Like a lot of people, It was intrigued by [“I Am Part of the Resistance Inside the Trump Administration”](https://www.nytimes.com/2018/09/05/opinion/trump-white-house-anonymous-resistance.html), an anonymous New York Times op-ed written by a “senior official in the Trump administration”. And like many data scientists, I was curious about what role text mining could play.

Ok NLP people, now’s your chance to shine. Just spitballing here but TF-IDF on “the op-ed” compared to the published writing of every senior Trump admin official? I want likelihood estimates with standard errors. GO!

Since my goal is R education more than it is political analysis, I show all the code in the post.

**Downloading data**

Getting the text of the op-ed is doable with the rvest package.

# setup

library(tidyverse)

library(tidytext)

library(rvest)

theme\_set(theme\_light())

url <- "https://www.nytimes.com/2018/09/05/opinion/trump-white-house-anonymous-resistance.html"

# tail(-1) removes the first paragraph, which is an editorial header

op\_ed <- read\_html(url) %>%

html\_nodes(".e2kc3sl0") %>%

html\_text() %>%

tail(-1) %>%

data\_frame(text = .)

The harder step is getting a set of documents representing “senior officials”. An imperfect but fast approach is to collect text from their Twitter accounts. (If you find an interesting dataset of, say, government FOIA documents, I recommend you try extending this analysis!)

We can look at a combination of two (overlapping) Twitter lists containing administration staff members:

* [CSPAN’s list of Cabinet accounts](https://twitter.com/cspan/lists/the-cabinet)
* [digiphile’s list of White House staff](https://twitter.com/cspan/lists/digiphile)

library(rtweet)

cabinet\_accounts <- lists\_members(owner\_user = "CSPAN", slug = "the-cabinet")

staff <- lists\_members(slug = "white-house-staff", owner = "digiphile")

# Find unique screen names from either account

accounts <- unique(c(cabinet\_accounts$screen\_name, staff$screen\_name))

# Download ~3200 from each account

tweets <- map\_df(accounts, get\_timeline, n = 3200)

This results in a set of 136,501 from 69 Twitter handles. There’s certainly no guarantee that the op-ed writer is among these Twitter accounts (or, if they are, that they even write their tweets themselves). But it still serves as an interesting case study of text analysis. How do we find the tweets with the closest use of language?

**Tokenizing tweets**

First, we need to tokenize the tweets: to turn them from full messages into individual words. We probably want to avoid retweets, and we need to use a custom regular expression for splitting it and remove links

# When multiple tweets across accounts are identical (common in government

# accounts), use distinct() to keep only the earliest

reg <- "([^A-Za-z\\d#@']|'(?![A-Za-z\\d#@]))"

tweet\_words <- tweets %>%

filter(!is\_retweet) %>%

arrange(created\_at) %>%

distinct(text, .keep\_all = TRUE) %>%

select(screen\_name, status\_id, text) %>%

mutate(text = str\_replace\_all(text, "https?://t.co/[A-Za-z\\d]+|&", "")) %>%

unnest\_tokens(word, text, token = "regex", pattern = reg) %>%

filter(str\_detect(word, "[a-z]"))

This parses the corpus of tweets into almost 1.5 million words.

Among this population of accounts, and ignoring “stop words” like “the” and “of”, what are the most common words? We can use ggplot2 to visualize this.

tweet\_words %>%

filter(!word %in% stop\_words$word) %>%

count(word, sort = TRUE) %>%

head(16) %>%

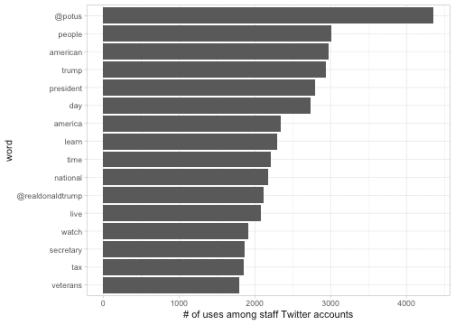
mutate(word = reorder(word, n)) %>%

ggplot(aes(word, n)) +

geom\_col() +

coord\_flip() +

labs(y = "# of uses among staff Twitter accounts")



No real surprises here. Many accounts mention @POTUS often, as well as words like “people”, “American”, and “Trump” that you’d expect from administration accounts.

