KENDRA OSBURN | 3-14-19 | IST707 | TEXT MINING

FAKE REVIEWS

Introduction

Text mining enables researchers to turn paragraphs of text into meaningful data by extracting individual words. Unlike many other models, text mining isn't a singular method that can be applied to data. Rather, it is a collection of many different methods — including sentiment analysis, topic modeling, and lie detection — of transforming blocks of text into data that can be processed, classified, and analyzed. As such, there are many different libraries that can help achieve this textual information retrieval. The input is any block of text, while the output is some variation of record data, often in the form of a term document matrix.

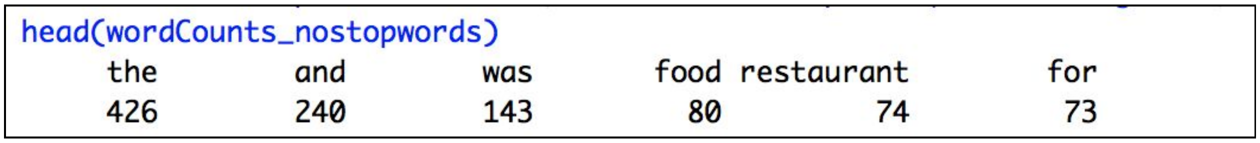
Analysis & Models

I. EXPLORATORY DATA ANALYSIS

Before any true data-tuning, the data was run through the most baseline text-mining program the author created during her first foray into text-mining. (She likes to create as many reusable code blocks as possible for her future self to use. This is useful about 2% of the time. This is one of those 2% times)

```r
library(tm)
library(wordcloud)
file <- "your-file-here"
fileData <- readLines(file)
words.vec <- VectorSource(fileData)
words.corpus <- Corpus(words.vec)
words.corpus <- tm_map(words.corpus, content_transformer(tolower))
words.corpus <- tm_map(words.corpus, removePunctuation)
words.corpus <- tm_map(words.corpus, removeNumbers)
words.corpus <- tm_map(words.corpus, removeWords, stopwords("english"))
tdm <- TermDocumentMatrix(words.corpus)
m <- as.matrix(tdm)
wordCounts <- rowSums(m)
wordCounts <- sort(wordCounts, decreasing=TRUE)
wordcloud(names(wordCounts), wordCounts)
```

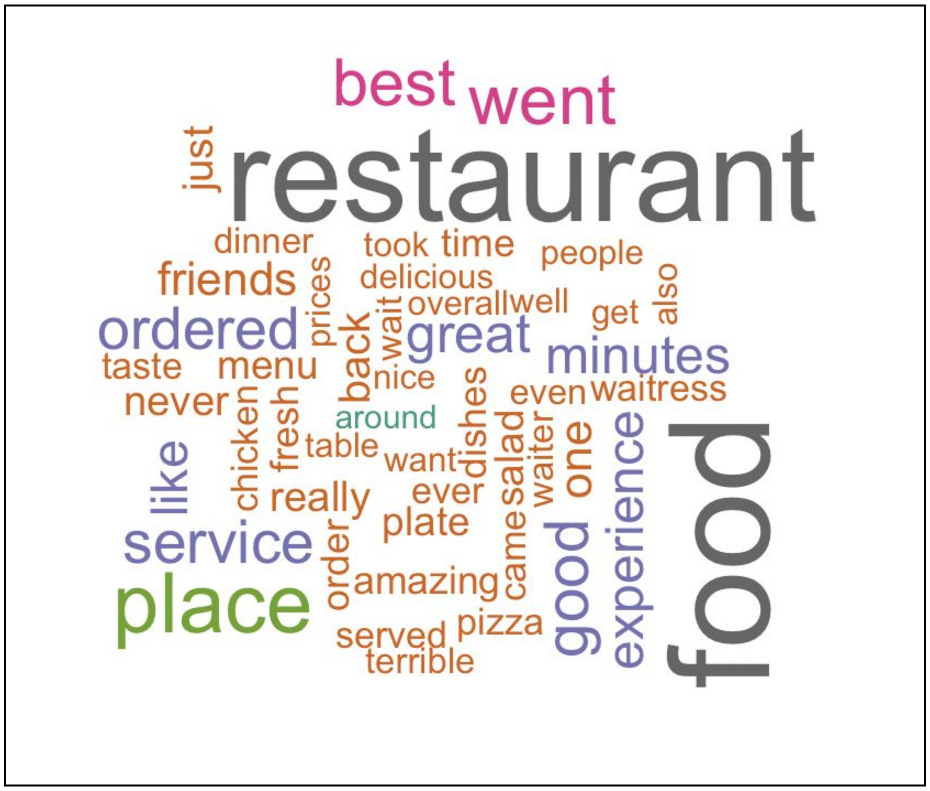
This produced these most frequent words and this illegible word cloud

