Week 13 Document Classification

MD SIMON CHOWDHURY

2024-12-8

Load Required Libraries

```
# Setting global chunk options to include echo for better understanding of code execution.
knitr::opts_chunk$set(echo = TRUE)

# Load essential libraries for data manipulation, visualization, and machine learning.
library(tidyverse)  # For data manipulation and visualization
library(dplyr)      # For data manipulation
library(stringr)    # For string operations
library(tidytext)   # For text mining
library(ggplot2)    # For data visualization
library(tm)         # For text mining and creating document-term matrices
library(readr)      # For reading text files
library(caret)      # For machine learning and data partitioning
library(randomForest) # For Random Forest classification
```

Introduction

Document classification is a powerful technique that enables the categorization of new “test” documents based on a pre-existing set of “training” documents. A common application of this technique is spam filtering, where emails are classified as either spam or ham (non-spam). For instance, a corpus of labeled spam and ham emails can be used to train a model to predict whether a new email is spam.

In this project, we utilize a spam/ham dataset to predict the class of new emails. The dataset can be sourced from public corpora such as the SpamAssassin Public Corpus, available at SpamAssassin Corpus (https://spamassassin.apache.org/old/publiccorpus/).

Loading the Dataset

```
# Define the paths to spam and ham email directories.
spam_folder <- "C:\\week 13\\spam_2"
ham_folder <- "C:\\week 13\\easy_ham"
```

Preparing and Tidying the Data

```
# Function to convert email files into a data frame with text and classification label.
to_df <- function(path, tag) {
  files <- list.files(path = path, full.names = TRUE, recursive = TRUE)
  email <- lapply(files, read_file)  # Read email contents
  email <- unlist(email)             # Convert list to vector
  data <- data.frame(email = email, tag = tag, stringsAsFactors = FALSE) # Create a data fram
e
  return(data)
}

# Create data frames for ham and spam emails.
ham_df <- to_df(ham_folder, tag = "ham")
spam_df <- to_df(spam_folder, tag = "spam")

# Combine ham and spam data frames into a single data frame.
df <- rbind(ham_df, spam_df)

# Print the distribution of spam and ham emails.
print(table(df$tag))
```

```
##
##  ham spam
## 2551 1397
```

Data Cleaning and Preprocessing

```
# Clean the email text by removing HTML tags, numbers, punctuation, and line breaks.
df <- df %>%
  mutate(email = str_remove_all(email, pattern = "<.*?>")) %>%  # Remove HTML tags
  mutate(email = str_remove_all(email, pattern = "[:digit:]")) %>%  # Remove numbers
  mutate(email = str_remove_all(email, pattern = "[:punct:]")) %>%  # Remove punctuation
  mutate(email = str_remove_all(email, pattern = "[\n]")) %>%      # Remove line breaks
  mutate(email = str_to_lower(email)) %>%                            # Convert text to lowerc
ase
  unnest_tokens(output = text, input = email, token = "paragraphs", format = "text") %>%
  anti_join(stop_words, by = c("text" = "word"))                  # Remove common stop words
```

Building a Document-Term Matrix

```
# Shuffle the dataset to ensure randomness.
set.seed(7614)
shuffled <- sample(nrow(df))
df <- df[shuffled, ]

# Convert class labels to factors.
df$tag <- as.factor(df$tag)

# Create a text corpus for further processing.
v_corp <- VCorpus(VectorSource(df$text))

# Apply text preprocessing steps to the corpus.
v_corp <- tm_map(v_corp, content_transformer(stringi::stri_trans_tolower))
v_corp <- tm_map(v_corp, removeNumbers)
v_corp <- tm_map(v_corp, removePunctuation)
v_corp <- tm_map(v_corp, stripWhitespace)
v_corp <- tm_map(v_corp, removeWords, stopwords("english"))
v_corp <- tm_map(v_corp, stemDocument)  # Apply stemming

# Create a document-term matrix and remove sparse terms.
dtm <- DocumentTermMatrix(v_corp)
dtm <- removeSparseTerms(dtm, 0.999)

# Convert DTM to binary format (1 for presence, 0 for absence of terms).
convert_count <- function(x) {
  y <- ifelse(x > 0, 1, 0)
  factor(y, levels = c(0, 1), labels = c(0, 1))
}

# Convert the sparse matrix to a data frame and add class labels.
df_matrix <- as.data.frame(as.matrix(dtm))
df_matrix$class <- df$tag

# Verify the structure of the resulting data frame.
str(df_matrix$class)
```

```
##  Factor w/ 2 levels "ham","spam": 2 2 1 1 1 1 1 2 1 1 ...
```

Building and Evaluating the Prediction Model

```
# Split the data into training (70%) and testing (30%) sets.
set.seed(7316)
prediction <- createDataPartition(df_matrix$class, p = 0.7, list = FALSE, times = 1)
training <- df_matrix[prediction, ]
testing <- df_matrix[-prediction, ]

# Train a Random Forest classifier with 400 trees.
classifier <- randomForest(x = training[, -ncol(training)],
                           y = training$class,
                           ntree = 400)

# Make predictions on the test dataset.
predicted <- predict(classifier, newdata = testing)

# Evaluate the performance using a confusion matrix.
confusionMatrix(predicted, testing$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ham spam
##       ham  764    0
##       spam   1  419
##
##                Accuracy : 0.9992
##                  95% CI : (0.9953, 1)
##     No Information Rate : 0.6461
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9982
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9987
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9976
##              Prevalence : 0.6461
##          Detection Rate : 0.6453
##    Detection Prevalence : 0.6453
##       Balanced Accuracy : 0.9993
##
##        'Positive' Class : ham
##
```

Conclusion

This project demonstrates the process of document classification using a spam/ham email dataset. Key steps included:

- Reading and processing email data from tar archives.
- Cleaning and preprocessing text to create a tidy dataset.

- Transforming the text data into a document-term matrix for analysis.
- Training a Random Forest classifier to achieve an impressive 99.92% accuracy on the test dataset.

Performance Summary:

- **Accuracy**: 99.92%
- **Sensitivity**: 99.87% (Recall for ham)
- **Specificity**: 100% (Recall for spam)
- **Kappa**: 0.9982
- **Positive Predictive Value**: 100%
- **Negative Predictive Value**: 99.76%

These metrics highlight the model's exceptional ability to classify both spam and ham emails with near-perfect precision and recall. Such high performance underscores the robustness of the Random Forest algorithm for text classification tasks.

While Random Forest performed exceptionally well, other classification methods, such as Naïve Bayes, could be explored for comparative analysis. The project highlights the practical challenges of working with text data and the potential of machine learning algorithms in solving real-world problems.