**Finding a text signature: TF-IDF vectors**

What words make up someone’s “signature”? What make up mine, or Trump’s, or Mike Pence’s, or the op-ed’s?

We could start with the most common words someone uses. But there are some words, like “the” and “of” that just about everyone uses, as well as words like “President” that everyone in our dataset will use. So we also want to downweight words that appear across many documents. A common tool for balancing these two considerations and turning them into a “signature” vector is tf-idf: **term-frequency inverse-document-frequency**. This takes how frequently someone uses a term, but divides it by (the log of) how many documents mention it.

The bind\_tf\_idf function from tidytext lets us compute tf-idf on a dataset of word counts like this. Before we do, we bring in the op-ed as an additional document (since we’re interesting in considering it as one “special” document in our corpus).

# Combine in the op\_ed wordswith the name "OP-ED"

op\_ed\_words <- op\_ed %>%

unnest\_tokens(word, text) %>%

count(word)

word\_counts <- tweet\_words %>%

count(screen\_name, word, sort = TRUE) %>%

bind\_rows(op\_ed\_words %>% mutate(screen\_name = "OP-ED"))

# Compute TF-IDF using "word" as term and "screen\_name" as document

word\_tf\_idf <- word\_counts %>%

bind\_tf\_idf(word, screen\_name, n) %>%

arrange(desc(tf\_idf))

word\_tf\_idf

## # A tibble: 226,410 x 6

## screen\_name word n tf idf tf\_idf

##

## 1 JoshPaciorek #gogreen 170 0.0204 4.25 0.0868

## 2 USTradeRep ustr 147 0.0213 3.56 0.0757

## 3 DeptVetAffairs #vantagepoint 762 0.0173 3.56 0.0614

## 4 DeptofDefense #knowyourmil 800 0.0185 3.15 0.0584

## 5 DanScavino #trumptrain 655 0.0201 2.86 0.0575

## 6 USUN @ambassadorpower 580 0.0154 3.56 0.0548

## 7 USTreasury lew 690 0.0183 2.86 0.0523

## 8 HUDgov hud 566 0.0196 2.30 0.0451

## 9 OMBPress omb 38 0.0228 1.95 0.0444

## 10 SecElaineChao 'can 1 0.00990 4.25 0.0421

## # ... with 226,400 more rows

We can now see the words with the strongest associations to a user. For example, [Josh Paciorek](https://twitter.com/joshpaciorek?lang=en) (the VP’s Deputy Press Secretary) uses the hashtag #gogreen (supporting Michigan State Football) quite often; it makes up 2% of the words (tf, term frequency). Since no one else uses it (leading to an inverse document frequency, idf, of 4.5), this makes it a critical part of his TF-IDF vector (his “signature”).

We could take a look at the “signatures” of a few selected Twitter accounts.

library(drlib)

selected <- c("realDonaldTrump", "mike\_pence", "DeptVetAffairs", "KellyannePolls")

word\_tf\_idf %>%

filter(screen\_name %in% selected) %>%

group\_by(screen\_name) %>%

top\_n(12, tf\_idf) %>%

ungroup() %>%

mutate(word = reorder\_within(word, tf\_idf, screen\_name)) %>%

ggplot(aes(word, tf\_idf, fill = screen\_name)) +

geom\_col(show.legend = FALSE) +

scale\_x\_reordered() +

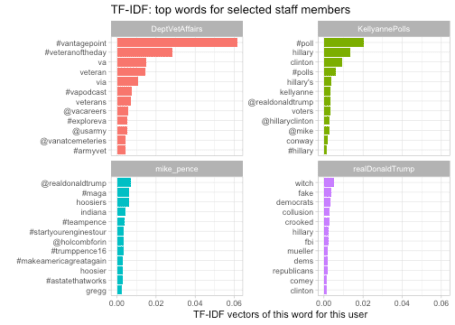
coord\_flip() +

facet\_wrap(~ screen\_name, scales = "free\_y") +

labs(x = "",

y = "TF-IDF vectors of this word for this user",

title = "TF-IDF: top words for selected staff members")



