

Maximal term length: 7

Weighting : term frequency (tf)

Sample :

Terms

Docs amazing food happy indian

1 0 0 0 0

10 1 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

5 0 0 0 0

6 0 1 0 0

7 0 0 0 1

8 0 0 0 0

9 0 0 0 0

>

> # fourth let's only get the words that are common (alternative could use removeSparseTerms(tmreviews,0.2))

> shorttermlist=findFreqTerms(tmcreviews,500) # let's find words that are used 500 times or more

> mcterms = tmcreviews[,shorttermlist] # let's create another term matrix with just common words

> dim(mcterms) # notice now we have the same number of rows but only 1984 columns (just frequent words)

[1] 87356 1298

> wordcount=apply(mcterms,1,sum) # count the number words in reviews (the '1' tells R to sum each row or user)

> summary(wordcount) # summary of the users

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.00 17.00 29.00 38.82 50.00 353.00

> wordcount.index = wordcount>10

> mcterms = mcterms[wordcount.index,] # let's only keep user reviews that have at least 10 words

> dim(mcterms) # notice that now we only have 109929 reviews

[1] 79984 1298

> y = yelp$stars[wordcount.index]

> mcterms = as.matrix(mcterms)

> mcterms = as.data.frame(mcterms)

> mcterms = cbind(mcterms, y)

> dim(mcterms)

[1] 79984 1299

>

> #shrink mcterms

> rs1 = sample(1:79984, size = 79984/2, replace = F)

> mcterms.shrink= mcterms[-rs1, ]

> dim(mcterms.shrink)

[1] 39992 1299

>

>

> #boosting

> library(gbm)

Loading required package: survival

Loading required package: splines

Loading required package: parallel

Loaded gbm 2.1.3

Warning message:

package 'gbm' was built under R version 3.3.2

> gbmModel.adaboost.depth.two<-gbm(y~.,data=mcterms,

+ interaction.depth=9,n.trees=100,shrinkage=.01,verbose=FALSE)

Distribution not specified, assuming gaussian ...

>

> test$business\_id=as.factor(test$business\_id)

> test$user\_id=as.factor(test$user\_id)

>

> test.creviews = VCorpus(VectorSource(test$text))

> meta(test.creviews,"business\_id")=test$business\_id # save the business\_id

> meta(test.creviews,"user\_id")=test$user\_id # save the user\_id

> writeLines(as.character(test.creviews[1])) # print specific review

list(list(content = "So me and the kids stop here on a layover. I should have read the previous reviews. You know you are in financial trouble when a cheese quesadilla is $7.39. My daughter gets street tacos, conveniently made from tortillas the size of a hockey puck. I have a burrito that is 80% refried beans and tortilla. Spanish rice is dry and flavorless. Aforementioned refritos are a watery paste, seemingly acquired from the same taste factory as the rice. Lady calling the orders as ready appears to have been a voice actor in the upcoming \"Unintelligible Helium Chihuahua Power Rangers\" release. The tortillas are decent, but everything else was a bust.",

meta = list(author = character(0), datetimestamp = list(sec = 50.8385519981384, min = 27, hour = 6, mday = 4, mon = 4, year = 117, wday = 4, yday = 123, isdst = 0), description = character(0), heading = character(0), id = "1", language = "en", origin = character(0))))

list()

list(business\_id = 1084, user\_id = 70584)

>

> test.creviews = tm\_map(test.creviews,stripWhitespace) # remove extra whitespace

> test.creviews = tm\_map(test.creviews,content\_transformer(tolower)) # convert all to lower case

> test.creviews = tm\_map(test.creviews,removeWords,stopwords("english")) # remove common english words

> test.creviews = tm\_map(test.creviews,removeNumbers) # remove numbers

> test.creviews = tm\_map(test.creviews,removePunctuation) # remove symbols

> writeLines(as.character(test.creviews[1]))

list(list(content = " kids stop layover read previous reviews know financial trouble cheese quesadilla daughter gets street tacos conveniently made tortillas size hockey puck burrito refried beans tortilla spanish rice dry flavorless aforementioned refritos watery paste seemingly acquired taste factory rice lady calling orders ready appears voice actor upcoming unintelligible helium chihuahua power rangers release tortillas decent everything else bust",

meta = list(author = character(0), datetimestamp = list(sec = 50.8385519981384, min = 27, hour = 6, mday = 4, mon = 4, year = 117, wday = 4, yday = 123, isdst = 0), description = character(0), heading = character(0), id = "1", language = "en", origin = character(0))))

list()

list(business\_id = 1084, user\_id = 70584)

>

> test.tmcreviews = DocumentTermMatrix(test.creviews)

> dim(test.tmcreviews) # notice that we have one row for each user review, and one column for each word

[1] 29118 37981

> inspect(test.tmcreviews[1:10,c("amazing","happy","indian","food")])

<<DocumentTermMatrix (documents: 10, terms: 4)>>

Non-/sparse entries: 8/32

Sparsity : 80%

Maximal term length: 7

Weighting : term frequency (tf)

Sample :

Terms

Docs amazing food happy indian

1 0 0 0 0

10 0 1 0 0

2 0 2 0 0

3 0 1 0 0

4 0 1 0 0

5 0 1 0 0

6 0 0 0 0

7 0 1 0 0

8 0 1 0 0

9 0 3 0 0

>

> test.mcterms = test.tmcreviews[, shorttermlist] # let's create another term matrix with just common words

> dim(test.mcterms) # notice now we have the same number of rows but only 1984 columns (just frequent words)

[1] 29118 1298

> #test.wordcount=apply(test.mcterms,1,sum) # count the number words in reviews (the '1' tells R to sum each row or user)

> #summary(test.wordcount) # summary of the users

> #test.wordcount.index = test.wordcount>10

> #test.mcterms = test.mcterms[test.wordcount.index,] # let's only keep user reviews that have at least 10 words

> #dim(test.mcterms)

> test.mcterms = as.data.frame(as.matrix(test.mcterms))

>

> #test.business\_id = test$business\_id[test.wordcount.index]

>

> gbm.Pred<-predict(gbmModel.adaboost.depth.two,newdata=test.mcterms,

+ type="response",n.trees=100)

>

> ###user.stars.regularize is a function to regularize individual user reviews to [-0.5, 6.5] range.

> #The range size can be adjusted later.

> user.stars.regularize = function(x) {if(x >= 6.5){return(6.5)} else if (x <= -0.5) {return(-0.5)} else {return(x)}}

> prediction.regularized = apply(as.matrix(gbm.Pred), user.stars.regularize, MARGIN = 1)

>

> #business.test = read.csv("yelp\_academic\_dataset\_business\_test.csv", header = TRUE)

>

> length(prediction.regularized)

[1] 29118

> #length(test.business\_id)

>

> error = abs(test$stars - prediction.regularized)

>

> sum(error<=0.5)

[1] 7817