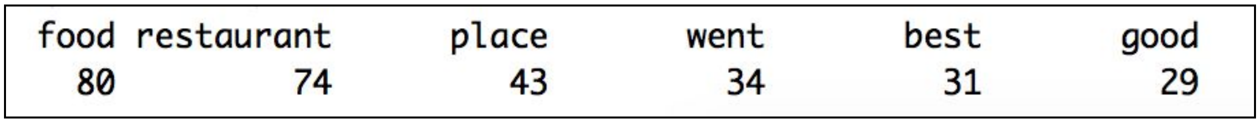


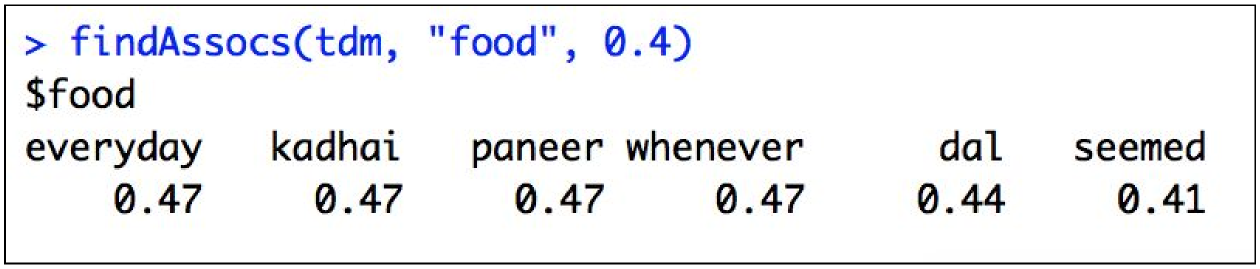
```
head(wordCounts_nostopwords)
      the          and          was         food  restaurant          for
      426          240          143           80          74           73
```

good
place
best
ordered
service
minutes
like
great
amazing
back
fresh
really
order
dishes
one
just
time
came
pizza
taste
waiter
definitely

Then punctuation and stop words were removed. The result is the colorful word cloud below.




```
food restaurant        place         went         best         good
  80         74           43           34           31           29
```



```
> findAssocs(tdm, "food", 0.4)
$food
everyday    kadhai    paneer whenever       dal    seemed
    0.47      0.47      0.47     0.47      0.44      0.41
```

After that, the researcher got down to business and buckled into her office chair, which is what the reader should do now because it's about to get lit like a 3.4 star romantic dinner.

The file data was read in and all of the excess columns were combined into one giant column. The file now had three columns: sentiment, lie, review. The reviews were turned into a corpus, but this was wrong. The reviews first needed to be turned into a vector and then into a corpus. All the words were converted to lowercase. Then punctuation was removed. Then Stopwords were removed. Stopwords are words that occur frequently and, as such, offer little help and a lot of noise.

The data was then split into lie and sentiment, and then further split into testing and training. The model was applied and got 50% accuracy! Normally, the researcher would be very concerned about this, but she squandered away any "concern" time being wayyy to excited about her final project.

