Predicting Business Rating Based on Yelp Tips Using Lasso model

Sammy Yu

November 2, 2015

### I Introduction

The tips on the Yelp web site is the insider information provided by the Yelp's users or reviewers. The text of the tips contain the sentiments of the users of how they would recomend for the other users when they use the service or buy a product from a particular business. They can positive or negative and should be consistent with review posted by the same user. In this study, I will try to predict a business overall rating based on only the text of tips to see how the tips can affect the rating or image of the business.

### II Methods and Data

To create and train a prediction model for business ratings, I use two datasets, "yelp\_academic\_dataset\_business.json" and "yelp\_academic\_dataset\_tips.json" from [Yelp Dataset Challenge](http://www.yelp.com/dataset_challenge) (Rounds 5 and 6).

Both files are in JSON format. The files will be read, merged and cleaned up before they are used for developing the prediction model.

The algorithm I use to develop the prediction model is called the LASSO (Least Absolute Shrinkage and Selection Operator) which was developed by Robert Tibshirani in 1996. The LASSO is derived from multiple linear regression and is similar to Ridge Regression which penalize the coefficients to minimize overfitting. The LASSO has an advantage over Ridge Regression though is that it not only shrinks the coefficients but also force some of the coefficients to be exactly zero when the tuning parameter is selected large enough.

Where the is the popular word or term appears in the tips and is a mean-zero random error term.  
The goal of the LASSO is to find the coefficients those can minimize the quantity below:

To get the most popular words from the tips as the features of the LASSO model, text mining technique is used. There are more than 80,000 words found in the text of the tips but many of them are not frequent used terms (high sparsity). So we filter out those terms with more than 99% sparsity and keep around 290 terms for the model development.

The study first estimates the predicted rating and mean square errors (MSE) from a naive model for comparison. Then we train and validate a LASSO model using *unigrams* (single word in each term) and *bigrams* (two adjacient words in each term).

## load required libraries  
library(jsonlite)

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

library(NLP)  
library(tm)  
library(SnowballC)  
library(wordcloud)

## Loading required package: RColorBrewer

library(glmnet)

## Loading required package: Matrix  
## Loading required package: foreach  
## Loaded glmnet 2.0-2

library(ggplot2)

##   
## Attaching package: 'ggplot2'  
##   
## The following object is masked from 'package:NLP':  
##   
## annotate

#### Reading data

bus <- stream\_in(file("yelp\_academic\_dataset\_business.json"))

## opening file input connection.  
## closing file input connection.

tips <- stream\_in(file("yelp\_academic\_dataset\_tip.json"))

## opening file input connection.  
## closing file input connection.

## user defined functions  
PlotWordCloud <- function (docTermMatrix, maxwords) {  
 ## Find frequent terms  
 freq <- colSums(docTermMatrix)  
 freq <- sort(freq, decreasing=TRUE)  
 ## make word cloud  
 word <- names(freq)  
 wordcloud(word, freq, max.words=maxwords, colors=brewer.pal(6,"Dark2"))  
   
 wf <- data.frame(word=word, freq=freq)  
 return(wf)  
}  
  
GetModelingData <- function (docTermMatrix) {  
 ## add the business ID column to the term matrix  
 dtm\_tips <- cbind(business\_id=tips$business\_id, as.data.frame(docTermMatrix))  
   
 ## merge the business ratings with tip terms dataframe  
 bustips <- merge(bus[,c("business\_id","stars")], dtm\_tips, by="business\_id")  
 ## business ID column is no longer needed  
 bustips$business\_id <- NULL  
  
 ## prepare the training and test data  
 x <- model.matrix(stars~., bustips)[,-1]  
 y <- bustips$stars  
 ## create a list of lambdas for regularization  
 lambdas <- 10^seq(10,-2, length=100)  
 set.seed(1)  
 ## randomly get 70% of the data for training, 30% for validation  
 train<-sample (1: nrow(x), nrow(x) \* 0.7)  
  
 return(list(x=x, y=y, train=train, lambdas=lambdas))  
}  
  
LassoRegression <- function (params) {  
 train <- params$train  
 test <- (-train)  
 lasso.mod <- glmnet(params$x[train,], params$y[train], alpha=1, lambda=params$lambdas)  
 set.seed(2)  
 cv.out <- cv.glmnet(params$x[train,], params$y[train], alpha=1)  
 bestlamda <- cv.out$lambda.min  
 lasso.pred <- predict(lasso.mod, s=bestlamda, newx=params$x[test,])  
 mse <- mean((lasso.pred - params$y[test])^2)  
 out <- glmnet (params$x, params$y, alpha=1, lambda=params$lambdas)  
 lasso.pred.coef <- predict(out, type="coefficients", s=bestlamda)  
 ## get the coefficients vector  
 lasso.coef <- lasso.pred.coef[1:dim(lasso.pred.coef)[1],]  
   
 return(list(mod=lasso.mod, bestlamda=bestlamda, mse=mse, coef=lasso.coef))  
}

#### Text mining on the tip texts

## set the tip texts as the source  
vsource <- VectorSource(tips$text)  
tiptexts <- Corpus(vsource)  
  
## cleaning  
tiptexts <- tm\_map(tiptexts, content\_transformer(tolower))  
tiptexts <- tm\_map(tiptexts, removeNumbers)  
tiptexts <- tm\_map(tiptexts, removePunctuation)  
tiptexts <- tm\_map(tiptexts, stripWhitespace)

