Email Classifier Optimised

Frederick Jones

2023-11-19

```r
library(tm)
```

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

```r
library(data.table)
```

```r
# Function to load texts from a directory
load_texts <- function(folder) {
  file_paths <- list.files(folder, full.names = TRUE)
  texts <- vector("list", length(file_paths))
  for (i in seq_along(file_paths)) {
    texts[[i]] <- tolower(readLines(file_paths[i], warn = FALSE, n = 10, encoding = "UTF-8"))  # Read o
  }
  return(unlist(texts))
}

clean_text <- function(text) {
  iconv(text, from = "UTF-8", to = "ASCII//TRANSLIT")
}
# Load spam and ham data
# Use a subset of data if necessary
spam <- load_texts("/cloud/project/spam")
ham <- load_texts("/cloud/project/easy_ham")


spam <- lapply(spam, clean_text)
ham <- lapply(ham, clean_text)

# Combine and label data
data <- data.table(text = c(spam, ham), label = c(rep(1, length(spam)), rep(0, length(ham))))

# Text preprocessing
corpus <- VCorpus(VectorSource(data$text))
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeWords, stopwords("en"))
corpus <- tm_map(corpus, stripWhitespace)

# Create a document-term matrix with more efficient settings
dtm_controls <- list(wordLengths = c(1, Inf), weighting = weightTfIdf, bounds = list(global = c(5, Inf)
dtm <- DocumentTermMatrix(corpus, control = dtm_controls)
```

```
## Warning in weighting(x): empty document(s): 360 370 400 410 437 438 3277 3408
## 3410 3477 3479 19167 19170 19178 19180 19189 19198 19210 19218 19220 19227
## 19228 19230 19238 19239 19248 19249 19899 19909 19959 20339 20909 21360 21367
## 21370 21378 21380 21389 21390 21397 21400 21408 21410 21417 21420 21428 21429
## 21438 21440 21448 21450 21460 21468 21479 21480 21490 21497 21500 21520 21540
## 21548 21550 21553 21558 21560 21579 21588 21590 21598 21600 21607 21609 21610
## 21628 21638 21647 21650 21658 21660 21670 21673 21679 21688 21690 21697 21700
## 21708 21710 21718 21720 21728 21730 21739 21748 21750 21759 21768 21770 21778
## 21780 21788 21789 21790 21798 21800 21808 21818 21828 21838 21848 21849 21850
## 21857 21859 21860 21868 21869 21878 21879 21880 21887 21897 21898 21908 21909
## 21919 21920 21927 21928 21930 21937 21940 21949 21968 21978 21980 21988 21990
## 21998 22000 22008 22009 22018 22020 22029 22037 22040 22048 22050 22058 22068
## 22078 22080 22087 22090 22097 22100 22108 22110 22120 22128 22130 22137 22140
## 22147 22150 22158 22160 22168 22170 22180 22187 22198 22208 22218 22228 22230
## 22238 22240 22248 22257 22259 22260 22268 22269 22278 22280 22288 22290 22298
## 22300 22310 22317 22320 22328 22347 22353 22358 22366 22368 22369 22378 22380
## 22386 22389 22400 22406 22416 22426 22430 22437 22438 22440 22448 22450 22458
## 22459 22467 22468 22470 22478 22487 22489 22490 22498 22508 22510 22517 22529
## 22538 22540 22548 22558 22559 22560 22568 22578 22588 22590 22598 22600 22607
## 22608
```

```r
library(e1071)

# Reduce the size of the DTM (select only top terms)
top_terms <- findFreqTerms(dtm, lowfreq = 50)
dtm_reduced <- dtm[, top_terms]
```

```r
# Split data into training and testing sets
set.seed(42)
train_indices <- sample(seq_len(nrow(dtm_reduced)), size = 0.8 * nrow(dtm_reduced))
train_data <- dtm_reduced[train_indices, ]
test_data <- dtm_reduced[-train_indices, ]
train_labels <- data$label[train_indices]
test_labels <- data$label[-train_indices]

# Train Naive Bayes model
model <- naiveBayes(as.matrix(train_data), as.factor(train_labels))

# Predict and evaluate
predictions <- predict(model, as.matrix(test_data))
confusionMatrix <- table(Predicted = predictions, Actual = test_labels)
print(confusionMatrix)
```

```
##          Actual
## Predicted    0    1
##         0  201    9
##         1 4791 1003
```

```r
# Predict and evaluate
predictions <- predict(model, as.matrix(test_data))
confusionMatrix <- table(Predicted = predictions, Actual = test_labels)
print(confusionMatrix)
```

```
##          Actual
## Predicted    0    1
##         0  201    9
##         1 4791 1003
```

```r
# Function to predict new document
predict_new <- function(new_doc) {
  new_corpus <- VCorpus(VectorSource(new_doc))
  new_corpus <- tm_map(new_corpus, content_transformer(tolower))
  new_corpus <- tm_map(new_corpus, removePunctuation)
  new_corpus <- tm_map(new_corpus, removeWords, stopwords("en"))
  new_corpus <- tm_map(new_corpus, stripWhitespace)

  # Create a DTM for the new document
  new_dtm <- DocumentTermMatrix(new_corpus, control = list(dictionary = Terms(dtm)))

  # Predict using the model
  prediction <- predict(model, as.matrix(new_dtm))

  # Return a user-friendly message
  if (length(prediction) > 0) {
    if (prediction[1] == 1) {
      return("The email given is spam.")
    } else {
      return("The email given is not spam.")
    }
  } else {
    return("Unable to classify the email.")
  }
}
```

```r
# Example usage
new_doc <- "Congratulations! You have been selected for an exclusive offer. Get a 70% discount on our ne

But hurry! This offer is valid for a limited time only. Visit our website now and use the code EXCLUSIVE

Don't miss out on this incredible deal. Act now and be the envy of all your friends with a stylish new w
print(predict_new(new_doc))
```

```
## [1] "The email given is spam."
```