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:

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	total_vars
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-]*[
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.

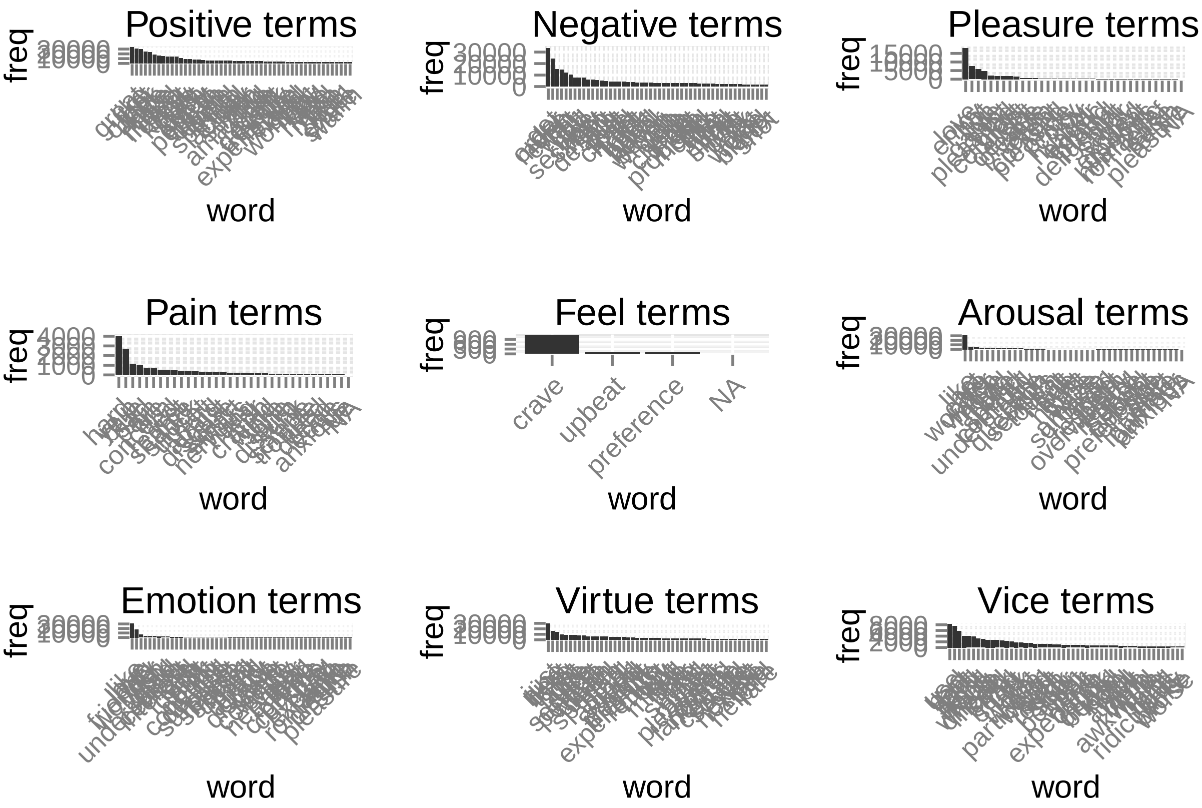
```
dtm <- DocumentTermMatrix(rcorpus, control = list(wordLengths = c(3,10), minDocFreq=50))
```

```
## Warning: Removed 16 rows containing missing values (position_stack).
```

```
## Warning: Removed 17 rows containing missing values (position_stack).
```

```
## Warning: Removed 47 rows containing missing values (position_stack).
```

```
## Warning: Removed 11 rows containing missing values (position_stack).
```



