Naive Bayes

Steve Pittard

June 8, 2016

Introduction

The purpose of the document is to reproduce the section on Naive Bayes from the book Data Smart. The data used in this document were downloaded from http://www.wiley.com/WileyCDA/WileyTitle/productCd-111866146X.html. In particular we will look at Chapter 3 "Naive Bayes and the Incredible Lightness of Being an Idiot". The concepts here are based largely on an understanding of conditional and joint probabbilities along with Bayes Theorem. The ultimate idea is to look at some tweets and determine if they relate to a software application named "Mandrill" or not. The chapter goes into some explanations of probability theory which we won't go into here as the goal is to reproduce the Excel computations with R code and in a way that is reproducible. Anyone with this document can generate a working version of this code and an associated PDF - you will need the RStudio IDE which can be obtained from https://www.rstudio.com/

Reading Excel Files

There are a number of ways to read an Excel workbook or worksheet into R. Here we we will use the **readxl** package which greatly simplifies this effort. This isn't the only way of course.

```
library(readxl)

mandrill1 <- read_excel("Mandrill.xlsx",1)

mandrill2 <- mandrill1[,1]     # Pull out the tweet

mandrill2 <- mandrill2[complete.cases(mandrill2),]   # Get only complete cases
```

Next we will create a function that will do some cleaning on each tweet. This is basically to pull out extra spaces and various punctuation marks. We'll stick with what the book outlined but there is a twitteR package that does lots of cleanup for you.

Cleaning the Tweets (Sort of)

```
# Let's create a function to do cleanup. There are some packages that have twitter specific
# functions to do this but let's mimic what we see in the Mandrill example

cleaners <- function(x) {
  x <- tolower(x)
  x <- gsub("\\. "," ",x)
  x <- gsub("\\: "," ",x)
  x <- gsub("\\?"," ",x)
  x <- gsub("\\!"," ",x)
  x <- gsub("\\;"," ",x)
  x <- gsub("\\,"," ",x)
```

```
#  x <- gsub("[[:punct:]]","",x)
  return(x)
}
```

So now we will apply this function to the tweets to clean them up. In a "real" example we might also pull out stop words that aren't significant to an overall understanding of the patterns (if any) contained in the body of text. But we are sticking to the script in the book.

```
mandrill2 <- sapply(mandrill2,cleaners)

tokens_per_tweet <-  apply(as.matrix(mandrill2),1,strsplit," ")

total_tokens <- unlist(tokens_per_tweet)

token_table <- table(total_tokens)

head(token_table,20)
```

```
## total_tokens
##                                        -                     -
##                   178                 13                     3
##                    :/                 ..                   ...
##                     2                  5                    18
##             'migrate'                 "i            "mandrill"
##                     1                  1                     1
##                     (       (@mandrillapp)               (0.0.3)
##                     2                  1                     1
##               (0.0.4)            (1.0.19)              (1.0.25)
##                     1                  1                     1
##                   (by                 (cc          (confirmação
##                     1                  1                     1
## (http://j.mp/10tohxc (http://j.mp/10tohxg
##                     1                  1
```

```
# Let's find the count for a given token and check it against
# the spreadsheet as a spot check

token_table[which(names(token_table)=="your")]
```

```
## your
##   11
```

```
# Now strip out the tokens that are <= 3 characters long

final_token_table <- token_table[nchar(names(token_table)) > 3]

final_token_table[1:20]
```

```
## total_tokens
##             'migrate'            "mandrill"       (@mandrillapp)
##                     1                     1                     1
```

```
##              (0.0.3)              (0.0.4)             (1.0.19)
##                    1                    1                    1
##             (1.0.25)         (confirmação (http://j.mp/10tohxc
##                    1                    1                    1
## (http://j.mp/10tohxg                (i.e                 (ils
##                    1                    1                    1
##                (line          (one-to-one                (some
##                    1                    1                    1
##               (their               [blog]          @aalbertson
##                    1                    2                    1
##          @abhijeetmk       @adrienneleigh
##                    1                    1
```

```
probs <- as.vector((final_token_table+1)/sum(final_token_table+1))
names(probs) <- names(final_token_table)

# According to the text the probability for the "support" token is .002074689
# Let's do a spot check

probs[which(names(probs)=="support")]
```

```
##     support
## 0.002074689
```

```
# support
# 0.002074689
```

Create a data frame for the APP related tweets

So next we will take the natural logarithm of these probabilities since they are small. We can then do summation on these. Let's also take the opportunity here to put things into a data frame which is the equivalent of an Excel worksheet. Strictly speaking this isn't necessary but is convneinet for readability.

```
ln_probs <- log(probs)

app_prob_df <- data.frame(token=names(probs),
                      token_count=as.vector(final_token_table),
                      probs=as.vector(probs),
                      ln_probs=as.vector(ln_probs),
                      stringsAsFactors = FALSE)

head(app_prob_df)
```

```
##            token token_count        probs  ln_probs
## 1      'migrate'           1 0.0008298755 -7.094235
## 2     "mandrill"           1 0.0008298755 -7.094235
## 3 (@mandrillapp)           1 0.0008298755 -7.094235
## 4        (0.0.3)           1 0.0008298755 -7.094235
## 5        (0.0.4)           1 0.0008298755 -7.094235
## 6       (1.0.19)           1 0.0008298755 -7.094235
```

