Report_final

rongl2-xdai12-zixingd2

2022-12-06

Contents

```
#test4
# df = read.csv("amazon_review_polarity_csv/train.csv", header = FALSE)
```

1 Amazon reviews

```
set.seed(1)
N <- 100000
N_t <- 0.8*N
reviews_text <- readLines("amazon_review_polarity_csv/train.csv", n = N)
reviews_text <- data.frame(reviews_text)
```

```
library(tidyr)
reviews_text <- separate(data = reviews_text, col = reviews_text,
                         into = c("Sentiment", "SentimentText"), sep = 4)
```

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

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

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

```
print(length(reviews_text))
```

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

```
# print(head(reviews_text,2))
# Replacing the positive sentiment value 2 with __label__2
reviews_text <- gsub("\\\"2\\\",","__label__2 ",reviews_text)
# Replacing the negative sentiment value 1 with __label__1
reviews_text <- gsub("\\\"1\\\",","__label__1 ",reviews_text)
# Removing the unnecessary \" characters
reviews_text <- gsub("\\\""," ",reviews_text)
# Replacing multiple spaces in the text with single space
reviews_text <- gsub("(?<=[\\s])\\s*|^\\s+|\\s+$", "", reviews_text, perl=TRUE)
# Basic EDA post the required processing to confirm input is as desired
print("EDA POST PROCESSING")
```

```
## [1] "EDA POST PROCESSING"
```

```
print(class(reviews_text))
```

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

```
print(length(reviews_text))
```

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

```
# print(head(reviews_text,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_text, fileConn)
close(fileConn)
```

1.1 BoW approach

```
library(SnowballC)
library(tm)
```

```
## Loading required package: NLP
```

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

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

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

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

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

```
## <<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
```

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

```
## <<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
```

```
# Splitting the train and test DTM
dtm_train_train <- dtm_train[1:N_t, ]
dtm_train_test <- dtm_train[(N_t+1):N, ]
dtm_train_train_labels <- as.factor(as.character(text[1:N_t, ]$Sentiment))
dtm_train_test_labels <- as.factor(as.character(text[(N_t+1):N, ]$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_train_train <- apply(dtm_train_train, MARGIN = 2,cellconvert)
dtm_train_test <- apply(dtm_train_test, MARGIN = 2,cellconvert)
```

```
# Training the naive bayes classifier on the training dtm
library(e1071)
nb_senti_classifier <- naiveBayes(dtm_train_train,dtm_train_train_labels)
# Printing the summary of the model created
summary(nb_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_predicts <- predict(nb_senti_classifier, dtm_train_test,type="class")
```

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

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

1.2 Pretrained word2vec word embedding

```
library(softmaxreg)
```

```
##
## Attaching package: 'softmaxreg'

## The following object is masked from 'package:e1071':
##
##     sigmoid
```

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

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

```
# Function to get word vector for each review
docVectors <- function(x) { wordEmbed(x, word2vec, meanVec = TRUE) }
text <- 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 <- t(sapply(text$SentimentText, docVectors))
dim(temp)
```

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

```
# Splitting the dataset into train and test
temp_train <- temp[1:N_t,]
temp_test <- temp[(N_t+1):N,]
labels_train <- as.factor(as.character(text[1:N_t,]$Sentiment))
labels_test <- as.factor(as.character(text[(N_t+1):N,]$Sentiment))
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
# Training a model using random forest classifier with training dataset
# Observe that we are using 20 trees to create the model
rf_senti_classifier <- randomForest(temp_train, labels_train,ntree=20)
print(rf_senti_classifier)
```

```
##
## Call:
##  randomForest(x = temp_train, y = labels_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: 39.82%
## Confusion matrix:
##       1     2 class.error
## 1 23699 15284   0.3920683
## 2 16570 24441   0.4040379
```

```
# Making predictions on the dataset
rf_predicts <- predict(rf_senti_classifier, temp_test)
library(rminer)
print(mmetric(rf_predicts, labels_test, c("ACC")))
```

```
## [1] 62.875
```

1.3 GloVe word embedding

```
# Including the required library
library(text2vec)
```

```
##
## Attaching package: 'text2vec'
```

```
## The following object is masked from 'package:rminer':
##
##     fit
```

