Sentiment Analysis for Amazon Review and Drug Review

fa22-prj-rongl2-xdai12-zixingd2

2022-12-07

Contents

1 Introduction

In recent years, machine learning techniques have become increasingly popular and relevant to solving text and sentiment-related problems. It has boosted performance on several tasks and significantly reduced the necessity for human efforts. For this project, we focused on text classification, especially sentiment analysis, on two datasets, *Amazon Review* and *Drug Review*. Although the *Amazon Review* dataset is popular and was being used for many research papers and projects, most code works were carried out using Python. Therefore, we decided to explore more on implementing four classic Natural Language Processing (NLP) methods using R packages. We first replicated the R code from existing literature for the *Amazon Review* dataset. We adapted them to a newer but less popular UCI Machine Learning Repository dataset. Our goal for the project is to compare classifiers, including `BoW`, `Word2Vec`, `GloVe`, `fastText` for two different datasets.

2 Data

2.1 Dataset Overview

For the *Amazon Review* dataset, we use the dataset constructed and made available by Zhang, Zhao, and LeCun (2015). The dataset contains about 1,800,000 training samples and 200,000 testing samples with three attributes, which are classification labels (1 for negative reviews and 2 for positive reviews), the title of each review text, and the review text body. Due to the limit of computer computation ability, we pulled out the first 100,000 data samples and split 80% of the data into the training set and 20% of the data into the testing set. ## Dataset Prepossessing

2.1.1 Amazon reviews

```
set.seed(1)
N_amazon <- 100000
N_train_amazon <- 0.8*N_amazon
reviews_amazon <- readLines("amazon_review_polarity_csv/train.csv", n = N_amazon)
reviews_amazon <- data.frame(reviews_amazon)
```

```
library(tidyr)
reviews_amazon <- separate(data = reviews_amazon, col = reviews_amazon,
                       into = c("Sentiment", "SentimentText"), sep = 4)
# Head rows of the data:
head(reviews_amazon)
```

```
##   Sentiment
## 1      "2",
## 2      "2",
## 3      "2",
## 4      "2",
## 5      "2",
## 6      "2",
##
## 1
## 2
## 3                                                    "Amazing!","This soundtrack is my favorite mus
## 4                                                                                "Excellent Soundt
```

```
## 5
## 6 "an absolute masterpiece","I am quite sure any of you actually taking the time to read this have pl
```

```
# Retaining only alphanumeric values in the sentiment column
reviews_amazon$Sentiment <- gsub("[^[:alnum:] ]","",reviews_amazon$Sentiment)
# Retaining only alphanumeric values in the sentiment text
reviews_amazon$SentimentText <- gsub("[^[:alnum:] ]"," ",reviews_amazon$SentimentText)
# Replacing multiple spaces in the text with single space
reviews_amazon$SentimentText <- gsub("(?<=[\\s])\\s*|^\\s+|\\s+$", "",
                                     reviews_amazon$SentimentText, perl=TRUE)
# Writing the output to a file that can be consumed in other projects
write.table(reviews_amazon,file = "Sentiment Analysis Dataset.csv",row.names = F,
            col.names = T,sep=',')
```

```
reviews_amazon <- readLines('amazon_review_polarity_csv/train.csv', n = N_amazon)
# Basic EDA to confirm that the data is read correctly
print(class(reviews_amazon))
```

```
## [1] "character"
```

```
print(length(reviews_amazon))
```

```
## [1] 100000
```

```
# print(head(reviews_amazon,2))
# Replacing the positive sentiment value 2 with __label__2
reviews_amazon <- gsub("\\\"2\\\",","__label__2 ",reviews_amazon)
# Replacing the negative sentiment value 1 with __label__1
reviews_amazon <- gsub("\\\"1\\\",","__label__1 ",reviews_amazon)
# Removing the unnecessary \" characters
reviews_amazon <- gsub("\\\""," ",reviews_amazon)
# Replacing multiple spaces in the text with single space
reviews_amazon <- gsub("(?<=[\\s])\\s*|^\\s+|\\s+$", "", reviews_amazon, perl=TRUE)
# Basic EDA post the required processing to confirm input is as desired
# print("EDA POST PROCESSING")
# print(class(reviews_amazon))
# print(length(reviews_amazon))
# print(head(reviews_amazon,2))
# Writing the revamped file to the directory so we could use it with
# fastText sentiment analyzer project
fileConn <- file("Sentiment Analysis Dataset_ft.txt")
writeLines(reviews_amazon, fileConn)
close(fileConn)
```