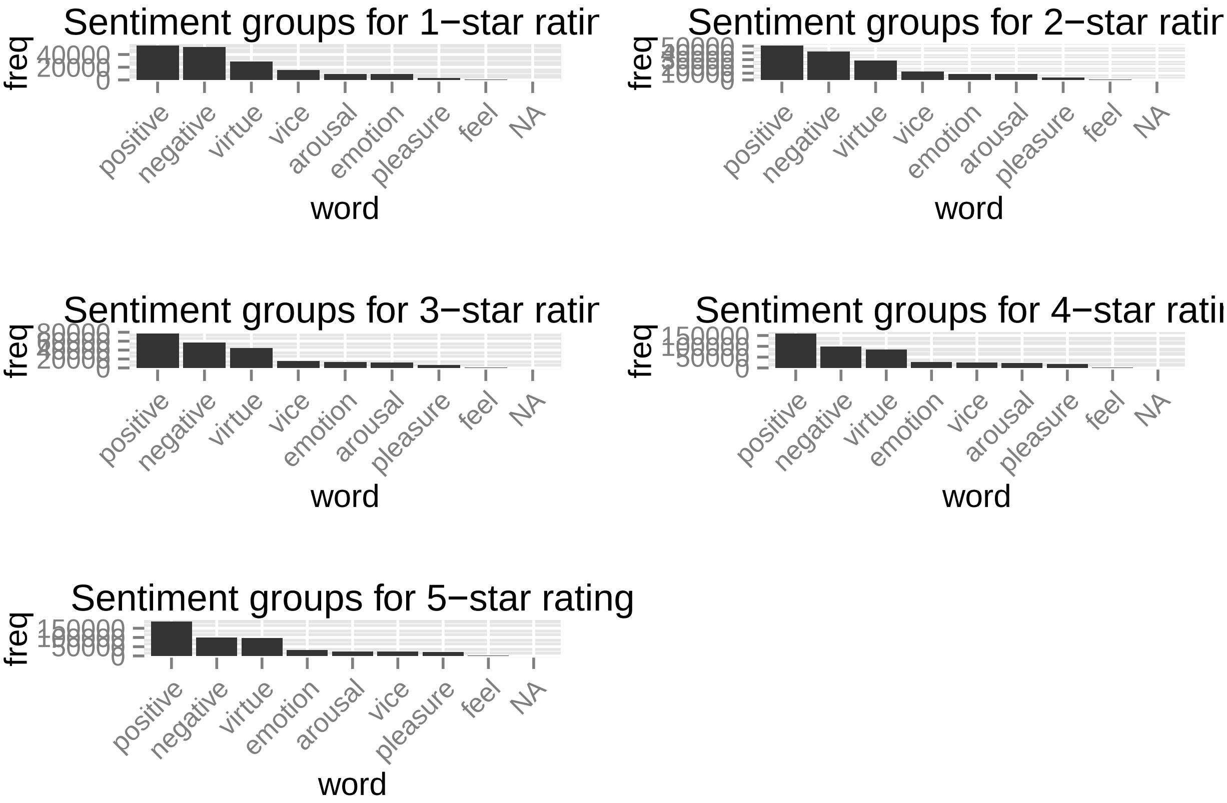
By star rating, the next plot shows the distribution of sentiment word groups. Notice that 5-star rating has more positive words thant the others. It decreases proportionaly with lower star rating.

```
## Warning: Removed 42 rows containing missing values (position_stack).
```

```
## Warning: Removed 42 rows containing missing values (position_stack).
## Warning: Removed 42 rows containing missing values (position_stack).

## Warning: Removed 42 rows containing missing values (position_stack).

## Warning: Removed 42 rows containing missing values (position_stack).
```



Also, notice that when star rating is increasing (from 1 to 5) the emotions group begins to scale up, i.e. the frequencies rises up instead of vice group. There's no insidences of pleasures and vice. ##Clusters of sentiment words

Hierarchical clustering is applied to the more than 50 frequent words in order to vizualice groups of words in each of the sentiment's groups. A cosine distance is used and counts are normalized.

