Jed Isom

Data Mining Capstone

September 20, 2015

## Overview

The University of Illinois – Urbana Champaign Data Mining Specialization is utilizing a real world data set from Yelp! to teach it’s students how to analyze data. This data has information about the businesses, the reviews for the businesses, the users who rate businesses, etc.

## Task 2.1

This task was about creating a simplistic similarity (or dissimilarity) matrix and visualization for the different cuisines in the data set. There are lots of different ways that similarity could be calculated based on this data set. Since this was the first part of the assignment and I saw it as a “warm-up”, I decided to use a simple term frequency vector space model to measure the angle between the term vectors for each cuisine. I am trying to do as much of this course work in the R statistical software package as possible, so I’m including my R scripts in the Appendices (see the R script in Appendix I for the script for task 2.1)

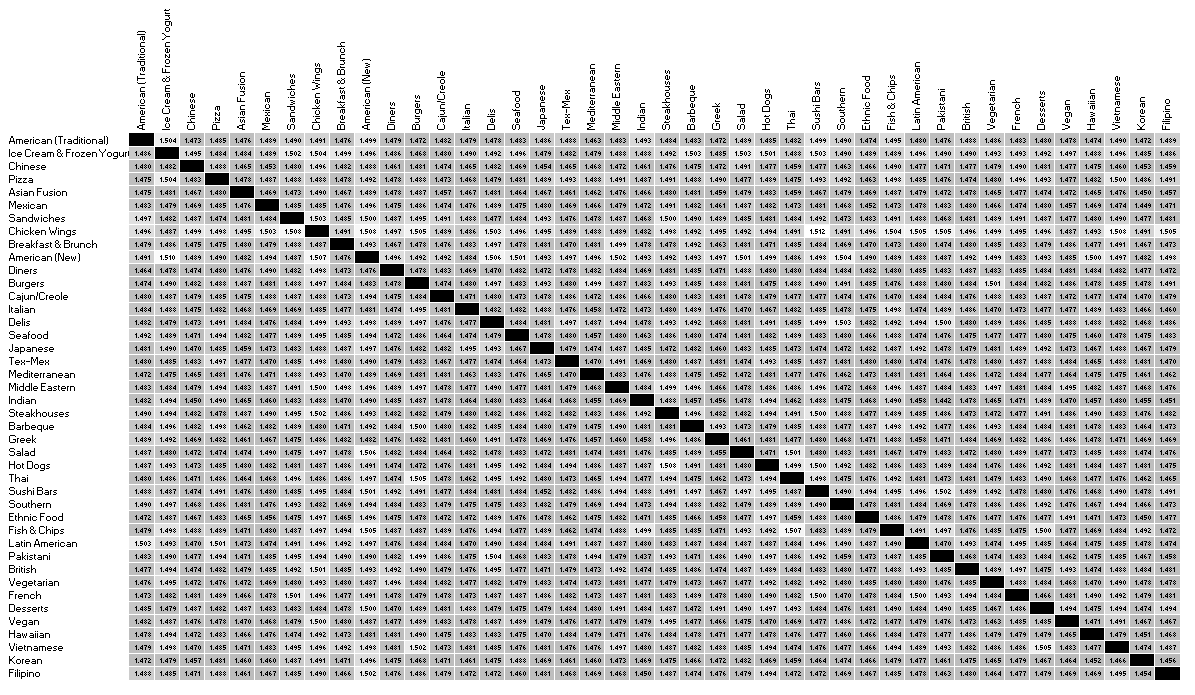
*Calculate the topic model term posterior probabilities*

1. Read in the JSON file into a variable called “review” and “business”
2. Filter the business to only get businesses that are “Restaurants”, and get rid of categories that either aren’t really cuisines, or have <50 reviews
3. Get a random sampling of all of the remaining cuisines (this is due to the limitations of the machine that I used to run this analysis).
4. Create an R corpus from these reviews
5. Scrub/clean up the text
   1. Remove numbers
   2. Turn all characters to lower case
   3. Remove stop words
   4. Remove punctuation
   5. Take out any white space
   6. Use word stemming to combine words with similar roots
6. Turn the corpus into a “DocumentTermMatrix” (this is a variable class in the tm package in R) that lists documents in rows and the words/terms in the columns
7. Compare all of the cuisines to each other by doing the following
   1. Pick one of the one of the 50 reviews from 2 cuisines
   2. Use the data from the DocumentTermMatrix to calculate an angle between the 2 reviews using the formula
   3. Return to step a for 50 other random samples and repeat
   4. Take the average of the angles between the 50 samples
   5. Continue on to another pair of cuisines to measure…
8. Output the data as a dissimilarity matrix in a .csv file to be visualized

*Create the visualization*

1. Open the .csv file in Excel
2. Use conditional formatting so that angle of 0 (radians) is black, and the maximum angle measured equals white. A medium gray color was also selected to equal the median value in the data set

Here is how the visualization for task 2.1 turned out:

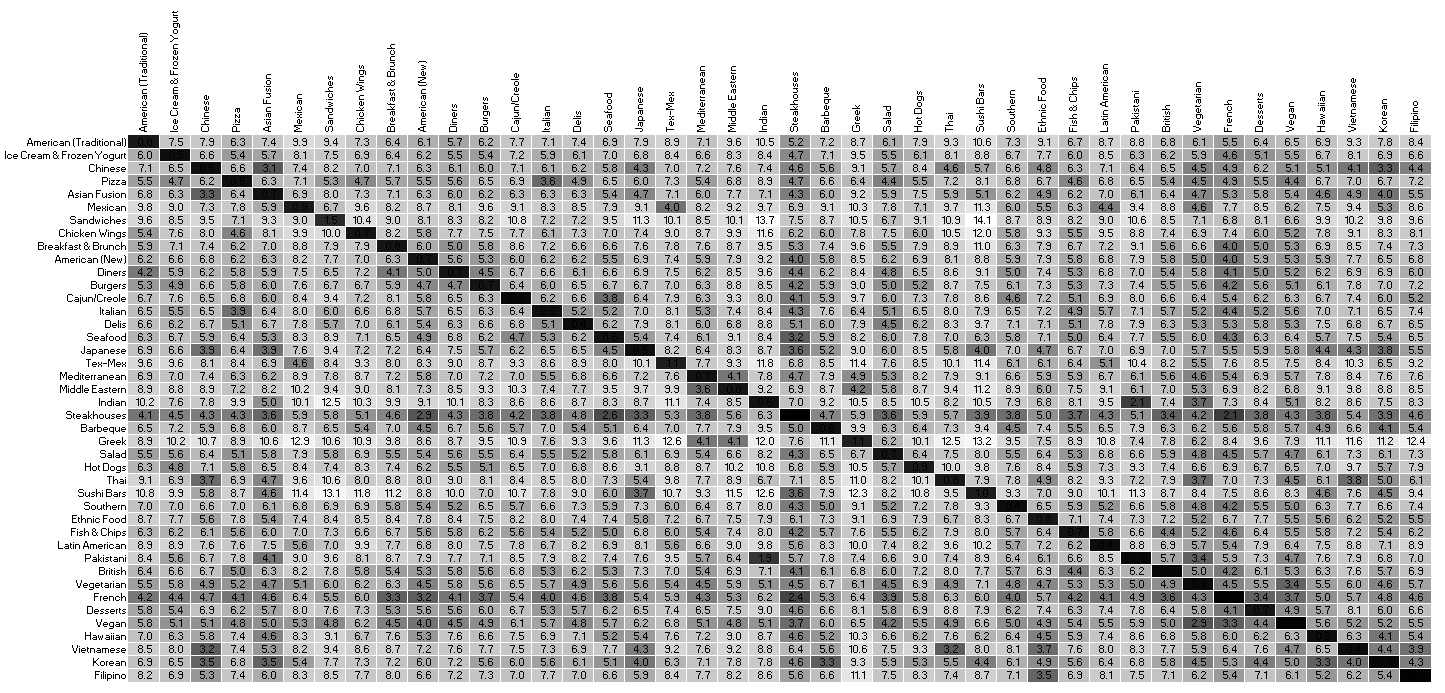


The first thing that I noticed with this was that there didn’t seem to be much distinction between all of the different cuisines measured (it all looks pretty gray other than the diagonals). This is probably happened for 2 reasons (1) the simple VSM distance measure I used and (2) the random sampling and averaging of the distances/angles of reviews. I knew I could improve on this in task 2.2 so I took this as a good first step and decided to move on.