2.1.2 Drug Data

```
##   Sentiment
## 2      "2",
## 3      "2",
## 4      "2",
## 5      "2",
```

```
## 6      "1",
## 7      "1",
##
## 2
## 3 "Guanfacine ADHD My son is halfway through his fourth week of Intuniv We became concerned when he
## 4
## 5                    "Buprenorphine naloxone Opiate Dependence Suboxone has completely turned my life
## 6
## 7
```

```
# Checking the summary of our label for Drug data
(Sentimentable = table(reviews_text_Drug$Sentiment))
```

```
##
##      1      2
##  40075 106866
```

```
# Balance our Drug data
minlabel <- names(which(Sentimentable == min(Sentimentable)))
maxlabel <- names(which(Sentimentable == max(Sentimentable)))

n_maxlabel <- min(Sentimentable)
minlabelid <- c(1:N_Drug)[reviews_text_Drug$Sentiment==minlabel]
maxlabelid <- sample(c(1:N_Drug)[reviews_text_Drug$Sentiment==maxlabel],n_maxlabel)
balanceid <- sample(c(minlabelid,maxlabelid))
reviews_text_Drug <- reviews_text_Drug[balanceid,]

N_Drug <- nrow(reviews_text_Drug)
N_train_Drug <- round(0.8*N_Drug)
```

```
## [1] "character"
```

```
## [1] 80150
```

3 BoW approach

3.1 Amazon reviews

```
library(SnowballC)
library(tm)
# Reading the transformed file as a dataframe
text_amazon <- read.table(file='Sentiment Analysis Dataset.csv', sep=',', header = TRUE)
# Checking the dataframe to confirm everything is in tact
print(dim(text_amazon))
```

```
## [1] 100000      2
```

```r
# Transforming the text into volatile corpus
amazon_corp <- VCorpus(VectorSource(text_amazon$SentimentText))
print(amazon_corp)
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 100000
```

```r
# Creating document term matrix
dtm_amazon <- DocumentTermMatrix(amazon_corp, control = list( tolower = TRUE,
   removeNumbers = TRUE, stopwords = TRUE, removePunctuation = TRUE, stemming = TRUE))
# Basic EDA on dtm
inspect(dtm_amazon)
```

```
## <<DocumentTermMatrix (documents: 100000, terms: 74760)>>
## Non-/sparse entries: 3399444/7472600556
## Sparsity           : 100%
## Maximal term length: 188
## Weighting          : term frequency (tf)
## Sample             :
##        Terms
## Docs    book get good great just like movi one read time
##   1250     0   0    0     0    0    0    0   0    0    0
##   56817    0   0    0     0    0    0    0   0    0    0
##   63995    0   2    1     1    0    2    0   2    0    1
##   6785     0   7    1     0    0    1    0   1    0    0
##   69262    0   0    0     0    0    0    0   0    0    0
##   73633    1   0    0     2    0    3    0   2    0    2
##   79144    0   0    0     0    0    0    0   0    0    0
##   80872    0   0    0     0    0    0    0   0    0    0
##   85894    0   1    0     0    0    1    0   1    0    0
##   87875    0   0    0     0    0    0    0   0    0    0
```

```r
# Removing sparse terms
dtm_amazon = removeSparseTerms(dtm_amazon, 0.99)
inspect(dtm_amazon)
```

```
## <<DocumentTermMatrix (documents: 100000, terms: 645)>>
## Non-/sparse entries: 2131029/62368971
## Sparsity           : 97%
## Maximal term length: 10
## Weighting          : term frequency (tf)
## Sample             :
##        Terms
## Docs    book get good great just like movi one read time
##   34297    0   1    0     0    2    5    0   3    0    0
##   38984    6   0    0     1    1    0    0   1    0    1
##   42051    3   0    1     0    2    1    0   3    5    1
##   56269    0   1    0     0    1    0    1   1    0    1
##   65117    0   1    1     1    1    0    0   1    0    0
##   65135    0   0    1     2    0    4    0   0    1    0
##   6785     0   7    1     0    0    1    0   1    0    0
```

```
##    80366    0   3    1     1    1    1    0   8    0    0
##    87149    6   0    1     0    0    1    0   2    1    0
##    90397    0   2    1     0    3    3    0   2    0    0
```

