Text Prediction With R

This paper explores text mining with R. Messages from Twitter, news and blogs are acquired and analyzed using R's TM package. The final objective is to generate n-gram probability models for the purpose of text prediction.

Acquisition

Three data sets (news, blogs, twitter) were provided. Command line operations (e.g., cat en_US.twitter.txt | wc -l) were used to quickly explore basic dimensions of the data sets. The twitter data set contains 2,360,148 lines and 30,374,206 words. The news data set contains 10,10,242 lines and 34,372,720 words. The blogs data set contains 899,288 lines and 37,334,690.

For illustration purposes, this milestone report is limited to Twitter data. That being said, the basic methodology detailed in the following section can be easily applied towards blogs and news if necessary.

Transformation

A random subset (representing 1% of total twitter data set) was used for this preliminary analysis.

The subsetted Twitter corpus was transformed with the tm package. This text file was converted into a corpus object, which treats as a separate entity or record. The transformation process removed stopwords, numbers and punctuation from tweets.

This transformed corpus was then converted into a bigram term document matrix. (It is entirely possible to create a tri-gram, or any n-gram, TDM.) A TDM is a 2D matrix where rows represent n-grams and columns represent messages. Each TDM cell represents a frequency count at the corresponding n-gram and message.

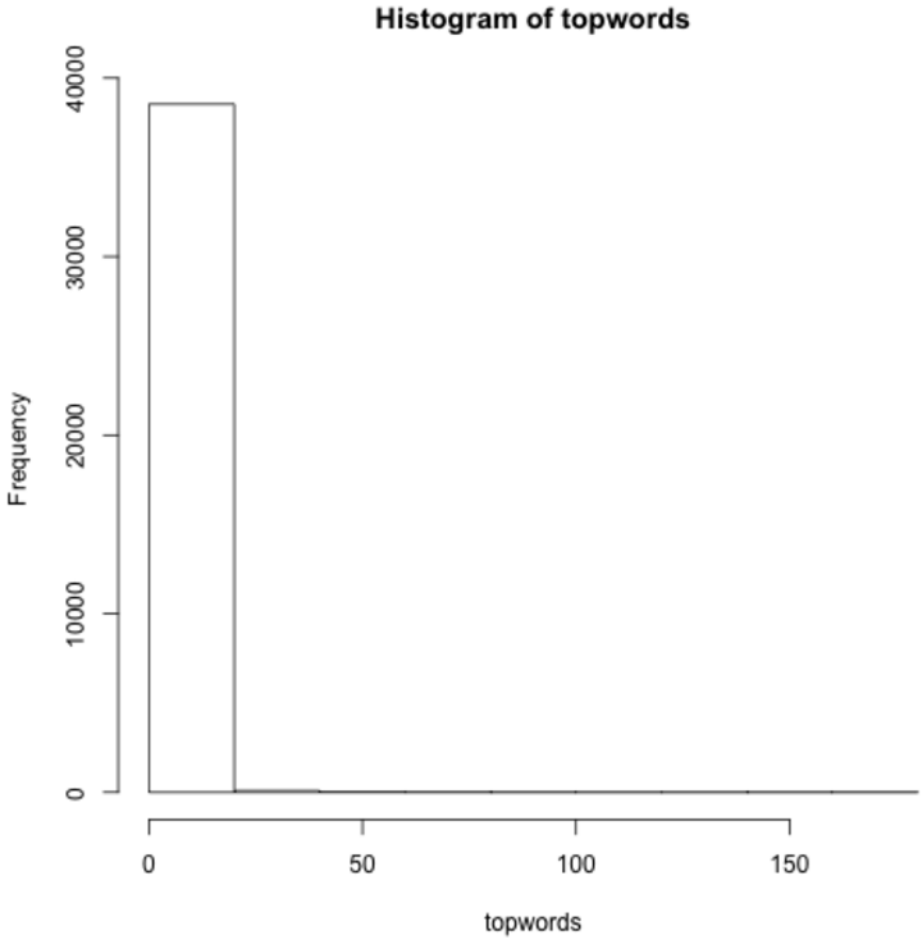
```
library(tm)
library(RWeka)
setwd("~/Dropbox/Development/Scripts/swiftcapstone")
data <- read.csv("tweetsdataset.csv", header=FALSE,
stringsAsFactors=FALSE)
data <- data[,2]
source("GenerateTDM.R") # generatetdm function in appendix
tdm <- tdm.generate(data, 2)
```

The resulting 14MB term document matrix captures 144,677 unique bigrams across 23,602 tweets. It is a highly sparse matrix, with only 267,529 entries across 3.4 trillion elements.

Exploration

It is clear that the distribution of ngrams is heavily skewed towards a handful of high frequency words.

```
tdm.matrix <- as.matrix(tdm)
topwords <- rowSums(tdm.matrix)
topwords <- as.numeric(topwords)
hist(topwords, breaks = 10)
```



Bigrams appearing more than 50 times and 100 times in the corpus were identified.

```
findFreqTerms(tdm, lowfreq = 50)
```

```
##  [1] "a great"    "at the"     "for a"      "for the"    "going
to"
##  [6] "have a"     "i am"       "i dont"     "i have"     "i
love"
## [11] "i was"      "if you"     "in the"     "it was"     "of
the"
## [16] "on the"     "thank you"  "thanks for" "to be"      "to
see"
## [21] "to the"     "want to"    "will be"    "you are"
```

```
findFreqTerms(tdm, lowfreq = 100)
```

```
## [1] "for the" "in the"  "of the"  "to be"
```

We can also identify high frequency terms.

```
tdm.matrix <- as.matrix(tdm)
topwords <- rowSums(tdm.matrix)
head(sort(topwords, decreasing = TRUE))
```

```
##     in the    for the     of the      to be     on the thanks
for
##        163        150        128        119         99
99
```

Application Design

The basic methodology for the n-gram text prediction is as follows:

1. Generate 1-gram, bigram, and trigram matrices.
2. By summing frequency counts, generate a 2-column table of unique ngrams by frequencies ("N-gram Frequency Table").
3. Match a n-word character string with the appropriate n+1 gram entry in the N-gram Frequency Table. For example, a two-word string should be matched with its corresponding entry in a tri-gram table.
4. If there is a match, propose high frequency words to the user. Continuing the previous example, a match should be the last word of the n-gram.

Appendix

```
tdm.generate <- function(string, ng){

  # tutorial on rweka - http://tm.r-forge.r-project.org/faq.html

  corpus <- Corpus(VectorSource(string)) # create corpus for TM
processing
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, stripWhitespace)
  # corpus <- tm_map(corpus, removeWords, stopwords("english"))
  options(mc.cores=1) # http://stackoverflow.com/questions/17703553
/bigrams-instead-of-single-words-in-termdocument-matrix-using-
r-and-rweka/20251039#20251039
  BigramTokenizer <- function(x) NGramTokenizer(x,
Weka_control(min = ng, max = ng)) # create n-grams
  tdm <- TermDocumentMatrix(corpus, control = list(tokenize =
BigramTokenizer)) # create tdm from n-grams
  tdm
}
```