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.

```r
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)
```

Cleaning procedures using gsub

Here I have done some initial cleaning of the data using `gsub`. I chose this function over other libraries for cleaning, as I wanted to perform the cleaning procedure more interactively, in order to better understand what is doing on in each step.

```r
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 <- removeWords(blogs, stopwords("english"))
blogs <- stripWhitespace(blogs)
head(blogs)
```

```
## [1] " years thereafter. oil fields platforms named pagan gods."
## [2] " love mr. brown."
## [3] "chad awesome kids holding fort work later usual. kids busy together playing skylander xbox toge
## [4] " anyways. going share home decor inspiration storing folder puter. amazing images stored away r
## [5] " graduation season right around corner. nancy whipped fun set help graduation cards gifts. occa
## [6] " alternative argument. hear . "
```

This cleaning procedure worked well for initial cleaning, although a few problems still remain. For example, I am not clear why the `stripWhitespace` function didn't work. And, why doesn't the `gsub` function call with the arguments `[])(;:#%$^\\~{}[&+=@/"<>_]+...` unable to pick up the smiley fact `:)` in the text? Also, I need a better understanding of the different encoding in R – that is, should I be converting an entire text file from one encoding format (e.g., UTF-8) to another format (e.g., ASCII)? This is on my to-do list to better understand what is going on in the programming environment.

Profanity filtering

The next task was to determine how to address the problem of profanity – that is, do I include or exclude profane language from the data. This represents an important decision point. On one hand, we might not want to create an application that makes predictions of profane language or uses profane language as part of the prediction model. In doing so, the application might not be appropriate for adolescents or other people who are bothered by profane language. On the other hand, profane language is deliberate language that is used to convey an idea. Such language cannot simply be replaced with another word while maintaining the same connotation. Therefore, excluding or replacing profane language could create problems in model performance.

To better understand this problem, a descriptive analysis of profane language was condcuted. This purpose of the descriptive analysis was to determine the actual amount of profane language contained in the documents and consider whether any exclusions or replacements of language would influence subsequent analyses.

This analysis was performed by using a pre-defined profanity dictionary that can be downloaded from http://www.bannedwordlist.com/. The following code chunk uses the xml file, which has been saved locally.

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

After the profanity dictionary has been loaded into memory, a procedure was written to identify the records that contain a profanity match. This is used to generate a percentage of blog posts that contain profane language.

```
swearNumber <- length(unique(grep(paste(swearWords[1:77,], collapse = "|"),
     blogs, value= TRUE)))

blogNumber <- length(blogs)
```

```
## [1] "Percent of posts with profanity: 14"
```

This is a fairly significant number, so it makes sense to visually inspect the occurences to determine how changes (e.g., replacing or excluding profane language) might influence subsequent analyses.

```
grep(paste(swearWords[1:77,], collapse = "|"), blogs, value= TRUE)
```

```
##  [1] " graduation season right around corner. nancy whipped fun set help graduation cards gifts. occa
##  [2] "pure large leaf assam. waffling leaf. thank . want strong dark herby frills. goodness sake frui
##  [3] "\" anc earlier approached south gauteng high court grant us leave appeal earlier ruling singing
##  [4] " costume. pricklewood. im real mccoy. got onto carpet. grasped feet armchair toes lifted ground
##  [5] " many times heard someone say -- nice tell person theyre seeing want call quits. nice address w
##  [6] " lovin people. benefit good backing band consisting pretty purdie drums. artie butler organ & p
##  [7] " worried wasnt going like . unhealthy love comic book heroes unrealistic expectations. . worrie
##  [8] " stars making magazines 2011 sexiest women alive list include model-turned-actress rosie huntin
##  [9] " napped studio class tonight. andrew really late last night. despite going bed somewhat decent
## [10] "actually. april 15. 1912. headline based preliminary news error. final tally shows 1.513 lives
## [11] " recently tried super fabulous restaurant must must try. usually give restaurant recommendation
## [12] "hell larger today yesterday. many us failed pray.-david smithers"
## [13] "valentines day falls february 14th year. holiday lovers express said love . traditionally flowe
## [14] "even dont like called screwball comedy. critic also called sex comedy without sex. whose troubl
```

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)
```

```
## [1] "14"
```