Process the non APP related tweets

So let's do the same thing for the non APP related tweets. This is easy and we can leverage code that we already have above.

```
mandrill1 <- read_excel("Mandrill.xlsx",2)

mandrill2 <- mandrill1[,1]
mandrill2 <- mandrill2[complete.cases(mandrill2),]


mandrill2 <- sapply(mandrill2,cleaners)

tokens_per_tweet <-  apply(as.matrix(mandrill2),1,strsplit," ")

total_tokens <- unlist(tokens_per_tweet)

token_table <- table(total_tokens)

#

mandrill1 <- read_excel("Mandrill.xlsx",2)

mandrill2 <- mandrill1[,1]
mandrill2 <- mandrill2[complete.cases(mandrill2),]


mandrill2 <- sapply(mandrill2,cleaners)

tokens_per_tweet <-  apply(as.matrix(mandrill2),1,strsplit," ")

total_tokens <- unlist(tokens_per_tweet)

token_table <- table(total_tokens)

# Let's find the count for a given token and check it against
# the spreadsheet as a spot check

token_table[which(names(token_table)=="#nameanamazingband")]
```

```
## #nameanamazingband
##                  3
```

```
# Now strip out the tokens that are <= 3 characters long

final_token_table <- token_table[nchar(names(token_table)) > 3]


probs <- as.vector((final_token_table+1)/sum(final_token_table+1))
names(probs) <- names(final_token_table)


ln_probs <- log(probs)
```

```
other_prob_df <- data.frame(token=names(probs),
                            token_count=as.vector(final_token_table),
                            probs=as.vector(probs),
                            ln_probs=as.vector(ln_probs),
                            stringsAsFactors = FALSE)

head(other_prob_df)
```

```
##           token token_count        probs  ln_probs
## 1   .@katie_phd           1 0.0009876543 -6.920178
## 2   .@theophani           1 0.0009876543 -6.920178
## 3          .hru           1 0.0009876543 -6.920178
## 4          .xzw           1 0.0009876543 -6.920178
## 5 'mandrillus'.           1 0.0009876543 -6.920178
## 6  'reproachful           1 0.0009876543 -6.920178
```

Now we process the Test Tweets

So all of the above was in preparation to predict whether a test tweet relates to the Mandrill application or not. In reality we already know since the test tweet data is labelled but let's see how our approach works. The idea is that if we look at the probabilities associated with the words in a tweet showing up in the APP related data drame and compare them to the probabilities of them showup up in the non APP related data frame then we can make a guess.

```
test_tweets <- read_excel("Mandrill.xlsx",7)

test_tweets <- test_tweets[,2:3]

# Normalize the test tweets like above.

just_tweets <- as.list(sapply(test_tweets[,2],cleaners))

# Split up the tweets into a list
split_tweets <- sapply(just_tweets,strsplit," ")
```

So next we will write a function to process of the word tokens contained within each test tweet. Some of the logic below is based on how R represents missing values. For example we need to look at each word in a tweet and see if it exists in the APP related and non APP related data frames. If the look up is successful then we are okay but if not then R doesn't return a NA but a zero length numeric. The below logic handles that.

```
# Next we write a function to process each of the tokens within each tweet

token_probs <- function(y,df) {
  tmp_probs <- vector()
  length(tmp_probs) <- length(y)
  for (ii in 1:length(y)) {

    # If the token is 3 chars or less then set prob to 0

    if (nchar(y[ii]) <= 3) {
```

```
        tmp_probs[ii] <- 0
    } else {
        tmp <- df[df$token==y[ii],]$ln_probs

     # If token is not in data frame then apply formula given from text

        if ( length(tmp) == 0 ) {
            tmp_probs[ii] <- log(1/sum(df$token_count+1))
        } else {

     # We found the token and pull out the associated log prob

            tmp_probs[ii] <- tmp
        }
    }
  }
  return(sum(tmp_probs))
}
```

So now we can make the predictions.

```
approbs    <- sapply(split_tweets,token_probs,app_prob_df)
otherprobs <- sapply(split_tweets,token_probs,other_prob_df)

# Here are the predictions - Looks like we were incorrect on Test tweet 19

test_tweets[,1] == ifelse(approbs > otherprobs,"APP","OTHER")
```

```
##       Class
##  [1,]  TRUE
##  [2,]  TRUE
##  [3,]  TRUE
##  [4,]  TRUE
##  [5,]  TRUE
##  [6,]  TRUE
##  [7,]  TRUE
##  [8,]  TRUE
##  [9,]  TRUE
## [10,]  TRUE
## [11,]  TRUE
## [12,]  TRUE
## [13,]  TRUE
## [14,]  TRUE
## [15,]  TRUE
## [16,]  TRUE
## [17,]  TRUE
## [18,]  TRUE
## [19,] FALSE
## [20,]  TRUE
```