Text Spam Claasifier / Email Spam Classifier

Terence Lau

12/2/2018

## Data Pre-Processing

Basic situation of the data set

As the beginning, we have to do the data pre-processing:the data were divided into training group and test group, and the training group was used to train the model, and the test group was used to evaluate the model. There are three types of e-mail: 1. The hams (normal emails) which are easy to classified; 2. The hams (normal emails) which are hard to classified; 3. The **spam** (inormal emails)

# Training Dataset  
spam\_train.path <- "Email-Spam-Classifier/data/spam/"  
easy\_ham\_train.path <- "Email-Spam-Classifier/data/easy\_ham/"  
hard\_ham\_train.path <- "Email-Spam-Classifier/data/hard\_ham/"  
  
# Testing Dataset  
spam\_test.path <- "Email-Spam-Classifier/data/spam\_2/"  
easy\_ham\_test.path <- "Email-Spam-Classifier/data/easy\_ham\_2/"  
hard\_ham\_test.path <- "Email-Spam-Classifier/data/hard\_ham\_2/"

## Function Writing

# Extract Email Body ; Single Element Vector  
get.msg <- function(path) {  
 f <- file(path, open ="rt", encoding = "latin1")   
 #rt: Open for reading in text mode   
 text <- readLines(f)   
 # transform the text as "UTF-8"  
 # The message always begins after the first full line break  
 msg <- tryCatch(text[seq(which(text == "")[1] + 1, length(text), 1)], error = function(e)e)  
 #add a 'tryCatch' to catch the Exception  
 close(f)  
 return(paste(msg, collapse ="\n"))  
 # setting msg as a single element vector with '\n'  
 }

## 构建一个文本资料库

量化垃圾邮件特征词项频率的方法之一就是构造一个词项-文档矩阵(Term Document Matrix,TDM), TDM是一个N\*M矩阵,矩阵的行对应在特定语料库的所有文档中抽取的词项,列对应该语料库中所有文档. 在这个矩阵中,一个[i,j]位置的元素表.示词项i在文档j中出现的次数

构建语料库(corpus对象不懂!!!) 本案例用到了VectorSource函数构建source对象以利用邮件向量构建语料库. conrol参数的解释:告诉tm该如何清洗和规整文本 stopwords=TRUE : 移除488个最常见的英文停用词. removePunctuation , removeNumbers: 移除标点和数字 minDocFreq=2 : 文本中出现次数大于1的词才能出现在TDM的行中.

# Term Document Matrix  
get.tdm <- function(doc.vec) { #doc.vec is all the files content  
 doc.corpus <- Corpus(VectorSource(doc.vec)) # set up a corpora/natural language database  
 control <- list(stopwords = TRUE, removePunctuation = TRUE, removeNumbers = TRUE, minDocFreq = 2)  
 # This control is the option to set how to extract text (extract text requirements)  
 # remove stop word + punctution + number   
 doc.tdm <- TermDocumentMatrix(doc.corpus, control)  
 return(doc.tdm)# returning the TMD text  
}

## 定义分类器并用不易识别的正常邮件测试

为了计算一封邮件是垃圾邮件还是正常邮件的概率，需要找出待分类邮件和训练集共有的词项。让背后用这些特征的概率计算这封邮件是训练集中对应类别的条件概率，当待分类邮件中的词汇未出现在训练集合中出现时，我们可以给一个固定值或者按照某种分布给一个概率，或者用自然语言处理技术估计一个词项在特定上下文中的“垃圾倾向”。这里给固定值1e-6.

# Naive Bayes Classifier  
classify.email <- function(path, train.df, prior = 0.8, q = 1e-6) {  
 # Here, we use many of the support functions to get the  
 # email text data in a workable format  
 msg <- get.msg(path) # Read the contents of the message you want to verify  
 msg.tdm <- get.tdm(msg) # tdm  
 msg.freq <- rowSums(as.matrix(msg.tdm))  
 # Counting the number of occurrences of each word  
 # Find intersections of words  
 msg.intersect <- intersect(names(msg.freq), train.df$term) # get the intersection  
 if(length(msg.intersect) < 1){  
 return(prior \* q ^ (length(msg.freq))) # A fixed probability is given if # does not appear  
 }  
 else{  
 match.prob <- train.df$occurrence[match(msg.intersect, train.df$term)]  
 return(prior \* prod(match.prob) \* q ^ (length(msg.freq)-length(msg.intersect)))  
 }  
}  
# Finally, we can get the probability of testing e-mail

