Dictionary-Based Text Analysis in R

- Introduction
- Word Counting
- tf-idf
- Dictionary-Based Quantitative Text Analysis
- Sentiment Analysis
- When Should I use a Dictionary-Based Approach?

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):

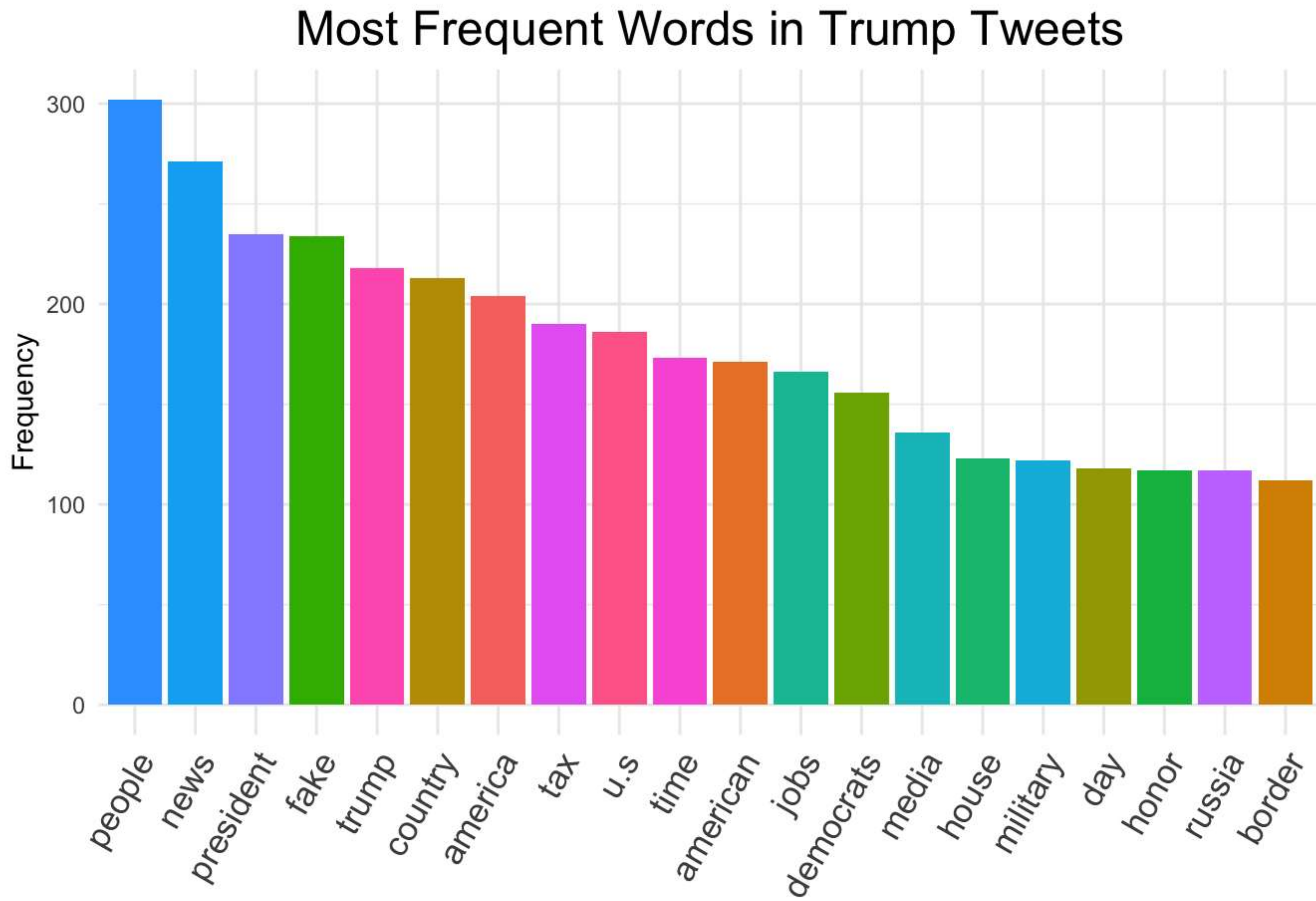
```
data("stop_words")

top_words<-
   tidy_trump_tweets %>%
      anti_join(stop_words) %>%
        filter(!(word=="https"|
                 word=="rt"|
                 word=="t.co"|
                 word=="amp")) %>%
            count(word) %>%
              arrange(desc(n))
```

