Dictionary-Based Text Analysis in R

Table of Contents

[**Text as Data Course**](https://cbail.github.io/textasdata/Text_as_Data.html)  
**Chris Bail, PhD**  
**Duke University**  
[www.chrisbail.net](http://www.chrisbail.net)  
[github.com/cbail](https://github.com/cbail)  
[twitter.com/chris\_bail](https://twitter.com/chris_bail)

# Introduction

Among the most basic forms of quantitative text analysis are word-counting techniques and dictionary-based methods. This tutorial will cover both of these topics, as well as sentiment analysis, which is a form of dictionary-based text analysis. This tutorial assumes basic knowledge about R and other skills described in previous tutorials at the link above.

# Word Counting

In the early days of quantitative text analysis, word-frequency counting in texts was a common mode of analysis. In this section, we’ll learn a few basic techniques for counting word frequencies and visualizing them. We’re going to work within the tidytext framework, so if you need a refresher on that, see my previous tutorial entitled “Basic Text Analysis in R.”

Let’s begin by loading the Trump tweets we extracted in a previous tutorial and transform them into tidytext format:

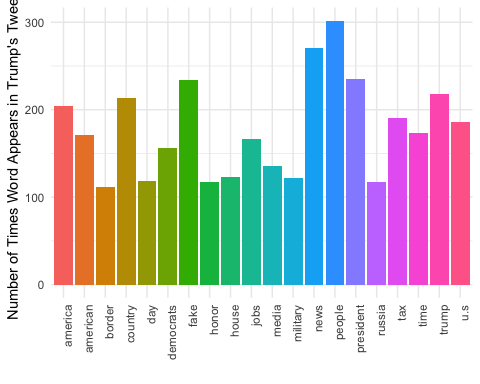
load(url("https://cbail.github.io/Trump\_Tweets.Rdata"))  
library(tidytext)  
library(dplyr)  
tidy\_trump\_tweets<- trumptweets %>%  
 select(created\_at,text) %>%  
 unnest\_tokens("word", text)

Next, let’s count the top words after removing stop words (frequent words such as “the”, and “and”) as well as other unmeaningful words (e.g. https):

data("stop\_words")  
  
trump\_tweet\_top\_words<-  
 tidy\_trump\_tweets %>%  
 anti\_join(stop\_words) %>%  
 count(word) %>%  
 arrange(desc(n))  
   
  
trump\_tweet\_top\_words<-  
 trump\_tweet\_top\_words[-grep("https|t.co|amp|rt",  
 trump\_tweet\_top\_words$word),]

Now let’s make a graph of the top 20 words

#select only top words  
top\_20<-trump\_tweet\_top\_words[1:20,]  
  
#create factor variable to sort by frequency  
trump\_tweet\_top\_words$word <- factor(trump\_tweet\_top\_words$word, levels = trump\_tweet\_top\_words$word[order(trump\_tweet\_top\_words$n,decreasing=TRUE)])  
  
  
library(ggplot2)  
ggplot(top\_20, aes(x=word, y=n, fill=word))+  
 geom\_bar(stat="identity")+  
 theme\_minimal()+  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1))+  
 ylab("Number of Times Word Appears in Trump's Tweets")+  
 xlab("")+  
 guides(fill=FALSE)



# tf-idf

Though we have already removed very common “stop words” from our analysis, it is common practice in quantitative text analysis to identify unusual words that might set one document apart from the others (this will become particularly important when we get to more advanced forms of pattern recognition in text later on). The metric most commonly used to identify these unusual words is “Term Frequency Inverse Document Frequency” (tf-idf). We can calculate the tf-idf for the Trump tweets databased in tidytext as follows:

tidy\_trump\_tfidf<- trumptweets %>%  
 select(created\_at,text) %>%  
 unnest\_tokens("word", text) %>%  
 anti\_join(stop\_words) %>%  
 count(word, created\_at) %>%  
 bind\_tf\_idf(word, created\_at, n)

Now let’s see what the most unusual words are:

top\_tfidf<-tidy\_trump\_tfidf %>%  
 arrange(desc(tf\_idf))  
  
top\_tfidf$word[1]

## [1] "standforouranthem"

The tfidf increases the more a term appears in a document but it is negatively weighted by the overall frequency of terms across all documents in the dataset or Corpus. In simpler terms, the tf-idf helps us capture which words are not only important within a given document but also distinctive vis-a-vis the broader corpus or tidytext dataset.

