biweekly_corpus_tokenization

Jack Lovell

9/4/2020

#Intro In the previous document we explored how to use the twitter API to scrape the web for large amounts of text data. Rather than searching for a given term within the twitter API, we look at replies from a specific account. The tutotial we followed was not necessarily geared towards cleaning the data for natural language processing. In this report, I will follow a tutorial that can be found here: https://www.earthdatascience.org/courses/earth-analytics/get-data-using-apis/text-mining-twitter-data-intro-r/ The objective is to get our data from the twitter API, after sesrching for the specific terms "computational neuroscience" and then use the tutorial attached above to clean the data specificsally for natural language processing.

So let's get to it!

We will check out comp neuro twitter

```r
#Import rtweet and other libs
library(rtweet)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# plotting and pipes - tidyverse!
library("ggplot2")
library(dplyr)
# text mining library
library(tidytext)
# plotting packages
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
library(ggraph)
#Use our own token
twitter_token <- create_token(
  app = 'meta_nlp',
  consumer_key="pzOo1DHoCGpIjElgpHzVDHgl4",
  consumer_secret="QpTDNngQEDv9CkroVTjBlpRQcoL4iJvGkrAqPNAmXrIFdEVSNE",
  access_token="1301874142359293952-K7O1mzxSh7PEPR799S9Db3cfHzLx97",
  access_secret="mOvMMSEmBGZwT9sqZzuX8Oe1K2PRFwohGEOA6KEkxXZ1w"
)
#search twitter
science_tweets <- search_tweets(q="computational+neuroscience", n = 50000, lang="en", include_rts = F)
head(science_tweets$text)
```

```
## [1] "#superneuromatch abstract submission deadline is October 2! (Yes, that fast!). *ALL NEUROSCIENC
## [2] "taking a neuroscience course so that I can take computational neuro and my math brain can’t han
## [3] "New Research: Braitenberg Vehicles as Computational Tools for Research in Neuroscience: Valentin
## [4] "The scientific study of the nervous system increased significantly during the second half of the
## [5] "Applications are open (until 29Sep) for a research fellowship in the Neuroengineering and Compu
## [6] "Biological Cybernetics journal has a new subtitle, \"Advances in Computational Neuroscience and
```

Preprocessing text

Our first step will be removing URLS from our tweets, as those are clearly not useful in a text analysis!

```
#lets do this manually & without the tidyverse package
science_tweets$stripped_text <- gsub("http.*","",  science_tweets$text)
science_tweets$stripped_text <- gsub("https.*","", science_tweets$stripped_text)
head(science_tweets$stripped_text)
```

```
## [1] "#superneuromatch abstract submission deadline is October 2! (Yes, that fast!). *ALL NEUROSCIENC
## [2] "taking a neuroscience course so that I can take computational neuro and my math brain can’t han
## [3] "New Research: Braitenberg Vehicles as Computational Tools for Research in Neuroscience: Valentin
## [4] "The scientific study of the nervous system increased significantly during the second half of the
## [5] "Applications are open (until 29Sep) for a research fellowship in the Neuroengineering and Compu
## [6] "Biological Cybernetics journal has a new subtitle, \"Advances in Computational Neuroscience and
```

Nice! We now have a new column called ‘stripped_text’ too which is great, as our original data are not tampered with at all.