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

A dataset is created merging all sentiment words matched above. From these, the most frequent (quantile above 85% of frequencies) are filtered. Finally, the new dataset is merged with the original variables from the reviews dataset, excluding text, ids and type covariates.

```
#merging all sentiment words
all.sentiment <- c(terms_in_General_Inquirer_categories("Positiv"),
                   terms_in_General_Inquirer_categories("Negativ"),
                   terms_in_General_Inquirer_categories("Pleasur"),
                   terms_in_General_Inquirer_categories("Pain"),
                   terms_in_General_Inquirer_categories("Feel"),
                   terms_in_General_Inquirer_categories("Arousal"),
                   terms_in_General_Inquirer_categories("EMOT"),
                   terms_in_General_Inquirer_categories("Virtue"),
                   terms_in_General_Inquirer_categories("Arousal"))
all.sentiment <- unique(all.sentiment)
df.terms <- dtm[, dtm$dimnames$Terms %in% all.sentiment]

#filtering words with frequencies above the 85% of the dense probability function (quantile = 85)
df.terms <- df.terms[,filterByFreq(df.terms, .85)]
```

For modelling, we take advantage of caret package for the modelling process. It provides a set of functions in which the mining process is divided in cross-validation step, transformation, training algorithms and evaluation. The following sections descibres each step.

Splitting step

Because of the dataset size and the number of variables found (316!), some problems arouse like the curse of dimensionality. To face this problem, the use of parallel processing is imperative. For that, we use doSNOW library that is applied by the train function of the caret package (see parallel processing).

So, the dataset is divided into training and testing sets, holding a random sample of 75% of size for the training set.

```
##Cross Validation
set.seed(1234)
inTrain <- createDataPartition(y=data.fimp$stars, p=3/4, list=FALSE)
training = data[inTrain,]
testing = data[-inTrain,]
```

Some transformations before training

Before training a model, there is a main issue: curse of dimensionality. To face this problem, near zero variance and less important covariates must be removed. To do that, **klaR** and **FSelector** tools must be loaded in order to filter the less important variables: those that are colinear with others (correlated in a manner) and in which the gain of information over the classes (ratings-stars) is the least of all covariates.

```
require(caret)
#identifying variables with the lowest variance value
nzv <- nearZeroVar(data, saveMetrics= TRUE)
#removing variables with near zero variance
data.nzv <- data[,-which(names(data) %in% c(rownames(nzv[nzv$nzv,])))]
##Identifying weights based on information gain for all variables
library(FSelector)
weights <- information.gain(stars~., data.nzv)
#Variables with weights lower than 75% are removed
subset <- cutoff.k.percent(weights, 0.75)
data.fimp <- data.nzv[, which(names(data.nzv) %in% c("stars",subset))]
```

Training classifiers

In order to get a competitive model, three types of techniques were compared: naive bayes, penalized discriminant analysis and random forest.

```
##             Length Class      Mode     
## apriori      5     table      numeric  
## tables      35     -none-     list     
## levels       5     -none-     character
## call         5     -none-     call     
## x           35     data.frame list     
## usekernel    1     -none-     logical  
## varnames    35     -none-     character
## xNames      35     -none-     character
## problemType  1     -none-     character
## tuneValue    2     data.frame list     
## obsLevels    5     -none-     character
```

```
##                 Length Class      Mode     
## call                 4 -none-     call     
## type                 1 -none-     character
## predicted        58847 factor     numeric  
## err.rate          3000 -none-     numeric  
## confusion           30 -none-     numeric  
## votes           294235 matrix     numeric  
## oob.times        58847 -none-     numeric  
## classes              5 -none-     character
## importance          35 -none-     numeric  
## importanceSD         0 -none-     NULL     
## localImportance      0 -none-     NULL     
## proximity            0 -none-     NULL     
## ntree                1 -none-     numeric  
## mtry                 1 -none-     numeric  
## forest              14 -none-     list     
## y                58847 factor     numeric  
## test                 0 -none-     NULL     
## inbag                0 -none-     NULL     
## xNames              35 -none-     character
## problemType          1 -none-     character
## tuneValue            1 data.frame list     
## obsLevels            5 -none-     character
```

```
##                 Length Class      Mode     
## call                 4 -none-     call     
## type                 1 -none-     character
## predicted        58847 factor     numeric  
## err.rate          3000 -none-     numeric  
## confusion           30 -none-     numeric  
## votes           294235 matrix     numeric  
## oob.times        58847 -none-     numeric  
## classes              5 -none-     character
## importance          35 -none-     numeric  
## importanceSD         0 -none-     NULL     
## localImportance      0 -none-     NULL     
## proximity            0 -none-     NULL     
## ntree                1 -none-     numeric  
## mtry                 1 -none-     numeric  
## forest              14 -none-     list     
## y                58847 factor     numeric  
## test                 0 -none-     NULL     
## inbag                0 -none-     NULL     
## xNames              35 -none-     character
## problemType          1 -none-     character
## tuneValue            1 data.frame list     
## obsLevels            5 -none-     character
```

