Corpus Exploration

Wally Thornton

December 15, 2015

Executive Summary

The ultimate goal of this project is to create a Shiny web app that predicts the next word you will need as you type, much like the predictive text that the SwiftKey apps feature. Such apps need to be trained, which is done by analyzing documents of real-world text. A collection of these training documents is called a corpus, and we look at the word sequences in a corpus to build a predictive model. For example, if "today" is usually the word that follows "it will rain" in the corpus, we can predict that a user of our app will want "today" if they type "it will rain" in the app. The best text predictors, like SwiftKey's, continue to learn as you use it, since you may have typing tendencies that differ from those in the corpus.

Before we build a predictive model, we first need to analyze the text in the corpus to get a sense of what we're dealing with. We want to answer questions like, "How many lines of text do we have? How long is each line? Does the length differ depending on what the source is? How much will we need to clean up the text to be able to use it in our modeling?" After this analysis, we'll process the text and then build and test our models.

This paper covers the efforts of that initial exploratory text analysis. The corpus provided was built with lines of random, anonymized news, blogs and tweets from the U.S. We'll find below that the sources vary greatly in length and need for processing, and the paper then ends with our plan for creating a prediction algorithm and Shiny app. (In the interest of keeping the paper concise, all code is in the Appendix.)

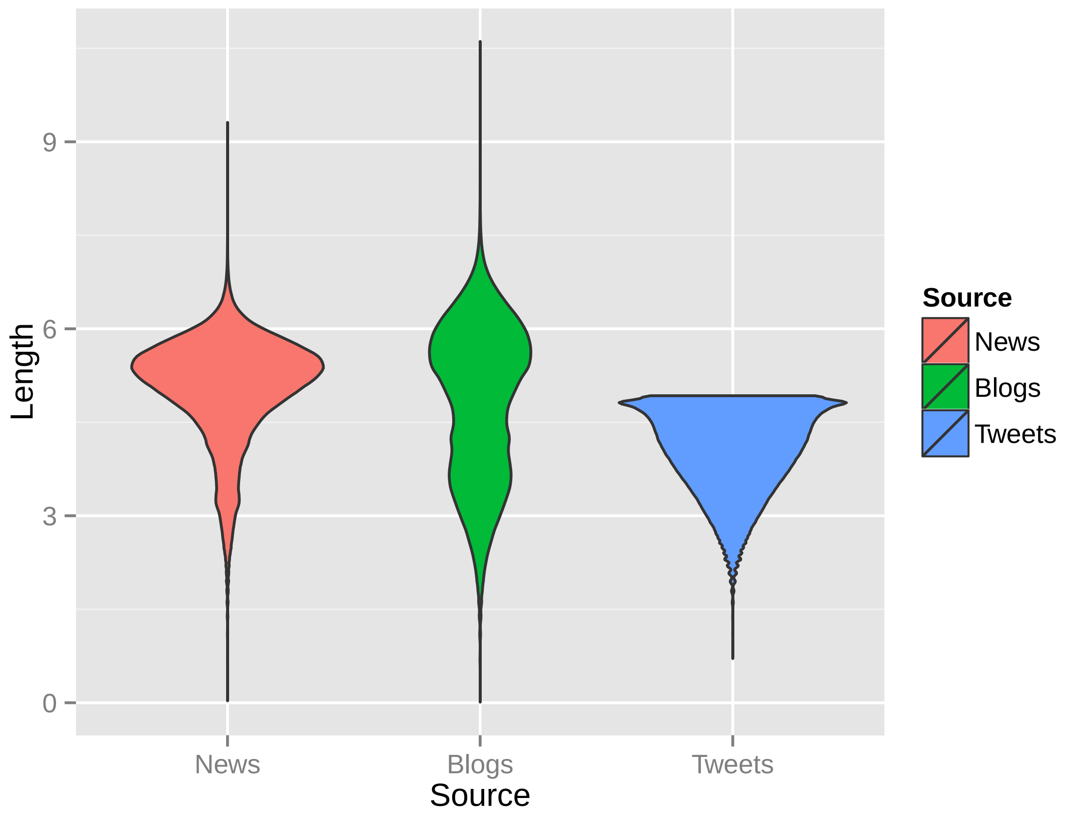
Exploratory Data Analysis

Characters

The US corpus consists of 1,010,242 lines of news excerpts, 899,288 lines from blogs and 2,360,148 tweets. Let's break these lines down to the characters themselves and compare the text sources. Examining the news, blog and tweet excerpts line by line, we get a sense for their lengths by counting the characters:

Source	Min	Median	Mean	Max
News	1	185	201.2	11,380
Blogs	1	156	230	40,830
Tweets	2	64	68.7	140

Not surprisingly, the length of the blog and news items are much longer than the tweets (three to four times as long, on average) and the longest tweets are only 140 characters, versus the tens of thousands of characters in the longest blog and news excerpts. This is evident in the violin plot below, which shows the log of the lengths of the sources on the y-axis and the distribution of the lengths illustrated by their shapes. (Given the huge range of line lengths, this plot is of the log of the lengths to help in visualizing the spreads.)