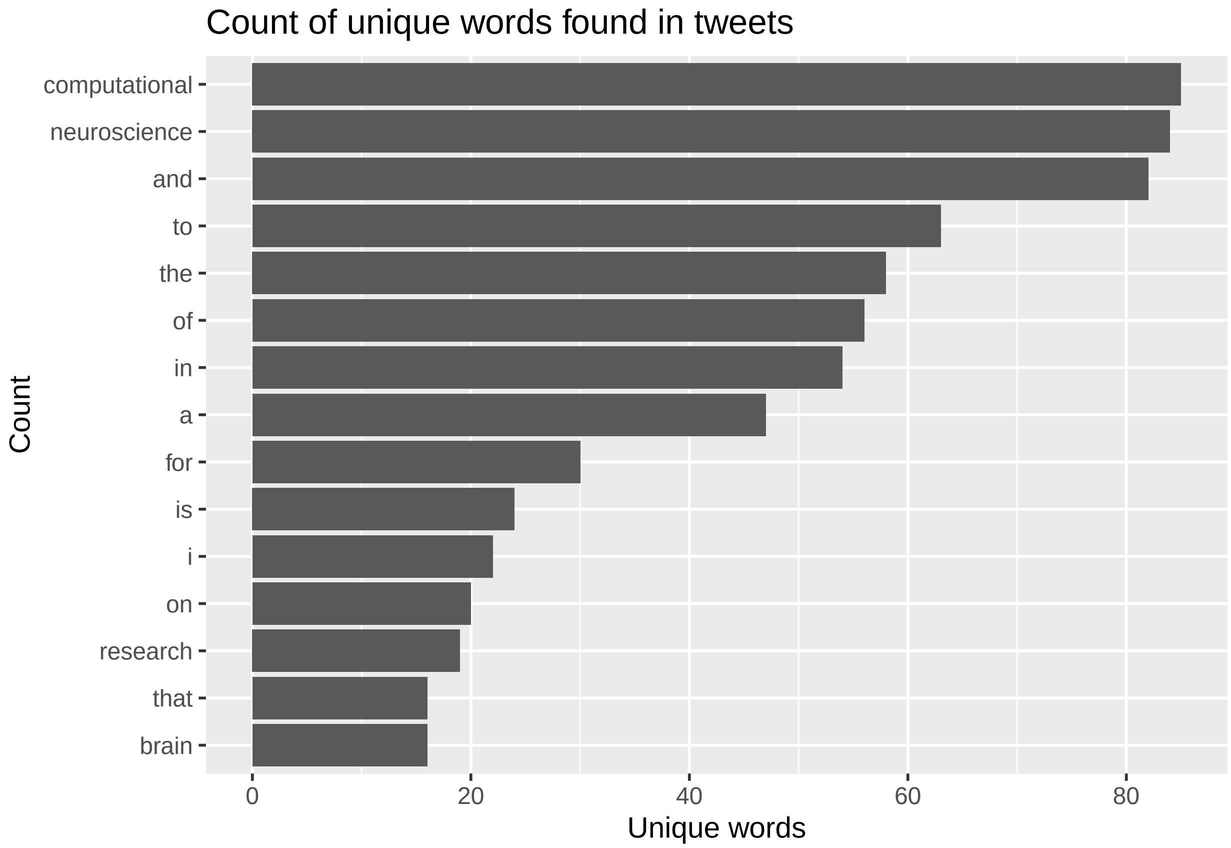
Rather than going through each step of cleaning the text like we did last time (i.e. convert to lowercase, remove punctuation, etc.) we can leverage the power of the tidytext and use the function unset_tokens(). Which is super useful in speeding up your precprocessing pipeline. Just make sure you know what it is doing!!

```
#lets store it in a new column!
science_tweets_clean <- science_tweets%>%
  dplyr::select(stripped_text)%>%
  tidytext::unnest_tokens(word, stripped_text)
```

Now let's visualize this data, as it give us a better understanding of what is really going on in our data.

```
# plot the top 15 words -- notice any issues?
science_tweets_clean %>%
  count(word, sort = TRUE) %>%
  top_n(15) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(x = word, y = n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip() +
      labs(x = "Count",
      y = "Unique words",
      title = "Count of unique words found in tweets")
```

```
## Selecting by n
```



As we saw last time, we don't want "stop words" as they give don't contribute to the smenatic meaning of the text at all... After plotting the number of unique words, it is obvious that some of the most frequent words are ‘stop words.’ Let’s leverage the power of tidytext and remove these! We can do this by using the stop_words data, and anti_join() to remove them.

```
data("stop_words")
#what is it?
head(stop_words)
```

```
## # A tibble: 6 x 2
##   word      lexicon
##   <chr>     <chr>
## 1 a         SMART
## 2 a's       SMART
## 3 able      SMART
## 4 about     SMART
## 5 above     SMART
## 6 according SMART
```

```
#how many rows w/ stop words
nrow(science_tweets_clean)
```

```
## [1] 2288
```

```
#remove
cleaned_tweet_words <- science_tweets_clean%>%
  anti_join(stop_words)
```