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

```
library(ggplot2)
```

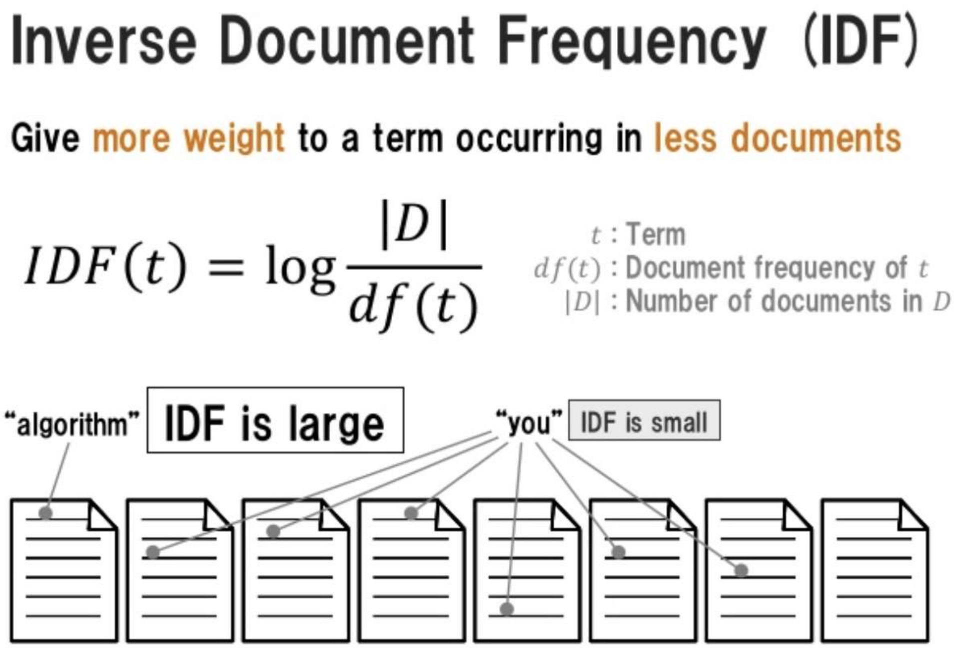
```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
top_words %>%
  slice(1:20) %>%
    ggplot(aes(x=reorder(word, -n), y=n, fill=word))+
      geom_bar(stat="identity")+
        theme_minimal()+
        theme(axis.text.x =
            element_text(angle = 60, hjust = 1, size=13))+
        theme(plot.title =
            element_text(hjust = 0.5, size=18))+
          ylab("Frequency")+
          xlab("")+
          ggtitle("Most Frequent Words in Trump Tweets")+
          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). As the figure below shows, the metric most commonly used to identify this type of 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 using the `str_detect` function within Hadley Wickham's `stringr` package:

```
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 3.5.2
```

```
economic_tweets<-trumptweets[str_detect(trumptweets$text, paste(economic_dictionary, collapse=
"|")),]
```

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 can produce very different results since they were created using very 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:

```
library(tidytext)
head(get_sentiments("bing"))
```

```
## # A tibble: 6 x 2
##   word       sentiment
##   <chr>      <chr>
## 1 2-faces    negative
## 2 abnormal   negative
## 3 abolish    negative
## 4 abominable negative
## 5 abominably negative
## 6 abominate  negative
```

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()`

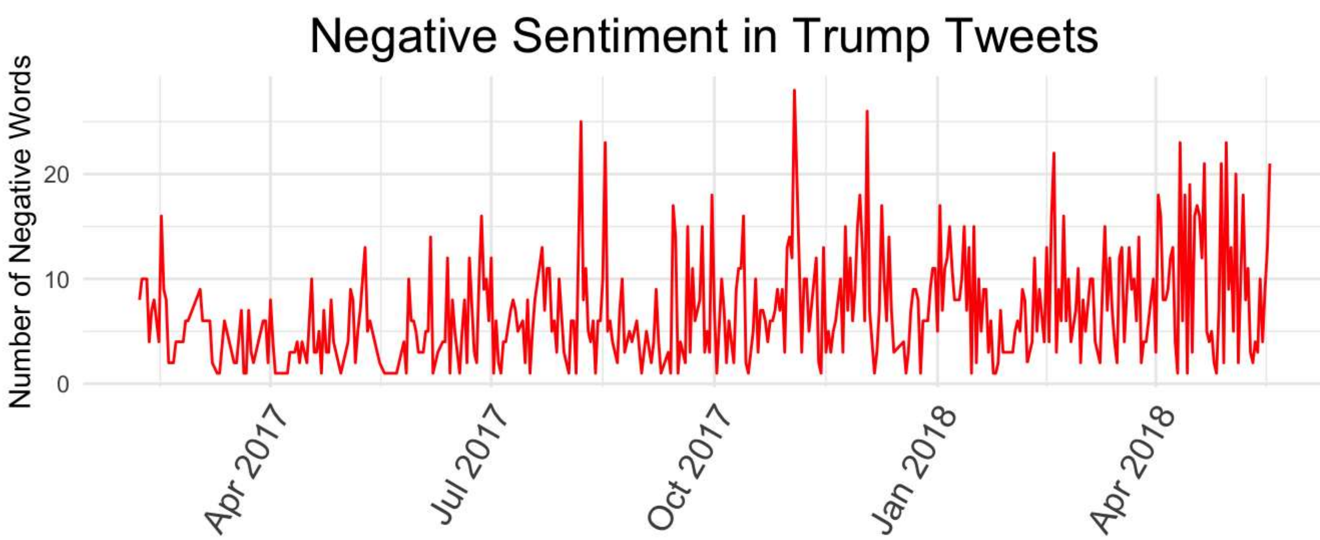
Now let's aggregate negative sentiment by day

```
trump_sentiment_plot <-
  tidy_trump_tweets %>%
    inner_join(get_sentiments("bing")) %>%
      filter(sentiment=="negative") %>%
          count(date, sentiment)
```