Evaluating trained models

Naive Bayes

```
##           X1           X2          X3           X4           X5 obs pred
## 1 0.01357197 0.0008158132 0.003303536 2.301168e-02 9.592970e-01  X1   X5
## 2 0.63862738 0.3456341987 0.015711004 2.741881e-05 6.334054e-11  X1   X1
## 5 0.94306354 0.0359984701 0.016981936 3.482418e-03 4.736355e-04  X2   X1
## 6 0.02047018 0.0017737457 0.007062944 3.985572e-02 9.308374e-01  X2   X5
## 7 0.84678634 0.1185861530 0.005835959 9.540682e-03 1.925087e-02  X2   X1
## 9 0.02345489 0.0761110465 0.422820301 4.478595e-01 2.975424e-02  X2   X4
```

```
##                logLoss               Mean_ROC               Accuracy 
##             3.26816750             0.66657311             0.39815429 
##                  Kappa       Mean_Sensitivity       Mean_Specificity 
##             0.15145090             0.31556627             0.82972587 
##    Mean_Pos_Pred_Value    Mean_Neg_Pred_Value    Mean_Detection_Rate 
##             0.32151481             0.83907842             0.07963086 
## Mean_Balanced_Accuracy 
##             0.57264607
```

Random forest

```
##      X1    X2    X3    X4    X5 obs pred
## 1 0.000 0.000 0.000 0.000 1.000  X1   X5
## 2 0.216 0.150 0.122 0.312 0.200  X1   X4
## 5 0.230 0.090 0.156 0.308 0.216  X2   X4
## 6 0.086 0.020 0.034 0.306 0.554  X2   X5
## 7 0.106 0.096 0.104 0.334 0.360  X2   X5
## 9 0.152 0.094 0.316 0.258 0.180  X2   X3
```

```
##                logLoss               Mean_ROC               Accuracy 
##             2.66724967             0.66690300             0.42614592 
##                  Kappa       Mean_Sensitivity       Mean_Specificity 
##             0.13097726             0.27839511             0.82492488 
##    Mean_Pos_Pred_Value    Mean_Neg_Pred_Value    Mean_Detection_Rate 
##             0.36366529             0.84628271             0.08522918 
## Mean_Balanced_Accuracy 
##             0.55165999
```

Penalized Discriminant Analysis

```
##           X1         X2         X3        X4         X5 obs pred
## 1 0.05535835 0.07541543 0.15350899 0.3246275 0.39108970  X1   X5
## 2 0.18512231 0.22631075 0.23075145 0.2617414 0.09607405  X1   X4
## 5 0.23245494 0.17909333 0.22445118 0.2069076 0.15709291  X2   X1
## 6 0.07365680 0.11170120 0.18296645 0.3077847 0.32389085  X2   X5
## 7 0.20913052 0.13558003 0.09351322 0.2165087 0.34526751  X2   X5
## 9 0.09916384 0.11959840 0.23301008 0.3158082 0.23241944  X2   X4
```