# Training :-  
  
# SPAM DATA SET  
  
# Get all the SPAM Emails into a single character vector  
spam.docs <- dir(spam\_train.path)  
spam.docs <- spam.docs[which(spam.docs != "cmds")]  
all.spam <- sapply(spam.docs, function(x) get.msg(file.path(spam\_train.path,x)))  
#every line of 'all.spam' is the context of email text

## 1.4用TDM构建一套垃圾邮件训练数据

用as.matrix将TDM转换成R的标准矩阵 rowSums创建一个向量, 每个特征在所有文档中的总频次 在构建数据框时注意stringsAsFacors=F occurrence : 计算一个特定特征词项所出现的文档在所有文档中所占比例 density: 统计整个语料库中每个词项的频次(不用此来分类, 如果想知道某些词是否影响结果,对比频次相当有用)

# DocumentTermMatrix from that vector  
spam.tdm <- get.tdm(all.spam) # training Corpus   
spam.matrix <- as.matrix(spam.tdm)  
spam.counts <- rowSums(spam.matrix)  
  
# Data frame   
spam.df <- data.frame(cbind(names(spam.counts),   
 as.numeric(spam.counts)), stringsAsFactors = FALSE)  
names(spam.df) <- c("term", "frequency")  
spam.df$frequency <- as.numeric(spam.df$frequency)  
spam.occurrence <- sapply(1:nrow(spam.matrix),function(i){  
 length(which(spam.matrix[i, ] > 0)) / ncol(spam.matrix)  
 })  
spam.density <- spam.df$frequency / sum(spam.df$frequency)  
spam.df <- transform(spam.df, density = spam.density, occurrence = spam.occurrence)

# HAM DATA SET  
  
# Get all the HAM Emails into a single character vector  
easy\_ham.docs <- dir(easy\_ham\_train.path)  
easy\_ham.docs <- easy\_ham.docs[which(easy\_ham.docs != "cmds")]  
all.easy\_ham <- sapply(easy\_ham.docs, function(x) get.msg(file.path(easy\_ham\_train.path,x)))  
  
# DocumentTermMatrix from that vector  
easy\_ham.tdm <- get.tdm(all.easy\_ham)  
easy\_ham.matrix <- as.matrix(easy\_ham.tdm)  
easy\_ham.counts <- rowSums(easy\_ham.matrix)  
  
# Data frame   
easy\_ham.df <- data.frame(cbind(names(easy\_ham.counts), as.numeric(easy\_ham.counts)), stringsAsFactors = FALSE)  
names(easy\_ham.df) <- c("term", "frequency")  
easy\_ham.df$frequency <- as.numeric(easy\_ham.df$frequency)  
easy\_ham.occurrence <- sapply(1:nrow(easy\_ham.matrix), function(i){  
 length(which(easy\_ham.matrix[i, ] > 0)) / ncol(easy\_ham.matrix)  
 })  
easy\_ham.density <- easy\_ham.df$frequency / sum(easy\_ham.df$frequency)  
easy\_ham.df <- transform(easy\_ham.df, density = easy\_ham.density, occurrence = easy\_ham.occurrence)

# HARD HAM   
  
hard\_ham.docs <- dir(hard\_ham\_train.path)  
hard\_ham.docs <- hard\_ham.docs[which(hard\_ham.docs != "cmds")]  
  
hard\_ham.spamtest <- sapply(hard\_ham.docs, function(x) classify.email(file.path(hard\_ham\_train.path, x), train.df = spam.df))  
hard\_ham.hamtest <- sapply(hard\_ham.docs, function(x) classify.email(file.path(hard\_ham\_train.path, x), train.df = easy\_ham.df))  
  
hard\_ham.res <- ifelse(hard\_ham.spamtest > hard\_ham.hamtest, TRUE, FALSE)  
# Check  
summary(hard\_ham.res)

## Mode FALSE TRUE   
## logical 238 11

# Test DATA  
spam.classifier <- function(path) #path指定所有文件路径，进行分类func  
{  
 pr.spam <- classify.email(path, spam.df)  
 pr.ham <- classify.email(path, easy\_ham.df)  
 return(c(pr.spam, pr.ham, ifelse(pr.spam > pr.ham, 1, 0)))  
}  
  