This gives us a set of words that are quite specific to each account. For instance, @DeptVetAffairs uses hashtags like “#vantagepoint” and “#veteranoftheday” that almost no other account in this set would use. Words that are specific to Trump include “witch” (as in “witch hunt”), “fake” (as in “fake news”) and other phrases that he tends to fixate on while other government officials don’t. (See [here](http://varianceexplained.org/r/trump-followup/) for my text analysis of Trump’s tweets as of August 2017).

This shows how TF-IDF offers us a vector (an association of each word with a number) that describes the unique signature of that document. To compare our documents (the op-ed with each Twitter account), we’ll be comparing those vectors.

**The widyr package: cosine similarity**

How can we compare two vectors to get a measure of document similarity? There are many approaches, but perhaps the most common for comparing TF-IDF vectors is [cosine similarity](https://en.wikipedia.org/wiki/Cosine_similarity). This is a combination of a dot product (multiplying the same term in document X and document Y together) and a normalization (dividing by the magnitudes of the vectors).

The widyr package offers a convenient way to compute pairwise similarities on a tidy dataset:

library(widyr)

# Find similarities between screen names

# upper = FALSE specifies that we don't want both A-B and B-A matches

word\_tf\_idf %>%

pairwise\_similarity(screen\_name, word, tf\_idf, upper = FALSE, sort = TRUE)

## # A tibble: 2,415 x 3

## item1 item2 similarity

##

## 1 VPComDir VPPressSec 0.582

## 2 DeptVetAffairs SecShulkin 0.548

## 3 ENERGY SecretaryPerry 0.472

## 4 HUDgov SecretaryCarson 0.440

## 5 usedgov BetsyDeVosED 0.417

## 6 Interior SecretaryZinke 0.386

## 7 SecPriceMD SecAzar 0.381

## 8 SecPompeo StateDept 0.347

## 9 VPPressSec VP 0.343

## 10 USDA SecretarySonny 0.337

## # ... with 2,405 more rows

The top results show that this elementary method is able to match people to their positions. The VP Press Secretary and VP Communications Director unsurprisingly work closely together and tweet on similar topics. Similarly, it matches Shulkin, Perry, Carson, DeVos, and Zinke to their (current or former) cabinet positions, and links the two consecutive Health and Human Services directors (Price and Azar) to each other.

It’s worth seeing this document similarity metric in action, but it’s not what you’re here for. We’re really excited about seeing comparisons between the *op-ed* and Twitter articles. We can

# Look only at the similarity of the op-ed to other documents

op\_ed\_similarity <- word\_tf\_idf %>%

pairwise\_similarity(screen\_name, word, tf\_idf, sort = TRUE) %>%

filter(item1 == "OP-ED")

library(drlib)

op\_ed\_similarity %>%

head(12) %>%

mutate(item2 = reorder(item2, similarity)) %>%

ggplot(aes(item2, similarity)) +

geom\_col() +

scale\_x\_reordered() +

coord\_flip() +

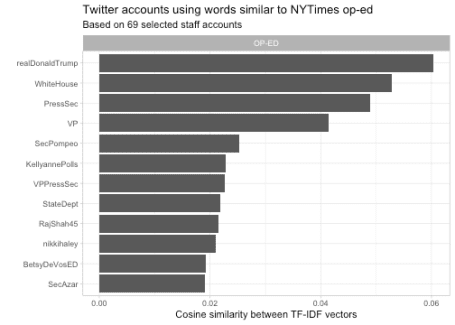
facet\_wrap(~ item1, scales = "free\_y") +

labs(x = "",

y = "Cosine similarity between TF-IDF vectors",

subtitle = "Based on 69 selected staff accounts",

title = "Twitter accounts using words similar to NYTimes op-ed")



This unveils the most similar writer as… Trump himself.

