Predictive classifier for Spam Email - Alvaro Bueno

preparing files

> Note: i'm leaving the local files in my presentation for spped purposes only, the online retrieval code is attached too but it's commented

```
hamdir <- '/Users/alvbueno/Sites/dataScience/ham'
spamdir <- '/Users/alvbueno/Sites/dataScience/spam'

#the online code
# base_url <- 'https://raw.githubusercontent.com/delagroove/dataScience/master/'
# req <- GET("https://api.github.com/repos/delagroove/dataScience/git/trees/master?recursive=1")
# stop_for_status(req)
# filelist <- unlist(lapply(content(req)$tree, "[", "path"), use.names = F)
# spamfiles <- grep("spam/", filelist, value = TRUE, fixed = TRUE)
# hamfiles <- grep("ham/", filelist, value = TRUE, fixed = TRUE)

hamfiles <- list.files(hamdir)
spamfiles <- list.files(spamdir)

hamlist <- NA
spamlist <- NA

for(i in 1:length(hamfiles)){
  thelines <- readLines(paste(hamdir, hamfiles[i], sep="/"), encoding = 'UTF-8')
  thelist <- list(paste(thelines, collapse="\n"))
  hamlist <- c(hamlist, thelist)
}

hamdataframe <- as.data.frame(unlist(hamlist), stringsAsFactors = FALSE)
hamdataframe$type <- "ham"
colnames(hamdataframe) <- c('data', 'type')

for(i in 1:length(spamfiles)){
  thelines <- readLines(paste(spamdir, spamfiles[i], sep="/"), encoding = 'UTF-8')
  thelist <- list(paste(thelines, collapse="\n"))
  spamlist <- c(spamlist, thelist)
}

spamdataframe <- as.data.frame(unlist(spamlist), stringsAsFactors = FALSE)
spamdataframe$type <- "spam"
colnames(spamdataframe) <- c('data', 'type')

thedataframe = rbind(hamdataframe, spamdataframe)
```

process train and test corpus

remove punctuation, numbers and whitespace, stop words did not see to affect much the result. after that create a document matrix to create the classifier

```
sample_size <- floor(0.75 * nrow(thedataframe))
set.seed(1000)
```

```
train_ind <- sample(seq_len(nrow(thedataframe)), size = sample_size)

train_df <- thedataframe[train_ind,]
test_df <- thedataframe[-train_ind,]

spam_count <-subset(thedataframe, thedataframe$type =='spam')
ham_count <-subset(thedataframe, thedataframe$type =='ham')

train_corpus <- Corpus(VectorSource(train_df$data))
test_corpus <- Corpus(VectorSource(test_df$data))

train_corpus<- tm_map(train_corpus,removePunctuation)
train_corpus <- tm_map(train_corpus, removeNumbers)
train_corpus <- tm_map(train_corpus, stripWhitespace)

test_corpus<- tm_map(test_corpus, removePunctuation)
test_corpus <- tm_map(test_corpus, removeNumbers)
test_corpus <- tm_map(test_corpus, stripWhitespace)

test_term_matrix <- DocumentTermMatrix(test_corpus)
train_term_matrix <- DocumentTermMatrix(train_corpus)
```

once the document matrixes are ready, we feed the train matrix to the algorithm, I'm using Maxent because it performs way better than Naive Bayes and there's no need to do a filtering function.

```
classifier <- maxent::maxent(train_term_matrix, factor(train_df$type))
test_predictions <- predict(classifier, feature_matrix = test_term_matrix)
```

the results on the data frame test_predictions will show a number between 0 and 1 depending on how accurate the algorithm thinks that the text is ham or spam, we use that information to show results in a table.

```
show_results <- function (x) {
  initial_ham <- 0
  initial_spam <- 0
  predicted_ham <- 0
  predicted_spam <- 0

  for(i in 1:nrow(x)) {
    if (x[i,1] == 'ham'){
      initial_ham <- initial_ham + 1
    }
    if (x[i,1] == 'spam'){
      initial_spam <- initial_spam + 1
    }
    if (as.numeric(x[i,2]) > 0.5){
      predicted_ham <- predicted_ham + 1
    }
    if (as.numeric(x[i,3]) > 0.5){
```

```
      predicted_spam <- predicted_spam + 1
    }
  }
  array = data.frame()
  array[1,1] <- initial_ham
  array[1,2] <- initial_spam
  array[1,3] <- predicted_ham
  array[1,4] <- predicted_spam
  array
}

arr <- show_results(test_predictions)
colnames(arr) <- c('Initial Ham', 'Initial Spam', 'predicted Ham', 'Predicted Spam')
arr
```

```
##   Initial Ham Initial Spam predicted Ham Predicted Spam
## 1         971            4           970              4
```

the algorithm seems very precise and the results are close to the real thing, some times the algorithm is not able to know between spam and ham, so it will give each a value of 0.5, that's why the conditions include that number.