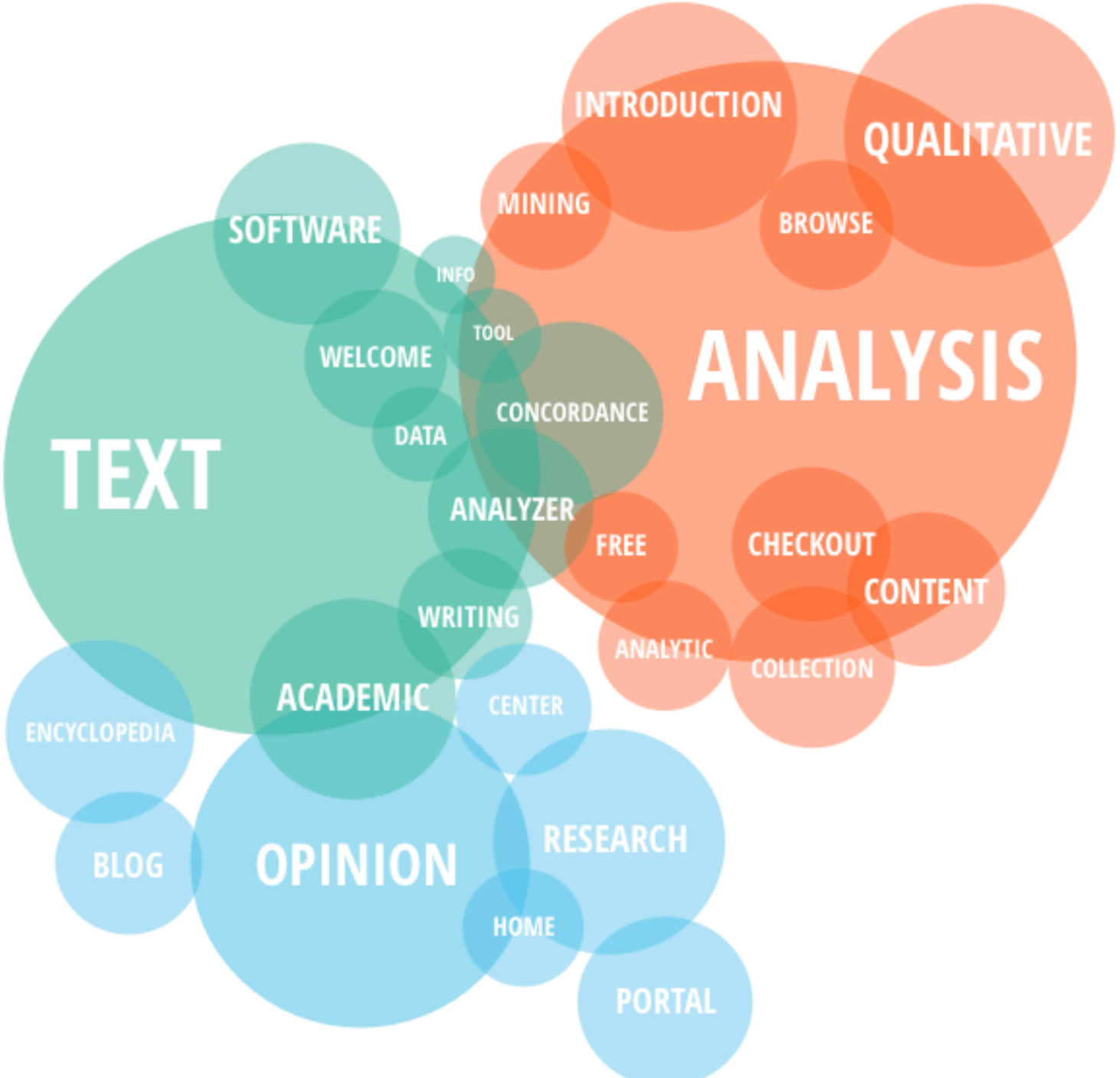


Taryn Morris
17 October 2019

Ladies CAPE TOWN

Who am I?