Hmmm. While that would certainly be a scoop, it doesn’t sound very likely to me. And the other top picks (the official White House account, the Press Secretary, and the Vice President) also seem like suspicious guesses.

**Interpreting machine learning: what words contributed to scores?**

The method of tf-idf is a fairly basic one for text mining, but as a result it has a useful trait: it’s based on a linear combination of one-score-per-word. This means we can say *exactly* how much each word contributed to a TF-IDF similarity between the article and a Twitter account. (Other machine learning methods allow *interactions* between words, which makes them harder to interpret).

We’ll try an approach of decomposing our TF-IDF similarity to see how much each . You could think of this as asking “if the op-ed hadn’t used this word, how much lower would the similarity score be?”

# This takes a little R judo, but it's worth the effort

# First we normalize the TF-IDF vector for each screen name,

# necessary for cosine similarity

tf\_idf <- word\_tf\_idf %>%

group\_by(screen\_name) %>%

mutate(normalized = tf\_idf / sqrt(sum(tf\_idf ^ 2))) %>%

ungroup()

# Then we join the op-ed words with the full corpus, and find

# the product of their TF-IDF with it in other documents

word\_combinations <- tf\_idf %>%

filter(screen\_name == "OP-ED") %>%

select(-screen\_name) %>%

inner\_join(tf\_idf, by = "word", suffix = c("\_oped", "\_twitter")) %>%

filter(screen\_name != "OP-ED") %>%

mutate(contribution = normalized\_oped \* normalized\_twitter) %>%

arrange(desc(contribution)) %>%

select(screen\_name, word, tf\_idf\_oped, tf\_idf\_twitter, contribution)

# Get the scores from the six most similar

word\_combinations %>%

filter(screen\_name %in% head(op\_ed\_similarity$item2)) %>%

mutate(screen\_name = reorder(screen\_name, -contribution, sum),

word = reorder\_within(word, contribution, screen\_name)) %>%

group\_by(screen\_name) %>%

top\_n(12, contribution) %>%

ungroup() %>%

mutate(word = reorder\_within(word, contribution, screen\_name)) %>%

ggplot(aes(word, contribution, fill = screen\_name)) +

geom\_col(show.legend = FALSE) +

scale\_x\_reordered() +

facet\_wrap(~ screen\_name, scales = "free\_y") +

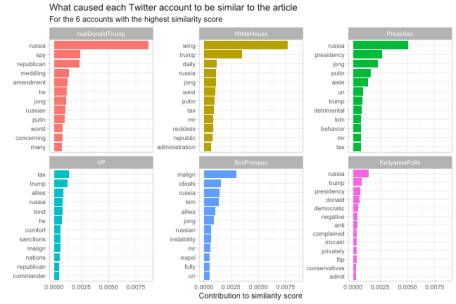
coord\_flip() +

labs(x = "",

y = "Contribution to similarity score",

title = "What caused each Twitter account to be similar to the article",

subtitle = "For the 6 accounts with the highest similarity score")



Now the reasons for the TF-IDF similarities become clearer.

The op-ed uses the words “Russia” five times. The Press Secretary and *especially* Trump mention Russia multiple times on their Twitter accounts, always within the context of defending Trump (as expected). Several accounts also get a high score because they mention the word “Trump” so frequently.

Unfortunately, with a document this short and topical, that’s all it takes to get a high similarity score (a bag of words method can’t understand the context, such as mentioning Russia in a negative or a defensive context). This is one reason it’s worth taking a closer look at what goes into an algorithm,

Having said that, there’s one signature I think is notable.

**“Malign behavior”**

None of the relevant documents included that. I’d like to focus on another word that did: **malign**. Emphasis mine:

He complained for weeks about senior staff members letting him get boxed into further confrontation with Russia, and he expressed frustration that the United States continued to impose sanctions on the country for its **malign** behavior.

“Malign” isn’t as rare a word as “lodestar”, but it’s notable for being used in the exact same context (discussing Russia or other countries’ behavior) in a number of tweets from both Secretary of State Pompeo and the @StateDepartment account.