```
## Joining, by = "word"
```

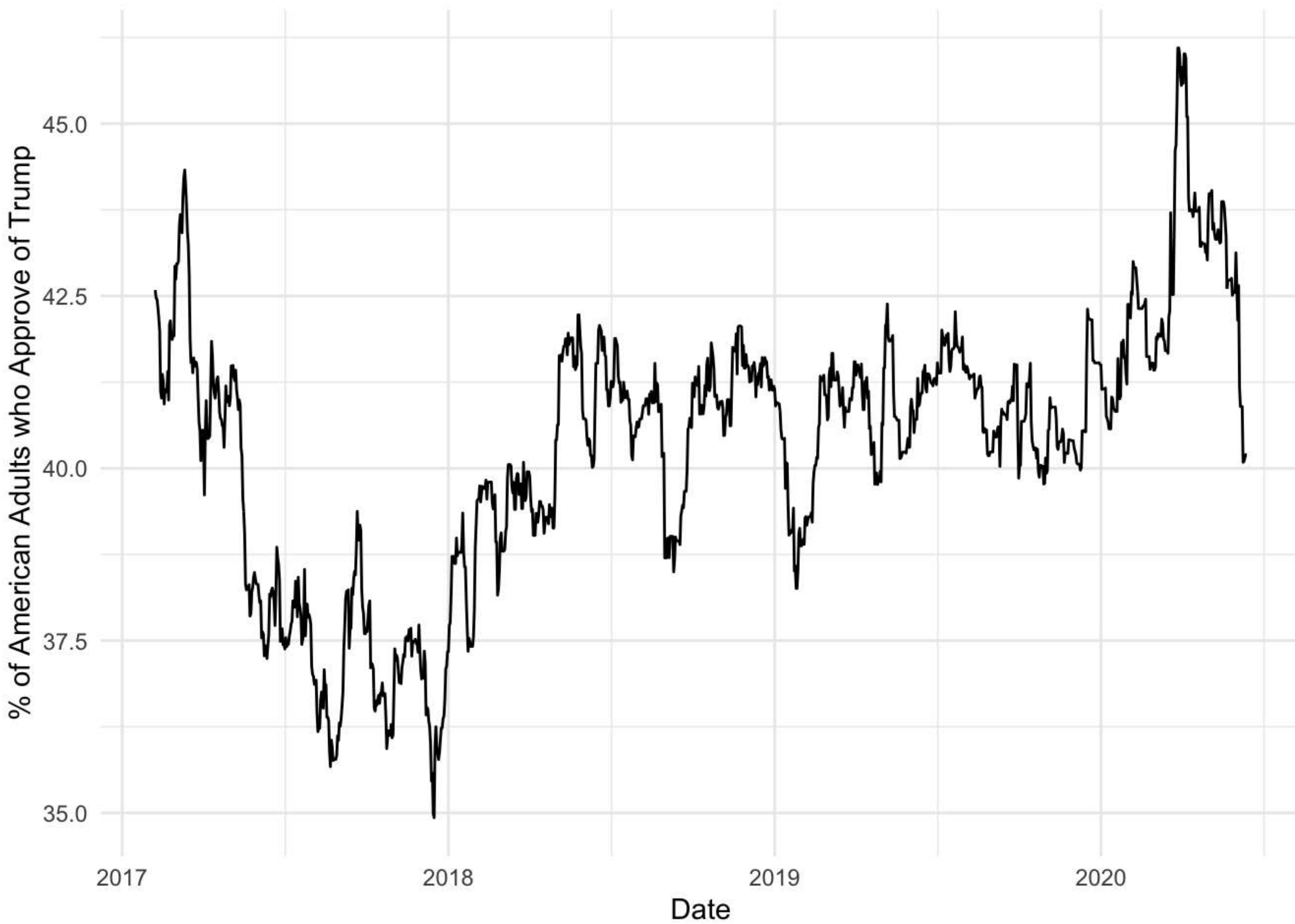
Now, let's plot it:

```
ggplot(trump_sentiment_plot, aes(x=date, y=n))+
  geom_line(color="red", size=.5)+
    theme_minimal()+
    theme(axis.text.x =
            element_text(angle = 60, hjust = 1, size=13))+
    theme(plot.title =
            element_text(hjust = 0.5, size=18))+
      ylab("Number of Negative Words")+
      xlab("")+
      ggtitle("Negative Sentiment in Trump Tweets")+
      theme(aspect.ratio=1/4)
```



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_topl
ine.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 (http://homepages.dcc.ufmg.br/~fabricio/download/cosn127-goncalves.pdf) that applies a variety of different sentiment dictionaries to different corpora:

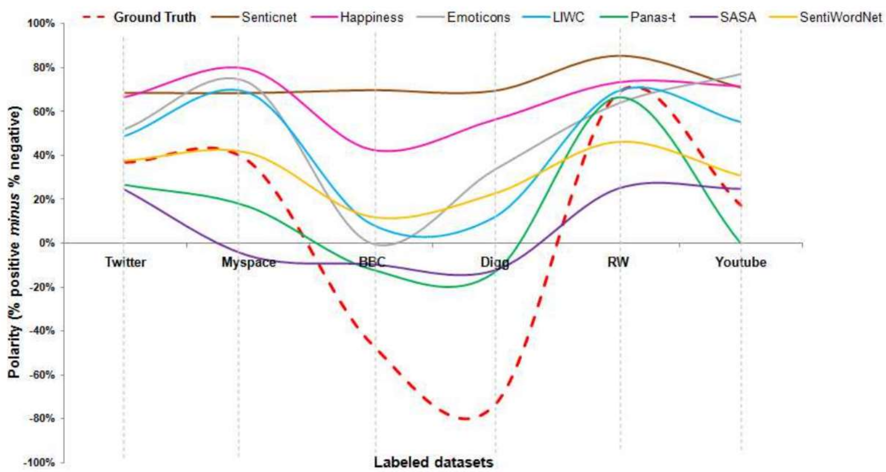


Figure 2: Polarity of the eight sentiment methods across the labeled datasets, indicating that existing methods vary widely in their agreement.

This paper (https://dl.acm.org/doi/abs/10.1145/2512938.2512951) also gives some comparative perspective on how different sentiment analysis tools perform on a database of tweets about different issues.

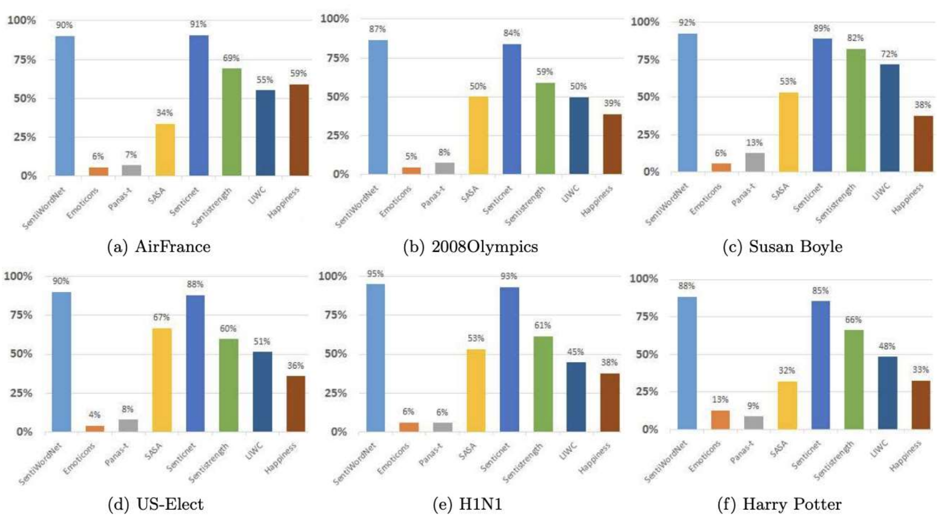


Figure 1: Coverage of six events.

Finally, other papers attempt to rank different sentiment classifiers. These analyses are helpful, but I think the most appropriate thing for you to consider when you are trying to choose which type of sentiment analysis to use is how the tool was created, and for what purpose. The tools that perform best will probably be those that were created for purposes that are most similar to your own desired purpose.

Table 8 Mean rank table for all datasets

3-classes			2-classes			
Pos	Method	Mean Rank	Pos	Method	Mean Rank	Coverage (%)
1	VADER	4.00 (4.17)	1	SentiStrength	2.33 (3.00)	29.30 (28.91)
2	LIWC15	4.62	2	Sentiment140	3.44	39.29
3	AFINN	4.69	3	Semantria	4.61	62.34
4	Opinion Lexicon	5.00	4	Opinion Lexicon	6.72	69.50
5	Semantria	5.31	5	LIWC15	7.33	68.28
6	Umigon	5.77	6	SO-CAL	7.61	72.64
7	SO-CAL	7.23	7	AFINN	8.11	73.05
8	Pattern.en	9.92	8	VADER	9.17 (9.79)	82.20 (83.18)
9	Sentiment140	10.92	9	Umigon	9.39	64.11
10	Emolex	11.38	10	PANAS-t	10.17	5.10
11	Opinion Finder	13.08	11	Emoticons	10.39	10.69
12	SentiWordNet	13.38	12	Pattern.en	12.61	65.02
13	Sentiment140_L	13.54	13	SenticNet	13.61	84.00
14	SenticNet	13.62	14	Emolex	14.50	66.12
15	SentiStrength	13.69 (13.71)	15	Opinion Finder	14.72	46.63
16	SASA	14.77	16	USent	14.89	44.00
17	Stanford DM	15.85	17	Sentiment140_L	14.94	93.36
18	USent	15.92	18	NRC Hashtag	17.17	93.52
19	NRC Hashtag	16.31	19	Stanford DM	17.39	87.32
20	LIWC	16.46	20	SentiWordNet	17.50	61.77
21	ANEW_SUB	18.54	21	SASA	18.94	60.12
22	Emoticons	21.00	22	LIWC	19.67	61.82
23	PANAS-t	21.77	23	ANEW_SUB	21.17	94.20
24	Emoticons DS	23.23	24	Emoticons DS	23.61	99.36

