NLP with Disaster Tweets

Priyanka

6/29/2021

Natural Language Processing with Disaster Tweets

- Predict which Tweets are about real disasters and which ones are not

```
library(dplyr)
library(tidyverse)
library(tidytext)
library(tm)
library(SnowballC)
library(readr)

train_data <- read.csv("train.csv")
test_data <- read.csv("test.csv")
```

```
complete_data <- bind_rows(train_data, test_data)
glimpse(complete_data)
```

```
## Rows: 10,876
## Columns: 5
## $ id       <int> 1, 4, 5, 6, 7, 8, 10, 13, 14, 15, 16, 17, 18, 19, 20, 23, 24,~
## $ keyword  <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "~
## $ location <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "~
## $ text     <chr> "Our Deeds are the Reason of this #earthquake May ALLAH Forgi~
## $ target   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0~
```

```
head(complete_data)
```

```
##   id keyword location
## 1  1
## 2  4
## 3  5
## 4  6
## 5  7
## 6  8
##
## 1                                                                    Our Deeds are the Reason of this #e
## 2                                                                                                   For
## 3 All residents asked to 'shelter in place' are being notified by officers. No other evacuation or sh
## 4                                                                            13,000 people receive #wildfir
```

```
## 5                                              Just got sent this photo from Ruby #Alaska as smoke fi
## 6                           #RockyFire Update => California Hwy. 20 closed in both directions due to La
##   target
## 1      1
## 2      1
## 3      1
## 4      1
## 5      1
## 6      1
```

```
#-------------------- test Corpus
#Train dataset text Procesiing
#creating corpus

Corpus  <- VCorpus(VectorSource(complete_data$text))

#Transformation to lowercase
Corpus <- tm_map(Corpus, content_transformer(tolower))

#removing numbers
Corpus <- tm_map(Corpus,removeNumbers)

#removing punctuation
Corpus <- tm_map(Corpus, removePunctuation)


#removing stopwords
Corpus <- tm_map(Corpus, removeWords, stopwords())


#stemming
Corpus <- tm_map(Corpus, stemDocument)

#remove whitespaces
Corpus <- tm_map(Corpus, stripWhitespace)

as.character(Corpus[[1]])
```

```
## [1] "deed reason earthquak may allah forgiv us"
```

```
#creating dtm (bag-of-words) model
Corpus_dtm <- DocumentTermMatrix(Corpus)
Corpus_dtm
```

```
## <<DocumentTermMatrix (documents: 10876, terms: 23920)>>
## Non-/sparse entries: 101048/260052872
## Sparsity           : 100%
## Maximal term length: 51
## Weighting          : term frequency (tf)
```

```
Corpus_dtm <- removeSparseTerms(Corpus_dtm, 0.99)
Corpus_dtm
```

```
## <<DocumentTermMatrix (documents: 10876, terms: 106)>>
## Non-/sparse entries: 19142/1133714
## Sparsity           : 98%
## Maximal term length: 10
## Weighting          : term frequency (tf)
```

```r
#converting as matrix
Corpus_dtm <- as.data.frame(as.matrix(Corpus_dtm))
```

```r
Corpus_dtm$id <- as.factor(complete_data$id )

Corpus_dtm$target <- as.factor(complete_data$target)
```

```r
# Fixing incomplete cases
incomplete.cases <- which(!complete.cases(Corpus_dtm))
Corpus_dtm[incomplete.cases,] <- rep(0.0, ncol(Corpus_dtm))
```

```r
#----------------------- Classification algorithm implementation ---------------

library(caret)
set.seed(3456)
trainIndex <- createDataPartition(Corpus_dtm$target , p = .70,
                                  list = FALSE,
                                  times = 1)
train <- Corpus_dtm[ trainIndex,]
test <- Corpus_dtm[-trainIndex,]
```

```r
#------------------Model Building------------------------------
set.seed(123)
library(e1071)
# nb.model <- naiveBayes
nb.model <- naiveBayes(x = train[,-108],
                        y = train$target )

summary(nb.model)
```

```
##           Length Class  Mode
## apriori     2    table  numeric
## tables    107    -none- list
## levels      2    -none- character
## isnumeric 107    -none- logical
## call        3    -none- call
```

```r
# Predicting the Test set results
nb_pred <-  predict(nb.model, newdata = test[,-108])

# Making the Confusion Matrix
cm  <- table(test$target , nb_pred)
cm
```

```
##    nb_pred
##        0    1
```

```
##   0 1948  333
##   1  417  564
```

```
error_metric <- function(cm){
  TN = cm[1,1]
  TP = cm[2,2]
  FP = cm[1,2]
  FN = cm[2,1]
  accuracy=(TP+TN)/(TP+FP+TN+FN)
  precision =(TP)/(FP+TP)
  recall=(TP)/(FN+TP)
  F_score = 2* ((precision*recall)/(precision+recall))
  print(paste("Naive Bayes Accuracy",round(accuracy,2)))
  print(paste("Naive Bayes Precision",round(precision,2)))
  print(paste("Naive Bayes Recall",round(recall,2)))
  print(paste("Naive Bayes F_score",round(F_score,2)))
}

rf_results <- error_metric(cm)
```

```
## [1] "Naive Bayes Accuracy 0.77"
## [1] "Naive Bayes Precision 0.63"
## [1] "Naive Bayes Recall 0.57"
## [1] "Naive Bayes F_score 0.6"
```

```
rf_results
```

```
## [1] "Naive Bayes F_score 0.6"
```

```
# Fitting Kernel SVM to the Training set
# install.packages('e1071')
library(e1071)
svm.model <- svm(formula = target ~ .,
                 data = train,
                 type = 'C-classification',
                 kernel = 'radial')
```

```
svm.model
```

```
##
## Call:
## svm(formula = target ~ ., data = train, type = "C-classification",
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  4500
```

```r
summary(svm.model)
```

```
##
## Call:
## svm(formula = target ~ ., data = train, type = "C-classification",
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  4500
##
##  ( 2204 2296 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
# Predicting the Test set results
svm_pred <-  predict(svm.model, newdata = test[,-108])

# Making the Confusion Matrix
cm2 <- table(test$target , svm_pred)
cm2
```

```
##    svm_pred
##        0    1
##   0 2259   22
##   1  755  226
```

```r
error_metric <- function(cm2){
  TN = cm2[1,1]
  TP = cm2[2,2]
  FP = cm2[1,2]
  FN = cm2[2,1]
  accuracy=(TP+TN)/(TP+FP+TN+FN)
  precision =(TP)/(FP+TP)
  recall=(TP)/(FN+TP)
  F_score = 2* ((precision*recall)/(precision+recall))
  print(paste("SVM Accuracy",round(accuracy,2)))
  print(paste("SVM Precision",round(precision,2)))
  print(paste("SVM Recall",round(recall,2)))
  print(paste("SVM F_score",round(F_score,2)))
}


SVM_results <- error_metric(cm2)
```

```
## [1] "SVM Accuracy 0.76"
## [1] "SVM Precision 0.91"
## [1] "SVM Recall 0.23"
## [1] "SVM F_score 0.37"
```

```r
SVM_results
```

```
## [1] "SVM F_score 0.37"
```

```r
sample_submission <-  data.frame("id"=test$id,"target"= nb_pred)
View(sample_submission)
```

```r
# Now creating new CSV file
write.csv(sample_submission, file = "sample_submission.csv",
          row.names = FALSE)
```