**Conclusion: Opening the black box**

**Text Analysis of Trump’s Tweets**

**The dataset**

First we’ll retrieve the content of Donald Trump’s timeline using the userTimeline function in the twitteR package:1

library(dplyr)

library(purrr)

library(twitteR)

*# You'd need to set global options with an authenticated app*

setup\_twitter\_oauth(getOption("twitter\_consumer\_key"),

getOption("twitter\_consumer\_secret"),

getOption("twitter\_access\_token"),

getOption("twitter\_access\_token\_secret"))

*# We can request only 3200 tweets at a time; it will return fewer*

*# depending on the API*

trump\_tweets **<-** userTimeline("realDonaldTrump", n **=** 3200)

trump\_tweets\_df **<-** tbl\_df(map\_df(trump\_tweets, as.data.frame))

*# if you want to follow along without setting up Twitter authentication,*

*# just use my dataset:*

load(url("http://varianceexplained.org/files/trump\_tweets\_df.rda"))

We clean this data a bit, extracting the source application. (We’re looking only at the iPhone and Android tweets- a much smaller number are from the web client or iPad).

library(tidyr)

tweets **<-** trump\_tweets\_df **%>%**

select(id, statusSource, text, created) **%>%**

extract(statusSource, "source", "Twitter for (.\*?)<") **%>%**

filter(source **%in%** **c**("iPhone", "Android"))

Overall, this includes 628 tweets from iPhone, and 762 tweets from Android.

One consideration is what time of day the tweets occur, which we’d expect to be a “signature” of their user. Here we can certainly spot a difference:

library(lubridate)

library(scales)

tweets **%>%**

count(source, hour **=** hour(with\_tz(created, "EST"))) **%>%**

mutate(percent **=** n **/** **sum**(n)) **%>%**

ggplot(aes(hour, percent, color **=** source)) **+**

geom\_line() **+**

scale\_y\_continuous(labels **=** percent\_format()) **+**

labs(x **=** "Hour of day (EST)",

y **=** "% of tweets",

color **=** "")

Trump on the Android does a lot more tweeting in the morning, while the campaign posts from the iPhone more in the afternoon and early evening.

Another place we can spot a difference is in Trump’s anachronistic behavior of “manually retweeting” people by copy-pasting their tweets, then surrounding them with quotation marks:

*"@trumplican2016: @realDonaldTrump @DavidWohl stay the course mr trump your message is resonating with the PEOPLE"*

*— Donald J. Trump (@realDonaldTrump) July 28, 2016*

Almost all of these quoted tweets are posted from the Android:

In the remaining by-word analyses in this text, I’ll filter these quoted tweets out (since they contain text from followers that may not be representative of Trump’s own tweets).

Somewhere else we can see a difference involves sharing links or pictures in tweets.

tweet\_picture\_counts **<-** tweets **%>%**

filter(**!**str\_detect(text, '^"')) **%>%**

count(source,

picture **=** ifelse(str\_detect(text, "t.co"),

"Picture/link", "No picture/link"))

ggplot(tweet\_picture\_counts, aes(source, n, fill **=** picture)) **+**

geom\_bar(stat **=** "identity", position **=** "dodge") **+**

labs(x **=** "", y **=** "Number of tweets", fill **=** "")

It turns out tweets from the iPhone were **38 times as likely to contain either a picture or a link.** This also makes sense with our narrative: the iPhone (presumably run by the campaign) tends to write “announcement” tweets about events, like this:

*Thank you Windham, New Hampshire! #TrumpPence16 #MAGA pic.twitter.com/ZL4Q01Q49s*

*— Donald J. Trump (@realDonaldTrump) August 7, 2016*

While Android (Trump himself) tends to write picture-less tweets like:

*The media is going crazy. They totally distort so many things on purpose. Crimea, nuclear, "the baby" and so much more. Very dishonest!*

*— Donald J. Trump (@realDonaldTrump) August 7, 2016*

**Comparison of words**

Now that we’re sure there’s a difference between these two accounts, what can we say about the difference in the *content*? We’ll use the tidytext package that Julia Silge and I developed.

