Sentiment Analysis over Yelp Business Reviews

paperezq

19 de noviembre de 2015

Introduction

This report aims at describing a set of classifiers in which a review vote or qualification is given based-on a set of words identified on a review. This is part of the Data Science capstone and the dataset used comes from the Yelp Dataset Challenge.

With the idea of connecting people to great local business, yelp provides five datasets describing business, checkin information, customers' reviews of businesses, tips and user data.

My question of interest for this project is how well can a tool guess a review's rating from its text alone?. Basically, the idea is measure the how well a classifier performs based-on some positive and negative words from text reviews. Nevertheless, words of other kind of lexicons were used. We will see that the models built used more than positive and negative words.

The following sections describes the analysis made over the reviews dataset in order to build classifiers of punctuations or stars qualifications. In the first section some basic concepts are mentioned. The second section explores tools and methods used over the dataset. An explanation and some performance measures of the classifiers built are describe in the third section. Finally, results are discussed and some conclusions are given.

Some concepts: Sentiment Analysis

Natural Language Processing (NLP) is a technology in which text, in a manner of documents, are processed aiming at to find new knowledge and to answer questions related to patterns that can be found. This is also named text mining. Some of the tasks in NLP includes words and sentence tokenazition, prediction on typing words, text classification, information extraction, and some others.

Text mining could be consider an extension of data mining because it includes the use of methods for classification and clustering. Thus, the variables that are likely manipulated in this kind of mining are counts of some words previously identified in a preprocessing step.

Inside this technology, exists a task so called sentiment analysis. This tasks has the challenge of predicting an election or rate based on some texts given by users of a particular matter. i.e. movies reviews, ease of use of an electronic device, qualifications of a service, etc. In sentiment analysis, the use of lexicons is important in order to identify affective states like emotions, attitudes, pleasure, pain, virtue, vice, and so on. For more information, see the coursera of NLP at standord.

In the next section, is described the text mining tools and the dataset of lexicons got for this project.

Data is explored: tools and methods used to get an exploratory data analysis

In order to get a good classifier model for ratings based on text, R tool is appropiate for such as job. The datasets are in json file format. The library *jsonlite* is suitable for treating this kind of files. Primeraly, jsonlite library has to main functions: toJSON and fromJSON. However, the dataset of reviews is the largest of the five datasets with 1.32GB. Because of its size and in terms of performance, the best function for reading

these json files is the **stream_in** with the name of the file as a parameter. This function lets read the json file faster than the fromJSON function does. Although, it considers a slightly different format.

```
#data_business <- stream_in(file("yelp_academic_dataset_business.json"))
#data_checkin <- stream_in(file("yelp_academic_dataset_checkin.json"))
data_review <- stream_in(file("yelp_academic_dataset_review.json"))
#data_tip <- stream_in(file("yelp_academic_dataset_tip.json"))
#data_user <- stream_in(file("yelp_academic_dataset_user.json"))
```

The following sections descibres the process of reading, sampling and transforming the dataset.

Datasets

The whole Yelp dataset is composed of five data frames in which business, users, reviews, chechins and tips are registered.

Dataset of business has 15 variables for 61184 observations, many of those with categories and schedules of operation. The checkin dataset is composed of three other dataframes for 45166 observations. Tips dataset is described by texts for each customer, business and date and the likes given for each tip. The users dataset has features of 366715 customers and 11 features. Those features are the votes (funny, useful and cool) made by the customer, the count of reviews, names and list of friends.

This project focus on business reviews dataset. It has for each customer, business and date a text review which will turn our gold mine to be exploited. A summary of this dataset is presented as follows:

```
s <- dget(file = "summary_data_review.txt")
s
```

```
##     userid            reviewid            stars           date
##  Length:1569264     Length:1569264     Min.   :1.000   Length:1569264
##  Class :character   Class :character   1st Qu.:3.000   Class :character
##  Mode  :character   Mode  :character   Median :4.000   Mode  :character
##                                        Mean   :3.743
##                                        3rd Qu.:5.000
##                                        Max.   :5.000
##      text               type            businessid
##  Length:1569264     Length:1569264     Length:1569264
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##    votesfunny        votesuseful        votescool
##  Min.   :  0.0000   Min.   :  0.000   Min.   :  0.0000
##  1st Qu.:  0.0000   1st Qu.:  0.000   1st Qu.:  0.0000
##  Median :  0.0000   Median :  0.000   Median :  0.0000
##  Mean   :  0.4789   Mean   :  1.072   Mean   :  0.5942
##  3rd Qu.:  0.0000   3rd Qu.:  1.000   3rd Qu.:  1.0000
##  Max.   :141.0000   Max.   :166.000   Max.   :137.0000
```

Notice that some other covariates exists in the reviews dataset. This features will be used in the model. Features like votes.funny, votes.cool, votes.useful. These last features came in a form of data frame. This means that *jsonlite* tool read some structures that involves data frames inside another. To reduce this

overload of data in memory, the function used to enhanced the reading is **flatten**.When flatten a data frame, the variables of their included data frames becomes part of the main and their names are concatenated with a prefix of their original dataframe name.

```
#flatten data frame to reduce memory and enhance performance
sDataReview <- flatten(sDataReview)
#cleaning variable names, removing all punctuation characters
colnames(sDataReview) <- gsub("[^A-Za-z]", "", colnames(sDataReview))
```

Summarizing stats of each dataset: Dataset | Size (MB) | Total Obs. | Number of variables ————- | ————- | ————- | ————- Reviews | 1393 | 1569264 | 8 Business | 53 | 61184 | 15 Checkin | 20 | 45166 | 3 Tips | 94 | 495107 | 6 User | 159 | 366715 | 11

Stratified sampling...

with a sample size of 5% over the 1569264 observations, a stratified sample (by class) is created in order to get an analysis and to get faster results, though. The sampling was **without replacement**.