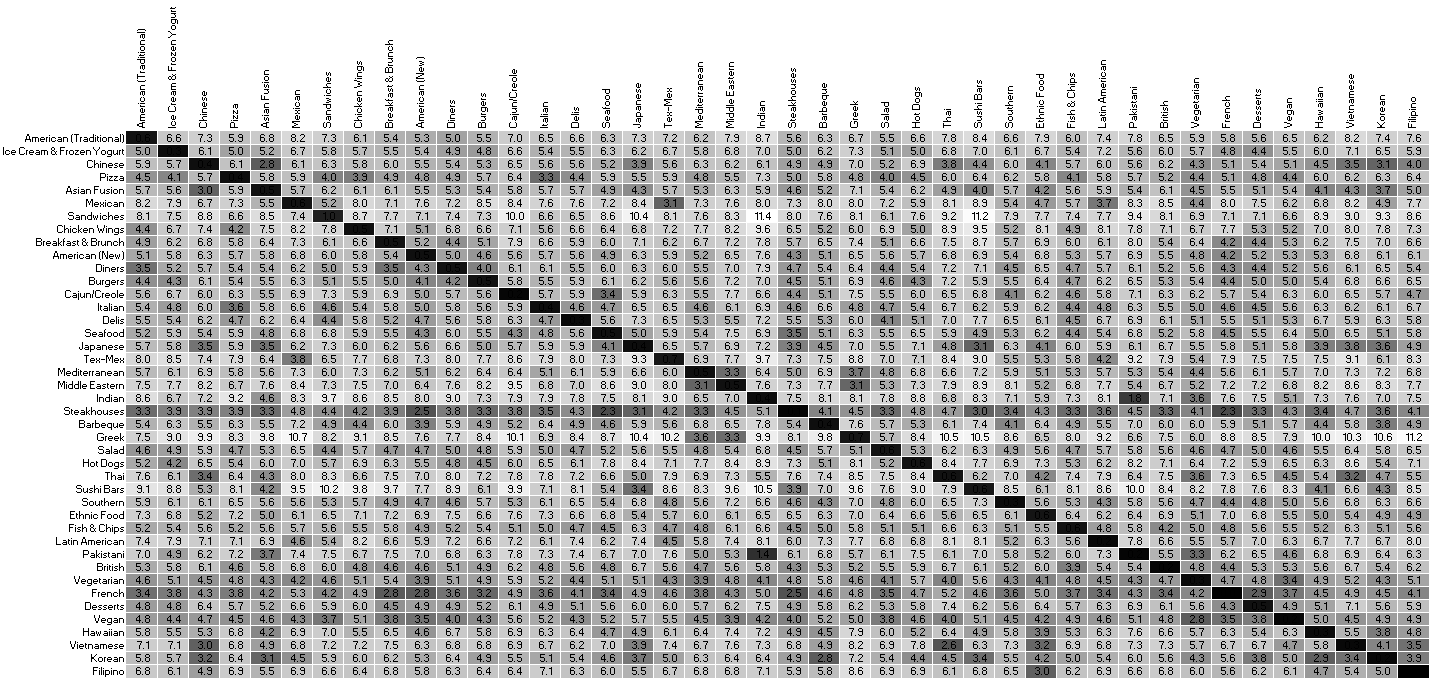
## Task 2.2

For task 2.2, I decided to vary the text representation by upgrading the vector space model to being a Pivoted Length Normalization (PLM) vector space model that would allow me to change the ‘b’ constant that rewards/punishes document length. This method treats the first cuisine as the “query” and the 2nd cuisine as the “document” to be scored. Because of this the score for f(q,d) is not the same for f(d,q). To compensate for this, I averaged the 2 so that the similarity matrix is still symmetric. I also went away from the method of randomly matching reviews of each cuisine to measure the difference between them and instead concatenated the reviews from each cuisine and compared that longer review to the other cuisines. Once I had accomplished this, I varied the ‘b’ parameter in the PLM model to see what appeared to give the most differentiation between cuisines. The code for all of this is in Appendix II

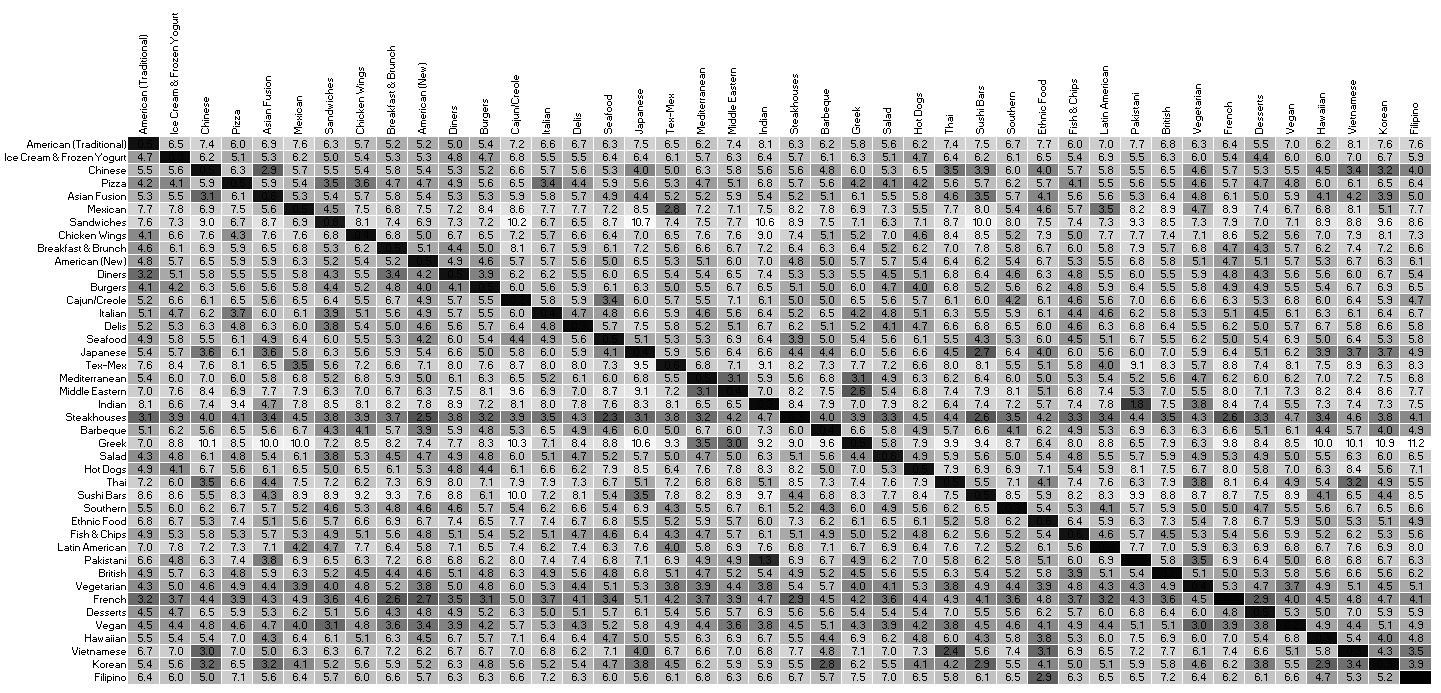
The next 3 visualizations (made in Excel) are all with the concatenated reviews for cuisines, but with varying values of ‘b’. Based on my review of the visualization, I decided to stick with ‘b’ = 0.9, because it appeared to have more variation in color suggesting differences in cuisine “closeness”.



