Predicting Rating Based on Yelp Tips Using LASSO

Ka Pang (Sammy) Yu

November 14, 2015

### I Introduction

The tips on the Yelp website is the insider information provided by the Yelp's users or reviewers. The tip text contain the sentiments of the users of how they would (or would not) recomend the other users the service or product they are going to buy from a particular business. The tips can be positive or negative and should be consistent with review, if any, posted by the same user. In this study, I am going to find the correlation between a business rating and its tips by developing a predictive model using the words (or terms) in the tips text as the independent variables (or features) and rating as the dependent variable.

### II Methods and Data

To create and train a predictive model for business ratings, I use two datasets, namely "yelp\_academic\_dataset\_business.json" and "yelp\_academic\_dataset\_tips.json" from the [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 to develop the model.

#### Exploratory Analysis

Here is the summary of the tips dataset:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. of Tips | Avg. Tip Length | No. of Bus. in Tips | No. of Users Provided Tips | Tips Covered Period |
| 495,107 | 58.73 chars | 42,111 | 80,836 | 4/15/2009 - 1/22/2015 |

#### Machine Learning Algorithm

The algorithm used to develop the predictive model is called LASSO (Least Absolute Shrinkage and Selection Operator) which was developed by Dr. Robert Tibshirani in 1996. The LASSO is derived from the multiple linear regression and is similar to Ridge Regression which penalizes the coefficients in order to minimize the overfitting. The LASSO has an advantage over Ridge Regression though is that it not only shrinks the coefficients but also forces some of the coefficients to be exactly zero when the regularization parameter is selected properly.

Here is the model:

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 a set of coefficients that can minimize the quantity below:

To get the most popular words, from the tips, text mining technique is used. There are more than 80,000 words found in the tips text but many of them are not frequently used in all the tips (high sparsity). So those terms will be filtered out when their sparsity are more than 99%. As a result, there are about 290 terms are left for the model development.

The study first estimates the predictive 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 adjacent words in each term) to see which one works better. The code is as follow:

## Loading libraries  
lapply(c("jsonlite","NLP","tm","SnowballC","wordcloud","glmnet","ggplot2"),  
 require, character.only = TRUE)

## Read data from the files assuming they reside in the working directory   
bus <- stream\_in(file("yelp\_academic\_dataset\_business.json"))  
tips <- stream\_in(file("yelp\_academic\_dataset\_tip.json"))

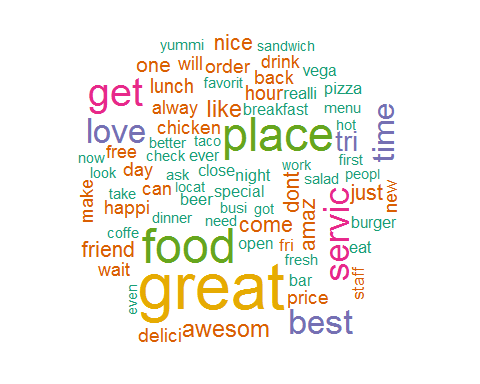
## user defined functions  
PlotWordCloud <- function (docTermMatrix, maxwords) {  
 ## Find frequent terms  
 freq <- colSums(docTermMatrix)  
 freq <- sort(freq, decreasing=TRUE)  
 ## make word cloud  
 word <- names(freq)  
 wordcloud(word, freq, max.words=maxwords, colors=brewer.pal(6,"Dark2"), rot.per=.15)  
 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$train  
 test <- (-train)  
 ## for Lasso, parameter alpha needs to be 1  
 lasso.mod <- glmnet(params$x[train,], params$y[train], alpha=1, lambda=params$lambdas)  
 set.seed(2)  
 ## perform cross validation  
 cv.out <- cv.glmnet(params$x[train,], params$y[train], alpha=1)  
 ## get the best lamdba from the validation result  
 bestlamda <- cv.out$lambda.min  
 ## predict the ratings using the test data  
 lasso.pred <- predict(lasso.mod, s=bestlamda, newx=params$x[test,])  
 ## calcuate the mean squre errors on the test predictions  
 mse <- mean((lasso.pred - params$y[test])^2)  
 ## use the full dataset to re-train the model and get the coeffients from it  
 out <- glmnet (params$x, params$y, alpha=1, lambda=params$lambdas)  
 lasso.pred.coef <- predict(out, type="coefficients", s=bestlamda)  
 ## get the vector of coefficients  
 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)  
## data cleaning  
tiptexts <- tm\_map(tiptexts, content\_transformer(tolower))  
tiptexts <- tm\_map(tiptexts, removeNumbers)  
tiptexts <- tm\_map(tiptexts, removePunctuation)  
tiptexts <- tm\_map(tiptexts, stripWhitespace)

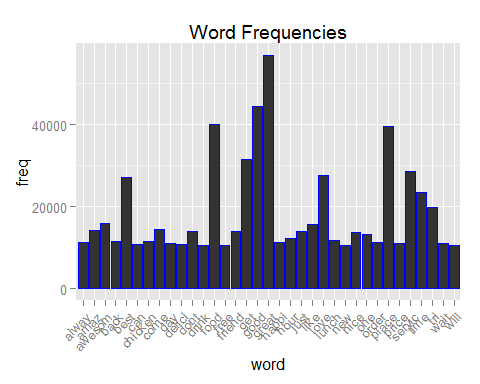
## Naive model. The predicted rating is simply the average of the ratings in the training data  
naive.pred <- mean(bus$stars)  
## Mean square errors (MSE) of the naive model  
naive.mse <- mean((naive.pred - bus$stars)^2)

Predictive rating from naive model: **3.6733051**

## Lasso Regression using Unigrams   
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 the model  
data <- GetModelingData(dtm2)  
out\_ugm <- LassoRegression(data)  
## plot word cloud and bar chart for most frequent used terms  
wf\_ugm <- PlotWordCloud(dtm2, 75)



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



## remove the big objects to save memory space  
rm(tiptexts\_ugm, dtm, dtm2, data)

## bigrams data (stop words are not removed)  
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)  
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 coef. |
| 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 coefficients:  
(***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***)

### IV Discussion

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

Ridge regression (code is not shown here) produces a better error rate which is about 0.3481998 of MSE, but it doesn't force some coefficients to be exactly zero. Principal Component Analysis (PCA) can reduce the dimensions but it is hard to interpret the model because the new features (principal components) are some kinds of transformation from the original features.

The Lasso model using bigrams is also trained here because I think using two adjacent words in a term may have a better sentiment and predicting accuracy, but it doesn't seem to have a much better prediction rate than using the unigrams by looking at the results above. However, it does have a shorter dimensions than the unigrams model.

One interesting finding from the unigrams 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 them are in the features of the final 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 review is also associated with the direct votes for a business rating. So the reviews can also be used to develop a predictive model. And actually there are some people have done the work in this direction.

This study does not try to compare the predictive models using the reviews and the tips, but try to understand if there is a correlation between the ratings and tips; and the results confirm that.

In conclusion, the predicting model based on the Yelp tips using the Lasso regression works pretty well, the error rate 0.3526149 is comparable to the ones of the models based on reviews from other studies.

### References

1. A. De Catro, A. Du and A. Manicka, "Predicting Ratings Based on Yelp Tips".
2. Yifei Feng and Zhengli Sun, "Yelp User Rating Prediction".
3. Mingming Fan and Maryam Khademi, "Predicting a Business Star in Yelp from Its Reviews Text Alone"