Results

```
## IST 707 | HW 8 | WEEK 9 | TEXT MINING
## ====================================
## STEP 1: IMPORT LIBRARIES & DATA
## ====================================
library(tm)
library(wordcloud)
library(tidyr)
library(tidytext)
library(dplyr)
setwd("/Users/kosburn 1/syracuse/IST707/WK9")
file <- "deception_data_converted.csv"
fileData <- read.csv(file)
str(fileData)
og_fileData <- fileData
```

```r
fileData <- tidyr::unite(fileData, review, c("review", "X", "X.1", "X.2",
"X.3", "X.4", "X.5", "X.6", "X.7", "X.8", "X.9", "X.10", "X.11", "X.12",
"X.13", "X.14", "X.15", "X.16", "X.17", "X.18", "X.19", "X.20"))

## ====================================
## STEP 2: TURN DATA INTO CORPUS
## ====================================
## 2i.TURN VARIABLE INTO A CORPUS "BAG OF WORDS"
## 2ia. FIRST VECTOR, THEN CORPUS
## (corpus needs a vector to make corpus)

reviews <- fileData$review
words.vec <- VectorSource(reviews)
words.corpus <- Corpus(words.vec)
words.corpus

## ====================================
## STEP 3: TRANSFORM DATA
## ====================================
## 3i. TRANSFORM ALL WORDS TO LOWERCASE
words.corpus <- tm_map(words.corpus, content_transformer(tolower))
## 3ii. REMOVE PUNCTUATION
words.corpus <- tm_map(words.corpus, removePunctuation)
## 3iii. REMOVE NUMBERS
words.corpus <- tm_map(words.corpus, removeNumbers)
## 3iv. REMOVE STOPWORDS
words.corpus <- tm_map(words.corpus, removeWords, stopwords("english"))

## ====================================
## STEP 4: CREATE TERM-DOCUMENT MATRIX
## ====================================
## WHAT IS A TERM? TERM = DIFFERENT WORD
tdm <- TermDocumentMatrix(words.corpus)
tdmOG <- TermDocumentMatrix(words.corpus)

## ====================================
## STEP 5:VIEW WORD CLOUD & SEE FREQ
## ====================================
## FOR TDM
m <- as.matrix(tdm)
wordCounts <- rowSums(m)
wordCounts <- sort(wordCounts, decreasing=TRUE)
head(wordCounts)
wordcloud(names(wordCounts), wordCounts)
wordcloud(names(wordCounts), wordCounts, min.freq=2, max.words=50,
rot.per=0.35, colors=brewer.pal(8, "Dark2"))

findAssocs(tdm, "food", 0.4)

## ====================================
## STEP 6: Create DTM
## ====================================

## Create DTM
dtm_reviews <- t(tdmOG)
matrix_reviews <- as.matrix(dtm_reviews)

df_reviews <- tidy(matrix_reviews)

df_reviews_normalized <- data.frame(t(apply(df_reviews, 1, function(i)
i/sum(i))))


## NON NORMALIZED TEST AND TRAIN
## TURN BACK INTO A MATRIX
lie <- fileData$lie
sentiment <- fileData$sentiment
alldata <-cbind(lie, sentiment, df_reviews)
alldata <- alldata[-c(83:84),]

alldata_lie_true_sentiment_positive <- subset(alldata, lie == 't' &
sentiment =='p')
alldata_lie_true_sentiment_negative <- subset(alldata, lie == 't' &
sentiment =='n')
alldata_lie_false_sentiment_positive <- subset(alldata, lie == 'f' &
sentiment =='p')
alldata_lie_false_sentiment_negative <- subset(alldata, lie == 'f' &
sentiment =='n')

## Ali's amaze functions!!
#Creating a function to create a training df
my_sample_fun <- function(df, n) {
  sample(nrow(df), n)
}

#Creating a function to create a training df
my_train_set <- function(df, vector) {
  df[vector,]
}

#Creating a function to create a testing df
my_test_set <- function(df, vector, n) {
  df <- df[-vector,]
  df[sample(nrow(df), n),]
}

sample_lie_true_sentiment_positive <-
my_sample_fun(alldata_lie_true_sentiment_positive, 15)
train_lie_true_sentiment_positive <-
my_train_set(alldata_lie_true_sentiment_positive,
sample_lie_true_sentiment_positive)
test_lie_true_sentiment_positive <-
my_test_set(alldata_lie_true_sentiment_positive,
sample_lie_true_sentiment_positive, 6)