*Task 2.2 similarity matrix (b=0.0)*



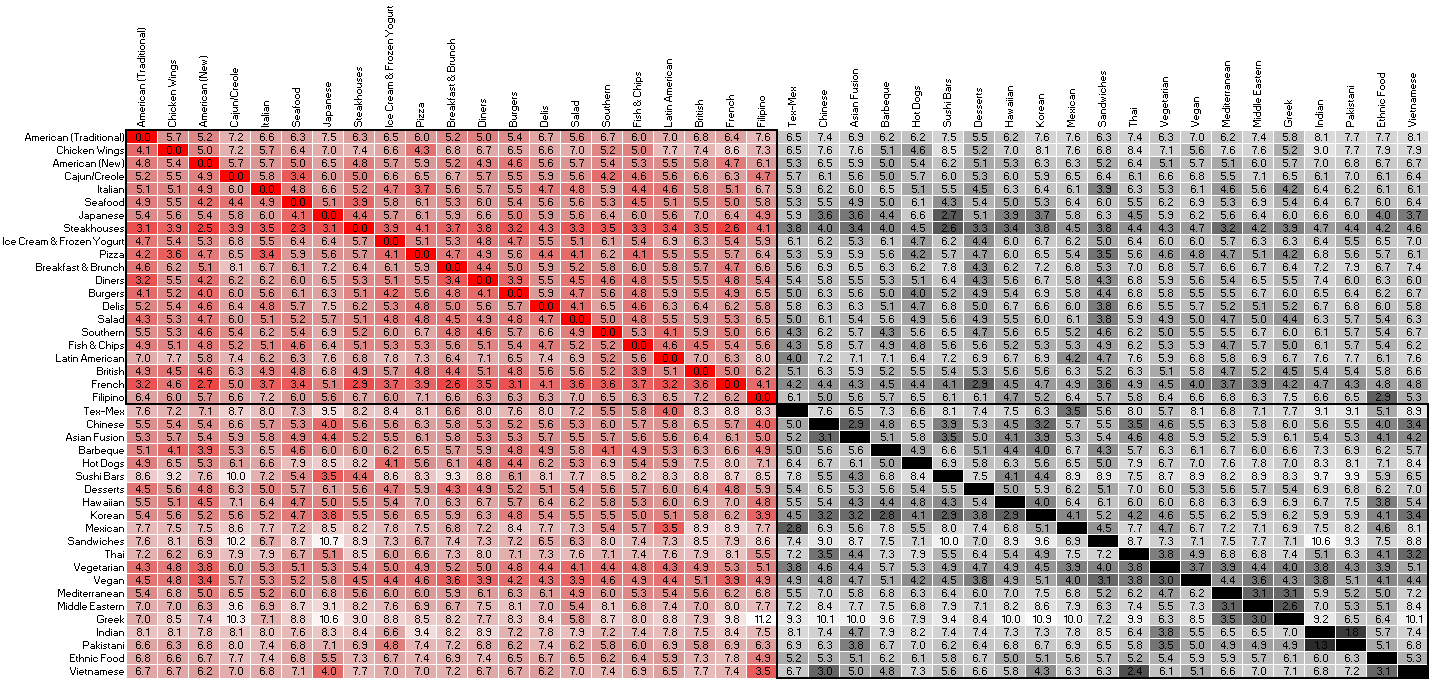
*Task 2.2 similarity matrix (b=0.5)*



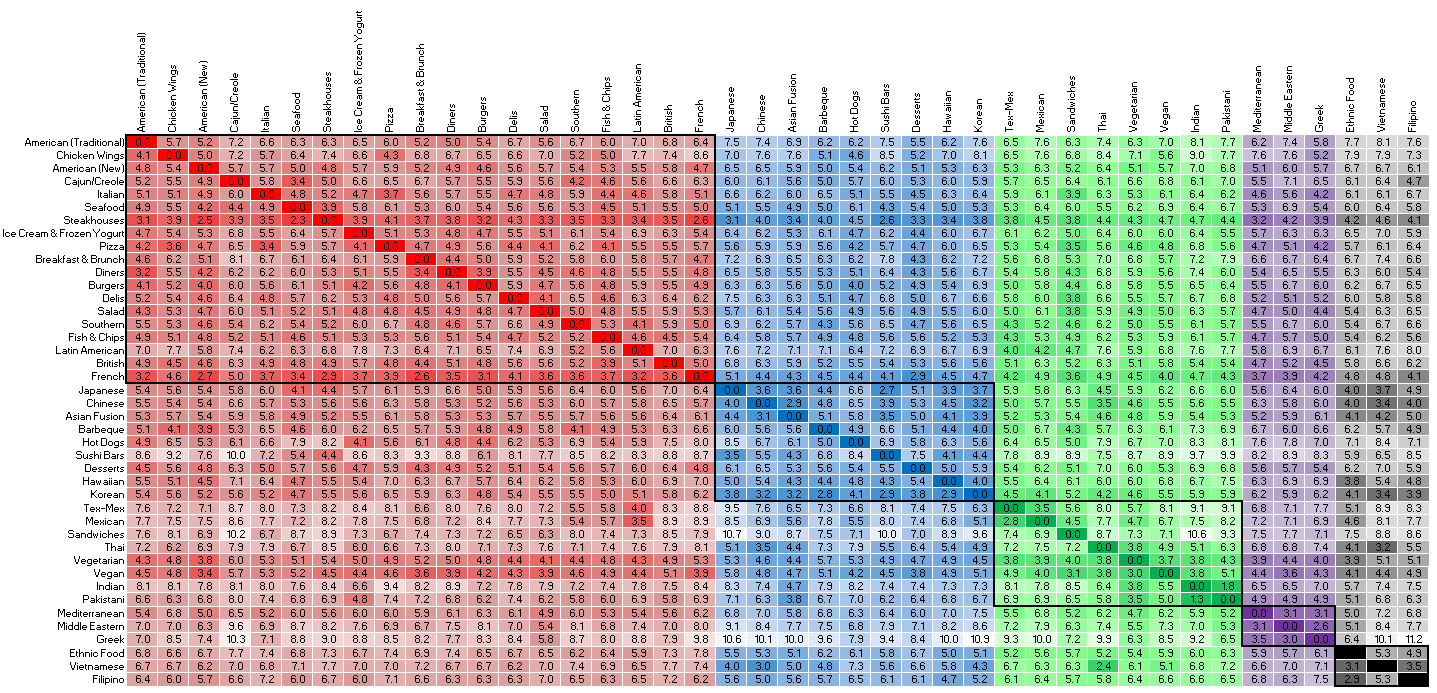
*Task 2.2 similarity matrix (b=0.9)*

## Task 2.3

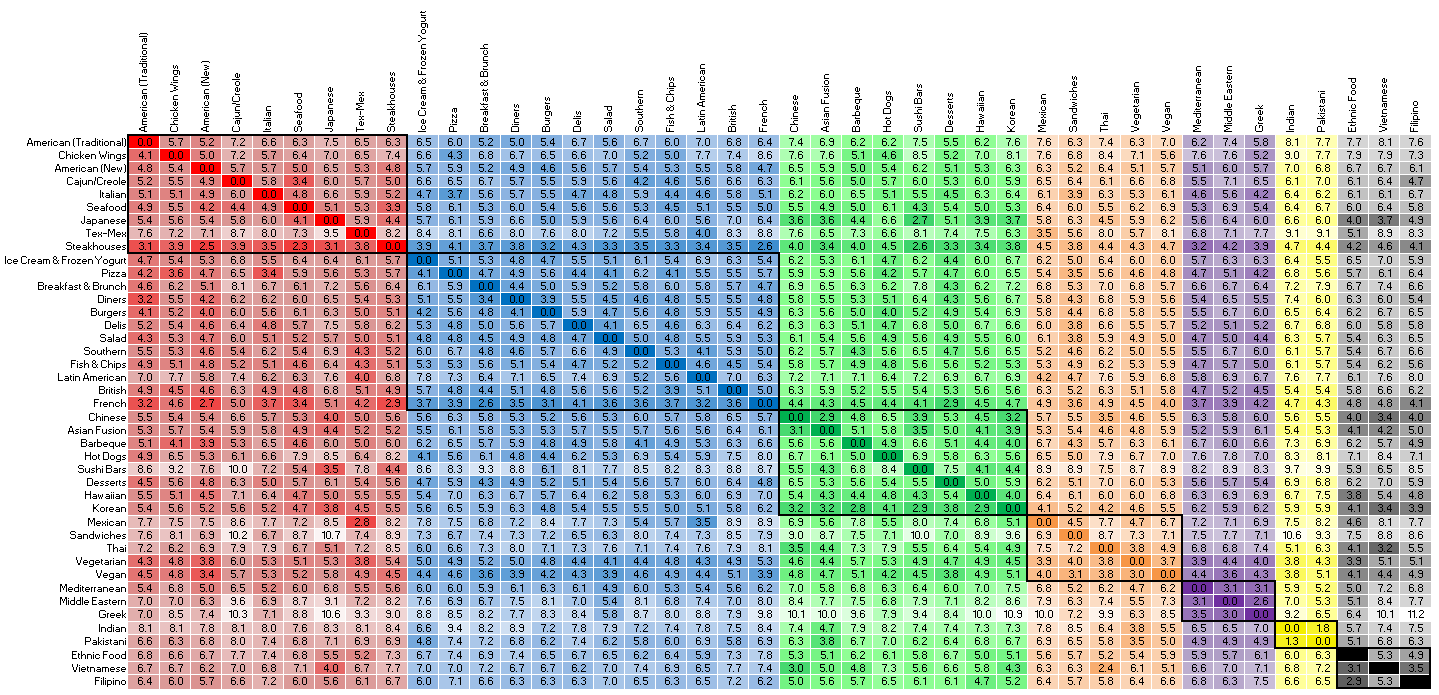
For the next section of this assignment, we’re supposed to cluster the cuisines to see which cuisines are similar to each other based on their reviews. One of the trickiest parts of clustering is often selecting how many clusters to choose. A trick I learned in a market segmentation class in business school was to use agglomerative clustering to figure out where there are big jumps in distances between clusters, and then use that information to pick a good number of clusters. I used that trick here, but before I could I had to transform the similarity matrix from task 2.2 with b=0.9 into a distance matrix by dividing all the values by the maximum value and then subtracting 1 from all of the data points. I used the “agnes” agglomeration clustering algorithm and saw that there were some big jumps in “height” (that’s what the agnes algorithm outputs) for clusters of 2, 5, and 7. Anything bigger than that gets hard to visualize due to color limitations. Once I had picked the number of clusters, I used the “pam” algorithm in R which is essentially a k-medoids algorithm. The results from k=2, k=5, and k=7 are shown below (again these visualizations were made in Excel). You can see that k=2 is not very interesting and basically just splits the cuisines in half. K=5 gets more interesting; you can see some very specific clusters near the right side (“Mediterranean” and “Ethnic”). K=7 appears to be the best clustering with “Mediterranean”, “Indian” and “Ethnic” being very distinct on the right side. There are also some good clusters of Asian style foods and vegetarian cuisines.



*Task 2.2 similarity matrix (b=0.9, k=2)*



*Task 2.2 similarity matrix (b=0.9, k=5)*



*Task 2.2 similarity matrix (b=0.9, k=7)*

## Appendix I: Task 2.1.R

## ##Task 2.1

## #by Jed Isom

## #Week of September 13th, 2015

## #import the applicable JSON files

## library("pacman")

## pacman::p\_load(jsonlite, tm, topicmodels, slam)

## #jsonlite for JSON file loading

## #tm and topicmodels for LDA topic modeling

## rm(list=ls())

## json\_file <- "yelp\_academic\_dataset\_business.JSON"

## #took this line of code from http://stackoverflow.com/questions/26519455/error-parsing-json-file-with-the-jsonlite-package

