Milestone Report

Brian Perron

August 28, 2014

# Overview

This report is submitted in partial fulfillement of the Data Science Specialization Swiftkey Capstone. The focus of the capstone project involves applying data science to the area of Natural Language Processing. Students are provided with text data from a corpus called HC Corpora (from www.corpora.heliohost.org). Using these data, students are expected to:

* Develop a predictive text model that is tested on a real data set
* Create a reproducible R markdown document describing your model building process
* Build a Shiny or Yhat application to demonstrate the use of the product

This is a milestone report that summarizes the major tasks and activities to date. These tasks include understanding the problem (Task 0); data acquisition and cleaning (Task 1); exploratory analysis (Task 2); modeling (Task 3); and prediction (Task 4). This report is organized around each of these major tasks.

## Task 0: Understanding the problem

What is the problem -- what is to be developed? What is the domain -- what is natural language processing (NLP)

Data were available in four different languages -- English, German, Russian, and Finnish. For this project, I chose to work with only the English text data because I do not have competencies with any of the other languages, which would place significant limits on my ability to clean the data and make meaningful interpretations of models.

For each language, three different sources of data were available: Twitter, blogs, and news reports. While these sources of data are all text-based communications, I was concerned that these data could not be used as a basis for a single prediction model. More specifically, the structure of the language of a tweet is limited to 140 characters, requiring authors to construct short sentences that commonly include abbreviations, emoticons, and special symbols (e.g., hashtags and ampersands). Tweets may have grammatical errors to meet the 140 character limit, but grammatically incorrect errors are rarely criticized. Writing a blog or a news story is a fundamentally different task than a tweet. For example, a blog or news story requires some development of an idea that extends far beyond 140 characters. Blogs and news stories are built on more more strict usage of grammatical rules.

In light of these differences, I have selected to use only one of the three data sources -- i.e., blogs -- as my primary data source. As time allows, I will also systematically explore and contrast the properties of the other data sources to help determine whether it is reasonable, from an empirical perspective, to build a final model on hetergenous data sources.

## Task 1: Data acquisition and cleaning

The goal of this first task is to obtain the data and produce a tidy data set to facilitate subsequent exploration and analysis. As noted, this will involve using text data from blogs written in English.

The following code chunk initializes the R work space by calling the necessary libraries and reading the blog data. Because no formal analyses are being conducted at this time, only a small subset of the data are being read into the R workspace. This is done to ensure fast processing time.

libs <- c("tm", "SnowballC", "XML", "ggplot2", "wordcloud", "tau")  
lapply(libs, require, character.only = TRUE)  
  
# Set the following path to the location of the text file  
blogs <- readLines("~/Git/capstoneCoursera/en\_US/blogs.txt", n = 100)

Do some initial cleaning of the text file:

blogs <- tolower(blogs)  
blogs <- gsub("/", "", blogs)  
blogs <- gsub("'", '', blogs)  
blogs <- gsub("’", '', blogs)  
blogs <- gsub("‘", '', blogs)  
blogs <- gsub("′", '', blogs)  
blogs <- gsub("”", '', blogs)  
blogs <- gsub("“", '', blogs)  
blogs <- gsub("@", "", blogs)  
blogs <- gsub("–", "", blogs)  
blogs <- gsub("\\|", "", blogs)  
blogs <- gsub("[!?,.]+", ".", blogs)  
blogs <- gsub("…", "", blogs)  
blogs <- gsub('[])(;:#%$^\\~{}[&+=@/"<>\_]+', "", blogs)  
blogs <- removeWords(blogs, stopwords("english"))  
blogs <- stripWhitespace(blogs)

Extract profanity by using a pre-defined dictionary from <http://www.bannedwordlist.com/>

swearWords <- xmlParse(file = "~/Git/capstoneCoursera/supplement/swearWords.xml")  
swearWords <- xmlToList(swearWords)  
swearWords <- data.frame(unlist(swearWords))

Identify how many records match a profanity match:

swearNumber <- length(unique(grep(paste(swearWords[1:77,], collapse = "|"),   
 blogs, value= TRUE)))  
  
blogNumber <- length(blogs)  
  
print(100\*(swearNumber / blogNumber))

## [1] 14

Create a profanity filter that removes profane language from the files. The results code checks the output.

for(i in 1:77)  
 { #Create a loop for the profanity filer  
 blogsLoop <- gsub(swearWords[i,1], "", blogs)  
 }  
  
swearNumber <- length(unique( #Double check the output  
 grep(paste(swearWords[1:77,], collapse = "|"),   
 blogsLoop, value= TRUE)))  
  
blogNumberLoop <- length(blogsLoop)  
  
print(100\*(swearNumber / blogNumberLoop))

## [1] 14

Create a corpus:

txt <- VectorSource(blogs);   
txt.corpus <- Corpus(txt)

