Statistical Language Models 2019 Week 3 part 2

Dr. Dave Campbell
davecampbell@math.carleton.ca

Approximate Course Outline

Week 1: ShinyApps and Dashboarding

Week 2: TidyText & obtaining data, dealing with time events

Week 3: Regular Expressions; Word co-occurrence explorations

Week 4: Sentiment Analysis; Stochastic process models

Week 5: Exponential models for time between events.

Week 6: Bayesian Basics; Author attribution models; hierarchical models

Week 7: MCMC Diagnostics

Week 8: Embeddings and Word2Vec; Cryptography

Week 9: Clustering; Latent Dirichlet Allocation and topic models.

Week 10: Variational Inference

Week 11: Getting Fancier with Language Models

Week 12: Student projects and presentations

https://www.tidytextmining.com/twitter.html

Great book.

Available irl and online

Get twitter data

```
Climateactivist  = get_timeline("GretaThunberg",    n=10000)

GasGroup          = get_timeline("@exxonmobil", n = 3500)

#Impeachedprez = get_timeline("realDonaldTrump", n = 3500)

#CanadianPM     = get_timeline("JustinTrudeau", n = 3500)

Tweets = rbind(Climateactivist, GasGroup)# ,Impeachedprez,
CanadianPM)
```

Tweet frequency:

```
plot(sort(Tweets$created_at[Tweets$screen_name=="exxonmobil"]),

ylab="Date",xlab="tweet #")

lines(sort(Tweets$created_at[Tweets$screen_name=="GretaThunberg"]),
col=2)
```

```
#ditch punctuation

nourls = str_replace_all(Tweets$text, pattern="[[:punct:]]", "")

#http+any word type characters now that punctuation is gone

nourls = str_replace_all(nourls , pattern="http[[:alnum:]]*", "")

(tidy_tweets = mutate(Tweets,textnourls = nourls) %>%

     filter(!str_detect(textnourls, "^RT")) %>% # no retweets

   unnest_tokens(word, textnourls, token = "words") %>%

   anti_join(stop_words,by="word"))
```

Word frequency per user

```
frequency = tidy_tweets %>%
  group_by(screen_name) %>%
  count(word, sort = TRUE) %>%
  left_join(tidy_tweets %>%
              group_by(screen_name) %>%
              summarise(total = n())) %>%
  mutate(freq = n/total)
```

Spreading data out so that it can be plotted

```
library(tidyr)


spreadwords = frequency %>%
  select(screen_name, word, freq) %>%
  spread(screen_name, freq)


plot(spreadwords$exxonmobil, spreadwords$GretaThunberg,type='p')
```

```
library(scales)

ggplot(spreadwords, aes(exxonmobil, GretaThunberg)) +
  geom_jitter(alpha = 0.1, size = 2.5, width = 0.25, height = 0.25) +
  geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
  scale_x_log10(labels = percent_format()) +
  scale_y_log10(labels = percent_format()) +
  geom_abline(color = "red")
```

Frequency of word usage in one month bins

```
library(lubridate) # has some other time functions including floor_date == trunc from last day

words_by_time <- tidy_tweets %>%

  mutate(time_floor = floor_date(created_at, unit = "1 month")) %>%

  count(time_floor, screen_name, word) %>%

  group_by(screen_name, time_floor) %>%

  mutate(time_total = sum(n)) %>%

  group_by(screen_name, word) %>%

  mutate(word_total = sum(n)) %>%

  ungroup() %>%

  rename(count = n) %>%

  filter(word_total > 30)


words_by_time
```

Subsetting the tweets by time

Keep just 2019-2020 tweets, how?

```
words_by_time = words_by_time[

                    !grepl(words_by_time$time_floor, pattern="_________") ,]
```

What about the probability of occurrence?

Consider the change in probability of word use as a function of time:

Event ~ Binomial(n,p)

p depends on time

Linear Regression models don’t work

Use logistic regression (aka take a course on Generalized Linear Models):

tidyr let’s us make mini data frames
purrr let’s us repeat a procedure on mini data frames

```
library(purrr)

library(tidyr)

# make a whole lot of tiny little data frames

nested_data <- words_by_time %>%

    nest(-word, -screen_name)

nested_data
```

Build a model within each month for the biggest change in tweet probability with respect to time

```
nested_models = nested_data %>%
  mutate(models = map(data, ~ glm(cbind(count, time_total) ~ time_floor, .,
                  family = "binomial")))
nested_models
```

```
#Extract all the slopes

library(broom)

slopes = nested_models %>%

  mutate(models = map(models, tidy)) %>%

  unnest(cols = c(models)) %>%

  filter(term == "time_floor") %>%

  mutate(adjusted.p.value = p.adjust(p.value))

slopes
```

```
top_slopes = slopes %>%

  filter(adjusted.p.value < 0.05)



top_slopes
```