We found that some of the selected terms are difficult to interpret. So we decided to first filter out a vocab with words that we can interpret. We used a simple screening method, which we described in discriminant analysis before: two-sample t-test. Assume we have one-dimensional observations from two groups

$$X_1, X_2, \ldots, X_m, \quad Y_1, Y_2, \ldots, Y_n X$$

to test whether the X population and the Y population have the same mean, we compute the following two-sample t-statistic $\frac{\bar{X}-\bar{Y}}{\sqrt{\frac{s_X^2}{m}+\frac{s_Y^2}{n}}}$.

where s_X^2 denotes the sample variance of X.

Again, we use the training data from the first split. Since dtm_train is a large sparse matrix, I use commands from the R package slam to efficiently compute the mean and var for each column of dtm_train.

```
# Word Cloud preparing
v.size = dim(dtm_amazon)[2]
ytrain = as.numeric(text_amazon$Sentiment)
```

```
# Using two-sample t-test to find the most represent words to show our Word Cloud
library(slam)
summ = matrix(0, nrow=v.size, ncol=4)
summ[,1] = colapply_simple_triplet_matrix(
  as.simple_triplet_matrix(dtm_amazon[ytrain==2, ]), mean)
summ[,2] = colapply_simple_triplet_matrix(
  as.simple_triplet_matrix(dtm_amazon[ytrain==2, ]), var)
summ[,3] = colapply_simple_triplet_matrix(
  as.simple_triplet_matrix(dtm_amazon[ytrain==1, ]), mean)
summ[,4] = colapply_simple_triplet_matrix(
  as.simple_triplet_matrix(dtm_amazon[ytrain==1, ]), var)

n1 = sum((ytrain)-1);
n = length(ytrain)
n0 = n - n1

myp = (summ[,1] - summ[,3])/
  sqrt(summ[,2]/n1 + summ[,4]/n0)
```

We ordered words by the magnitude of their t-statistics, which are then divided into two lists: positive words and negative words.

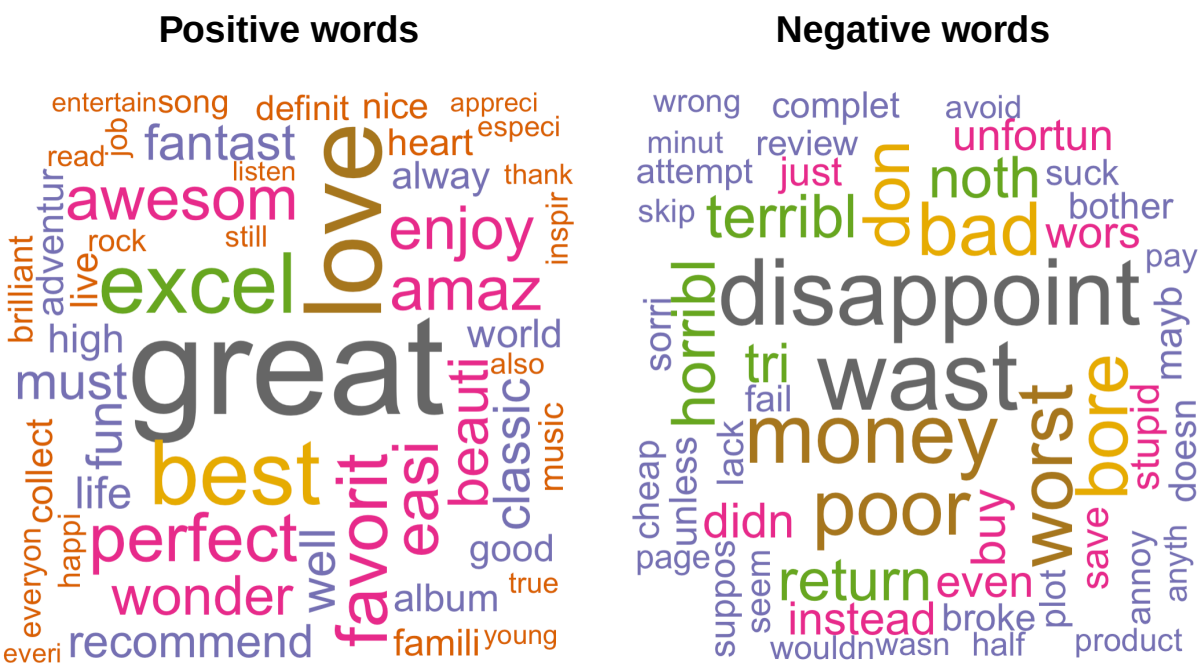
```
words = colnames(dtm_amazon)
id = order(abs(myp), decreasing=TRUE)
pos.list = words[id[myp[id]>0]]
posvalue = myp[id][myp[id]>0][1:50]
neg.list = words[id[myp[id]<0]]
negvalue = myp[id][myp[id]<0][1:50]
```