## business <- fromJSON(sprintf("[%s]", paste(readLines(json\_file), collapse=",")))

## #this lists information about the businesses (location, hours, category, name, some attributes)

## json\_file <- "yelp\_academic\_dataset\_review.JSON"

## review <- fromJSON(sprintf("[%s]", paste(readLines(json\_file), collapse=",")))

## #list reviews by businesses

## #add boolean variable to business is.restaurant

## #quick and dirty code for this, but only takes a couple seconds

## for (i in 1:dim(business)[1]){

## business[i,"is.restaurant"] = is.element("Restaurants", business[i,"categories"][[1]])

## }

## #subset business dataframe for just restaurants

## business <- business[business[,"is.restaurant"]==TRUE,]

## #get only the restaurants that have a cuisine as a category (and remove low freq. cuisines...)

## non.cuisine.list <- c("Nightlife", "Lounges", "Party & Event Planning", "Event Planning & Services",

## "Venues & Event Spaces", "Active Life", "Bowling", "Beer, Wine & Spirits", "Grocery",

## "Meat Shops", "Dance Clubs", "Arts & Entertainment", "Music Venues", "Karaoke",

## "Shopping Centers", "Shopping", "Outlet Stores", "Golf", "Convenience Stores",

## "Drugstores", "Hotels & Travel", "Hotels", "Jazz & Blues", "Performing Arts", "Fashion",

## "Sporting Goods", "Sports Wear", "Cinema", "Pool Halls", "Arcades", "Casinos",

## "Health Markets", "Social Clubs", "Food Delivery Services", "Gift Shops",

## "Flowers & Gifts", "Health & Medical", "Hospitals", "Hookah Bars", "Amusement Parks",

## "Gas & Service Stations", "Automotive", "Adult Entertainment", "Beauty & Spas",

## "Gyms", "Medical Spas", "Fitness & Instruction", "Day Spas", "Taxis", "Transportation",

## "Auto Repair", "Colleges & Universities", "Education", "Specialty Schools",

## "Cooking Schools", "RV Parks", "Home Decor", "Home & Garden", "Kitchen & Bath",

## "Appliances", "Airports", "Tours", "Do-It-Yourself Food", "Cafeteria",

## "Swimming Pools", "Wineries", "Art Galleries", "Bed & Breakfast", "Arts & Crafts",

## "Landmarks & Historical Buildings", "Personal Shopping", "Public Services & Government",

## "Street Vendors", "Dry Cleaning & Laundry", "Local Services", "Festivals",

## "Farmers Market", "Internet Cafes", "Leisure Centers", "Kids Activities", "Car Wash",

## "Horseback Riding", "Butcher", "Country Dance Halls", "Cultural Center", "Delicatessen",

## "Home Services", "Real Estate", "Apartments", "Mass Media", "Print Media",

## "Food", "Fast Food", "Bars", "Bakeries", "Coffee & Tea", "Donuts", "Caterers",

## "Dive Bars", "Pubs", "Buffets", "Cafes", "Sports Bars", "Specialty Food",

## "Gluten-Free", "Wine Bars", "Comfort Food", "Bagels", "Gastropubs",

## "Juice Bars & Smoothies", "Breweries", "Pretzels", "Food Stands", "Island Pub",

## "Tapas Bars", "Cheese Shops", "Gay Bars", "Herbs & Spices", "Hot Pot", "Local Flavor",

## "Brasseries", "Shaved Ice", "Food Trucks", "Food Court", "Champagne Bars",

## "Bubble Tea", "Piano Bars", "Poutineries", "Beer Bar", "Distilleries", "Lebanese",

## "Soup", "Caribbean", "Tea Rooms", "Cheesesteaks", "Soul Food", "Salvadoran", "Kosher",

## "Polish", "Creperies", "Cuban", "Russian", "Irish", "Fruits & Veggies", "Fondue",

## "Arabian", "Seafood Markets", "Peruvian", "Halal", "Dim Sum", "Mongolian",

## "Persian/Iranian", "German", "Cantonese", "Taiwanese", "Argentine",

## "Himalayan/Nepalese", "Moroccan", "Falafel", "Ethiopian", "African", "Indonesian",

## "Turkish", "Afghan", "Tapas/Small Plates", "Basque", "Spanish", "Cocktail Bars",

## "Brazilian", "Personal Chefs", "Laotian", "Szechuan", "Belgian", "Gelato",

## "Live/Raw Food", "Bistros", "Chocolatiers & Shops", "Malaysian", "Singaporean",

## "Burmese", "Scandinavian", "Canadian (New)", "Czech", "Slovakian", "Scottish",

## "Modern European", "Bangladeshi", "Ramen", "Portuguese", "Ukrainian", "Shanghainese",

## "Cambodian", "Venezuelan", "Colombian", "Dominican", "Patisserie/Cake Shop",

## "Australian", "Egyptian")

## #create dataframe of cuisines and the business\_id's associated with them

## cat.bus <- as.data.frame(matrix(c(NA,NA),nrow=1,ncol=2))

## names(cat.bus) <- c("cuisine", "business\_id")

## cuisine.found <- FALSE

## for (i in 1:dim(business)[1]){ #go through all businesses

## temp <- as.list(business[i,"categories"][[1]]) #get list of categories for that busines

## if (length(temp)>=1) { #Make sure the list isn't empty

## for (j in 1:length(temp)){ #cycle through all categories for the business

## if ((temp[j] != "Restaurants")&(!is.element(temp[j],non.cuisine.list))){

## #filter out non-cuisines and the word "Restaurants"

## if (cuisine.found==FALSE){

## cuisine.found <- TRUE

## cat.bus[1,1] <- temp[j]

## cat.bus[1,2] <- business[i,"business\_id"]

## cat.bus <- as.data.frame(cat.bus)

## names(cat.bus) <- c("cuisine", "business\_id")

## } else {

## cat.bus <- rbind(cat.bus, c(temp[j], business[i,"business\_id"]))

## }

## }

## }

## }

## }

## #get full list of all remaining cuisines

## cuisine.list <- unique(unlist(cat.bus$cuisine, use.names = FALSE))

## #randomly pick subset of reviews for each cuisine for future comparison

## cuisine.review <- NULL

## for (cuisine in cuisine.list){

## #get list of businesses that have that cuisine

## bus.list <- cat.bus[cat.bus[,"cuisine"]==cuisine, "business\_id"]

## 

## #get list of reviews for those businesses

## rev.list <- review[is.element(review[,"business\_id"],bus.list),"text"]

## 

## #randomly select maximum number of reviews to use as representation of the cuisine

## #I've made sure all remaining cuisines have at least 50 reviews, but just in case...