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

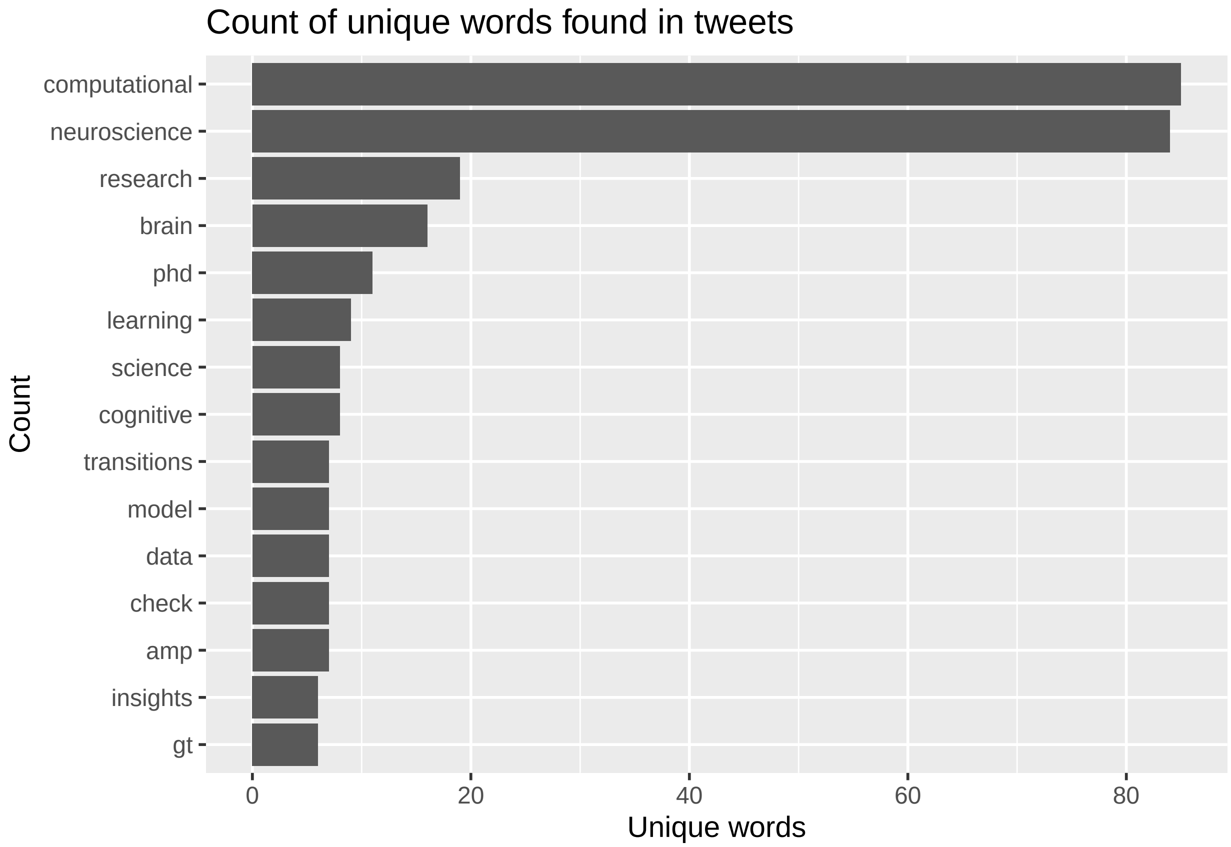
```
#now how many rows?
nrow(cleaned_tweet_words)
```

```
## [1] 1269
```

Wow, our data were nearly halfed. Let’s take another look at the data now that it’s a bit more clean

```
# plot the top 15 words
cleaned_tweet_words %>%
  count(word, sort = TRUE) %>%
  top_n(15) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(x = word, y = n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip() +
      labs(x = "Count",
      y = "Unique words",
      title = "Count of unique words found in tweets")
```

```
## Selecting by n
```



Nice, so now it's a bit more obvious what our tweets are saying. Rather than trying to nterpret what people were saying with the word "and", we can test meaningful differences in our data, such as if cognitive or biology was mentioned more. Or what topics of interest were mentioned most (emotion, gene transcription, etc.)

One piece of insight that would be good to gain is to see what words occur with eachother. Some call this looking for "networks" of words, and i often known as searching for "bigrams" within your data. This will be an important concept when we get to modeling, but for now all we will need to know is that it is telling us which words are said in succession.

```
library(devtools)
```

```
## Loading required package: usethis
```

```
install_github("dgrtwo/widyr")
```

```
## Skipping install of 'widyr' from a github remote, the SHA1 (13b3b3c4) has not changed since last ins
##   Use `force = TRUE` to force installation
```

```
library(widyr)

# remove punctuation, convert to lowercase, add id for each tweet!
science_tweets_paired_words <- science_tweets %>%
  dplyr::select(stripped_text) %>%
  unnest_tokens(paired_words, stripped_text, token = "ngrams", n = 2)
```

```r
science_tweets_paired_words %>%
  count(paired_words, sort = TRUE)
```

```
## # A tibble: 1,928 x 2
##    paired_words                    n
##    <chr>                       <int>
##  1 computational neuroscience    55
##  2 in computational              11
##  3 neuroscience and               9
##  4 of the                         9
##  5 in the                         7
##  6 and computational              6
##  7 and transitions                6
##  8 brain states                   6
##  9 from computational             6
## 10 insights from                  6
## # ... with 1,918 more rows
```

```r
library(tidyr)
```