Clean-up the corpus with the {tm\_map} function:

txt.corpus <- tm\_map(txt.corpus, content\_transformer(tolower))  
txt.corpus <- tm\_map(txt.corpus, removeNumbers)  
txt.corpus <- tm\_map(txt.corpus, removePunctuation)  
txt.corpus <- tm\_map(txt.corpus, removeWords, stopwords("english"))  
txt.corpus <- tm\_map(txt.corpus, stripWhitespace)  
txt.corpus <- tm\_map(txt.corpus, stemDocument) #From SnowballC library

Create a Document Term Matrix {dtm / tdm}:

dtm <- DocumentTermMatrix(txt.corpus)  
tdm <- TermDocumentMatrix(txt.corpus)

Do an initial inspection of the first 20 rows of the tdm:

inspect(dtm[1:20,1:10])

## <<DocumentTermMatrix (documents: 20, terms: 10)>>  
## Non-/sparse entries: 1/199  
## Sparsity : 100%  
## Maximal term length: 7  
## Weighting : term frequency (tf)  
##   
## Terms  
## Docs abil abl acceler accid accord achiev across act action actress  
## 1 0 0 0 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 0 0 0  
## 7 0 0 0 0 0 0 0 0 0 0  
## 8 0 0 0 0 0 0 0 0 0 0  
## 9 0 0 0 0 0 0 0 0 0 0  
## 10 0 0 0 0 0 0 0 0 0 0  
## 11 0 0 0 0 0 0 0 0 0 0  
## 12 0 0 0 0 0 0 0 0 0 0  
## 13 0 0 0 0 0 0 0 0 0 0  
## 14 0 0 0 0 0 0 0 0 0 0  
## 15 0 0 0 0 0 0 0 0 0 0  
## 16 0 0 0 0 0 0 0 0 0 0  
## 17 0 1 0 0 0 0 0 0 0 0  
## 18 0 0 0 0 0 0 0 0 0 0  
## 19 0 0 0 0 0 0 0 0 0 0  
## 20 0 0 0 0 0 0 0 0 0 0

inspect(tdm[1:20, 1:10])

## <<TermDocumentMatrix (terms: 20, documents: 10)>>  
## Non-/sparse entries: 2/198  
## Sparsity : 99%  
## Maximal term length: 7  
## Weighting : term frequency (tf)  
##   
## Docs  
## Terms 1 2 3 4 5 6 7 8 9 10  
## abil 0 0 0 0 0 0 0 0 0 0  
## abl 0 0 0 0 0 0 0 0 0 0  
## acceler 0 0 0 0 0 0 0 0 0 0  
## accid 0 0 0 0 0 0 0 0 0 0  
## accord 0 0 0 0 0 0 0 0 0 0  
## achiev 0 0 0 0 0 0 0 0 0 0  
## across 0 0 0 0 0 0 0 0 0 0  
## act 0 0 0 0 0 0 0 0 0 0  
## action 0 0 0 0 0 0 0 0 0 0  
## actress 0 0 0 0 0 0 0 0 0 0  
## actual 0 0 0 0 0 0 0 0 0 0  
## addict 0 0 0 0 0 0 0 0 0 0  
## addit 0 0 0 0 0 0 0 0 0 0  
## address 0 0 0 0 0 0 0 0 0 0  
## adjac 0 0 0 0 0 0 0 0 0 0  
## admiss 0 0 0 0 0 0 0 0 0 0  
## adopt 0 0 0 0 0 0 0 0 0 0  
## afford 0 0 0 0 0 0 0 0 1 1  
## age 0 0 0 0 0 0 0 0 0 0  
## aggre 0 0 0 0 0 0 0 0 0 0

Examine the most frequent terms in the tdm:

findFreqTerms(x = tdm, lowfreq=500, highfreq=Inf)

## character(0)

Exploring the Document Term Matix

freq <- colSums(as.matrix(dtm))  
length(freq)

## [1] 1375

#Least frequent terms  
ord <- order(freq)  
freq[head(ord)]

## abil acceler accid accord achiev across   
## 1 1 1 1 1 1

#Most frequent terms  
freq[tail(ord)]

## peopl get one will time like   
## 14 15 15 19 20 25

Conversion to Matrix and Save to CSV

m <- as.matrix(dtm)

findFreqTerms(dtm, lowfreq=100)

## character(0)

findAssocs(dtm, "like", corlimit=.15)