# Dictionary-Based Quantitative Text Analysis

Though word frequency counts and tf-idf can be an informative way to examine text-based data, another very popular techniques involves counting the number of words that appear in each document that have been assigned a particular meaning or value to the researcher. There are numerous examples that we shall discuss below— some of which are more sophisticated than others.

**Creating your own dictionary**

To begin, let’s make our own dictionary of terms we want to examine from the Trump tweet dataset. Suppose we are doing a study of economic issues, and want to subset those tweets that contain words associated with the economy. To do this, we could first create a list or “dictionary” or terms that are associated with the economy.

economic\_dictionary<-c("economy","unemployment","trade","tariffs")

Having created a very simple/primitive dictionary, we can now subset the parts of our tidytext dataframe that contain these words. We could do all of this in dplyr, but there is another very useful package for string manipulation that will be worth learning about called stringr

library(stringr)  
economic\_tweets<-trumptweets[str\_detect(trumptweets$text, economic\_dictionary),]  
head(economic\_tweets$text)

## [1] "Great talk with my friend President Mauricio Macri of Argentina this week. He is doing such a good job for Argentina. I support his vision for transforming his country’s economy and unleashing its potential!"   
## [2] "The Washington Post and CNN have typically written false stories about our trade negotiations with China. Nothing has happened with ZTE except as it pertains to the larger trade deal. Our country has been losing hundreds of billions of dollars a year with China..."   
## [3] "China and the United States are working well together on trade, but past negotiations have been so one sided in favor of China, for so many years, that it is hard for them to make a deal that benefits both countries. But be cool, it will all work out!"   
## [4] "The Fake News is working overtime. Just reported that, despite the tremendous success we are having with the economy &amp; all things else, 91% of the Network News about me is negative (Fake). Why do we work so hard in working with the media when it is corrupt? Take away credentials?"  
## [5] "Looking forward to my meeting with Tim Cook of Apple. We will be talking about many things, including how the U.S. has been treated unfairly for many years, by many countries, on trade."   
## [6] "So much Fake News about what is going on in the White House. Very calm and calculated with a big focus on open and fair trade with China, the coming North Korea meeting and, of course, the vicious gas attack in Syria. Feels great to have Bolton &amp; Larry K on board. I (we) are"

# Sentiment Analysis

The example above was somewhat arbitrary and mostly designed to introduce you to the concept of dictionary-base text analysis. The list of economic terms that I came up with was very ad hoc—and though the tweets identified above each mention the economy, there are probably many more tweets in our dataset that reference economic issues that do not include the words I identified.

Dictionary-based approaches are often most useful when a high-quality dictionary is available that is of interest to the researcher or analyst. One popular type of dictionary is a sentiment dictionary which can be used to assess the valence of a given text by searching for words that describe affect or opinion. Some of these dictionaries are created by examining comparing text-based evaluations of products in online forums to ratings systems. Others are created via systematic observation of people writing who have been primed to write about different emotions.