sample_lie_true_sentiment_negative <-
my_sample_fun(alldata_lie_true_sentiment_negative, 15)
train_lie_true_sentiment_negative <-
my_train_set(alldata_lie_true_sentiment_negative,
sample_lie_true_sentiment_negative)
test_lie_true_sentiment_negative <-
my_test_set(alldata_lie_true_sentiment_negative,
sample_lie_true_sentiment_negative, 6)

sample_lie_false_sentiment_positive <-
my_sample_fun(alldata_lie_false_sentiment_positive, 15)
train_lie_false_sentiment_positive <-
my_train_set(alldata_lie_false_sentiment_positive,
sample_lie_false_sentiment_positive)
test_lie_false_sentiment_positive <-
my_test_set(alldata_lie_false_sentiment_positive,
sample_lie_false_sentiment_positive, 6)

sample_lie_false_sentiment_negative <-
my_sample_fun(alldata_lie_false_sentiment_negative, 15)
```

```r
train_lie_false_sentiment_negative <-
my_train_set(alldata_lie_false_sentiment_negative,
sample_lie_false_sentiment_negative)
test_lie_false_sentiment_negative <-
my_test_set(alldata_lie_false_sentiment_negative,
sample_lie_false_sentiment_negative, 6)


##############################################
## RUN MODELS: Naive Bayes
##############################################
library(e1071)

train_set <- rbind(train_lie_true_sentiment_positive,
train_lie_true_sentiment_negative, train_lie_false_sentiment_positive,
train_lie_false_sentiment_negative)
test_set <- rbind(test_lie_true_sentiment_positive,
test_lie_true_sentiment_negative, test_lie_false_sentiment_positive,
test_lie_false_sentiment_negative)

train_set_lie <- train_set[,!colnames(train_set) %in% c('sentiment')]
test_set_lie <- test_set[,!colnames(test_set) %in% c('sentiment')]

train_set_sentiment <- train_set[,!colnames(train_set) %in% c('lie')]
test_set_sentiment <- test_set[,!colnames(test_set) %in% c('lie')]

##********************************************
## LIE
##********************************************
train_set <- train_set_lie
test_set <- test_set_lie

test_label <- c('lie')
test_set_no_label <- test_set[,!colnames(test_set) %in% test_label]

NB_e1071<-naiveBayes(lie~., data=train_set, na.action = na.pass)
NB_e1071_Pred <- predict(NB_e1071, test_set_no_label)
(pred_table <- table(NB_e1071_Pred,test_set$lie))
correct <- sum(diag(pred_table))
(accuracy <- correct/sum(pred_table))

##********************************************
```

```
## SENTIMENT
##*********************************************
train_set <- train_set_sentiment
test_set <- test_set_sentiment

test_label <- c('sentiment')
test_set_no_label <- test_set[,!colnames(test_set) %in% test_label]
test_sentiment_lie_label <- test_set[,colnames(test_set) %in% test_label]

NB_e1071<-naiveBayes(sentiment~., data=train_set, na.action = na.pass)
NB_e1071_Pred <- predict(NB_e1071, test_set_no_label)
(pred_table <- table(NB_e1071_Pred,test_set$sentiment))
correct <- sum(diag(pred_table))
(accuracy <- correct/sum(pred_table))
```

Conclusion

The researchers had grand plans regarding text mining and sentiment analysis. Their ultimate goal with this dataset was to determine whether reviews were falsified or not. Sentiment analysis was applied for color. They can predict with 50% accuracy, the same accuracy as a coin(!) whether the review is true or false, positive or negative. This is depressing and again the researchers find themselves wishing for the ability to bend space as well as the ability to mine text. They are also hoping Dr. Gates hasn't gotten this far because a new paper has made its way into her hands. If she is here, please know this is a reflection of (1) having a huge work project in addition to school projects (2) over committing to our project!! Regardless, I learned so much. Thank you!!