Note how much more varied blogs and news item lengths are than tweet lengths, which cluster close to their maximum of 140. Also, the blog lines have much greater variance in length than news items.

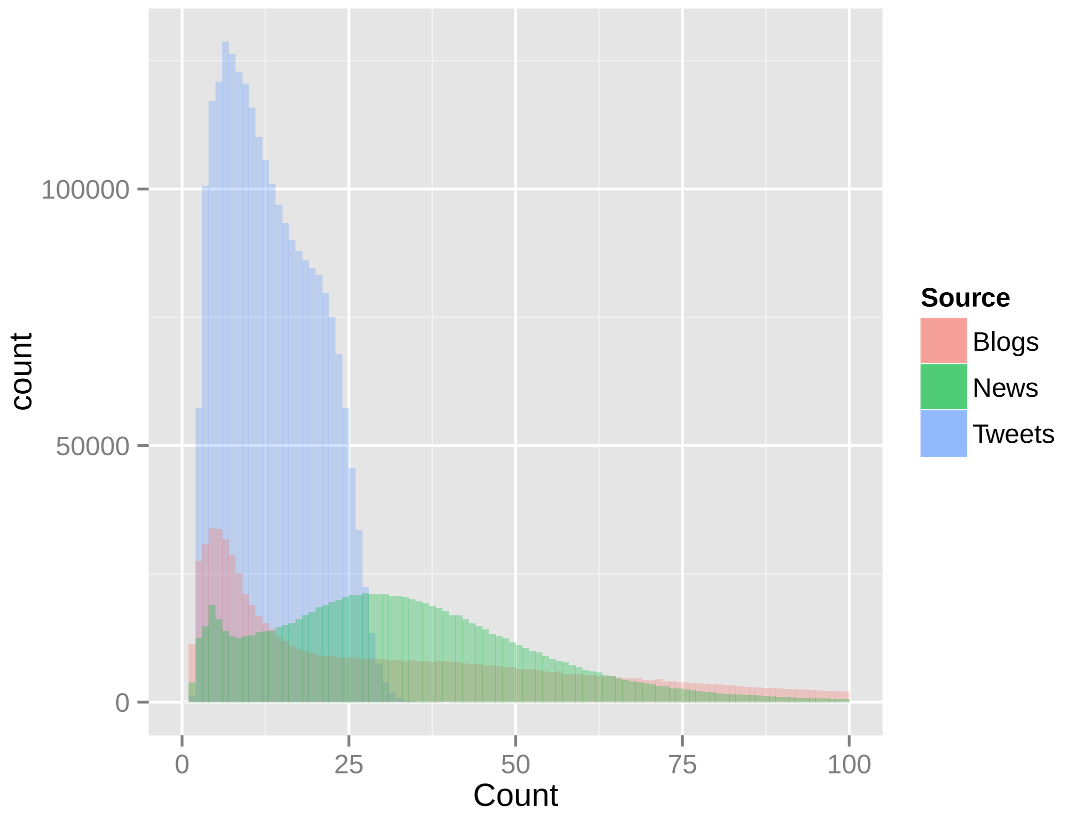
Words

Now let's look at the word counts per line per raw, unprocessed text source:

Source	Min	Median	Mean	Max
News	1	31	34	1,792
Blogs	1	28	41.5	6,630
Tweets	1	12	12.9	47

These words counts will be different after processing the data (e.g., removing profanities), but the counts above help us see again how much longer the news items are than tweets, and how blog excerpts are even longer. (Interesting trivia: the tweet with the most words in the corpus is the word 'hi' repeated 47 times. Not very useful for our analysis, but perhaps indicative of the overall usefulness of tweets for training a model.)

The histogram below shows the distribution of word counts per line between the three sources. (The plot is truncated at a count of 100 to help compare word count distributions since the word counts are clustered at the lower end of the scale.)