Using the "wordcloud" package to plot the Word Cloud for most represent words, both positive and negative.

```
# Word Cloud for positive words
library(wordcloud)
wordcloud(words = pos.list[1:50], freq = posvalue, scale=c(6,.2), min.freq = 5,
          random.order=FALSE, rot.per=0.35, colors = brewer.pal(8, "Dark2"))
```

```
# Word Cloud for negative words
wordcloud(words = neg.list[1:50], freq = abs(negvalue), scale=c(4.3,.2), min.freq = 5,
          random.order=FALSE, rot.per=0.35, colors = brewer.pal(8, "Dark2"))
```

```
library(png)
par(mfrow=c(1, 2), mar=c(1, 0, 3, 0))
plot.new()
plot.window(xlim=c(0, 1), ylim=c(0, 1), asp=1)
rasterImage(readPNG("amazonpos"), 0, 0, 1, 1)
title('Positive words', line = -0.5)
plot.new()
plot.window(xlim=c(0, 1), ylim=c(0, 1), asp=1)
rasterImage(readPNG("amazonneg"), 0, 0, 1, 1)
title('Negative words', line = -0.5)
title("Word Clouds from Amazon reviews", line = -22, outer = TRUE)
```



Word Clouds from Amazon reviews

```
# Splitting the train and test DTM
dtm_amazon_train <- dtm_amazon[1:N_train_amazon, ]
dtm_amazon_test <- dtm_amazon[(N_train_amazon+1):N_amazon, ]
```

```
dtm_amazon_train_labels <- as.factor(as.character(text_amazon[1:N_train_amazon, ]$Sentiment))
dtm_amazon_test_labels <- as.factor(as.character(text_amazon[(N_train_amazon+1):N_amazon, ]$Sentiment))
```

```
# Convert the cell values with a non-zero value to Y, and in case of a zero we convert it to N
cellconvert<- function(x) { x <- ifelse(x > 0, "Y", "N") }
```

```
# Applying the function to rows in training and test datasets
dtm_amazon_train <- apply(dtm_amazon_train, MARGIN = 2,cellconvert)
dtm_amazon_test <- apply(dtm_amazon_test, MARGIN = 2,cellconvert)
```

```
# Training the naive bayes classifier on the training dtm
library(e1071)
nb_amazon_senti_classifier <- naiveBayes(dtm_amazon_train,dtm_amazon_train_labels)
# Printing the summary of the model created
summary(nb_amazon_senti_classifier)
```

```
##           Length Class  Mode
## apriori     2    table  numeric
## tables    645    -none- list
## levels      2    -none- character
## isnumeric 645    -none- logical
## call        3    -none- call
```

```
# Making predictions on the test data dtm
nb_amazon_predicts <- predict(nb_amazon_senti_classifier, dtm_amazon_test, type="class")
```

```
# Computing accuracy of the model
library(rminer)
print(mmetric(nb_amazon_predicts, dtm_amazon_test_labels, c("ACC")))
```

```
## [1] 81.19
```