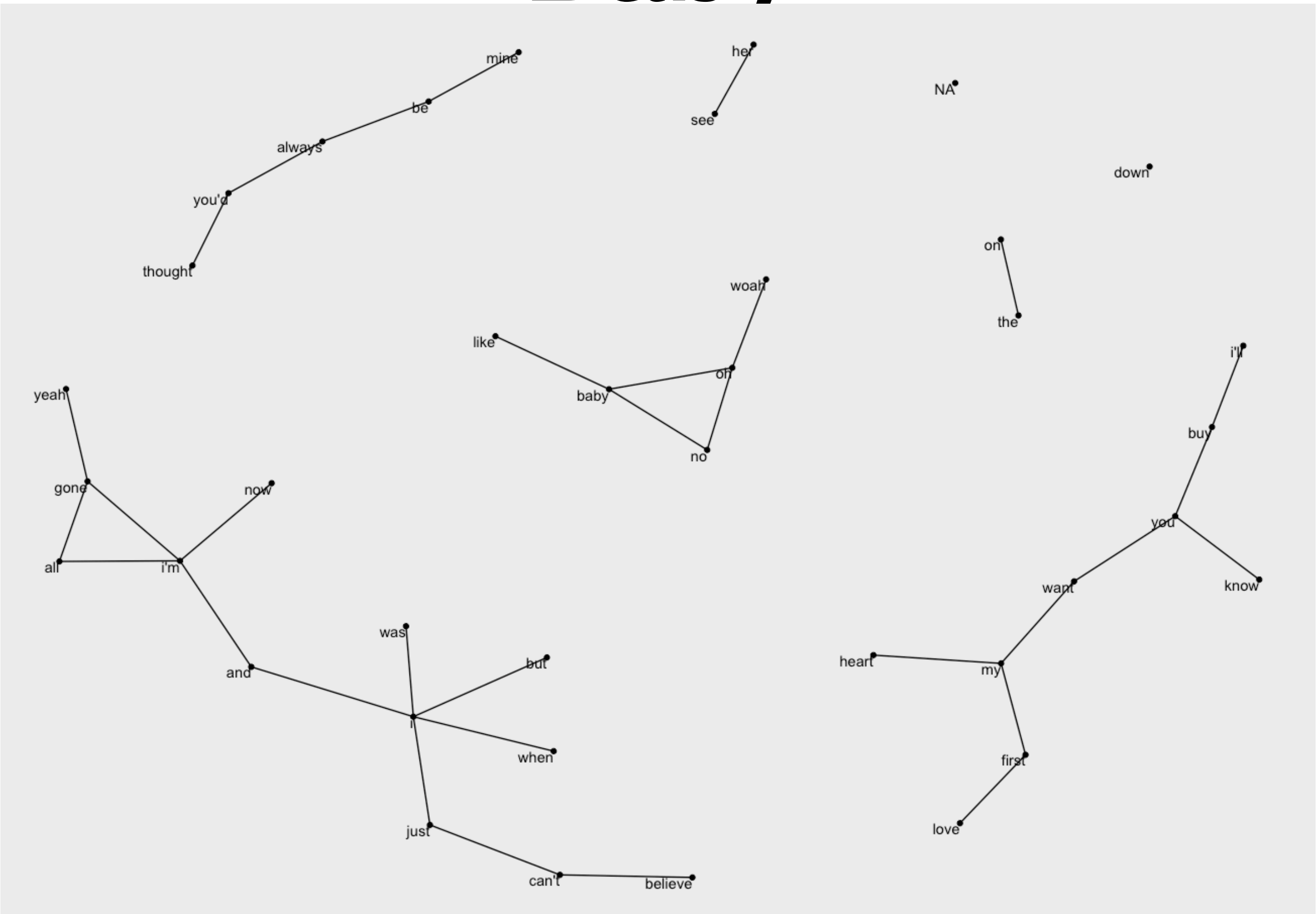
Greta changes in word use

```
words_by_time %>%
  inner_join(top_slopes, by = c("word", "screen_name")) %>%
  filter(screen_name == "GretaThunberg") %>%
  ggplot(aes(time_floor, count/time_total, color = word)) +
  geom_line(size = 1.3) +
  labs(x = NULL, y = "Word frequency",title="Word use changes for Greta")
```

Exxon changes in word use

```
words_by_time %>%

  inner_join(top_slopes, by = c("word", "screen_name")) %>%

  filter(screen_name == "exxonmobil") %>%

  ggplot(aes(time_floor, count/time_total, color = word)) +

  geom_line(size = 1.3) +

  labs(x = NULL, y = "Word frequency",title="Word use changes for
Exxon")
```

Baby


mine
be
always
you'd
thought
her
see
NA
down
on
the
woah
like
oh
baby
no
i'll
buy
yeah
gone
now
all
i'm
you
want
know
was
but
and
i
when
heart
my
first
love
just
can't
believe

https://genius.com song lyrics:

```
library(genius)

library(tidytext)

library(tidyverse)
```

Lyrics as a tibble —> examine bigrams and word co-occurrence

```
Artist = "Justin Bieber"

Song = "Baby"

Lyrics = genius_lyrics(artist=Artist ,song=Song)

LyricsTib =  Lyrics %>% unnest_tokens(output = word,input = lyric, token = "words")

LyricsTib %>% count(word,sort=TRUE)
```

```
lyrics_bigrams = Lyrics    %>%

  unnest_tokens(bigram, lyric, token = "ngrams", n = 2)


lyrics_bigramsCount =  lyrics_bigrams %>% count(bigram,sort=TRUE)
```

```r
#seperate bigrams into two columns:

bigrams_separated = lyrics_bigramsCount %>%

        separate(bigram, c("word1", "word2"), sep = " ")


#Examine bigrams in context:

bigrams_separated %>%

  filter(word1 == "baby") %>%

  count(word1, word2, sort = TRUE)
```

Build a graph of relationships between words

```
bigram_graph = bigrams_separated %>%

  filter(n > 1) %>%

  graph_from_data_frame() #examines relationship between first and
second column co-occurrence

bigram_graph
```

Making random plots of graphs
Plots edges and nodes (no direction)

```
library(ggraph) # makes nicer graph plots



ggraph(bigram_graph, layout = "fr") +

  geom_edge_link() +

  geom_node_point() +

  geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```

Distances between words are arbitrary
Edges (Lines) = connections, Arrows = directions,

```
a = grid::arrow(type = "closed", length = unit(.15, "inches"))


ggraph(bigram_graph, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.07, 'inches')) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  theme_void()
```