## rev.max <- min(50,length(rev.list))

## set.seed(1)

## rev.list <- sample(rev.list, rev.max, replace=FALSE)

## 

## #Add these reviews to cuisine.review

## cuisine.num <- match(cuisine,cuisine.list)

## if (is.null(cuisine.review)){ #initialize dataframe

## cuisine.review <- as.data.frame(matrix(c(1:length(rev.list),

## rep(cuisine,length(rev.list)),

## rev.list), ncol=3))

## names(cuisine.review) <- c("id", "cuisine", "reviews")

## cuisine.review[,1] <- as.character(cuisine.review[,1])

## cuisine.review[,2] <- as.character(cuisine.review[,2])

## } else {

## len <- dim(cuisine.review)[1]

## new.matrix <- as.data.frame(matrix(c((len+1):(len+length(rev.list)),

## rep(cuisine,length(rev.list)),

## rev.list), ncol=3))

## names(new.matrix) <- c("id", "cuisine", "reviews")

## cuisine.review <- rbind(cuisine.review, new.matrix)

## }

## }

## #Turn text into Corpus and clean up before creating document term matrix

## myReader <- readTabular(mapping=list(content="reviews", id="id", cuisine = "cuisine"))

## corp <- VCorpus(DataframeSource(cuisine.review), readerControl=list(reader=myReader))

## corp <- tm\_map(corp, removeNumbers)

## corp <- tm\_map(corp, content\_transformer(tolower)) #lower case needs to be before stopwords

## corp <- tm\_map(corp, removeWords, rev.default(stopwords('english'))) #reverse order to get contractions

## corp <- tm\_map(corp, removePunctuation) #remove after stopwords because many contractions are stop words

## corp <- tm\_map(corp, stripWhitespace)

## corp <- tm\_map(corp, stemDocument)

## #turn Corpus into "DocumentTermMatrix" class

## dtm <- DocumentTermMatrix(corp)

## rowTotals <- as.data.frame(as.matrix(rollup(dtm, 2, na.rm=TRUE, FUN = sum)))

## dtm <- dtm[rowTotals> 0, ] #remove all docs without words

## wordMatrix = as.data.frame(t(as.matrix(dtm)))

## #rm(corp) #remove corpus to save RAM

## ###functions that calculate similarity score based on 2 cuisines' texts

## #This function uses a simple cosine angle between texts using term frequency vectors

## Vector\_Cos <- function(tdm, q, d){

## 

## #subset tdm to only include "query" and the "document"

## if (q==d){

## return (0)

## } else {

## in.qd <- ((tdm[,q]>=1)|(tdm[,d]>=1))

## tdm\_qd <- tdm[in.qd,]

## }

## 

## # get q and d vectors for terms left

## a <- tdm\_qd[,q]

## b <- tdm\_qd[,d]

## 

## theta <- acos( sum(a\*b) / ( sqrt(sum(a \* a)) \* sqrt(sum(b \* b)) ) )

## 

## #get vector lengths to normalize angle calculation

## #q.len <- sqrt(sum(q.vect \* q.vect))

## #d.len <- sqrt(sum(d.vect \* d.vect))

## 

## #dot.prod <- sum(q.vect \* d.vect)

## 

## return (theta)

## }

## #Create similarity matrix by comparing different cuisines to each other

## cuisine.length <- length(cuisine.list)

## sim.matrix <- as.data.frame(matrix(nrow=cuisine.length, ncol=cuisine.length))

## names(sim.matrix) <- cuisine.list

## row.names(sim.matrix) <- cuisine.list

## set.seed(1)

## for (c1 in cuisine.list){

## #get cuisine's number in the list

## cuisine.num <- match(c1,cuisine.list)

## 

## #show progress to user

## print(paste(cuisine.num," of ", cuisine.length, " cuisines", sep=""))

## 

## #get review id's for cuisine 1 (c1) and randomize them

## c1.id <- cuisine.review[cuisine.review[,"cuisine"]==c1,1]

## c1.id <- sample(c1.id, length(c1.id), replace=FALSE)

## 

## for(c2 in cuisine.list){

## #get review id's for cuisine 2 (c2)

## if (c1==c2){

## c2.id <- c1.id #gives 0 angle if same cuisine

## } else {

## c2.id <- cuisine.review[cuisine.review[,"cuisine"]==c2,1]

## c2.id <- sample(c2.id, length(c2.id), replace=FALSE)

## }

## 

## #calculate angle between 50 review vectors from each cuisine

## temp <- rep(NA,50)

## for (i in 1:length(temp)){

## temp[i] <- Vector\_Cos(wordMatrix, c1.id[i], c2.id[i])

## }

## 

## #average the angles and report that to similarity matrix

## sim.matrix[c1,c2] <- mean(temp)

## }

## }

## write.csv(sim.matrix,"task 2.1 similiarity matrix.csv")

## Appendix II: Task 2.2.R

##Task 2.2

#by Jed Isom

#Week of September 13th, 2015

#import the applicable JSON files

library("pacman")

pacman::p\_load(jsonlite, tm, topicmodels, slam)

#jsonlite for JSON file loading

#tm and topicmodels for LDA topic modeling

rm(list=ls())

json\_file <- "yelp\_academic\_dataset\_business.JSON"

#took this line of code from http://stackoverflow.com/questions/26519455/error-parsing-json-file-with-the-jsonlite-package

business <- fromJSON(sprintf("[%s]", paste(readLines(json\_file), collapse=",")))

#this lists information about the businesses (location, hours, category, name, some attributes)

json\_file <- "yelp\_academic\_dataset\_review.JSON"

review <- fromJSON(sprintf("[%s]", paste(readLines(json\_file), collapse=",")))

#list reviews by businesses

#add boolean variable to business is.restaurant

#quick and dirty code for this, but only takes a couple seconds

for (i in 1:dim(business)[1]){

business[i,"is.restaurant"] = is.element("Restaurants", business[i,"categories"][[1]])