3.2 Drug Data

```
## [1] 80150     2
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 80150
```

```
## <<DocumentTermMatrix (documents: 80150, terms: 44610)>>
## Non-/sparse entries: 2919201/3572572299
## Sparsity           : 100%
## Maximal term length: 95
## Weighting          : term frequency (tf)
## Sample             :
##        Terms
## Docs    day effect get month pain start take week work year
##   14443   0      7   0     0    0     1    1    0    0    0
##   21739   3      4   3     1   10     4    3    5    6    1
```

```
##   32948   9      1   1     1    1     3    5    1    1    5
##   35157   7      7   4     1    3     1    6    4    4    0
##   39889   0      2   2     4    3     4    1    0    2    1
##   4810    0      7   0     0    0     1    1    0    0    0
##   48674   7      7   4     1    3     1    6    4    4    0
##   50714   7      5   3     2    0     4    9    6    2    1
##   56489   0      2   2     4    3     4    1    0    2    1
##   79862   6      0   2     0    7     0    2    0    4    2
```

```
## <<DocumentTermMatrix (documents: 80150, terms: 645)>>
## Non-/sparse entries: 2177799/49518951
## Sparsity           : 96%
## Maximal term length: 14
## Weighting          : term frequency (tf)
## Sample             :
##        Terms
## Docs    day effect get month pain start take week work year
##   1194    2      3   4     0    0     1    3    3    1    0
##   21739   3      4   3     1   10     4    3    5    6    1
##   32948   9      1   1     1    1     3    5    1    1    5
##   35157   7      7   4     1    3     1    6    4    4    0
##   35179   2      3   4     0    0     1    3    3    1    0
##   39889   0      2   2     4    3     4    1    0    2    1
##   48674   7      7   4     1    3     1    6    4    4    0
##   50714   7      5   3     2    0     4    9    6    2    1
##   56489   0      2   2     4    3     4    1    0    2    1
##   79862   6      0   2     0    7     0    2    0    4    2
```

Word Clouds from Drug reviews

```
##           Length Class  Mode
## apriori      2    table  numeric
## tables     645    -none- list
## levels       2    -none- character
## isnumeric  645    -none- logical
## call         3    -none- call
```

```
## [1] 74.7723
```

4 Pretrained word2vec word embedding

4.1 Amazon reviews

```
library(softmaxreg)
# Importing the word2vec pretrained vector into memory
data(word2vec)
dim(word2vec)
```

```
## [1] 12853    21
```

```r
# Function to get word vector for each review
docVectors <- function(x) { wordEmbed(x, word2vec, meanVec = TRUE) }
text_amazon <- read.csv(file='Sentiment Analysis Dataset.csv', header = TRUE)
# Applying the docVector function on each of the reviews
# Storing the matrix of word vectors as temp
temp_amazon <- t(sapply(text_amazon$SentimentText, docVectors))
dim(temp_amazon)
```

```
## [1] 100000     20
```

```r
# Splitting the dataset into train and test
temp_amazon_train <- temp_amazon[1:N_train_amazon,]
temp_amazon_test <- temp_amazon[(N_train_amazon+1):N_amazon,]
labels_amazon_train <- as.factor(as.character(text_amazon[1:N_train_amazon,]$Sentiment))
labels_amazon_test <- as.factor(as.character(text_amazon[(N_train_amazon+1):N_amazon,]$Sentiment))
library(randomForest)
# Training a model using random forest classifier with training dataset
# Observe that we are using 20 trees to create the model
rf_amazon_senti_classifier <- randomForest(temp_amazon_train, labels_amazon_train, ntree=20)
print(rf_amazon_senti_classifier)
```

```
##
## Call:
##  randomForest(x = temp_amazon_train, y = labels_amazon_train,      ntree = 20)
##                Type of random forest: classification
##                      Number of trees: 20
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 40%
## Confusion matrix:
##       1     2 class.error
## 1 23547 15436   0.3959675
## 2 16563 24448   0.4038673
```

```r
# Making predictions on the dataset
rf_amazon_predicts <- predict(rf_amazon_senti_classifier, temp_amazon_test)
library(rminer)
print(mmetric(rf_amazon_predicts, labels_amazon_test, c("ACC")))
```

```
## [1] 62.555
```