BirdLife
SOUTH AFRICA
Giving Conservation Wings

University of Cape Town • iYunivesithi yaseKapa • Universiteit van Kaapstad

University of the Witwatersrand, Johannesburg

Bard CEP
CENTER FOR ENVIRONMENTAL POLICY

IXIO

PREDICT. BETTER. GROW.

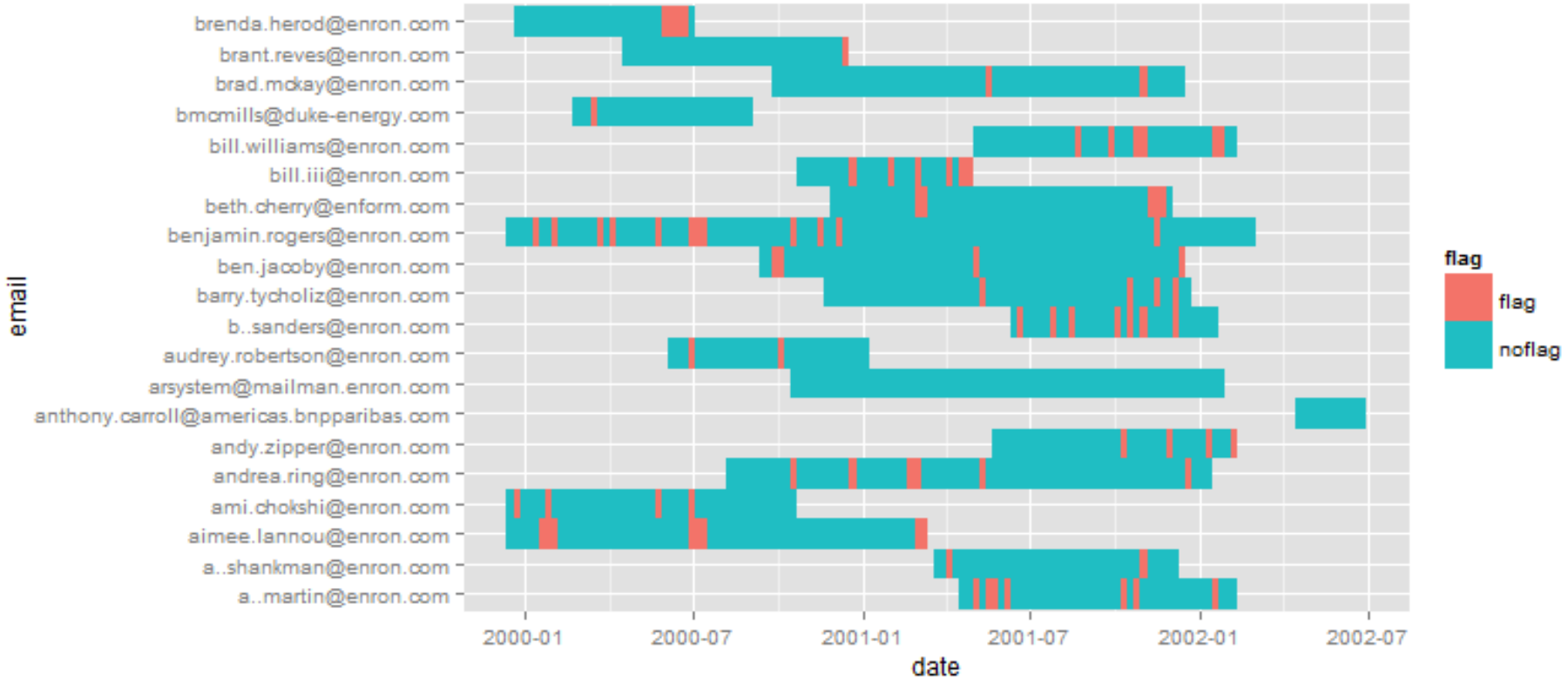
Text Analysis

- Text is important
  - Books, social media, Voice assistants, email

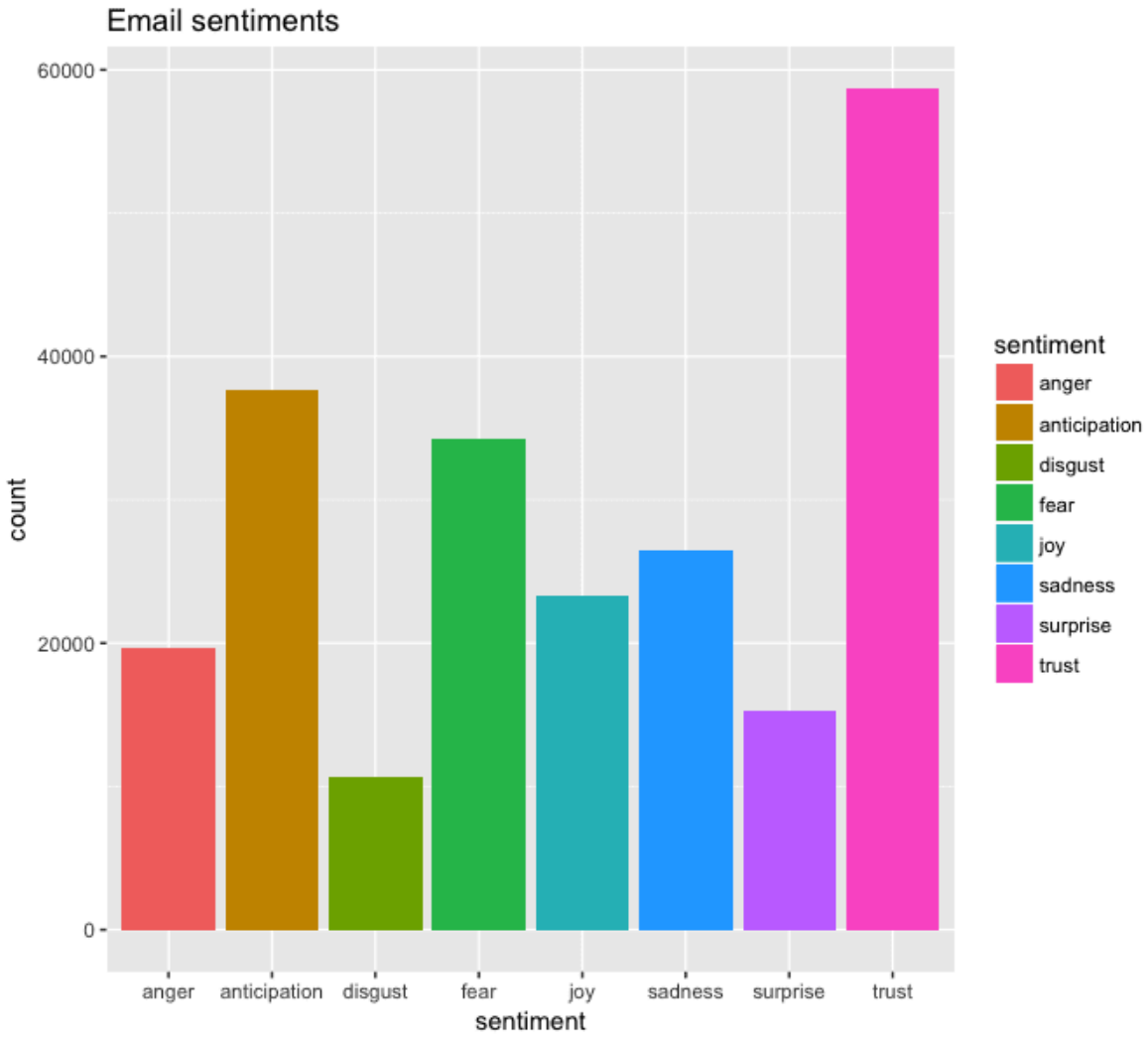
- Text is difficult
  - Unstructured
  - Dirty
  - Context matters

What is text analysis useful for?