#### Naive model

## The predicted rating is simply the average of the ratings in the training data  
naive.pred <- mean(bus$stars)  
## Mean square errors (MSE) of the naive model  
naive.mse <- mean((naive.pred - bus$stars)^2)

Business average overall rating: **3.6733051**

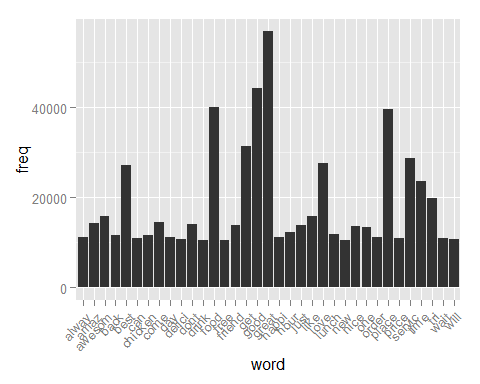
#### The Lasso Regression using Unigrams

## prepare data  
tiptexts\_ugm <- tm\_map(tiptexts, removeWords, stopwords("english"))  
tiptexts\_ugm <- tm\_map(tiptexts\_ugm, stemDocument)  
## create a document-term matrix from the tip texts  
dtm <- DocumentTermMatrix(tiptexts\_ugm)  
## remove the terms with more than 99.6% sparcity from the dtm  
dtm <- removeSparseTerms(dtm, 0.996)  
## save the dtm as a regular matrix for later processing  
dtm2 <- as.matrix(dtm)  
## train model  
data <- GetModelingData(dtm2)  
out\_ugm <- LassoRegression(data)  
  
## plot word cloud and bar chart for most frequent used terms  
wf\_ugm <- PlotWordCloud(dtm2, 100)

## Warning in wordcloud(word, freq, max.words = maxwords, colors =  
## brewer.pal(6, : place could not be fit on page. It will not be plotted.



ggplot(data = subset(wf\_ugm, freq>10000), aes(word, freq)) +   
 geom\_bar(stat="identity") + theme(axis.text.x=element\_text(angle=45, hjust=1))



rm(tiptexts\_ugm, dtm, dtm2, data)

#### The Lasso Regression using Bigrams

## bigrams data (stop words are kept in the matrix)  
tiptexts\_bgm <- tm\_map(tiptexts, stemDocument)  
BigramTokenizer <- function(x) {unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)}  
dtm <- DocumentTermMatrix(tiptexts\_bgm, control = list(tokenize = BigramTokenizer))  
dtm <- removeSparseTerms(dtm, 0.998)  
dtm2 <- as.matrix(dtm)  
data <- GetModelingData(dtm2)  
out\_bgm <- LassoRegression(data)  
wf\_bgm <- PlotWordCloud(dtm2, 50)

## Warning in wordcloud(word, freq, max.words = maxwords, colors =  
## brewer.pal(6, : this place could not be fit on page. It will not be  
## plotted.

## Warning in wordcloud(word, freq, max.words = maxwords, colors =  
## brewer.pal(6, : happi hour could not be fit on page. It will not be  
## plotted.

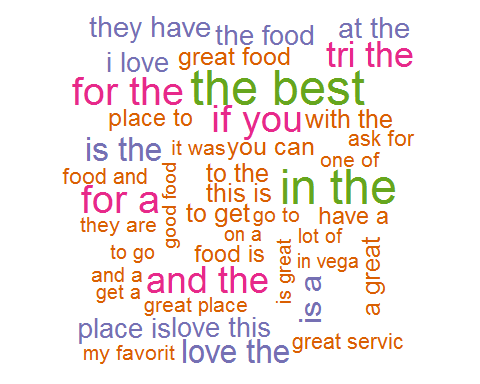
## Warning in wordcloud(word, freq, max.words = maxwords, colors =  
## brewer.pal(6, : on the could not be fit on page. It will not be plotted.

## Warning in wordcloud(word, freq, max.words = maxwords, colors =  
## brewer.pal(6, : get the could not be fit on page. It will not be plotted.

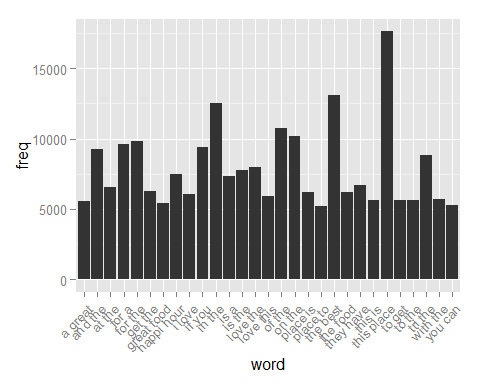
## Warning in wordcloud(word, freq, max.words = maxwords, colors =  
## brewer.pal(6, : of the could not be fit on page. It will not be plotted.

