Jed Isom

Data Mining Capstone

September 27, 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. The third assignment was about finding dishes in the reviews by using pattern and word/phrase mining.

**Task 3.1**

For this first task we were supposed to take a list of terms that might represent dishes and manually tag them as dishes or not. The original file had 0’s and 1’s next to each term (1=dish, 0=not dish). I categorized them by hand and then removed any dishes that were 1’s but were actually 0’s. However, I kept terms that were 0’s, but became 1’s. I submitted this file, for points and also used it later in task 3.2

**Task 3.2**

I decided to continue using the R script language to do my analysis for this course. I also wanted to use the word2vec algorithm created by a team at Google. To get an augmented list of cuisines greater than the one I submitted in task 3.1, I did the following (see the R script in appendix I):

1. Read in the business and review Yelp files into R
2. Filtered the business file to only include businesses with the category “Chinese”
3. Used that list of businesses to create a list of all reviews for Chinese restaurants
4. Concatenated all of the Chinese reviews into one long text
5. Used an R “Corpus” to
   1. Remove numbers
   2. Lower case all of the text
   3. Remove punctuation
   4. Strip out any whitespace
   5. [I decided not to remove stop words or do stemming because that might have messed up phrases I would be looking for in future steps]
6. Output this “Corpus” to a text file
7. Downloaded the word2vec and word2phrase code and compiled it in cygwin64 (a Unix based terminal window for Windows machines)
   1. Ran word2phrase on the corpus text file looking for common phrases in the corpus (this puts and underscore in between the words for phrases that meet the criteria which makes multiple words in effect 1 term)
   2. Ran word2vec on the output of word2phrase. word2vec essentially turns word patterns in document into numerical vectors that can be analyzed for similarity
8. Back in R, I loaded the output of word2vec
9. Loaded the list of dishes from task 3.1
10. Ran through all of the dishes in task 3.1 and used cosine similarity to the vectors created by word2vec to find other terms would likely also be dishes
11. I sorted this list by with highest similarity to lowest
12. Then pulled out only unique terms, because some terms might be similar to several of the dishes that I used in task 3.1

Overall, I thought that the process worked pretty well. When I look at my final list, the top several terms are dishes that I recognize or ingredients to food. That is likely because I was uncertain during task 3.1 whether some ingredients could be considered on their own to be dishes (is potato its own dish???). So I probably got a list of dishes and ingredients, but based on what I input into the process, that’s exactly what I should have expected. I think that it was pretty cool being able to feed the system a little human knowledge and then have the system find more items that also make sense to a human.

### Appendix I: Task 3.2.R

## #Coursera Data Mining Capstone - UIUC

## #Task 3.2

## #Jed Isom

## #September 22, 2015

## library("pacman")

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

## rm(list=ls())

## setwd("./Capstone/Raw Data")

## 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.chinese

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

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

## business[i,"is.chinese"] = is.element("Chinese", business[i,"categories"][[1]])

## }

## #subset business dataframe for just restaurants

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

## #consolidate all of the reviews for Chinese into 1 "document"

## #get list of businesses that have that Chinese

## bus.list <- business[,"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(5000,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 Chinese.reviews

## Chinese.reviews <- paste(rev.list, collapse = " - ")

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

## corp <- VCorpus(VectorSource(Chinese.reviews))

## 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)

## #hopefully this writes all that text to 1 file...

## write(as.character(corp[[1]][1],"Chinese\_reviews.txt"))

## #This section of code inspired by the class forum (thanks Juan Luis Herrera Cortijo!)

## #https://class.coursera.org/dataminingcapstone-001/forum/thread?thread\_id=124

## #word2vec doesn't work without pthread.h which isn't used by windows so use cygwin64

## #terminal window instead

## #before this works you have to download, compile word2vec.c and word2phrase.c from Google:

## #https://code.google.com/p/word2vec/

## #compile this by opening cygwin64 terminal, navigating to the folder and run 'make'

## #Add the folder where the compiled .exe files are to the list of PATH variables

## #http://stackoverflow.com/questions/10235125/linux-custom-executable-globally-available

## # confirm PATH variables this way 'echo $PATH'

## #This code just generates the text I coplied into the cygwin64 terminal

## training.file <- "Chinese\_reviews.txt"

## output.file.phrases <- "Chinese\_Phrases.txt"

## output.file.text <- "Chinese\_Model.txt"

## # Run the output of these next 2 lines in the cygwin64 terminal

## paste0("word2phrase -train ",training.file," -output ",output.file.phrases)

## paste0("word2vec -train ",output.file.phrases," -output ",output.file.text," -binary 0")

## # Read the vectors

## vectors <- read.table(output.file.text,skip = 1, stringsAsFactors = FALSE)

## # Get the terms

## words <- vectors[,1]

## # Transform to a matrix

## vectors <- as.matrix(vectors[,-1])

## # Get most similar terms to other dishes found

## dishes <- read.table("./manualAnnotationTask/Chinese\_Flat.txt")

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

## dishes.final <- as.data.frame(matrix(c("Chinese",100),nrow=1,ncol=2))

## names(dishes.final) <- c("dish", "cosine")

## dishes.final[,"cosine"] <- as.numeric(dishes.final[,"cosine"])

## #dishes.final <- rbind(dishes.final,as.data.frame(matrix(c(dishes,

## # rep(1, length(dishes))),ncol=2)))

## temp <- as.data.frame(matrix(c(dishes,rep(0, length(dishes))),ncol=2))

## names(temp) <- c("dish", "cosine")

## dishes.final <- rbind(dishes.final,temp)

## for (dish in dishes){

## if(is.element(dish,words)){

## v <- vectors[which(words==dish),]

## sim <- apply(vectors,1,function(d) cosine(v,d))

## sim.dishes <- head(data.frame(Word=words[order(sim,decreasing = TRUE)],

## Cosine=sim[order(sim,decreasing = TRUE)])[-1,],50)

## names(sim.dishes) <- c("dish", "cosine")

## dishes.final <- rbind(dishes.final, sim.dishes)

## }

## }

## dishes.final[,'cosine'] <- as.numeric(dishes.final[,'cosine'])

## dishes.final <- dishes.final[with(dishes.final, order(-cosine)), ]

## dishes.final <- unique(unlist(as.character(dishes.final[,1]), use.names = FALSE))

## write(dishes.final,"Task 3.2 Submission.txt")