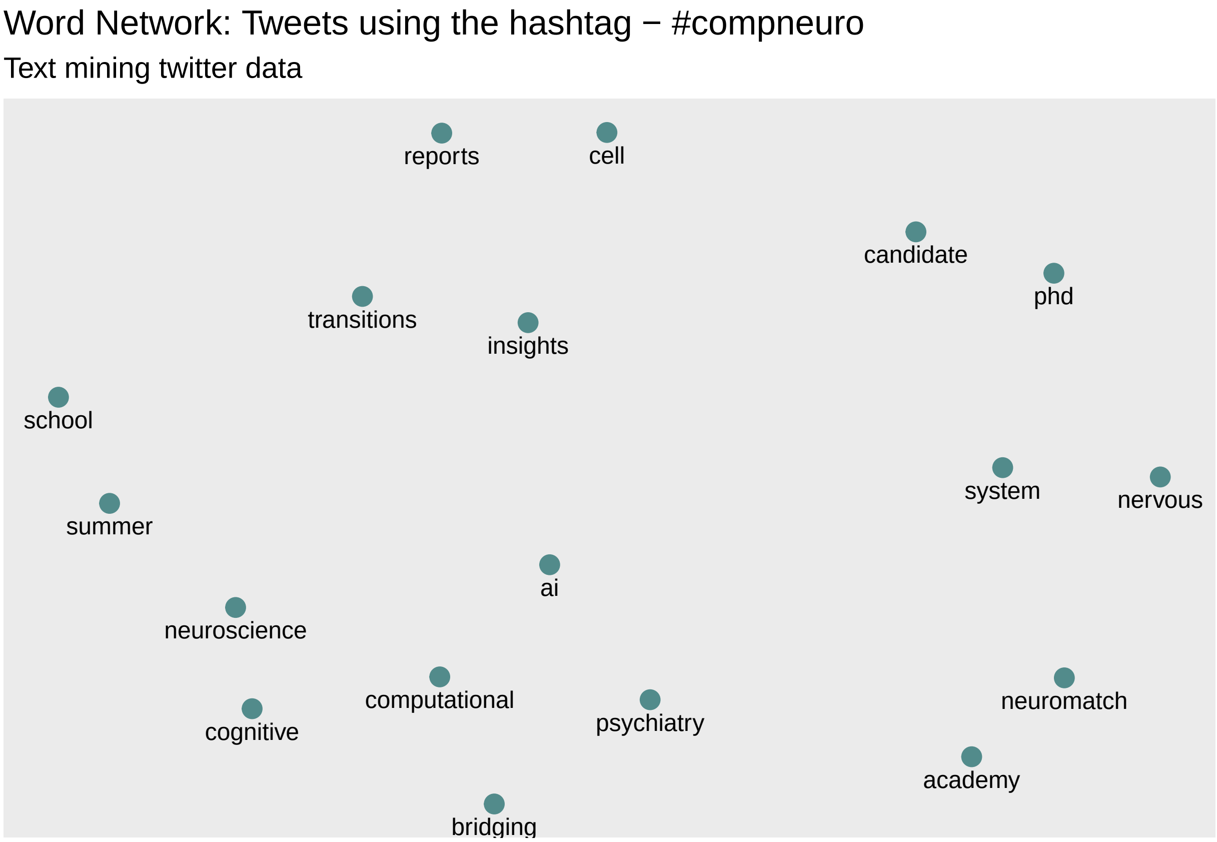
```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:igraph':
##
##     crossing
```

```r
science_tweets_separated_words <- science_tweets_paired_words %>%
  separate(paired_words, c("word1", "word2"), sep = " ")

science_tweets_filtered <- science_tweets_separated_words %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

# new bigram counts:
science_words_counts <- science_tweets_filtered %>%
  count(word1, word2, sort = TRUE)

head(science_words_counts)
```

```
## # A tibble: 6 x 3
##   word1         word2            n
##   <chr>         <chr>        <int>
## 1 computational neuroscience    55
## 2 transitions   insights         6
## 3 cognitive     neuroscience     5
## 4 cell          reports          4
## 5 computational cognitive        4
## 6 phd           candidate        4
```

Now let's visualize our network

```r
library(igraph)
library(ggraph)

# plot comp neuro word network
# (plotting graph edges is currently broken)
science_words_counts %>%
        filter(n >= 3) %>%
        graph_from_data_frame() %>%
        ggraph(layout = "fr") +
        # geom_edge_link(aes(edge_alpha = n, edge_width = n))
        # geom_edge_link(aes(edge_alpha = n, edge_width = n)) +
        geom_node_point(color = "darkslategray4", size = 3) +
        geom_node_text(aes(label = name), vjust = 1.8, size = 3) +
        labs(title = "Word Network: Tweets using the hashtag - #compneuro",
             subtitle = "Text mining twitter data ",
             x = "", y = "")
```



Awesome! It's intuitive that certain terms occur together, like neuro, cognitive, psychiatry, and other ones like cell and system, or summer and school. Although useful we will next want to understand how we can model such data, and also practice our cleaning on some actual meta-analytic text!!