}

#subset business dataframe for just restaurants

business <- business[business[,"is.restaurant"]==TRUE,]

#get only the restaurants that have a cuisine as a category (and remove low freq. cuisines...)

non.cuisine.list <- c("Nightlife", "Lounges", "Party & Event Planning", "Event Planning & Services",

"Venues & Event Spaces", "Active Life", "Bowling", "Beer, Wine & Spirits", "Grocery",

"Meat Shops", "Dance Clubs", "Arts & Entertainment", "Music Venues", "Karaoke",

"Shopping Centers", "Shopping", "Outlet Stores", "Golf", "Convenience Stores",

"Drugstores", "Hotels & Travel", "Hotels", "Jazz & Blues", "Performing Arts", "Fashion",

"Sporting Goods", "Sports Wear", "Cinema", "Pool Halls", "Arcades", "Casinos",

"Health Markets", "Social Clubs", "Food Delivery Services", "Gift Shops",

"Flowers & Gifts", "Health & Medical", "Hospitals", "Hookah Bars", "Amusement Parks",

"Gas & Service Stations", "Automotive", "Adult Entertainment", "Beauty & Spas",

"Gyms", "Medical Spas", "Fitness & Instruction", "Day Spas", "Taxis", "Transportation",

"Auto Repair", "Colleges & Universities", "Education", "Specialty Schools",

"Cooking Schools", "RV Parks", "Home Decor", "Home & Garden", "Kitchen & Bath",

"Appliances", "Airports", "Tours", "Do-It-Yourself Food", "Cafeteria",

"Swimming Pools", "Wineries", "Art Galleries", "Bed & Breakfast", "Arts & Crafts",

"Landmarks & Historical Buildings", "Personal Shopping", "Public Services & Government",

"Street Vendors", "Dry Cleaning & Laundry", "Local Services", "Festivals",

"Farmers Market", "Internet Cafes", "Leisure Centers", "Kids Activities", "Car Wash",

"Horseback Riding", "Butcher", "Country Dance Halls", "Cultural Center", "Delicatessen",

"Home Services", "Real Estate", "Apartments", "Mass Media", "Print Media",

"Food", "Fast Food", "Bars", "Bakeries", "Coffee & Tea", "Donuts", "Caterers",

"Dive Bars", "Pubs", "Buffets", "Cafes", "Sports Bars", "Specialty Food",

"Gluten-Free", "Wine Bars", "Comfort Food", "Bagels", "Gastropubs",

"Juice Bars & Smoothies", "Breweries", "Pretzels", "Food Stands", "Island Pub",

"Tapas Bars", "Cheese Shops", "Gay Bars", "Herbs & Spices", "Hot Pot", "Local Flavor",

"Brasseries", "Shaved Ice", "Food Trucks", "Food Court", "Champagne Bars",

"Bubble Tea", "Piano Bars", "Poutineries", "Beer Bar", "Distilleries", "Lebanese",

"Soup", "Caribbean", "Tea Rooms", "Cheesesteaks", "Soul Food", "Salvadoran", "Kosher",

"Polish", "Creperies", "Cuban", "Russian", "Irish", "Fruits & Veggies", "Fondue",

"Arabian", "Seafood Markets", "Peruvian", "Halal", "Dim Sum", "Mongolian",

"Persian/Iranian", "German", "Cantonese", "Taiwanese", "Argentine",

"Himalayan/Nepalese", "Moroccan", "Falafel", "Ethiopian", "African", "Indonesian",

"Turkish", "Afghan", "Tapas/Small Plates", "Basque", "Spanish", "Cocktail Bars",

"Brazilian", "Personal Chefs", "Laotian", "Szechuan", "Belgian", "Gelato",

"Live/Raw Food", "Bistros", "Chocolatiers & Shops", "Malaysian", "Singaporean",

"Burmese", "Scandinavian", "Canadian (New)", "Czech", "Slovakian", "Scottish",

"Modern European", "Bangladeshi", "Ramen", "Portuguese", "Ukrainian", "Shanghainese",

"Cambodian", "Venezuelan", "Colombian", "Dominican", "Patisserie/Cake Shop",

"Australian", "Egyptian")

#create dataframe of cuisines and the business\_id's associated with them

cat.bus <- as.data.frame(matrix(c(NA,NA),nrow=1,ncol=2))

names(cat.bus) <- c("cuisine", "business\_id")

cuisine.found <- FALSE

for (i in 1:dim(business)[1]){ #go through all businesses

temp <- as.list(business[i,"categories"][[1]]) #get list of categories for that busines

if (length(temp)>=1) { #Make sure the list isn't empty

for (j in 1:length(temp)){ #cycle through all categories for the business

if ((temp[j] != "Restaurants")&(!is.element(temp[j],non.cuisine.list))){

#filter out non-cuisines and the word "Restaurants"

if (cuisine.found==FALSE){

cuisine.found <- TRUE

cat.bus[1,1] <- temp[j]

cat.bus[1,2] <- business[i,"business\_id"]

cat.bus <- as.data.frame(cat.bus)

names(cat.bus) <- c("cuisine", "business\_id")

} else {

cat.bus <- rbind(cat.bus, c(temp[j], business[i,"business\_id"]))

}

}

}

}

}

#get full list of all remaining cuisines

cuisine.list <- unique(unlist(cat.bus$cuisine, use.names = FALSE))

#consolidate all of the reviews for a cuisine into 1 "document" each for future comparison

cuisine.review <- NULL

for (cuisine in cuisine.list){

#get list of businesses that have that cuisine

bus.list <- cat.bus[cat.bus[,"cuisine"]==cuisine, "business\_id"]

#get list of reviews for those businesses

rev.list <- review[is.element(review[,"business\_id"],bus.list),"text"]

#randomly select maximum number of reviews to use as representation of the cuisine

rev.max <- min(50,length(rev.list))

set.seed(1)

rev.list <- sample(rev.list, rev.max, replace=FALSE)

#Combine all the reviews for these businesses and store in cuisine.review

cuisine.num <- match(cuisine,cuisine.list)

if (is.null(cuisine.review)){ #initialize dataframe

cuisine.review <- as.data.frame(matrix(c(cuisine,

paste(rev.list, collapse = " - ")), nrow=1, ncol=2))

names(cuisine.review) <- c("cuisine", "combined\_reviews")

cuisine.review[,1] <- as.character(cuisine.review[,1])

cuisine.review[,2] <- as.character(cuisine.review[,2])

} else {

cuisine.review <- rbind(cuisine.review, c(cuisine, paste(rev.list, collapse=" - ")))

}

}

#Turn text into Corpus and clean up before creating document term matrix

myReader <- readTabular(mapping=list(content="combined\_reviews", id="cuisine"))

corp <- VCorpus(DataframeSource(cuisine.review), readerControl=list(reader=myReader))

corp <- tm\_map(corp, removeNumbers)

corp <- tm\_map(corp, content\_transformer(tolower)) #lower case needs to be before stopwords

corp <- tm\_map(corp, removeWords, rev.default(stopwords('english'))) #reverse order to get contractions

corp <- tm\_map(corp, removePunctuation) #remove after stopwords because many contractions are stop words

corp <- tm\_map(corp, stripWhitespace)

corp <- tm\_map(corp, stemDocument)

#turn Corpus into "DocumentTermMatrix" class

dtm <- DocumentTermMatrix(corp)

rowTotals <- as.data.frame(as.matrix(rollup(dtm, 2, na.rm=TRUE, FUN = sum)))

dtm <- dtm[rowTotals> 0, ] #remove all docs without words

wordMatrix = as.data.frame(t(as.matrix(dtm)))

#rm(corp) #remove corpus to save RAM

#function that calculates similarity score based on 2 cuisines' texts

#This one uses Pivoted Length Normalization - Vector Space Model

PLM\_VSM <- function (tdm, q, d, M, avdl, b=0.5){

#find document length

dl <- sum(tdm[,d])