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

```
## INFO  [17:56:12.912] epoch 1, loss 0.0502
## INFO  [17:56:18.318] epoch 2, loss 0.0317
## INFO  [17:56:23.895] epoch 3, loss 0.0266
## INFO  [17:56:29.403] epoch 4, loss 0.0239
## INFO  [17:56:34.956] epoch 5, loss 0.0221
## INFO  [17:56:40.455] epoch 6, loss 0.0209
## INFO  [17:56:46.006] epoch 7, loss 0.0199
## INFO  [17:56:51.447] epoch 8, loss 0.0191
## INFO  [17:56:56.978] epoch 9, loss 0.0185
## INFO  [17:57:02.532] epoch 10, loss 0.0179
```

```r
# 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 <- wv_main + t(wv_context)
# Converting the word_vector to a dataframe for visualization
word_vectors <- data.frame(word_vectors)
# 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 <- rownames_to_column(word_vectors, var = "words")
```

```r
library(softmaxreg)
docVectors = function(x) { wordEmbed(x, word_vectors, 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 <- t(sapply(text$SentimentText, docVectors))
```

```r
# Splitting the dataset into train and test portions
temp_train <- temp[1:N_t,]
temp_test <- temp[(N_t+1):N,]
labels_train <- as.factor(as.character(text[1:N_t,]$Sentiment))
labels_test <- as.factor(as.character(text[(N_t+1):N,]$Sentiment))
# Using randomforest to build a model on train data
library(randomForest)
rf_senti_classifier <- randomForest(temp_train, labels_train,ntree=20)
print(rf_senti_classifier)
```

```
##
## Call:
##  randomForest(x = temp_train, y = labels_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: 31.77%
## Confusion matrix:
##       1     2 class.error
## 1 26911 12070   0.3096380
## 2 13340 27671   0.3252786
```

```r
# Predicting labels using the randomforest model created
rf_predicts <- predict(rf_senti_classifier, temp_test)
# Estimating the accuracy from the predictions
library(rminer)
print(mmetric(rf_predicts, labels_test, c("ACC")))
```

```
## [1] 71.995
```

1.4 FastText word embedding

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

```r
# Dividing the reviews into training and test
temp_train <- text[1:N_t]
temp_test <- text[(N_t+1):N]
```

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

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

```
# Obtain word vectors from a previously trained model.
word_vec <- ft_word_vectors(model, words)
```

```
# 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 <- ft_predict(model, newdata = temp_test_nolabel)
# Reading the test file to extract the actual labels
reviewstestfile <- readLines("test.ft.txt")
# Extracting just the labels frm each line
library(stringi)
actlabels <- stri_extract_first(reviewstestfile, regex="\\w+")
# Converting the actual labels and predicted labels into factors
actlabels <- as.factor(as.character(actlabels))
ft_preds <- as.factor(as.character(ft_preds$label))
# Getting the estimate of the accuracy
library(rminer)
print(mmetric(actlabels, ft_preds, c("ACC")))
```

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

2 Drug Data

```
set.seed(1)
N_Drug <- 146942
reviews_text_Drug <- readLines("Drug Train.csv", n = N_Drug)
reviews_text_Drug <- data.frame(reviews_text_Drug)
```

```
library(tidyr)
reviews_text_Drug <- separate(data = reviews_text_Drug, col = reviews_text_Drug,
                              into = c("Sentiment", "SentimentText"), sep = 4)
reviews_text_Drug <- reviews_text_Drug[-1,]
N_Drug <- N_Drug - 1
```

```
# Retaining only alphanumeric values in the sentiment column
reviews_text_Drug$Sentiment <- gsub("[^[:alnum:] ]","",reviews_text_Drug$Sentiment)
# Retaining only alphanumeric values in the sentiment text
reviews_text_Drug$SentimentText <- gsub("[^[:alnum:] ]"," ",reviews_text_Drug$SentimentText)
# Replacing multiple spaces in the text with single space
reviews_text_Drug$SentimentText <- gsub("(?<=[\\s])\\s*|^\\s+|\\s+$", "",
                                        reviews_text_Drug$SentimentText, perl=TRUE)
```

```
# Balance our data
minlabel <- names(which(table(reviews_text_Drug$Sentiment)==
                        min(table(reviews_text_Drug$Sentiment))))
maxlabel <- names(which(table(reviews_text_Drug$Sentiment)==
                        max(table(reviews_text_Drug$Sentiment))))

n_maxlabel <- min(table(reviews_text_Drug$Sentiment))
minlabelid <- c(1:N_Drug)[reviews_text_Drug$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)
```

```
# Writing the output to a file that can be consumed in other projects
write.table(reviews_text_Drug,file = "Sentiment Analysis Dataset_Drug.csv",
            row.names = F, col.names = T,sep=',')
```

```
reviews_text_Drug <- readLines("Drug Train.csv", n = 146942)
reviews_text_Drug <- reviews_text_Drug[-1]
reviews_text_Drug <- reviews_text_Drug[balanceid]
# Basic EDA to confirm that the data is read correctly
print(class(reviews_text_Drug))
```