The number of words per tweet is on average much lower than in news and blogs, with the vast majority of tweets consisting of fewer than 25 words. Blog and news item per-line word counts are much more varied, with both having long tails with hundreds and thousands of words per line.

Textual Analysis

Next we want to get a sense of the data itself. How clean is the text? How many extraneous words, numbers, emoji and punctuation marks will we have to take care of? Starting with news and picking five at random:

```
## [[1]]
## [1] Chicago will turn Tuesday to Philip Humber, who went 9-9 with a 3.75 ERA in 28 a
## [2] ppearances for the White Sox last season. While he faded late in his first full
## [3] season as a starter, he had a strong spring training this year, giving up seven
## [4] runs and 10 hits in 17 innings.
##
## [[2]]
## [1] Coolio, best remembered for his 1995 No. 1 hit "Gangsta's Paradise," originally
## [2] was also on the bill. The rapper, whose real name is Artis Leon Ivey, dropped of
## [3] f the show after he was arrested last month in Los Angeles and charged with batt
## [4] ery and possession of crack cocaine.
##
## [[3]]
## [1] The Ohio Girls Golf Foundation will host "Diamonds in the Rough" July 17-18 at A
## [2] von Oaks C.C. in Avon. Select girls will play on July 17 in a nine-hole scramble
## [3] , followed by a scholarship reception. All golfers will participate July 18 in a
## [4] n 18-hole tournament with prizes awarded for best team and top individuals. Regi
## [5] ster by July 1. All female high school players, including seniors who have just
## [6] graduated and incoming freshmen, are eligible to participate. For more informati
## [7] on, visit www.oggf.org to download a registration form or contact Judd Stephenso
## [8] n at juddstephenson@yahoo.com, or 440-871-4638.
##
## [[4]]
## [1] The authority voted today for these four cost-cutting measures, all of which wer
## [2] e in the works before the audit was released:
##
## [[5]]
## [1] One of our outfielders on the Legacy Team was Gary Matthews Jr. His dad played o
## [2] n the great team of 1984, with Sandberg, Rick Sutcliffe and Keith Moreland.
```

We see in these five examples that the news items are typically full sentences, although not exclusively. Items vary from sports reports to local news to entertainment. Accordingly, there are many proper nouns ('Chicago', 'Coolio'), numbers ('9-9 with a 3.75', '1995'), topic-specific items ('ERA') and even email addresses that will not be useful in our predictive text app. On the plus side, the text tends to adhere to standard grammatical rules and will therefore be relatively easy to parse and useful for our modeling.

```
## [[1]]
## [1] sea salt and cracked pepper
##
## [[2]]
## [1] I believe this is an experience that every single study abroad student has on th
## [2] e first Friday that they spend in Morocco:
##
## [[3]]
## [1] But back to the on-point-former-sewing-machine-cover. I ended up needing to trim
## [2]  it quite a bit and add borders in order to square it up, but I think I like it
## [3] even more bordered than without. I think it will eventually become a mini wall-h
## [4] anging, probably to be hung near my sewing machine.
##
## [[4]]
## [1] They were.
##
## [[5]]
## [1] Even though there’s no risk of Montezuma’s revenge, Fransisco still drowns the d
## [2] ish in lime juice, a tradition in Mexico to keep bad bacteria at bay.
```

The five blog examples are much less structured than the news items, with grammatical errors, misspellings and incomplete sentences. Topics are all over the place and there is a lack of comment delineators. Punctuation is also much more liberally used ('*hug!!!!!!*').

```
## [[1]]
## [1] Old people don't deserve a sportscar
##
## [[2]]
## [1] "Guilt is the source of sorrows, the avenging fiend that follows us behind with
## [2] whips and stings." - Nicholas Rowe The past is past - grow!
##
## [[3]]
## [1] Thanks! I found that personification amusing. As if..! Not on libprof AFAIK.
##
## [[4]]
## [1] You're welcome Oppa! ^^
##
## [[5]]
## [1] Try our SIMPLE & FIT 2-Egg Breakfast Scrambled egg substitute, two strips of tur
## [2] key bacon, whole wheat toast and fresh fruit.350 Calories
```

If the news items are the most structured and parseable and blog excerpts slightly less so, the tweets reside at the opposite end of the spectrum with incomplete sentences, grammatical errors, misspellings (intentional and otherwise), abbreviations, and inscrutable references. Punctuation is used liberally, but not to standard, complicating the relationships between terms. The tweets will have less usefulness in building our predictive models.