4.2 Drug Data

```
## [1] 12853    21
```

```
## [1] 80150    20
```

```
##
## Call:
##  randomForest(x = temp_train_Drug, y = labels_train_Drug, ntree = 20)
```

```
##                Type of random forest: classification
##                      Number of trees: 20
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 31.81%
## Confusion matrix:
##       1     2 class.error
## 1 23040  8945   0.2796623
## 2 11453 20677   0.3564581
```

```
## [1] 70.98565
```

5 GloVe word embedding

5.1 Amazon reviews

```
# Including the required library
library(text2vec)
# Reading the dataset
text_amazon <- read.csv(file='Sentiment Analysis Dataset.csv', header = TRUE)
# Subsetting only the review text so as to create Glove word embedding
wiki_amazon <- as.character(text_amazon$SentimentText)
# Create iterator over tokens
tokens_amazon <- space_tokenizer(wiki_amazon)
# Create vocabulary. Terms will be unigrams (simple words).
it_amazon <- itoken(tokens_amazon, progressbar = FALSE)
vocab_amazon <- create_vocabulary(it_amazon)
# Consider a term in the vocabulary if and only if the term has appeared at least
# three times in the dataset
vocab_amazon <- prune_vocabulary(vocab_amazon, term_count_min = 3L)
# Use the filtered vocabulary
vectorizer_amazon <- vocab_vectorizer(vocab_amazon)
# Use window of 5 for context words and create a term co-occurance matrix
tcm_amazon <- create_tcm(it_amazon, vectorizer_amazon, skip_grams_window = 5L)
# Create the glove embedding for each in the vocab and
# the dimension of the word embedding should set to 50
# x_max is the maximum number of co-occurrences to use in the weighting function
glove <- GlobalVectors$new(rank = 50, x_max = 100)
wv_main_amazon <- glove$fit_transform(tcm_amazon, n_iter = 10, convergence_tol = 0.01)
```

```
## INFO  [16:54:19.703] epoch 1, loss 0.0502
## INFO  [16:54:26.068] epoch 2, loss 0.0318
## INFO  [16:54:32.983] epoch 3, loss 0.0267
## INFO  [16:54:39.983] epoch 4, loss 0.0239
## INFO  [16:54:46.295] epoch 5, loss 0.0222
## INFO  [16:54:52.638] epoch 6, loss 0.0209
## INFO  [16:54:58.947] epoch 7, loss 0.0199
## INFO  [16:55:05.310] epoch 8, loss 0.0191
## INFO  [16:55:11.794] epoch 9, loss 0.0184
## INFO  [16:55:18.253] epoch 10, loss 0.0179
```

```
# Glove model learns two sets of word vectors - main and context.
# Both matrices may be added to get the combined word vector
wv_context <- glove$components
word_vectors_amazon <- wv_main_amazon + t(wv_context)
# Converting the word_vector to a dataframe for visualization
word_vectors_amazon <- data.frame(word_vectors_amazon)
# The word for each embedding is set as row name by default
# Using the tibble library rownames_to_column function, the rownames is copied
# as first column of the dataframe
# We also name the first column of the dataframe as words
library(tibble)
word_vectors_amazon <- rownames_to_column(word_vectors_amazon, var = "words")
```

```
library(softmaxreg)
docVectors_amazon = function(x) { wordEmbed(x, word_vectors_amazon, meanVec = TRUE) }
# Applying the function docVectors function on the entire reviews dataset
# This will result in word embedding representation of the entire reviews dataset
temp_amazon <- t(sapply(text_amazon$SentimentText, docVectors_amazon))
```

```
# Splitting the dataset into train and test portions
temp_amazon_train <- temp_amazon[1:N_train_amazon,]
temp_amazon_test <- temp_amazon[(N_train_amazon+1):N_amazon,]
labels_amazon_train <- as.factor(as.character(text_amazon[1:N_train_amazon,]$Sentiment))
labels_amazon_test <- as.factor(as.character(text_amazon[(N_train_amazon+1):N_amazon,]$Sentiment))
# Using randomforest to build a model on train data
library(randomForest)
rf_amazon_senti_classifier <- randomForest(temp_amazon_train, labels_amazon_train,ntree=20)
print(rf_amazon_senti_classifier)
```

```
##
## Call:
##  randomForest(x = temp_amazon_train, y = labels_amazon_train,      ntree = 20)
##                Type of random forest: classification
##                      Number of trees: 20
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 30.7%
## Confusion matrix:
##       1     2 class.error
## 1 27275 11706   0.3003001
## 2 12849 28163   0.3132985
```

```
# Predicting labels using the randomforest model created
rf_amazon_predicts <- predict(rf_amazon_senti_classifier, temp_amazon_test)
# Estimating the accuracy from the predictions
library(rminer)
print(mmetric(rf_amazon_predicts, labels_amazon_test, c("ACC")))
```

```
## [1] 72.72
```