Text mining tool

A powerful tool to process text in R is **tm**.With tm we can build a corpus in which a set of task can be applied. In linguistics, a corpus (plural corpora) is a large and structured set of texts in which statistical analysis and hypothesis testing can be done.

The following, is a list of some other tools loaded in the project in order to do the sentiment analysis: * tm.plugin.sentiment : score of lexicons tool * tm.lexicon.GeneralInquirer : Harvard open lexicons * SnowballC : stemming tool * slam : manipulating term document matrix objects * bigmemory : converting term document matrix objects into matrix * stringi : manipulating strings

Preprocessing

A corpus is loaded from the sample of data reviews previously extracted.

```
#header of the corpus
m <- list(id = "id", content = "text")
myReader <- readTabular(mapping = m)
#loading corpus from random sample of reviews
corpus <- Corpus(DataframeSource(sDataReview), readerControl = list(reader = myReader))
```

Some functions are defined to get data cleaned from characters that are not in the english alphabet, i.e. just letters. Functions that catch english stop words, urls, emails, twitter tags, duplicated quotes, non-ascii characters, non english character, duplicated letters and duplicated words.

```
### preprocessing functions #####
skipWords <- function(x) removeWords(x, stopwords("english"))
# removing URLs
urlPat<-function(x) gsub("(ftp|http)s?:(\\/)+[\\d\\w.]*\\b", " ", x, perl = T)
# removing Emails
emailRgx <- "[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a
emlPat<-function(x) gsub(emailRgx, " ", x)
# removing Twitter tags
```

```
tags<-function(x) gsub("(RT )|via", " ", x)
#replace curly brace with single quote
singleQuote <- function(x) gsub("\u2019|`", "'", x, perl = T)
#replace non printable, except ' - and space with empty string
nonprint <- function(x) gsub("[^\\p{L}\\s'-]", "", x, perl = T)
#remove non English words
nonEng <- function(x) gsub("[^A-Za-z']", " ", x)
#remove duplicated letters in words
dupLetters <- function(x) gsub("(\\w)\\1+", "\\1", x, perl = T)
#remove duplicated consecutive words
dupWords <- function(x) gsub("(\\w+)\\s+\\1", "\\1", x, perl = T)
```

At the end, a list of this functions is created and an additional tm function is append in order to stem words. The latter is done loading the Snowballc library.

```
#list of functions to preprocess the corpus
funcs <- list(tolower, urlPat, emlPat, tags, singleQuote, nonprint, nonEng, dupLetters, dupWords, remov
#cleansing process is done
rcorpus <- tm_map(corpus, FUN = tm_reduce, tmFuns = funcs)
#fix a bug of tm library
rcorpus <- tm_map(rcorpus, PlainTextDocument)
```

Next, frequent words are identified.

The wordcloud for each rate

A document term matrix is created. This is also done with a tm function. As control parameters, the set of words to be extracted might be those no larger than 10 characters and with a minimum frequencie of total 50 counts.

```
dtm <- DocumentTermMatrix(rcorpus, control = list(wordLengths = c(3,10), minDocFreq=50))
```

```
require(slam)
```

```
## Loading required package: slam
```

```
pos.terms <- read.csv(file = "pos_terms.csv", encoding="UTF8")
#neg.terms <- read.csv(file = "neg_terms.csv")
#pleasur.terms <- read.csv(file = "pleasur_terms.csv")
#pain.terms <- read.csv(file = "pain_terms.csv")
#feel.terms <- read.csv(file = "feel_terms.csv")
#arousal.terms <- read.csv(file = "arousal_terms.csv")
#emot.terms <- read.csv(file = "emot_terms.csv")
#virtue.terms <- read.csv(file = "virtue_terms.csv")
#vice.terms <- read.csv(file = "vice_terms.csv")

#terms <- list(pos.terms, neg.terms, pleasur.terms, pain.terms, feel.terms, arousal.terms, emot.terms,
source("GlobalFun.R")
#lapply(terms, function(t) {getWordCloud(t); getWordHist(t);})
```

Clusters of sentiment words

```
######################
#clustering
#http://michael.hahsler.net/SMU/CSE7337/install/tm.R

## do document clustering

m <- as.matrix(df.terms)
rownames(m) <- 1:nrow(m)

### don't forget to normalize the vectors so Euclidean makes sense
norm_eucl <- function(m) m/apply(m, MARGIN=1, FUN=function(x) sum(x^2)^.5)
m_norm <- norm_eucl(m)

## hierarchical clustering
library(proxy)
library(cluster)

### this is going to take 4-ever (O(n^2))
distances <- dist(t(m_norm), method="cosine") ##warning: t() is the transpose function, we look for clus
#hc <- hclust(d, method="average")
hc <- hclust(d=distances, method="ward.D2")
hc
plot(hc, hang=-1)

groups <- cutree(hc, k=5)   # "k=" defines the number of clusters you are using
rect.hclust(hc, k=5, border="red") # draw dendogram with red borders around the 5 clusters

cl <- cutree(hc, 30)
table(cl)
##############################
```

Building classifiers: how well a tool can predict a 5-star rating based on any review

doSNOW

Results and discussion

Wrapping up