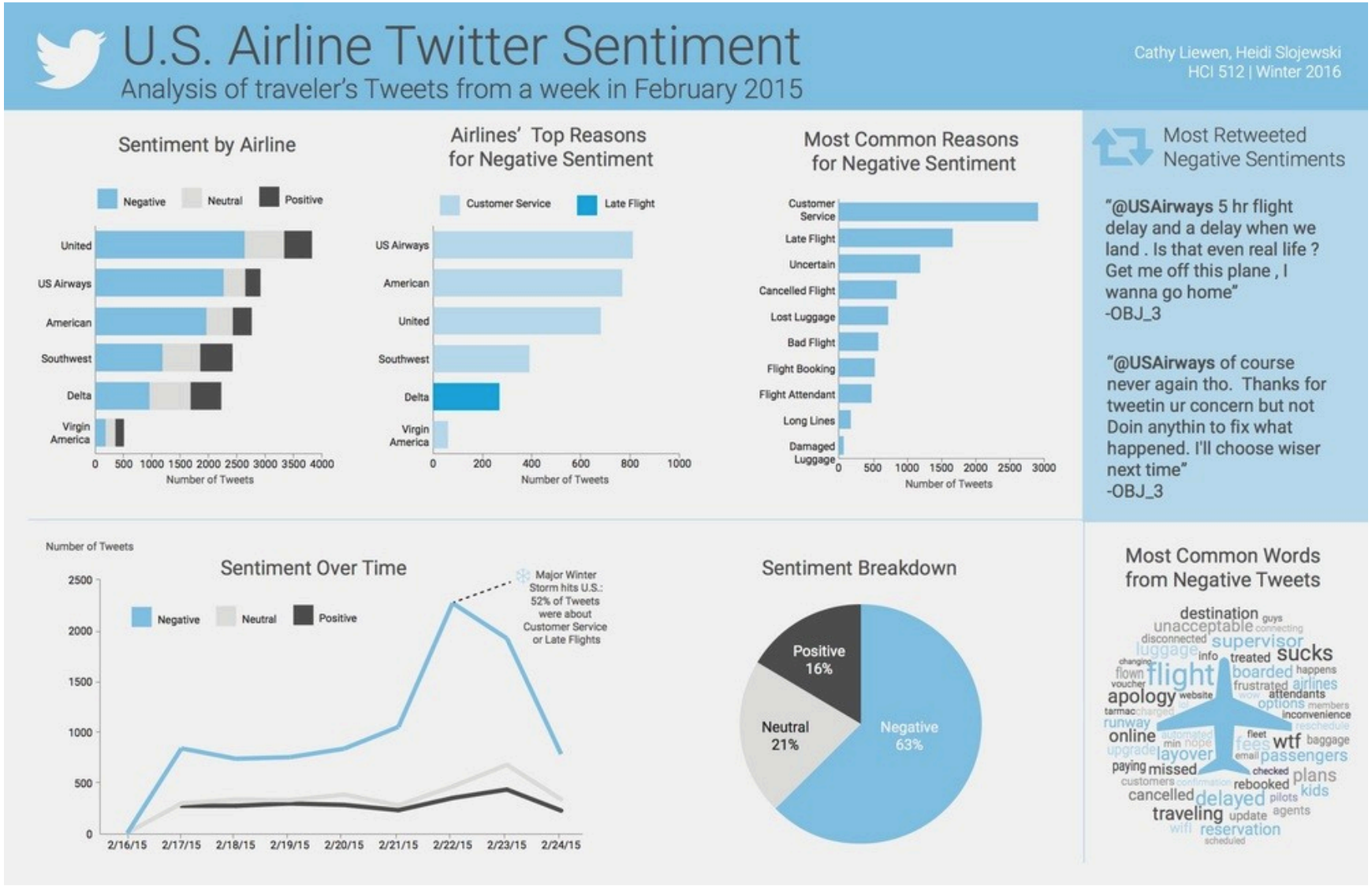
Spam filters …



Customer service



Social media analysis



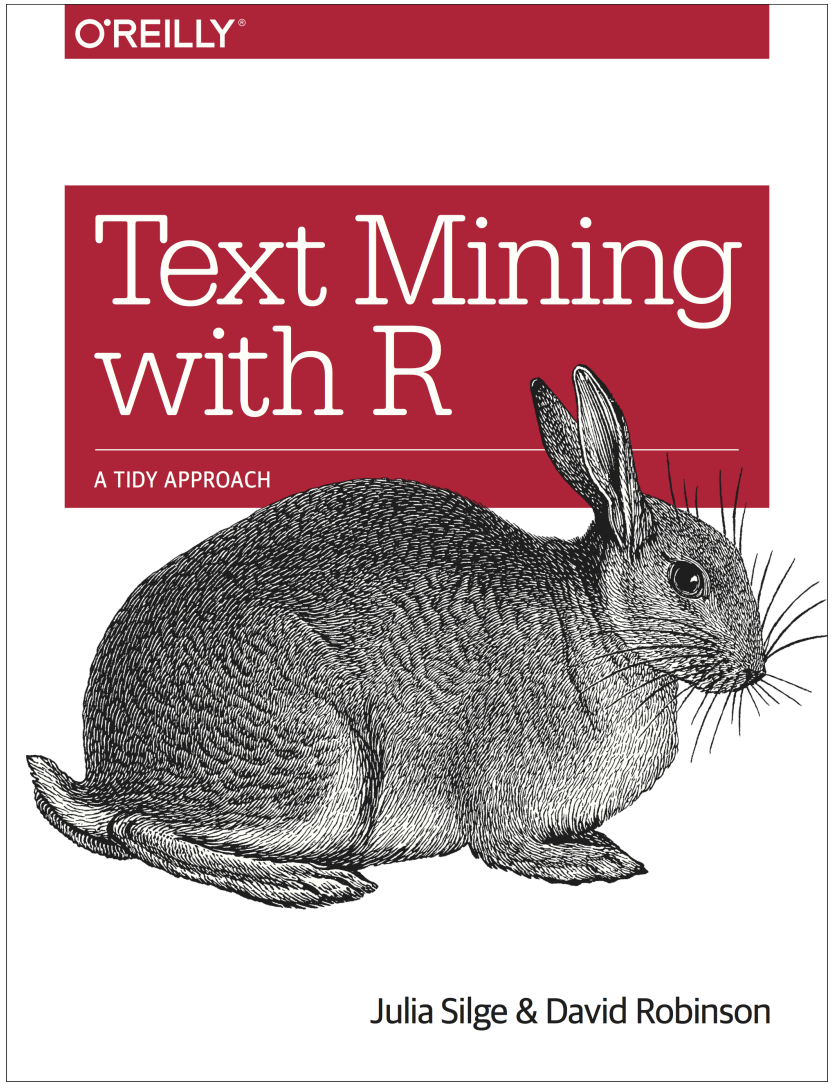
Some useful terms

- Document
  - One piece of text
  - Can be 1 sentence or 100 pages...
- Tokens or terms
  - Individual words (usually)
- Corpus
  - Collection of documents