#subset tdm to only include words contained in cuisine 1 and cuisine 2

in.qd <- ((tdm[,q]>=1)|(tdm[,d]>=1))

if (q==d){

tdm\_qd <- tdm[in.qd,]

df\_ws <- rowSums(tdm\_qd != 0) # num. of documents with each term

} else {

tdm\_qd <- tdm[in.qd,]

df\_ws <- rowSums(tdm\_qd != 0) # num. of documents with each term

}

##calculate score using vectorized approach

#vectorize word counts by terms

q.word.counts <- tdm\_qd[,q]

if (q==d){

d.word.counts <- q.word.counts

} else {

d.word.counts <- tdm\_qd[,d]

}

#Use vector math to calculate and sum the score

num <- (log(1+log(1+d.word.counts)))

den <- (1-b+b\*dl/avdl)

log\_term <-log((M+1)/df\_ws)

f\_qd <- sum((q.word.counts\*num/den)\*log\_term)

return (f\_qd)

}

M <- dim(cuisine.review)[1] #number of documents in corpus

avdl <- mean(colSums(wordMatrix)) #find average document length

#Create similarity matrix by comparing different cuisines to each other

cuisine.length <- length(cuisine.list)

sim.matrix <- as.data.frame(matrix(nrow=cuisine.length, ncol=cuisine.length))

names(sim.matrix) <- cuisine.list

row.names(sim.matrix) <- cuisine.list

for (c1 in cuisine.list){

cuisine.num <- match(c1,cuisine.list)

print(paste(cuisine.num," of ", cuisine.length, " cuisines", sep=""))

for(c2 in cuisine.list){

#First try with b=0.5

#sim.matrix[c1,c2] <- PLM\_VSM(wordMatrix, c1, c2, M, avdl, b=0.5)

#2nd Try with b=0.9

#sim.matrix[c1,c2] <- PLM\_VSM(wordMatrix, c1, c2, M, avdl, b=0.9)

#3rd Try with b=0

sim.matrix[c1,c2] <- PLM\_VSM(wordMatrix, c1, c2, M, avdl, b=0)

}

}

#Turn similarity score into a cuisine distance with match = 0 distance

sim.matrix <- (max(sim.matrix)/sim.matrix)-1

#write.csv(sim.matrix,"task 2.2 similiarity matrix.csv")

#write.csv(sim.matrix,"task 2.2(b=0.9) similiarity matrix.csv")

write.csv(sim.matrix,"task 2.2(b=0) similiarity matrix.csv")

## Appendix III: Task 2.3.R

##Task 2.3

#by Jed Isom

#Week of September 13th, 2015

#import the applicable JSON files

library("pacman")

pacman::p\_load(jsonlite, tm, topicmodels, slam, cluster)

#jsonlite for JSON file loading

#tm and topicmodels for LDA topic modeling

rm(list=ls())

json\_file <- "yelp\_academic\_dataset\_business.JSON"

#took this line of code from http://stackoverflow.com/questions/26519455/error-parsing-json-file-with-the-jsonlite-package

business <- fromJSON(sprintf("[%s]", paste(readLines(json\_file), collapse=",")))

#this lists information about the businesses (location, hours, category, name, some attributes)

json\_file <- "yelp\_academic\_dataset\_review.JSON"

review <- fromJSON(sprintf("[%s]", paste(readLines(json\_file), collapse=",")))

#list reviews by businesses

#add boolean variable to business is.restaurant

#quick and dirty code for this, but only takes a couple seconds

for (i in 1:dim(business)[1]){

business[i,"is.restaurant"] = is.element("Restaurants", business[i,"categories"][[1]])

}

#subset business dataframe for just restaurants

business <- business[business[,"is.restaurant"]==TRUE,]

#get only the restaurants that have a cuisine as a category (and remove low freq. cuisines...)

non.cuisine.list <- c("Nightlife", "Lounges", "Party & Event Planning", "Event Planning & Services",

"Venues & Event Spaces", "Active Life", "Bowling", "Beer, Wine & Spirits", "Grocery",

"Meat Shops", "Dance Clubs", "Arts & Entertainment", "Music Venues", "Karaoke",

"Shopping Centers", "Shopping", "Outlet Stores", "Golf", "Convenience Stores",

"Drugstores", "Hotels & Travel", "Hotels", "Jazz & Blues", "Performing Arts", "Fashion",

"Sporting Goods", "Sports Wear", "Cinema", "Pool Halls", "Arcades", "Casinos",

"Health Markets", "Social Clubs", "Food Delivery Services", "Gift Shops",

"Flowers & Gifts", "Health & Medical", "Hospitals", "Hookah Bars", "Amusement Parks",

"Gas & Service Stations", "Automotive", "Adult Entertainment", "Beauty & Spas",

"Gyms", "Medical Spas", "Fitness & Instruction", "Day Spas", "Taxis", "Transportation",

"Auto Repair", "Colleges & Universities", "Education", "Specialty Schools",

"Cooking Schools", "RV Parks", "Home Decor", "Home & Garden", "Kitchen & Bath",

"Appliances", "Airports", "Tours", "Do-It-Yourself Food", "Cafeteria",

"Swimming Pools", "Wineries", "Art Galleries", "Bed & Breakfast", "Arts & Crafts",

"Landmarks & Historical Buildings", "Personal Shopping", "Public Services & Government",

"Street Vendors", "Dry Cleaning & Laundry", "Local Services", "Festivals",

"Farmers Market", "Internet Cafes", "Leisure Centers", "Kids Activities", "Car Wash",

"Horseback Riding", "Butcher", "Country Dance Halls", "Cultural Center", "Delicatessen",

"Home Services", "Real Estate", "Apartments", "Mass Media", "Print Media",

"Food", "Fast Food", "Bars", "Bakeries", "Coffee & Tea", "Donuts", "Caterers",

"Dive Bars", "Pubs", "Buffets", "Cafes", "Sports Bars", "Specialty Food",

"Gluten-Free", "Wine Bars", "Comfort Food", "Bagels", "Gastropubs",

"Juice Bars & Smoothies", "Breweries", "Pretzels", "Food Stands", "Island Pub",

"Tapas Bars", "Cheese Shops", "Gay Bars", "Herbs & Spices", "Hot Pot", "Local Flavor",

"Brasseries", "Shaved Ice", "Food Trucks", "Food Court", "Champagne Bars",

"Bubble Tea", "Piano Bars", "Poutineries", "Beer Bar", "Distilleries", "Lebanese",

"Soup", "Caribbean", "Tea Rooms", "Cheesesteaks", "Soul Food", "Salvadoran", "Kosher",

"Polish", "Creperies", "Cuban", "Russian", "Irish", "Fruits & Veggies", "Fondue",

"Arabian", "Seafood Markets", "Peruvian", "Halal", "Dim Sum", "Mongolian",

"Persian/Iranian", "German", "Cantonese", "Taiwanese", "Argentine",

"Himalayan/Nepalese", "Moroccan", "Falafel", "Ethiopian", "African", "Indonesian",

"Turkish", "Afghan", "Tapas/Small Plates", "Basque", "Spanish", "Cocktail Bars",

"Brazilian", "Personal Chefs", "Laotian", "Szechuan", "Belgian", "Gelato",

"Live/Raw Food", "Bistros", "Chocolatiers & Shops", "Malaysian", "Singaporean",

"Burmese", "Scandinavian", "Canadian (New)", "Czech", "Slovakian", "Scottish",

"Modern European", "Bangladeshi", "Ramen", "Portuguese", "Ukrainian", "Shanghainese",

"Cambodian", "Venezuelan", "Colombian", "Dominican", "Patisserie/Cake Shop",

"Australian", "Egyptian")