Let’s begin by examining some of the sentiment dictionaries that are built into tidytext. These include the afinn which includes a list of sentiment-laden words that appeared in [Twitter discussions of climate change](http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/6006/pdf/imm6006.pdf); bing which includes sentiemnt words identified on online forums; and nrc which is a dictionary that was created by having workers on Amazon mechanical Turk code the emotional valence of a [long list of terms](https://arxiv.org/pdf/1308.6297.pdf). These algorithims often produce similar results, even though they are trained on different datasets (meaning they identify sentiment laden words using different corpora). Each of these dictionaries only describe sentiment-laden words in the English language. They also have different scales. We can browse the content of each dictionary as follows:

head(get\_sentiments("afinn"))

## # A tibble: 6 x 2  
## word score  
## <chr> <int>  
## 1 abandon -2  
## 2 abandoned -2  
## 3 abandons -2  
## 4 abducted -2  
## 5 abduction -2  
## 6 abductions -2

Let’s apply the bing sentiment dictionary to our database of tweets by Trump:

trump\_tweet\_sentiment <- tidy\_trump\_tweets %>%  
 inner\_join(get\_sentiments("bing")) %>%  
 count(created\_at, sentiment)   
  
head(trump\_tweet\_sentiment)

## # A tibble: 6 x 3  
## created\_at sentiment n  
## <dttm> <chr> <int>  
## 1 2017-02-05 22:49:42 positive 2  
## 2 2017-02-06 03:36:54 positive 4  
## 3 2017-02-06 12:01:53 negative 3  
## 4 2017-02-06 12:01:53 positive 1  
## 5 2017-02-06 12:07:55 negative 2  
## 6 2017-02-06 16:32:24 negative 3

Now let’s make a visual that compares the frequency of positive and negative tweets by day. To do this, we’ll need to work a bit with the created\_at variable—more specifically, we will need to transform it into a “date” object that we can use to pull out the day during which each tweet was made:

tidy\_trump\_tweets$date<-as.Date(tidy\_trump\_tweets$created\_at,   
 format="%Y-%m-%d %x")

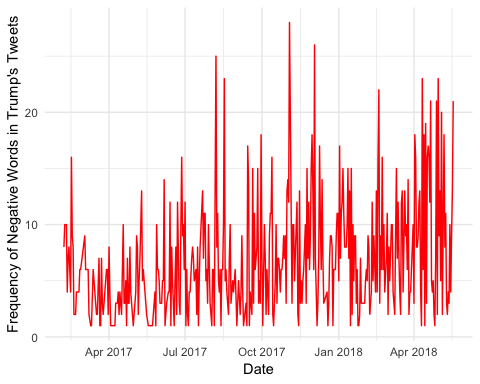
The format argument here tells R how to read in the date character string, since dates can appear in a number of different formats, time zones, etc. For more information about how to format data with other dates, see ?as.Date()

Now let’s aggregate negative sentiment by day

trump\_sentiment\_plot <-  
 tidy\_trump\_tweets %>%  
 inner\_join(get\_sentiments("bing")) %>%   
 filter(sentiment=="negative") %>%  
 count(date, sentiment)

## Joining, by = "word"

library(ggplot2)  
  
ggplot(trump\_sentiment\_plot, aes(x=date, y=n))+  
 geom\_line(color="red")+  
 theme\_minimal()+  
 ylab("Frequency of Negative Words in Trump's Tweets")+  
 xlab("Date")