TIDYTEXT

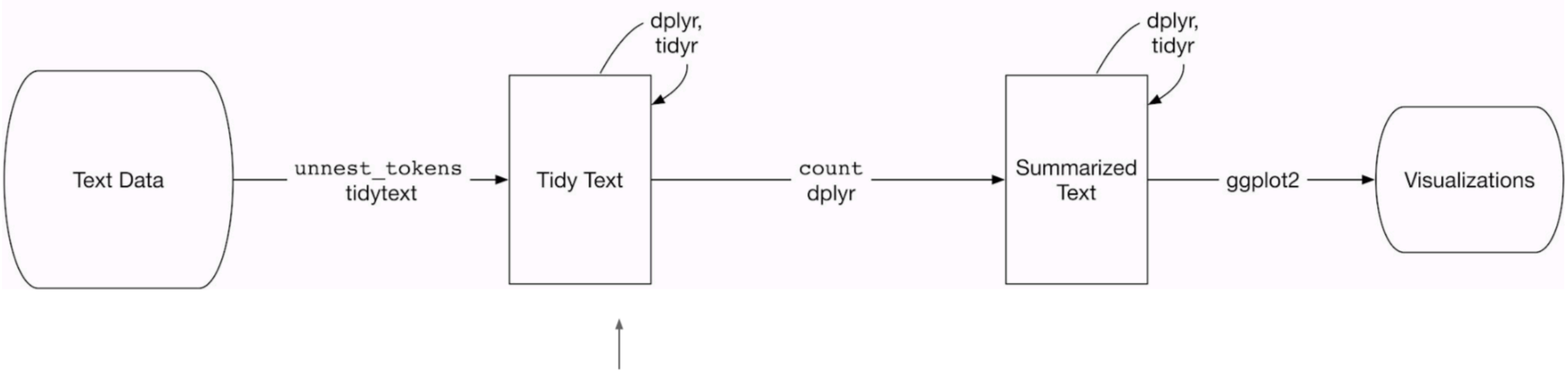
JULIA SILGE

BLOG ABOUT RESUME



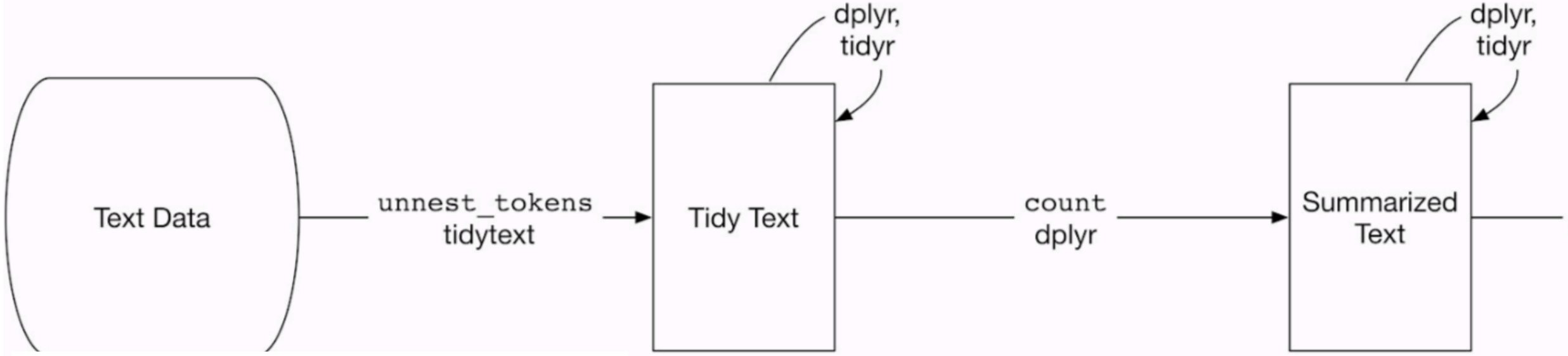
tidytextmining.com
@juliasilge
@juliasilge
juliasilge.com
@dataandme
@batpigandme
maraaverick.rbind.io

A ‘tidytext’ approach



One-token-per-document-per-row

Basically each token has its own line but can be linked to document




```
text <- c("Because I could not stop for Death -",
          "He kindly stopped for me -",
          "The Carriage held but just Ourselves -",
          "and Immortality")

text
```

```
## [1] "Because I could not stop for Death -"
## [2] "He kindly stopped for me -"
## [3] "The Carriage held but just Ourselves -"
## [4] "and Immortality"
```

```
library(tidyverse)
text_df <- data_frame(line = 1:4, text = text)

text_df
```

```
## # A tibble: 4 x 2
##    line text
##   <int> <chr>
## 1     1 Because I could not stop for Death -
## 2     2 He kindly stopped for me -
## 3     3 The Carriage held but just Ourselves -
## 4     4 and Immortality
```

```
library(tidytext)

text_df %>%
  unnest_tokens(word, text)
```

```
## # A tibble: 20 x 2
##     line word
##    <int> <chr>
##  1     1 because
##  2     1 i
##  3     1 could
##  4     1 not
##  5     1 stop
##  6     1 for
##  7     1 death
##  8     2 he
##  9     2 kindly
## 10     2 stopped
## 11     2 for
## 12     2 me
## 13     3 the
## 14     3 carriage
## 15     3 held
## 16     3 but
## 17     3 just
## 18     3 ourselves
## 19     4 and
## 20     4 immortality
```

Example

Time to tidy your text!

```
tidy_book <- full_text %>%
  mutate(line = row_number()) %>%
  unnest_tokens(word, text)

tidy_book
```

```
## # A tibble: 122,204 x 3
##    gutenberg_id  line word
##           <int> <int> <chr>
##  1         1342     1 pride
##  2         1342     1 and
##  3         1342     1 prejudice
##  4         1342     3 by
##  5         1342     3 jane
##  6         1342     3 austen
##  7         1342     7 chapter
##  8         1342     7 1
##  9         1342    10 it
## 10         1342    10 is
## # … with 122,194 more rows
```

What are the most common words?