Word Frequency

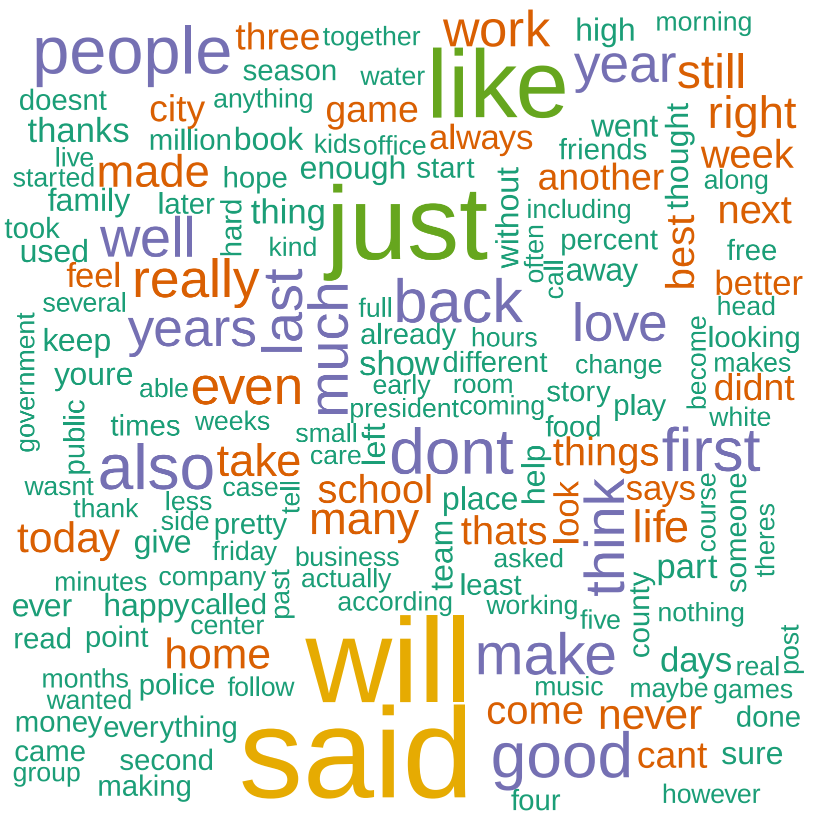
Given the clutter described above, we first do some general cleaning before analyzing further. We don't want to affect any relationships between the words, so we focus on converting to lowercase, removing punctuation and numbers, and stripping extra spaces. We'll also take this opportunity to remove profanity. After this, we'll create a document-term matrix, which will allow us to see which words occur most frequently across all three sources. We will also analyze with and without common words (e.g., "and") for this analysis, but we'll leave them in when we build our model since those frequently are the correct words to predict. Finally, we are using 20,000 lines from each of the sources in the corpus which let's us infer word frequency and relationships while reducing computational time.

After doing the above, we find the 10 most-frequent words across all sources to be: the, and, that, for, you, with, was, this, have, but

If we strip out common words (e.g., "the", "can"), the 10 most-frequent words are: said, will, just, like, time, people, know, also, good, dont

It's often just as useful to look at the least-frequent words as it is the most. In this case, we find at the bottom some Chinese characters that will have to be removed before building our model: .

Finally, just for fun, we plot a wordcloud of the 200 most-frequently appearing words in our 60,000-line sample:



Next Steps

Even with the pre-processing of the text performed above, we saw that further cleaning will be necessary before trying to build an algorithm. The next steps will be:

1. Remove foreign characters and words, as well as fixing the contractions that are now incorrect because of punctuation removal. This includes hashtags, email addresses and '@' names.
2. I won't use stemming (the reduction of words to their roots) because the ultimate goal is to predict the word you need as you type, so the full word makes all the difference. For example, 'operative' and 'operational' are not interchangeable, despite their shared root of 'oper.'
3. I will experiment with lemmatization, which is similar to stemming but takes grammatical context into account.
4. More.

