Spring Semester 2018
Iowa State University

LING 410X - Language as Data

Helpfile for Assignment 4

Author: Sowmya Vajjala

The purpose of this document is to get you started on performing text classification using RTexttools and tm libraries in R. We will be working with a dataset called SMS Spam Collection, which contains a two columns: whether the SMS is ham/spam and SMS text.

Your job in this is to perform spam vs ham text classification i.e., given a SMS (a new SMS, that is), you should be able to predict whether it is spam or ham:

We will be doing a bag of words classification. I will walk you through one bag of words classifier, and you have to think through and repeat the process with changing preprocessing (e.g., stemming, removing punctuation etc) and/or changing the classifier algorithm in train_model step. You may have to look at RTextTools and tm documentation from cran website. Most of this step by step procedure I wrote below is derived from the document written by RTextTools creators and is available here: `https://journal.r-project.org/archive/2013-1/collingwood-jurka-boydstun-etal.pdf`

First, let us start with setting the working directory to the folder where you have your data file. The below path is on my computer. Don't put it as is on your computer!

```
library(RTextTools)
library(tm)
setwd("~/Dropbox/ClassroomSlides-BothCourses/LING410X/Assignments/A4-Corpus")
```

Before you start doing other stuff, there is a small hack you need to do. RTextTools for some reason hasn't fixed some of its existing (reported) bugs. Here are two things you have to do to fix these before running the program:

Type this command in R console:

```
trace("create_matrix", edit=T)
```

This opens a file editor. In line 12, where you see textcnt(x, ...), change it to textcnt(x$content ..). So, the line becomes:

```
tokenize_ngrams <- function(x, n = ngramLength)
    return(rownames(as.data.frame(unclass(textcnt(x$content,method
    = "string", n = n)))))
```

In line 42, change "Acronym" to "acronym".

Let us now read the training and testing files into R:

```
data_train <- read.csv("sentiment_sentences_trainingdata.csv",
    header = TRUE)
data_test <- read.csv("sentiment_sentences_testdata.csv", header =
    TRUE)
names(data_train)
nrow(data_train)
nrow(data_test)
```

data_train$text is the column that contains all sentences, one per row. So, each line in this dataset is our text. Our first task now is to create a document term matrix. RTextTools has a create_matrix function for that. We give the text column name as input and it creates the matrix.

```
dtMatrix <- create_matrix(data_train$text)
```

In order to use the document term matrix to train a classification model, you need to put it into a "container" where you can also tell which is the column you will be using as predictor variable (in our case, it is vote.). This is shown below. 1:892 below just means we are using all data in gallup_train as our training data.

```
container <- create_container(dtMatrix, data_train$class,
    trainSize=1:nrow(data_train),virgin=FALSE)
```

The next step is to "train" a classification model. You can use the train_model function in tm for that. Below, I am training the model using the container I just created, and a Support Vector Machine algorithm.

```
model <- train_model(container, "SVM", kernel="linear", cost=1)
```

Model training takes time. So, please be patient.

Now, we have to test this model using the test set. Before that, we need to create the document-term matrix for test data as well, and this should match the training matrix.

```
predictionData <-data_test$text
predSize <- length(predictionData)
predMatrix <- create_matrix(predictionData, originalMatrix=dtMatrix)
```

Once the document-term matrix for test data is created, you should again create a container for this, and use that to test your classification model.

```
predictionContainer <- create_container(predMatrix,
    data_test$class, testSize=1:predSize, virgin=FALSE)
results <- classify_model(predictionContainer, model)
```

The results can be written into a csv file as follows:

```
df <- data.frame(predictionData,data_test$class,results)
write.csv(df,file = "myresults.csv")
```

You can give any filename you want. Change the name each time you train and test with a new setting (e.g., changing classifier, changing stemming settings etc) so that you can compare model predictions later.

This above tutorial used only words as features. Let us say you want to use bigrams and trigrams as well. The following code shows you how to change the document term matrix construction accordingly.

```
dtMatrixNgram <- create_matrix(gallup_train$X.text., ngramLength=3)
```

What other steps will change?