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

```
## # A tibble: 6,538 x 2
##    word      n
##    <chr> <int>
##  1 the    4331
##  2 to     4162
##  3 of     3610
##  4 and    3585
##  5 her    2203
##  6 i      2065
##  7 a      1954
##  8 in     1880
##  9 was    1843
## 10 she    1695
## # … with 6,528 more rows
```

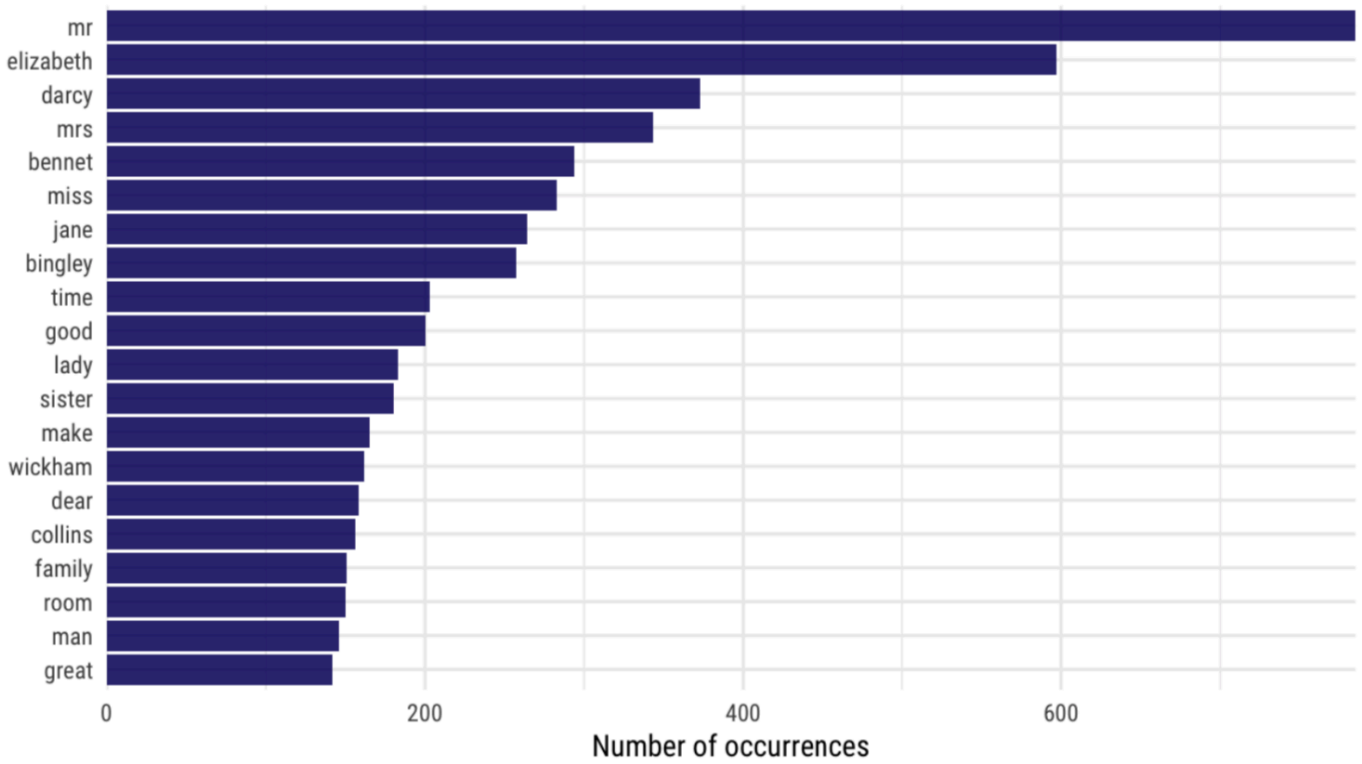
Stop words

```
get_stopwords()
```

```
## # A tibble: 175 x 2
##    word      lexicon
##    <chr>     <chr>
##  1 i         snowball
##  2 me        snowball
##  3 my        snowball
##  4 myself    snowball
##  5 we        snowball
##  6 our       snowball
##  7 ours      snowball
##  8 ourselves snowball
##  9 you       snowball
## 10 your      snowball
## # … with 165 more rows
```