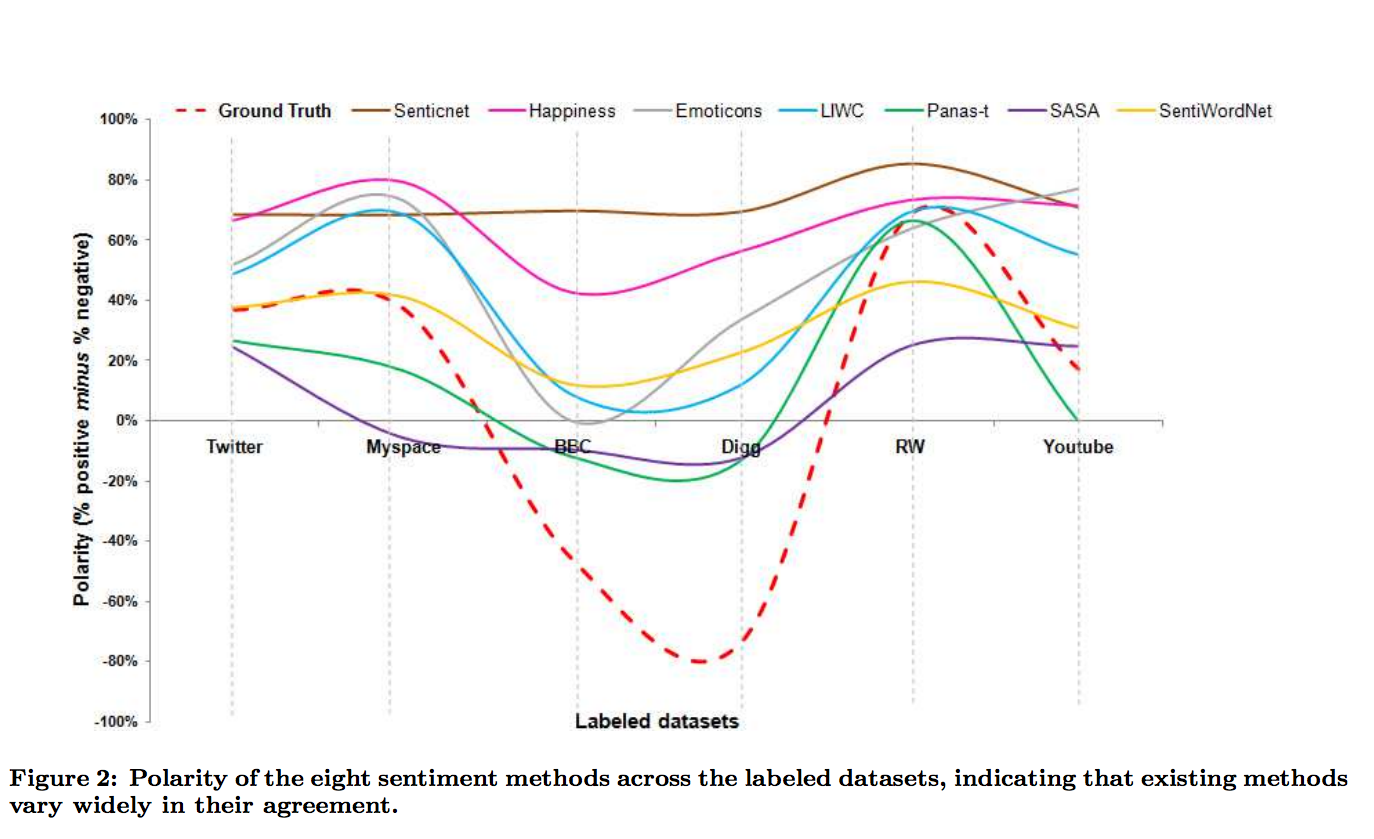
There appears to be an upward trend. Is it possible that this increase is being shaped by Trump’s approval rating? Let’s take a look, downloading data from the survey polling group 538 for the same time period as our Twitter data above:

trump\_approval<-read.csv("https://projects.fivethirtyeight.com/trump-approval-data/approval\_topline.csv")  
  
trump\_approval$date<-as.Date(trump\_approval$modeldate, format="%m/%d/%Y")  
  
approval\_plot<-  
 trump\_approval %>%  
 filter(subgroup=="Adults") %>%  
 filter(date>min(trump\_sentiment\_plot$date)) %>%   
 group\_by(date) %>%  
 summarise(approval=mean(approve\_estimate))  
  
#plot  
ggplot(approval\_plot, aes(x=date, y=approval))+  
 geom\_line(group=1)+  
 theme\_minimal()+  
 ylab("% of American Adults who Approve of Trump")+  
 xlab("Date")



Quite obviously we have some scaling issues we would need to address if we wanted to make a proper comparison, but for now, let’s move on.