## Warning in wordcloud(word, freq, max.words = maxwords, colors =  
## brewer.pal(6, : with a could not be fit on page. It will not be plotted.

## Warning in wordcloud(word, freq, max.words = maxwords, colors =  
## brewer.pal(6, : to be could not be fit on page. It will not be plotted.



ggplot(data = subset(wf\_bgm, freq>5000), aes(word, freq)) +   
 geom\_bar(stat="identity") + theme(axis.text.x=element\_text(angle=45, hjust=1))



rm(tiptexts\_bgm, dtm, dtm2, data)

### III Results

Here are the results from the three different models:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | MSE | RMSE | Total No. of Features | No. of Features with non-zero coefficient |
| Naive | 0.7942372 | 0.8911999 | N/A | N/A |
| Lasso (unigrams) | 0.3526149 | 0.5938138 | 294 | 43 |
| Lasso (bigrams) | 0.3594339 | 0.599528 | 291 | 19 |

Features in unigram model with non-zero coefficient:  
(*amaz, awesom, bad, beauti, best, buffet, car, delici, dont, drink, drive, everyth, excel, fantast, favorit, flavor, fresh, friend, great, hotel, hour, line, locat, long, love, minut, must, never, night, owner, pay, recommend, room, server, servic, slow, stay, tri, wine, wing, wonder, yelp*)  
Features in bigram model with non-zero coefficient:  
(*a must, at all, custom servic, do not, drive thru, great food, great servic, high recommend, in town, is amaz, love this, my favorit, my new, the best, the owner, this locat, tri the, tri to*)

### IV Discussion

As you can see from the results, the LASSO regression model (unigrams or bigrams) has a lower mean square error (MSE) rate. So the LASSO model is a better model than the baseline naive model. Also, for the Lasso unigrams model, the original number of features used for the model training is 43. Now the actual number of features is 294. So the Lasso algorithm not only minimize the overfitting but also reduce the dimensions of the final model.

Ridge regression can produce a better error rate which is about 0.3481998 in MSE, but it doesn't force some coefficients to be exactly zero. Principal Component Analysis (PCA) can reduce the dimensions but it is harder to interpret because the new features (principal components) are no longer the original features.

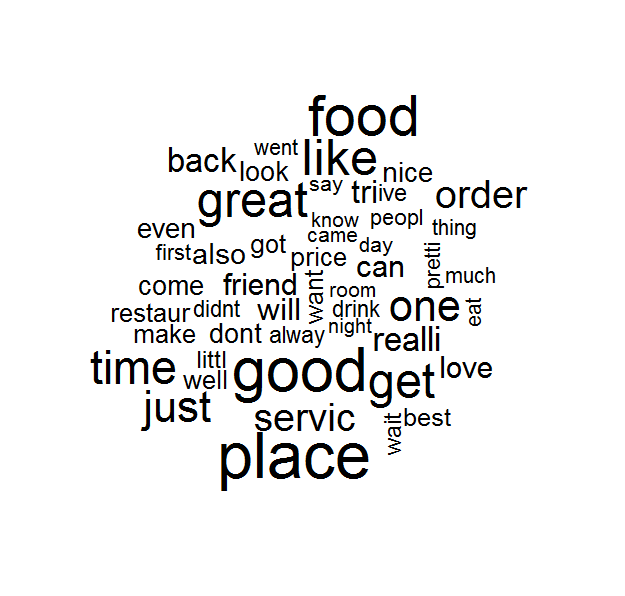
I train the model using bigrams because I think using two adjacent words in a term should have a better sentiment, but it doesn't seem to have a better prediction rate than using the unigrams.

One interesting finding from the model is that the top 42 most frequent terms in the tips are not necessarily the features in the Lasso final model:

42 most frequent terms:  
(*alway, amaz, awesom, back, best, can, chicken, come, day, delici, dont, drink, eat, food, free, fri, friend, get, good, great, happi, hour, just, like, love, lunch, make, new, nice, night, one, order, pizza, place, price, realli, servic, staff, time, tri, wait, will*)

There are only 13 out of 42 most frequent terms are in the features of the final Lasso model. So in other words, most frequent terms do not necessary contribute to the rating prediction.

In the Yelp dataset, the Reviews contain similar sentiment data to the Tips, and each reviews associates with a direct votes for the business rating. So it may be a better tool to develop a predicting model; and actually there are some people have done some work in this direction. The following word cloud show the top 50 most frequent terms in the review text.



This study does not try to override the work done by other people using the Reviews data, but just try to understand the correlation between the ratings and tips.

In conclusion, the predicting model based on the Tips using the Lasso regression works pretty well, the error rate 0.3526149 is comparable to the ones of the models based on Reviews.