Simple Star Rating Prediction based on review text (Data Science Capstone).

DSCourse001 User

November 22, 2015

Introduction

This document contains research for Data Science Capstone Project which is based on Yelp DataSet Challenge.

The questions that I've asked himself in this Data Science Capstone was:

1. Can we predict the star rating based on review text?
2. Has the review star rating correlation with weather in particular region?

This document is an attemp to ask on first question using R language as a tool.

Methods and Data

Assumption and Limitation

We need to take some asumtions and limitations because of lack of information or other resources.

1. We will take under consideration data only for US region.

Please note that text analysis was made using simple Word Count algorithm and is very rude.

Loading the Data

Data provided in JSON format, so we can use suggested jsonlite R package. To ask on first and second question we need only information from *review* and *business* datasets.

```
require(jsonlite)       # load require package
business <- flatten(stream_in(file("yelp_academic_dataset_business.json")),recursive=T) # load business
review <- flatten(stream_in(file("yelp_academic_dataset_review.json")),recursive=T) # load review
```

We can see that reviews is available from 2004 to 2015 years.

```
require(lubridate)
require(knitr)
kable(as.data.frame(table(year(review$date))))
```

Var1	Freq
2004	13
2005	680

Var1	Freq
2006	4239
2007	17724
2008	45117
2009	72948
2010	137764
2011	209429
2012	244106
2013	336273
2014	486306
2015	14665

It's more convinient to combine this two data set together. But before this we need clear some data.

```
business <- business[,c("business_id","full_address","open","name","city","state","longitude","latitude

business <- business[business$state %in% c("AL","MO","AK","MT","AZ","NE","AR","NV","CA","NH","CO","NJ",

review <- review[,c("review_id","business_id","user_id","date","stars","text")] # getting only that col
business_review <- merge(review,business,by="business_id") # merging two datasets
business_review <- business_review[!is.na(business_review$open),] # clear NA values
```

```
plot(table(business_review$stars),type = "b",main="Review Star Rating Summary", xlab="Number of Stars",
```

Processing the Data

According to documenation Text mining infrastucture in R we using *tm* package for Text Analysis, and also use *qdap* for more simple representation of *tm* package classes. Classification was done with help of *caret* package.

```
require(qdap)
require(tm)
require(SnowballC)

Sys.setlocale("LC_ALL", 'en_US.UTF-8') # Set Locale to avoid errors
```

```
## [1] "LC_CTYPE=en_US.UTF-8;LC_NUMERIC=C;LC_TIME=en_US.UTF-8;LC_COLLATE=en_US.UTF-8;LC_MONETARY=en_US.U
```

```
options(mc.cores=1) # set number of cores to 1 to avoid errors

business_review$words <- tolower(business_review$text)
business_review$words <- rm_stopwords(business_review$words,tm::stopwords("english"),separate=F,strip=TR
business_review$words <- gsub('([[:alnum:]])\\1+', '\\1', business_review$words) # remove repeated char
business_review$words <- replaceforeignchars(business_review$words,fromto) # replace foreign characters
business_review$words <- gsub('\\b\\w\\w?\\b', '', business_review$words) # remove one character words
business_review$words <- wordStem(business_review$words, language = "english") # stem words
```

Now we can build a model.

```
require(caret)
require(klaR)
require(MASS)
index <- createDataPartition(business_review$stars, p=0.7, list=FALSE)

br_train <- business_review[index,]
br_test <- business_review[-index,]

br_wfdf_train <- with(br_train,wfdf(words,stars))
br_wfdf_test <- with(br_test,wfdf(words,stars))

# Remove words with low frequency
freq <- 5
br_wfdf_train <- br_wfdf_train[apply(as.wfm(br_wfdf_train),1,function(x) {if (max(x)>freq) TRUE else FA
br_wfdf_test<- br_wfdf_test[apply(as.wfm(br_wfdf_test),1,function(x) {if (max(x)>freq) TRUE else FALSE})
```

A little summary in word cloud.

```
require(qdap)
require(wordcloud)
br_wfdf <- with(business_review,wfdf(words,stars))
br_wfdf <- br_wfdf[apply(as.wfm(br_wfdf),1,function(x) {if (max(x)>freq) TRUE else FALSE}),]
br_freq <- rowSums(as.wfm(br_wfdf))

set.seed(1)
wordcloud(names(br_freq), br_freq, max.words=100, rot.per=0.25, colors=brewer.pal(8, "Paired"))
```

Processing Data Futher.

```
br_wfm_train_norm <- as.wfm(br_wfdf_train) %*% diag(1/colSums(as.wfm(br_wfdf_train))) # normalize matri
br_wfm_test_norm <- as.wfm(br_wfdf_test) %*% diag(1/colSums(as.wfm(br_wfdf_test))) # normalize matrix

# Getting Normilized Rate
colname <- as.character(c(1:5))
br_wfdf_train$rate <- apply(br_wfm_train_norm,1,function(x) {round(sum(x*as.integer(colname))/sum(x))})
br_wfdf_test$rate <- apply(br_wfm_test_norm,1,function(x) {round(sum(x*as.integer(colname))/sum(x))})

br_wfdf_train <- as.data.frame(br_wfdf_train[c("Words","rate")]) # filter
br_wfdf_test <- as.data.frame(br_wfdf_test[c("Words","rate")]) # filter

br_wfdf_train$Words <- as.factor(br_wfdf_train$Words) # convert to factor
br_wfdf_train$rate <- factor(br_wfdf_train$rate,levels=c(1:5),labels=c(1:5)) # convert to factor
br_wfdf_test$Words <- as.factor(br_wfdf_test$Words) # convert to factor
br_wfdf_test$rate <- factor(br_wfdf_test$rate,levels=c(1:5),labels=c(1:5)) # convert to factor

br_model_nb <- train(br_wfdf_train, br_wfdf_train$rate, method="nb", trControl=trainControl(method="cv"
br_result_nb <- predict(br_model_nb,br_wfdf_test)
br_cm_nb <- confusionMatrix(br_result_nb, br_wfdf_test$rate)
```

Accuracy is equal to 1, but don't shine, this is a very simple algorithm. It's a normal to have high accuracy on a training Data.

Let's do some examples with Text Analysis on the fly.

```
require(qdap)
require(tm)
require(SnowballC)

getPredictedStars <- function(x=character) {
  words <- tolower(x)
  words <- rm_stopwords(words,tm::stopwords("english"),separate=F,strip=TRUE,apostrophe.remove=TRUE)
  words <- gsub('([[:alnum:]])\\1+', '\\1',words)
  words <- replaceforeignchars(words,fromto)
  words <- gsub('\\b\\w\\w?\\b', '', words)
  words <- wordStem(words, language = "english")
  round(mean(as.integer(predict(br_model_nb,wfdf(words)$Words))))
}

getPredictedStars("Thery nice place. Good kitchen.")
```

```
## [1] 4
```

Results

We can see that analysis was made using Regression to the Mean and for all words. So we are dial with Gaussian Distribution. This is a good starting point to do futher analysis.

Discussion

This research can help to do futher analysis with Yelp data, more precise.

Notes

replaceforeignchars function that was used in analysis can be obtained from the following URL: http://stackoverflow.com/questions/17517319/r-replacing-foreign-characters-in-a-string