There are many other types of sentiment analysis, which we do not have time to cover here. An important thing for you to know, however, is that different sentiment analysis tools work better for some corpuses than others. Here is a figure from a recent paper that applies a vareity of different sentiment dictionaries to different corpora:



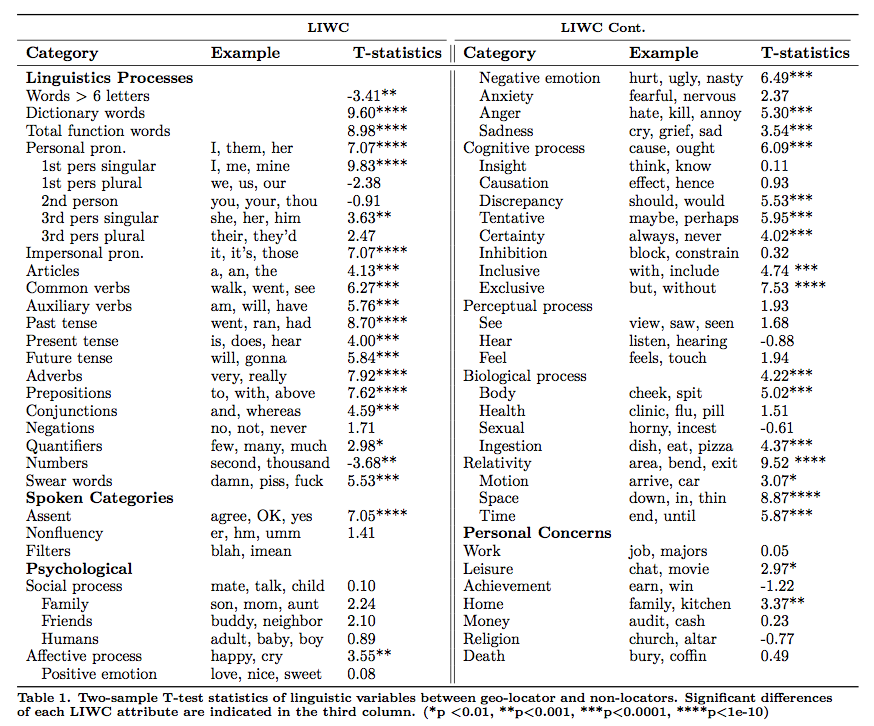
Source: <http://homepages.dcc.ufmg.br/~fabricio/download/cosn127-goncalves.pdf>

**Now you Try it**

Let’s try to build together several of the skills you’ve learned in the course thus far: 1) Pick another politician or celebrity of interest, such as Hilary Clinton; 2) Grab her 3,000 most recent tweets; 3) See if her approval rating tracks the sentiment of her tweets, or—better yet— create your own custom dictionary to track a variable or variables of interest over time.

**Linguistic Inquiry Word Count (LIWC)**

Before I wrap up, I just want to highlight another class of dictionary-based approaches: those that attempt to classify not only sentiment but a range of different types of psychometric properties and substantive properties of a text. A popular example is Linguistic Inquiry Word Count, which was developed by the social psychologist James Penebaker. As the figure below shows, LIWC is a large dictionary that classifies words into dozens of categories:



Unlike some of the other approaches above, LIWC was built in a very systematic fashion— through both observation of natural language use in a variety of settings as well as empirical observation of people who were primed to write about different subjetions. For these reasons, it has become one of the more popular dictionary-based approaches in the past decade. At the same time, it—like all dictionary approaches—is ultimately limited insofar as it assumes that each word has an intrinisic meaning. As we will soon see, a more useful assumption is often that words assume different meanings based upon their apperance alongside other words.

# When Should I use a Dictionary-Based Approach?

The quality of dictionary-based methods depends heavily upon the match between the learning-corpus and the one you want to code. Creating your own is often a good solution, but it is very time intensive. On the other hand, as we will see in future tutorials, dictionary-based approaches often perform better than more sophisticated techniques such as topic modeling, depending upon the task at hand.