## like  
## jelli 0.65  
## accord 0.62  
## appli 0.62  
## creami 0.62  
## dirt 0.62  
## dri 0.62  
## effort 0.62  
## emuls 0.62  
## layer 0.62  
## liquid 0.62  
## lukewarm 0.62  
## milki 0.62  
## product 0.62  
## requir 0.62  
## rub 0.62  
## skin 0.62  
## spread 0.62  
## thick 0.62  
## wash 0.62  
## start 0.56  
## water 0.54  
## your 0.54  
## either 0.51  
## face 0.49  
## actual 0.41  
## begin 0.41  
## involv 0.41  
## minut 0.41  
## remain 0.41  
## white 0.41  
## within 0.39  
## bit 0.38  
## seem 0.34  
## almost 0.31  
## end 0.31  
## quit 0.31  
## sure 0.31  
## take 0.31  
## come 0.30  
## acceler 0.29  
## achiev 0.29  
## addit 0.29  
## along 0.29  
## amount 0.29  
## ankl 0.29  
## axi 0.29  
## backward 0.29  
## bent 0.29  
## blog 0.29  
## blogrol 0.29  
## chronolog 0.29  
## clap 0.29  
## code 0.29  
## comic 0.29  
## energi 0.29  
## expect 0.29  
## extrem 0.29  
## extrema 0.29  
## felt 0.29  
## fin 0.29  
## fish 0.29  
## flexibl 0.29  
## forc 0.29  
## frog 0.29  
## grab 0.29  
## hero 0.29  
## hydrofoil 0.29  
## impuls 0.29  
## joint 0.29  
## kick 0.29  
## kinet 0.29  
## kitten 0.29  
## knee 0.29  
## leg 0.29  
## lower 0.29  
## maintain 0.29  
## maxim 0.29  
## mimic 0.29  
## muscl 0.29  
## nozzl 0.29  
## opinion 0.29  
## outer 0.29  
## outward 0.29  
## perman 0.29  
## pfftshite 0.29  
## ponder 0.29  
## posit 0.29  
## precis 0.29  
## prove 0.29  
## punch 0.29  
## rather 0.29  
## reader 0.29  
## rotat 0.29  
## smaller 0.29  
## sole 0.29  
## solut 0.29  
## stand 0.29  
## swim 0.29  
## therefor 0.29  
## transfer 0.29  
## twist 0.29  
## type 0.29  
## unfound 0.29  
## unhealthi 0.29  
## unlik 0.29  
## unrealist 0.29  
## upper 0.29  
## wade 0.29  
## wasnt 0.29  
## worri 0.29  
## littl 0.28  
## enjoy 0.25  
## fun 0.25  
## tri 0.25  
## feet 0.24  
## insid 0.24  
## point 0.24  
## small 0.24  
## togeth 0.24  
## best 0.22  
## fear 0.22  
## much 0.22  
## think 0.22  
## realli 0.21  
## turn 0.20  
## alreadi 0.18  
## brought 0.18  
## close 0.18  
## color 0.18  
## creat 0.18  
## distanc 0.18  
## effect 0.18  
## fit 0.18  
## fort 0.18  
## high 0.18  
## human 0.18  
## lift 0.18  
## perfect 0.18  
## popular 0.18  
## possibl 0.18  
## power 0.18  
## rear 0.18  
## sat 0.18  
## seen 0.18  
## separ 0.18  
## sever 0.18  
## situat 0.18  
## strong 0.18  
## term 0.18  
## thoma 0.18  
## thus 0.18  
## wife 0.18  
## bomb 0.17  
## design 0.17  
## dont 0.17  
## enough 0.17  
## money 0.17  
## restaur 0.17  
## special 0.17  
## two 0.17  
## usual 0.17  
## also 0.16  
## winter 0.16  
## work 0.16  
## time 0.15

#plot(dtm,   
 # terms = findFreqTerms(dtm, lowfreq=100)[1:50],   
 # corThreshold = .10)  
   
freq <- sort(colSums(as.matrix(dtm)), decreasing=TRUE)  
head(freq, 14)

## like time will get one peopl also come make know thing just   
## 25 20 19 15 15 14 13 11 11 10 10 9   
## love work   
## 9 9

wf <- data.frame(word = names(freq), freq=freq)  
  
p <- ggplot(subset(wf, freq > 500), aes(word, freq))  
  
p <- p + geom\_bar(stat="identity")  
p <- p + theme(axis.text.x=element\_text(angle=45, hjust=1))  
print(p)



#library(wordcloud)  
#set.seed(123)  
#wordcloud(names(freq), freq, min.freq=100)

blogs <- readLines("~/Git/capstoneCoursera/en\_US/blogs.txt", n=15000)  
blogs <- stemDocument(blogs)  
blogs <- stripWhitespace(blogs)  
blogs <- tolower(blogs)  
blogs <- removeWords(blogs, stopwords("english"))  
  
bigrams <- textcnt(blogs, n = 2, method="string")  
bigrams <- bigrams[order(bigrams, decreasing = TRUE)]  
  
trigrams <- textcnt(blogs, n = 3, method="string")  
trigrams <- trigrams[order(trigrams, decreasing = TRUE)]  
print(bigrams[1:10])

## new york can see year old first time right now last year   
## 94 89 82 81 81 79   
## years ago make sure feel like even though   
## 78 77 68 67

print(trigrams[1:10])

## mother s day new york city new york times couple years ago   
## 14 12 9 8   
## new york n year old daughter york n y ” said “   
## 8 8 8 8   
## happy new year m looking forward   
## 7 7