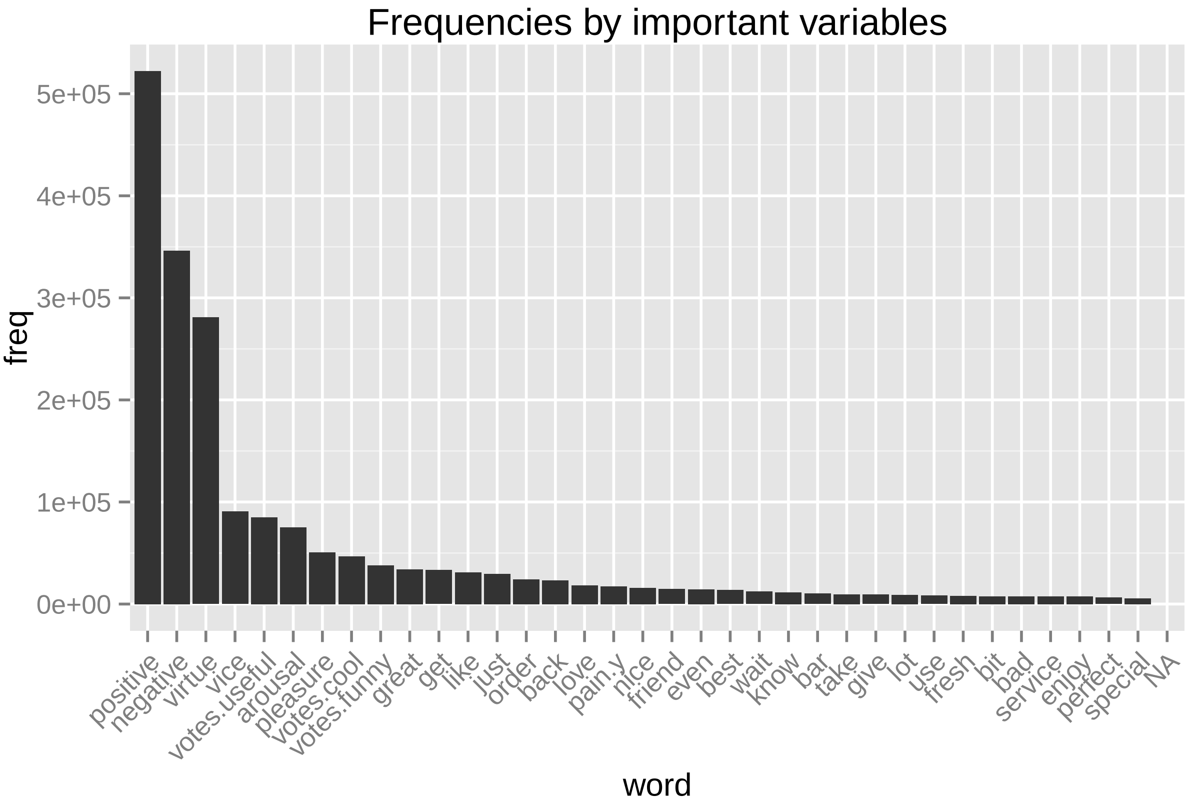
```
##                logLoss               Mean_ROC               Accuracy
##             1.28770296             0.72182841             0.44434814
##                  Kappa       Mean_Sensitivity       Mean_Specificity
##             0.18566035             0.32074626             0.83601017
##    Mean_Pos_Pred_Value    Mean_Neg_Pred_Value    Mean_Detection_Rate
##             0.38392795             0.84915481             0.08886963
## Mean_Balanced_Accuracy
##             0.57837822
```

Results and discussion

The near zero variance reduction remove most of feelings and emotions. They had the lowest variance. The words with the highest gain of information with respect of the classes are plotted next:

```
hs <- getWordHist(data.fimp[,-1], "Frequencies by important variables")
hs
```

```
## Warning: Removed 15 rows containing missing values (position_stack).
```



Notice that words like great, get, like, order, back, love, pain, nice and friend were more important than the others.

Wrapping up

There's no insidences of pleasures and vice.