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

```
print(length(reviews_text_Drug))
```

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

```
# print(head(reviews_text_Drug,2))
# Replacing the positive sentiment value 2 with __label__2
reviews_text_Drug<-gsub("\\\"2\\\"","__label__2 ",reviews_text_Drug)
# Replacing the negative sentiment value 1 with __label__1
reviews_text_Drug<-gsub("\\\"1\\\"","__label__1 ",reviews_text_Drug)
# Removing the unnecessary \" characters
reviews_text_Drug<-gsub("\\\""," ",reviews_text_Drug)
# Replacing multiple spaces in the text with single space
reviews_text_Drug<-gsub("(?<=[\\s])\\s*|^\\s+|\\s+$", "", reviews_text_Drug, perl=TRUE)
# Basic EDA post the required processing to confirm input is as desired
print("EDA POST PROCESSING")
```

```
## [1] "EDA POST PROCESSING"
```

```
print(class(reviews_text_Drug))
```

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

```
print(length(reviews_text_Drug))
```

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

```
# print(head(reviews_text_Drug,2))
# Writing the revamped file to the directory so we could use it with
# fastText sentiment analyzer project
fileConn<-file("Sentiment Analysis Dataset_ft_Drug.txt")
writeLines(reviews_text_Drug, fileConn)
close(fileConn)
```

2.1 BoW approach

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

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

```
# Transforming the text into volatile corpus
train_corp_Drug <- VCorpus(VectorSource(text_Drug$SentimentText))
print(train_corp_Drug)
```

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

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

```
## <<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
```

```
# Removing sparse terms
dtm_train_Drug <- removeSparseTerms(dtm_train_Drug, 0.99)
inspect(dtm_train_Drug)
```

```
## <<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
```

```
# Splitting the train and test DTM
dtm_train_train_Drug <- dtm_train_Drug[1:N_train_Drug, ]
dtm_train_test_Drug <- dtm_train_Drug[(N_train_Drug+1):N_Drug, ]
dim(dtm_train_Drug)
```

```
## [1] 80150   645
```

```
dtm_train_train_Drug_labels <- as.factor(as.character(text_Drug[1:N_train_Drug, ]$Sentiment))
dtm_train_test_Drug_labels <- as.factor(as.character(text_Drug[(N_train_Drug+1):N_Drug, ]$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_train_train_Drug <- apply(dtm_train_train_Drug, MARGIN = 2,cellconvert)
dtm_train_test_Drug <- apply(dtm_train_test_Drug, MARGIN = 2,cellconvert)
```

```
# Training the naive bayes classifier on the training dtm
library(e1071)
nb_senti_classifier_Drug <- naiveBayes(dtm_train_train_Drug, dtm_train_train_Drug_labels)
# Printing the summary of the model created
summary(nb_senti_classifier_Drug)
```

```
##           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_predicts_Drug <- predict(nb_senti_classifier_Drug, dtm_train_test_Drug, type="class")
```

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

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

2.2 Pretrained word2vec word embedding

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

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

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

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

```
# Splitting the dataset into train and test
temp_train_Drug <- temp_Drug[1:N_train_Drug,]
temp_test_Drug <- temp_Drug[(N_train_Drug+1):N_Drug,]
labels_train_Drug <- as.factor(as.character(text_Drug[1:N_train_Drug,]$Sentiment))
labels_test_Drug <- as.factor(as.character(text_Drug[(N_train_Drug+1):N_Drug,]$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_senti_classifier_Drug <- randomForest(temp_train_Drug, labels_train_Drug,ntree=20)
print(rf_senti_classifier_Drug)
```

```
##
## 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: 31.92%
## Confusion matrix:
##       1     2 class.error
## 1 23000  8985   0.2809129
## 2 11479 20649   0.3572896
```

```
# Making predictions on the dataset
rf_predicts_Drug <- predict(rf_senti_classifier_Drug, temp_test_Drug)
library(rminer)
print(mmetric(rf_predicts_Drug, labels_test_Drug, c("ACC")))
```

```
## [1] 71.01684
```