Appendix

```r
local <- TRUE
ifelse(local, setwd("~/Documents/DataScience/Capstone"), setwd("./r-projects/Capstone"))
knitr::opts_chunk$set(fig.width=6, fig.align='center')
str_break = function(x, width = 80L) {
  n = nchar(x)
  if (n <= width) return(x)
  n1 = seq(1L, n, by = width)
  n2 = seq(width, n, by = width)
  if (n %% width != 0) n2 = c(n2, n)
  substring(x, n1, n2)
}
options(scipen=999)
ensurePkg <- function(x) {
    if (!require(x,character.only = TRUE)) {
        install.packages(x,dep=TRUE, repos="http://cran.r-project.org")
        if(!require(x,character.only = TRUE)) stop("Package not found")
    }
}

ensurePkg("tm")
ensurePkg("dplyr")
ensurePkg("ggplot2")
ensurePkg("scales")
ensurePkg("wordcloud")
ensurePkg("stringi")
ensurePkg("slam")
```

```r
USnews <- readLines("corpus/final/en_US/en_US.news.txt", skipNul = TRUE)
USblogs <- readLines("corpus/final/en_US/en_US.blogs.txt", skipNul = TRUE)
UStweets <- readLines("corpus/final/en_US/en_US.twitter.txt", skipNul = TRUE)

newsChars <- nchar(USnews)
blogsChars <- nchar(USblogs)
tweetsChars <- nchar(UStweets)
newsSum <- summary(newsChars)
blogsSum <- summary(blogsChars)
tweetsSum <- summary(tweetsChars)
```

```r
a <- data.frame(Source = "News", Length = log(newsChars))
b <- data.frame(Source = "Blogs", Length = log(blogsChars))
c <- data.frame(Source = "Tweets", Length = log(tweetsChars))
plot.data <- rbind(a, b, c)
g <- ggplot(plot.data, aes(x = Source, y = Length, fill = Source))
g + geom_violin()
```

```r
newsWordsPre <- stri_count(USnews, regex="\\S+")
blogsWordsPre <- stri_count(USblogs, regex="\\S+")
tweetsWordsPre <- stri_count(UStweets, regex="\\S+")
newsWordsSum <- summary(newsWordsPre)
blogsWordsSum <- summary(blogsWordsPre)
```

```r
tweetsWordsSum <- summary(tweetsWordsPre)
```

```r
a <- data.frame(Source = "News", Count = newsWordsPre)
b <- data.frame(Source = "Blogs", Count = blogsWordsPre)
c <- data.frame(Source = "Tweets", Count = tweetsWordsPre)
plot.data <- rbind(a, b, c)
plot.data <- filter(plot.data, Count < 100)
g <- ggplot(plot.data, aes(Count, fill = Source))
g + geom_histogram(data = subset(plot.data, Source == "Tweets"), alpha = 0.3, binwidth = 1) +
    geom_histogram(data = subset(plot.data, Source == "Blogs"), alpha = 0.3, binwidth = 1) +
    geom_histogram(data = subset(plot.data, Source == "News"), alpha = 0.3, binwidth = 1)
```

```r
set.seed(42)
print(lapply(USnews[sample(length(USnews),5)], str_break), quote = F)
```

```r
set.seed(42)
print(lapply(USblogs[sample(length(USblogs),5)], str_break), quote = F)
```

```r
set.seed(42)
print(lapply(UStweets[sample(length(UStweets),5)], str_break), quote = F)
```

```r
# Take random 20,000-line samples from each source and then combine into a corpus
set.seed(42)
n <- USnews[sample(length(USnews), 20000)]
set.seed(42)
b <- USblogs[sample(length(USblogs), 20000)]
set.seed(42)
t <- UStweets[sample(length(UStweets), 20000)]
corpus <- Corpus(VectorSource(c(n,b,t)))
# From Shutterstock list at https://github.com/shutterstock/List-of-Dirty-Naughty-Obscene-and-Otherwise
curseDict <- read.csv("profanities.csv", stringsAsFactor=F)

ensurePkg("doParallel")
corecount <- detectCores()
cl <- makeCluster(corecount * 0.75)
registerDoParallel(cl)
# getDoParWorkers()

corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, as.matrix(curseDict))
dtm <- DocumentTermMatrix(corpus, control = list(minWordLength = 2))

corpus2 <- tm_map(corpus, removeWords, stopwords("english"))
dtm2 <- DocumentTermMatrix(corpus2, control = list(wordLengths = c(4, Inf)))

# Convert to a normal matrix from the sparse matrix
# Package 'slam' needed for larger dtm's
freq <- slam::col_sums(dtm, na.rm = T)
freq2 <- slam::col_sums(dtm2, na.rm = T)
```

```r
# Column sums of the matrix to get a named vector
freq <- sort(freq, decreasing=TRUE)
freq2 <- sort(freq2, decreasing=TRUE)

stopCluster(cl)
```

```r
# Plot a word cloud with top 200
words <- names(freq2)
wordcloud(words[1:200], freq2[1:200], colors=brewer.pal(6, "Dark2"))
```