#create dataframe of cuisines and the business\_id's associated with them

cat.bus <- as.data.frame(matrix(c(NA,NA),nrow=1,ncol=2))

names(cat.bus) <- c("cuisine", "business\_id")

cuisine.found <- FALSE

#

for (i in 1:dim(business)[1]){ #go through all businesses

temp <- as.list(business[i,"categories"][[1]]) #get list of categories for that busines

if (length(temp)>=1) { #Make sure the list isn't empty

for (j in 1:length(temp)){ #cycle through all categories for the business

if ((temp[j] != "Restaurants")&(!is.element(temp[j],non.cuisine.list))){

#filter out non-cuisines and the word "Restaurants"

if (cuisine.found==FALSE){

cuisine.found <- TRUE

cat.bus[1,1] <- temp[j]

cat.bus[1,2] <- business[i,"business\_id"]

cat.bus <- as.data.frame(cat.bus)

names(cat.bus) <- c("cuisine", "business\_id")

} else {

cat.bus <- rbind(cat.bus, c(temp[j], business[i,"business\_id"]))

}

}

}

}

}

#get full list of all remaining cuisines

cuisine.list <- unique(unlist(cat.bus$cuisine, use.names = FALSE))

#consolidate all of the reviews for a cuisine into 1 "document" each for future comparison

cuisine.review <- NULL

for (cuisine in cuisine.list){

#get list of businesses that have that cuisine

bus.list <- cat.bus[cat.bus[,"cuisine"]==cuisine, "business\_id"]

#get list of reviews for those businesses

rev.list <- review[is.element(review[,"business\_id"],bus.list),"text"]

#randomly select maximum number of reviews to use as representation of the cuisine

rev.max <- min(50,length(rev.list))

set.seed(1)

rev.list <- sample(rev.list, rev.max, replace=FALSE)

#Combine all the reviews for these businesses and store in cuisine.review

cuisine.num <- match(cuisine,cuisine.list)

if (is.null(cuisine.review)){ #initialize dataframe

cuisine.review <- as.data.frame(matrix(c(cuisine,

paste(rev.list, collapse = " - ")), nrow=1, ncol=2))

names(cuisine.review) <- c("cuisine", "combined\_reviews")

cuisine.review[,1] <- as.character(cuisine.review[,1])

cuisine.review[,2] <- as.character(cuisine.review[,2])

} else {

cuisine.review <- rbind(cuisine.review, c(cuisine, paste(rev.list, collapse=" - ")))

}

}

#Turn text into Corpus and clean up before creating document term matrix

myReader <- readTabular(mapping=list(content="combined\_reviews", id="cuisine"))

corp <- VCorpus(DataframeSource(cuisine.review), readerControl=list(reader=myReader))

corp <- tm\_map(corp, removeNumbers)

corp <- tm\_map(corp, content\_transformer(tolower)) #lower case needs to be before stopwords

corp <- tm\_map(corp, removeWords, rev.default(stopwords('english'))) #reverse order to get contractions

corp <- tm\_map(corp, removePunctuation) #remove after stopwords because many contractions are stop words

corp <- tm\_map(corp, stripWhitespace)

corp <- tm\_map(corp, stemDocument)

#turn Corpus into "DocumentTermMatrix" class

dtm <- DocumentTermMatrix(corp)

rowTotals <- as.data.frame(as.matrix(rollup(dtm, 2, na.rm=TRUE, FUN = sum)))

dtm <- dtm[rowTotals> 0, ] #remove all docs without words

wordMatrix = as.data.frame(t(as.matrix(dtm)))

#rm(corp) #remove corpus to save RAM

#function that calculates similarity score based on 2 cuisines' texts

#This one uses Pivoted Length Normalization - Vector Space Model

PLM\_VSM <- function (tdm, q, d, M, avdl, b=0.5){

#find document length

dl <- sum(tdm[,d])

#subset tdm to only include words contained in cuisine 1 and cuisine 2

in.qd <- ((tdm[,q]>=1)|(tdm[,d]>=1))

if (q==d){

tdm\_qd <- tdm[in.qd,]

df\_ws <- rowSums(tdm\_qd != 0) # num. of documents with each term

} else {

tdm\_qd <- tdm[in.qd,]

df\_ws <- rowSums(tdm\_qd != 0) # num. of documents with each term

}

##calculate score using vectorized approach

#vectorize word counts by terms

q.word.counts <- tdm\_qd[,q]

if (q==d){

d.word.counts <- q.word.counts

} else {

d.word.counts <- tdm\_qd[,d]

}

#Use vector math to calculate and sum the score

num <- (log(1+log(1+d.word.counts)))

den <- (1-b+b\*dl/avdl)

log\_term <-log((M+1)/df\_ws)

f\_qd <- sum((q.word.counts\*num/den)\*log\_term)

return (f\_qd)

}

M <- dim(cuisine.review)[1] #number of documents in corpus

avdl <- mean(colSums(wordMatrix)) #find average document length

#Create similarity matrix by comparing different cuisines to each other

cuisine.length <- length(cuisine.list)

sim.matrix <- as.data.frame(matrix(nrow=cuisine.length, ncol=cuisine.length))

names(sim.matrix) <- cuisine.list

row.names(sim.matrix) <- cuisine.list

for (c1 in cuisine.list){

cuisine.num <- match(c1,cuisine.list)

print(paste(cuisine.num," of ", cuisine.length, " cuisines", sep=""))

for(c2 in cuisine.list){

#Decided to use b=0.9 based on visual review of task 2.2 results

sim.matrix[c1,c2] <- PLM\_VSM(wordMatrix, c1, c2, M, avdl, b=0.9)

}

}

#Turn similarity score into a cuisine distance with match = 0 distance

dist.matrix <- (max(sim.matrix)/sim.matrix)-1

#get average of (i,j) and (j,i) because they're slightly different when using PLM\_VSM

dist.matrix2 <- dist.matrix

for (i in 1:cuisine.length){

for (j in 1:cuisine.length){

if (i==j){

dist.matrix2[i,j] <- 0

} else{

dist.matrix2[i,j] <- mean(dist.matrix[i,j], dist.matrix[j,i])

}

}

}

dist.matrix <- dist.matrix2

#use agglomerative clustering to guess a good # of clusters

cluster.agnes <- agnes(dist.matrix, diss = TRUE)

cluster.agnes$height #look at the height to see where a good break point is (k=2?, 5?, 7?)

#cluster based on the distance matrix

cluster.pam.7 <- pam(dist.matrix, 7, diss = TRUE, keep.diss = FALSE)

cluster.pam.5 <- pam(dist.matrix, 5, diss = TRUE, keep.diss = FALSE)

cluster.pam.2 <- pam(dist.matrix, 2, diss = TRUE, keep.diss = FALSE)

#Assign cluster numbers to each cuisine and sort them

#7 clusters

cluster.num.7 <- as.data.frame(cluster.pam.7[3])

cluster.num.7[,2] <- row.names(cluster.num.7)

names(cluster.num.7) <- c("number", "cuisine")

cluster.num.7 <- cluster.num.7[ order(cluster.num.7[,1]), ]

row.names(cluster.num.7) <- NULL

#5 clusters

cluster.num.5 <- as.data.frame(cluster.pam.5[3])

cluster.num.5[,2] <- row.names(cluster.num.5)

names(cluster.num.5) <- c("number", "cuisine")

cluster.num.5 <- cluster.num.5[ order(cluster.num.5[,1]), ]

row.names(cluster.num.5) <- NULL

#2 clusters

cluster.num.2 <- as.data.frame(cluster.pam.2[3])

cluster.num.2[,2] <- row.names(cluster.num.2)

names(cluster.num.2) <- c("number", "cuisine")

cluster.num.2 <- cluster.num.2[ order(cluster.num.2[,1]), ]

row.names(cluster.num.2) <- NULL

#sort rows and columns of dist.matrix to get like cuisines next to each other

dist.matrix.7 <- dist.matrix[cluster.num.7$cuisine,cluster.num.7$cuisine]

dist.matrix.5 <- dist.matrix[cluster.num.5$cuisine,cluster.num.5$cuisine]

dist.matrix.2 <- dist.matrix[cluster.num.2$cuisine,cluster.num.2$cuisine]

#write clustered distance matrix to file

write.csv(dist.matrix.7,"task 2.3(k=7) distance matrix.csv")

write.csv(dist.matrix.5,"task 2.3(k=5) distance matrix.csv")

write.csv(dist.matrix.2,"task 2.3(k=2) distance matrix.csv")