2.3 GloVe word embedding

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

```
## INFO  [18:06:50.483] epoch 1, loss 0.0758
## INFO  [18:06:52.751] epoch 2, loss 0.0489
## INFO  [18:06:54.939] epoch 3, loss 0.0402
## INFO  [18:06:57.169] epoch 4, loss 0.0355
## INFO  [18:06:59.327] epoch 5, loss 0.0324
## INFO  [18:07:01.490] epoch 6, loss 0.0302
## INFO  [18:07:03.661] epoch 7, loss 0.0285
## INFO  [18:07:05.844] epoch 8, loss 0.0272
## INFO  [18:07:08.006] epoch 9, loss 0.0262
## INFO  [18:07:10.246] epoch 10, loss 0.0253
```

```
# 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_Drug <- wv_main_Drug + t(wv_context)
# Converting the word_vector to a dataframe for visualization
word_vectors_Drug <- data.frame(word_vectors_Drug)
# 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_Drug <- rownames_to_column(word_vectors_Drug, var = "words")
```

```
library(softmaxreg)
docVectors_Drug = function(x) { wordEmbed(x, word_vectors_Drug, 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_Drug <- t(sapply(text_Drug$SentimentText, docVectors_Drug))
```

```
# Splitting the dataset into train and test portions
temp_train_Drug <- temp_Drug[1:N_train_Drug,]
temp_test_Drug <- temp_Drug[(N_train_Drug+1):N_Drug,]
labels_train_Drug <- as.factor(as.character(text_Drug[1:N_train_Drug,]$Sentiment))
labels_test_Drug <- as.factor(as.character(text_Drug[(N_train_Drug+1):N_Drug,]$Sentiment))
# Using randomforest to build a model on train data
library(randomForest)
rf_senti_classifier_Drug <- randomForest(temp_train_Drug, labels_train_Drug,ntree=20)
print(rf_senti_classifier_Drug)
```

```
##
## 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%
## Confusion matrix:
##       1     2 class.error
## 1 23384  8600   0.2688844
## 2  9994 22133   0.3110779
```

```
# Predicting labels using the randomforest model created
rf_predicts_Drug<-predict(rf_senti_classifier_Drug, temp_test_Drug)
# Estimating the accuracy from the predictions
library(rminer)
print(mmetric(rf_predicts_Drug, labels_test_Drug, c("ACC")))
```

```
## [1] 74.82845
```

2.4 FastText word embedding

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

```
# Dividing the reviews into training and test
temp_train_Drug <- text_Drug[1:N_train_Drug]
temp_test_Drug <- text_Drug[(N_train_Drug+1):N_Drug]
```

```
# Creating txt file for train and test dataset
fileConn <- file("train_Drug.ft.txt")
writeLines(temp_train_Drug, fileConn)
close(fileConn)
fileConn <- file("test_Drug.ft.txt")
writeLines(temp_test_Drug, fileConn)
close(fileConn)
# Creating a test file with no labels
temp_test_Drug_nolabel <- gsub("__label__1", "", temp_test_Drug, perl=TRUE)
temp_test_Drug_nolabel <- gsub("__label__2", "", temp_test_Drug_nolabel, perl=TRUE)
```

```
fileConn <- file("test_Drug_nolabel.ft.txt")
writeLines(temp_test_Drug_nolabel, fileConn)
close(fileConn)
# Training a supervised classification model with training dataset file
model_Drug <- ft_train("train_Drug.ft.txt", method = "supervised", control = ft_control(nthreads = 3L, s
# Obtain all the words from a previously trained model=
words_Drug <- ft_words(model_Drug)
```

```
# Obtain word vectors from a previously trained model.
word_vec_Drug <- ft_word_vectors(model_Drug, words_Drug)
```

```
# 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_Drug <- ft_predict(model_Drug, newdata = temp_test_Drug_nolabel)
# Reading the test file to extract the actual labels
reviewstestfile_Drug <- readLines("test_Drug.ft.txt")
# Extracting just the labels frm each line
library(stringi)
actlabels_Drug <- stri_extract_first(reviewstestfile_Drug, regex="\\w+")
# Converting the actual labels and predicted labels into factors
actlabels_Drug <- as.factor(as.character(actlabels_Drug))
ft_preds_Drug <- as.factor(as.character(ft_preds_Drug$label))
# Getting the estimate of the accuracy
library(rminer)
print(mmetric(actlabels_Drug, ft_preds_Drug, c("ACC")))
```

```
## [1] 78.68996
```