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’s Twitter account; 3) See if the politician’s approval rating tracks the sentiment of their 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:

LIWC			LIWC Cont.		
Category	**Example**	**T-statistics**	**Category**	**Example**	**T-statistics**
Linguistics Processes			Negative emotion	hurt, ugly, nasty	6.49***
Words > 6 letters		-3.41**	Anxiety	fearful, nervous	2.37
Dictionary words		9.60****	Anger	hate, kill, annoy	5.30***
Total function words		8.98****	Sadness	cry, grief, sad	3.54***
Personal pron.	I, them, her	7.07****	Cognitive process	cause, ought	6.09***
1st pers singular	I, me, mine	9.83****	Insight	think, know	0.11
1st pers plural	we, us, our	-2.38	Causation	effect, hence	0.93
2nd person	you, your, thou	-0.91	Discrepancy	should, would	5.53***
3rd pers singular	she, her, him	3.63**	Tentative	maybe, perhaps	5.95***
3rd pers plural	their, they'd	2.47	Certainty	always, never	4.02***
Impersonal pron.	it, it's, those	7.07****	Inhibition	block, constrain	0.32
Articles	a, an, the	4.13***	Inclusive	with, include	4.74 ***
Common verbs	walk, went, see	6.27***	Exclusive	but, without	7.53 ****
Auxiliary verbs	am, will, have	5.76***	Perceptual process		1.93
Past tense	went, ran, had	8.70****	See	view, saw, seen	1.68
Present tense	is, does, hear	4.00***	Hear	listen, hearing	-0.88
Future tense	will, gonna	5.84***	Feel	feels, touch	1.94
Adverbs	very, really	7.92****	Biological process		4.22***
Prepositions	to, with, above	7.62****	Body	cheek, spit	5.02***
Conjunctions	and, whereas	4.59***	Health	clinic, flu, pill	1.51
Negations	no, not, never	1.71	Sexual	horny, incest	-0.61
Quantifiers	few, many, much	2.98*	Ingestion	dish, eat, pizza	4.37***
Numbers	second, thousand	-3.68**	Relativity	area, bend, exit	9.52 ****
Swear words	damn, piss, fuck	5.53***	Motion	arrive, car	3.07*
Spoken Categories			Space	down, in, thin	8.87****
Assent	agree, OK, yes	7.05****	Time	end, until	5.87***
Nonfluency	er, hm, umm	1.41	**Personal Concerns**		
Filters	blah, imean		Work	job, majors	0.05
Psychological			Leisure	chat, movie	2.97*
Social process	mate, talk, child	0.10	Achievement	earn, win	-1.22
Family	son, mom, aunt	2.24	Home	family, kitchen	3.37**
Friends	buddy, neighbor	2.10	Money	audit, cash	0.23
Humans	adult, baby, boy	0.89	Religion	church, altar	-0.77
Affective process	happy, cry	3.55**	Death	bury, coffin	0.49
Positive emotion	love, nice, sweet	0.08			

Table 1. Two-sample T-test statistics of linguistic variables between geo-locator and non-locators. Significant differences of each LIWC attribute are indicated in the third column. (*p <0.01, **p<0.001, ***p<0.0001, ****p<1e-10)

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.