We start by dividing into individual words using the unnest\_tokens function (see this vignette for more), and removing some common “stopwords”2:

library(tidytext)

reg **<-** "([^A-Za-z\\d#@']|'(?![A-Za-z\\d#@]))"

tweet\_words **<-** tweets **%>%**

filter(**!**str\_detect(text, '^"')) **%>%**

mutate(text **=** str\_replace\_all(text, "https://t.co/[A-Za-z\\d]+|&amp;", "")) **%>%**

unnest\_tokens(word, text, token **=** "regex", pattern **=** reg) **%>%**

filter(**!**word **%in%** stop\_words**$**word,

str\_detect(word, "[a-z]"))

tweet\_words

## # A tibble: 8,753 x 4

## id source created word

## <chr> <chr> <time> <chr>

## 1 676494179216805888 iPhone 2015-12-14 20:09:15 record

## 2 676494179216805888 iPhone 2015-12-14 20:09:15 health

## 3 676494179216805888 iPhone 2015-12-14 20:09:15 #makeamericagreatagain

## 4 676494179216805888 iPhone 2015-12-14 20:09:15 #trump2016

## 5 676509769562251264 iPhone 2015-12-14 21:11:12 accolade

## 6 676509769562251264 iPhone 2015-12-14 21:11:12 @trumpgolf

## 7 676509769562251264 iPhone 2015-12-14 21:11:12 highly

## 8 676509769562251264 iPhone 2015-12-14 21:11:12 respected

## 9 676509769562251264 iPhone 2015-12-14 21:11:12 golf

## 10 676509769562251264 iPhone 2015-12-14 21:11:12 odyssey

## # ... with 8,743 more rows

What were the most common words in Trump’s tweets overall?

These should look familiar for anyone who has seen the feed. Now let’s consider which words are most common from the Android relative to the iPhone, and vice versa. We’ll use the simple measure of log odds ratio, calculated for each word as:3

log2(# in Android+1Total Android+1# in iPhone+1Total iPhone+1)log2⁡(# in Android+1Total Android+1# in iPhone+1Total iPhone+1)

android\_iphone\_ratios **<-** tweet\_words **%>%**

count(word, source) **%>%**

filter(**sum**(n) **>=** 5) **%>%**

spread(source, n, fill **=** 0) **%>%**

ungroup() **%>%**

mutate\_each(funs((. **+** 1) **/** **sum**(. **+** 1)), **-**word) **%>%**

mutate(logratio **=** log2(Android **/** iPhone)) **%>%**

arrange(desc(logratio))

Which are the words most likely to be from Android and most likely from iPhone?

A few observations:

* **Most hashtags come from the iPhone**. Indeed, almost no tweets from Trump’s Android contained hashtags, with some rare exceptions like this one. (This is true only because we filtered out the quoted “retweets”, as Trump does sometimes quote tweets like this that contain hashtags).
* **Words like “join” and “tomorrow”, and times like “7pm”, also came only from the iPhone**. The iPhone is clearly responsible for event announcements like this one (“Join me in Houston, Texas tomorrow night at 7pm!”)
* **A lot of “emotionally charged” words, like “badly”, “crazy”, “weak”, and “dumb”, were overwhelmingly more common on Android.** This supports the original hypothesis that this is the “angrier” or more hyperbolic account.

**Sentiment analysis: Trump’s tweets are much more negative than his campaign’s**

Since we’ve observed a difference in sentiment between the Android and iPhone tweets, let’s try quantifying it. We’ll work with the NRC Word-Emotion Association lexicon, available from the tidytext package, which associates words with 10 sentiments: **positive**, **negative**, **anger**, **anticipation**, **disgust**, **fear**, **joy**, **sadness**, **surprise**, and **trust**.