5.2 Drug Data

```
## INFO  [17:01:54.406] epoch 1, loss 0.0755
```

```
## INFO  [17:01:56.868] epoch 2, loss 0.0487
## INFO  [17:01:59.315] epoch 3, loss 0.0401
## INFO  [17:02:01.831] epoch 4, loss 0.0354
## INFO  [17:02:04.218] epoch 5, loss 0.0324
## INFO  [17:02:06.689] epoch 6, loss 0.0302
## INFO  [17:02:09.124] epoch 7, loss 0.0285
## INFO  [17:02:11.427] epoch 8, loss 0.0273
## INFO  [17:02:13.648] epoch 9, loss 0.0262
## INFO  [17:02:15.819] epoch 10, loss 0.0254
```

```
##
## Call:
##  randomForest(x = temp_train_Drug, y = labels_train_Drug, ntree = 20)
##                Type of random forest: classification
##                      Number of trees: 20
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 29.08%
## Confusion matrix:
##       1     2 class.error
## 1 23380  8604   0.2690095
## 2 10039 22089   0.3124689
```

```
## [1] 74.95945
```

6 FastText word embedding

6.1 Amazon reviews

```
library(fastTextR)
# Input reviews file
text_amazon <- readLines("Sentiment Analysis Dataset_ft.txt")
```

```
# Dividing the reviews into training and test
temp_amazon_train <- text_amazon[1:N_train_amazon]
temp_amazon_test <- text_amazon[(N_train_amazon+1):N_amazon]
```

```
# Creating txt file for train and test dataset
fileConn <- file("train.ft.txt")
writeLines(temp_amazon_train, fileConn)
close(fileConn)
fileConn <- file("test.ft.txt")
writeLines(temp_amazon_test, fileConn)
close(fileConn)
# Creating a test file with no labels
temp_amazon_test_nolabel <- gsub("__label__1", "", temp_amazon_test, perl=TRUE)
temp_amazon_test_nolabel <- gsub("__label__2", "", temp_amazon_test_nolabel, perl=TRUE)
```

```
fileConn <- file("test_nolabel.ft.txt")
writeLines(temp_amazon_test_nolabel, fileConn)
close(fileConn)
# Training a supervised classification model with training dataset file
model_amazon <- ft_train("train.ft.txt", method = "supervised",
                control = ft_control(nthreads = 3L, seed = 1))
# Obtain all the words from a previously trained model
words_amazon <- ft_words(model_amazon)
```

```
# Obtain word vectors from a previously trained model.
word_vec_amazon <- ft_word_vectors(model_amazon, words_amazon)
```

```
# Predicting the labels for the reviews in the no labels test dataset
# Getting the predictions into a dataframe so as to compute performance measurement
ft_preds_amazon <- ft_predict(model_amazon, newdata = temp_amazon_test_nolabel)
# Reading the test file to extract the actual labels
reviewstestfile_amazon <- readLines("test.ft.txt")
# Extracting just the labels frm each line
library(stringi)
actlabels_amazon <- stri_extract_first(reviewstestfile_amazon, regex="\\w+")
# Converting the actual labels and predicted labels into factors
actlabels_amazon <- as.factor(as.character(actlabels_amazon))
ft_preds_amazon <- as.factor(as.character(ft_preds_amazon$label))
# Getting the estimate of the accuracy
library(rminer)
print(mmetric(actlabels_amazon, ft_preds_amazon, c("ACC")))
```

```
## [1] 86.48
```

6.2 Drug Data

```
## [1] 78.68372
```

References

Zhang, Xiang, Junbo Zhao, and Yann LeCun. 2015. "Character-Level Convolutional Networks for Text Classification." *Advances in Neural Information Processing Systems* 28.