# Get lists of all the email messages  
easy\_ham2.docs <- dir(easy\_ham\_test.path)  
easy\_ham2.docs <- easy\_ham2.docs[which(easy\_ham2.docs != "cmds")]  
  
hard\_ham2.docs <- dir(hard\_ham\_test.path)  
hard\_ham2.docs <- hard\_ham2.docs[which(hard\_ham2.docs != "cmds")]  
  
spam2.docs <- dir(spam\_test.path)  
spam2.docs <- spam2.docs[which(spam2.docs != "cmds")]  
  
# Classify them all!  
#易于辨别的正常邮件  
easy\_ham2.class <- suppressWarnings(lapply(easy\_ham2.docs,  
 function(x)  
 {  
 spam.classifier(file.path(easy\_ham\_test.path, x))  
 #不易于辨别的正常邮件  
 }))  
hard\_ham2.class <- suppressWarnings(lapply(hard\_ham2.docs,  
 function(x)  
 {  
 spam.classifier(file.path(hard\_ham\_test.path, x))  
 }))  
#垃圾邮件   
spam2.class <- suppressWarnings(lapply(spam2.docs,  
 function(x)  
 {  
 spam.classifier(file.path(spam\_test.path, x))  
 }))  
# Create a single, final, data frame with all of the classification data in it数据整合  
easy\_ham2.matrix <- do.call(rbind, easy\_ham2.class)  
easy\_ham2.final <- cbind(easy\_ham2.matrix, "EASYHAM")  
  
hard\_ham2.matrix <- do.call(rbind, hard\_ham2.class)  
hard\_ham2.final <- cbind(hard\_ham2.matrix, "HARDHAM")  
  
spam2.matrix <- do.call(rbind, spam2.class)  
spam2.final <- cbind(spam2.matrix, "SPAM")  
  
class.matrix <- rbind(easy\_ham2.final, hard\_ham2.final, spam2.final)  
class.df <- data.frame(class.matrix, stringsAsFactors = FALSE)  
names(class.df) <- c("Pr.SPAM" ,"Pr.HAM", "Class", "Type")  
class.df$Pr.SPAM <- as.numeric(class.df$Pr.SPAM)  
class.df$Pr.HAM <- as.numeric(class.df$Pr.HAM)  
class.df$Class <- as.logical(as.numeric(class.df$Class))  
class.df$Type <- as.factor(class.df$Type)  
  
# Create final plot of results  
class.plot <- ggplot(class.df, aes(x = log(Pr.HAM), log(Pr.SPAM))) +  
 geom\_point(aes(shape = Type, alpha = 0.5)) +  
 scale\_shape\_manual(values = c( "EASYHAM" = 1, "HARDHAM" = 2, "SPAM" = 3), name = "Email Type") +  
 scale\_alpha(guide = "none") +  
 xlab("log[P(HAM)]") +  
 ylab("log[P(SPAM)]") +  
 #theme\_bw() +  
 theme(axis.text.x = element\_blank(), axis.text.y = element\_blank())  
  
 ggsave(plot = class.plot, filename = file.path("Email-Spam-Classifier/images", "result.png"), height = 10, width = 10)  
  
get.results <- function(bool.vector)  
{  
 results <- c(length(bool.vector[which(bool.vector == FALSE)]) / length(bool.vector),  
 length(bool.vector[which(bool.vector == TRUE)]) / length(bool.vector))  
 return(results)  
}  
  
# Save results as a 2x3 table  
EasyHam <- get.results(subset(class.df, Type == "EASYHAM")$Class)  
HardHam <- get.results(subset(class.df, Type == "HARDHAM")$Class)  
Spam <- get.results(subset(class.df, Type == "SPAM")$Class)  
  
class.res <- rbind(EasyHam, HardHam, Spam)  
colnames(class.res) <- c("NOT SPAM", "SPAM")  
print(class.res)

## NOT SPAM SPAM  
## EasyHam 0.9871429 0.01285714  
## HardHam 0.9556452 0.04435484  
## Spam 0.2870437 0.71295634

write.csv(spam.df, file.path("Email-Spam-Classifier/data", "spam\_df.csv"), row.names = FALSE)  
write.csv(easy\_ham.df, file.path("Email-Spam-Classifier/data", "easyham\_df.csv"), row.names = FALSE)  
  
write.csv(class.df, file.path("Email-Spam-Classifier/Results", "result\_df.csv"), row.names = FALSE)  
write.table(class.res, file.path("Email-Spam-Classifier/Results", "classify.csv"), row.names=FALSE)