nrc **<-** sentiments **%>%**

filter(lexicon **==** "nrc") **%>%**

dplyr**::**select(word, sentiment)

nrc

## # A tibble: 13,901 x 2

## word sentiment

## <chr> <chr>

## 1 abacus trust

## 2 abandon fear

## 3 abandon negative

## 4 abandon sadness

## 5 abandoned anger

## 6 abandoned fear

## 7 abandoned negative

## 8 abandoned sadness

## 9 abandonment anger

## 10 abandonment fear

## # ... with 13,891 more rows

To measure the sentiment of the Android and iPhone tweets, we can count the number of words in each category:

sources **<-** tweet\_words **%>%**

group\_by(source) **%>%**

mutate(total\_words **=** n()) **%>%**

ungroup() **%>%**

distinct(id, source, total\_words)

by\_source\_sentiment **<-** tweet\_words **%>%**

inner\_join(nrc, by **=** "word") **%>%**

count(sentiment, id) **%>%**

ungroup() **%>%**

complete(sentiment, id, fill **=** **list**(n **=** 0)) **%>%**

inner\_join(sources) **%>%**

group\_by(source, sentiment, total\_words) **%>%**

summarize(words **=** **sum**(n)) **%>%**

ungroup()

head(by\_source\_sentiment)

## # A tibble: 6 x 4

## source sentiment total\_words words

## <chr> <chr> <int> <dbl>

## 1 Android anger 4901 321

## 2 Android anticipation 4901 256

## 3 Android disgust 4901 207

## 4 Android fear 4901 268

## 5 Android joy 4901 199

## 6 Android negative 4901 560

(For example, we see that 321 of the 4901 words in the Android tweets were associated with “anger”). We then want to measure how much more likely the Android account is to use an emotionally-charged term relative to the iPhone account. Since this is count data, we can use a Poisson test to measure the difference:

library(broom)

sentiment\_differences **<-** by\_source\_sentiment **%>%**

group\_by(sentiment) **%>%**

do(tidy(poisson.test(.**$**words, .**$**total\_words)))

sentiment\_differences

## Source: local data frame [10 x 9]

## Groups: sentiment [10]

##

## sentiment estimate statistic p.value parameter conf.low

## (chr) (dbl) (dbl) (dbl) (dbl) (dbl)

## 1 anger 1.492863 321 2.193242e-05 274.3619 1.2353162

## 2 anticipation 1.169804 256 1.191668e-01 239.6467 0.9604950

## 3 disgust 1.677259 207 1.777434e-05 170.2164 1.3116238

## 4 fear 1.560280 268 1.886129e-05 225.6487 1.2640494

## 5 joy 1.002605 199 1.000000e+00 198.7724 0.8089357

## 6 negative 1.692841 560 7.094486e-13 459.1363 1.4586926

## 7 positive 1.058760 555 3.820571e-01 541.4449 0.9303732

## 8 sadness 1.620044 303 1.150493e-06 251.9650 1.3260252

## 9 surprise 1.167925 159 2.174483e-01 148.9393 0.9083517

## 10 trust 1.128482 369 1.471929e-01 350.5114 0.9597478

## Variables not shown: conf.high (dbl), method (fctr), alternative (fctr)

And we can visualize it with a 95% confidence interval:

Thus, Trump’s Android account uses about 40-80% more words related to **disgust**, **sadness**, **fear**, **anger**, and other “negative” sentiments than the iPhone account does. (The positive emotions weren’t different to a statistically significant extent).

We’re especially interested in which words drove this different in sentiment. Let’s consider the words with the largest changes within each category:

This confirms that lots of words annotated as negative sentiments (with a few exceptions like “crime” and “terrorist”) are more common in Trump’s Android tweets than the campaign’s iPhone tweets.

I was fairly skeptical from the start that we could get strong results with document-comparison methods like this, especially on such a small article. That opinion mirrored people with much more expertise than I have:

But I’m satisfied with this analysis both as a demonstration of tidytext methods and one on the importance of model interpretability. When we ran a TF-IDF comparison, we knew it was wrong because @realDonaldTrump appeared at the top. But what if Trump *hadn’t* been the one to mention Russia the most, or if another false positive had caused an account to rise to the top? Breaking similarity scores down by word is a useful way to interrogate our model and understand its output.