What are the most common words?

```
tidy_book %>%
  anti_join(get_stopwords(source = "smart")) %>%
  count(word, sort = TRUE) %>%
  top_n(20) %>%
  ggplot(aes(fct_reorder(word, n), n)) +
  geom_col() +
  coord_flip()
```

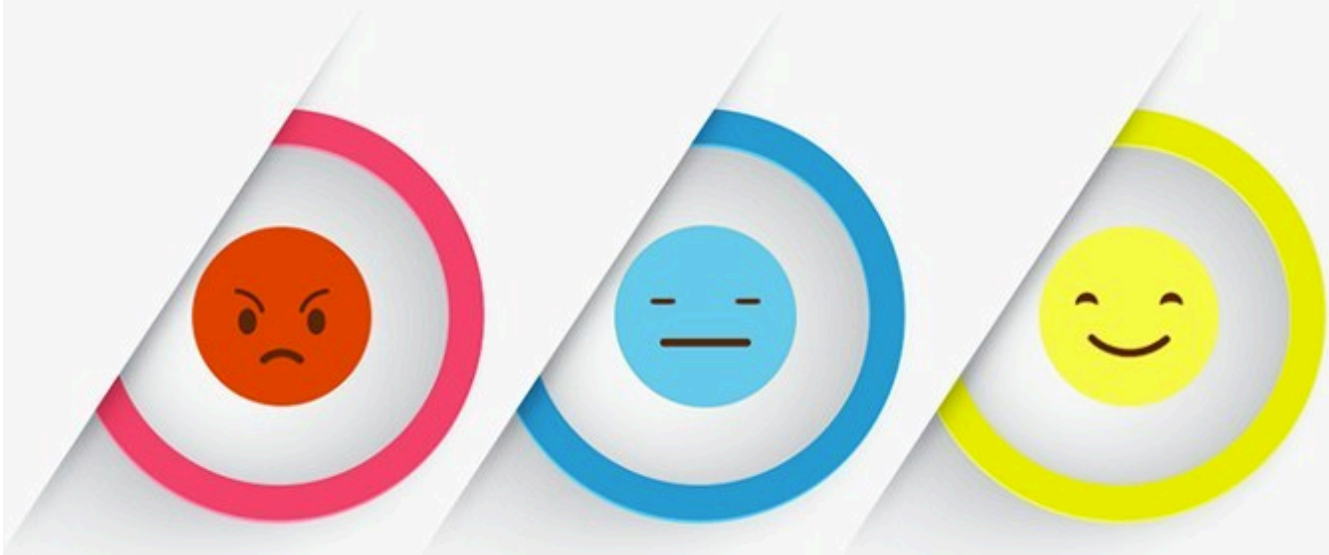


Stop words

```
get_stopwords(source = "smart")
```

```
## # A tibble: 571 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
##  7 accordingly smart
##  8 across     smart
##  9 actually   smart
## 10 after      smart
## # … with 561 more rows
```

SENTIMENT ANALYSIS



NEGATIVE

Totally dissatisfied with the service. Worst customer care ever.

NEUTRAL

Good Job but I will expect a lot more in future.

POSITIVE

Brilliant effort guys! Loved Your Work.

Sentiment lexicons

```
get_sentiments("bing")
```

```
## # A tibble: 6,788 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faced     negative
##  2 2-faces     negative
##  3 a+          positive
##  4 abnormal    negative
##  5 abolish     negative
##  6 abominable  negative
##  7 abominably  negative
##  8 abominate   negative
##  9 abomination negative
## 10 abort       negative
## # … with 6,778 more rows
```

```
get_sentiments("afinn")
```

```
## # A tibble: 2,476 x 2
##    word       score
##    <chr>      <int>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # … with 2,466 more rows
```

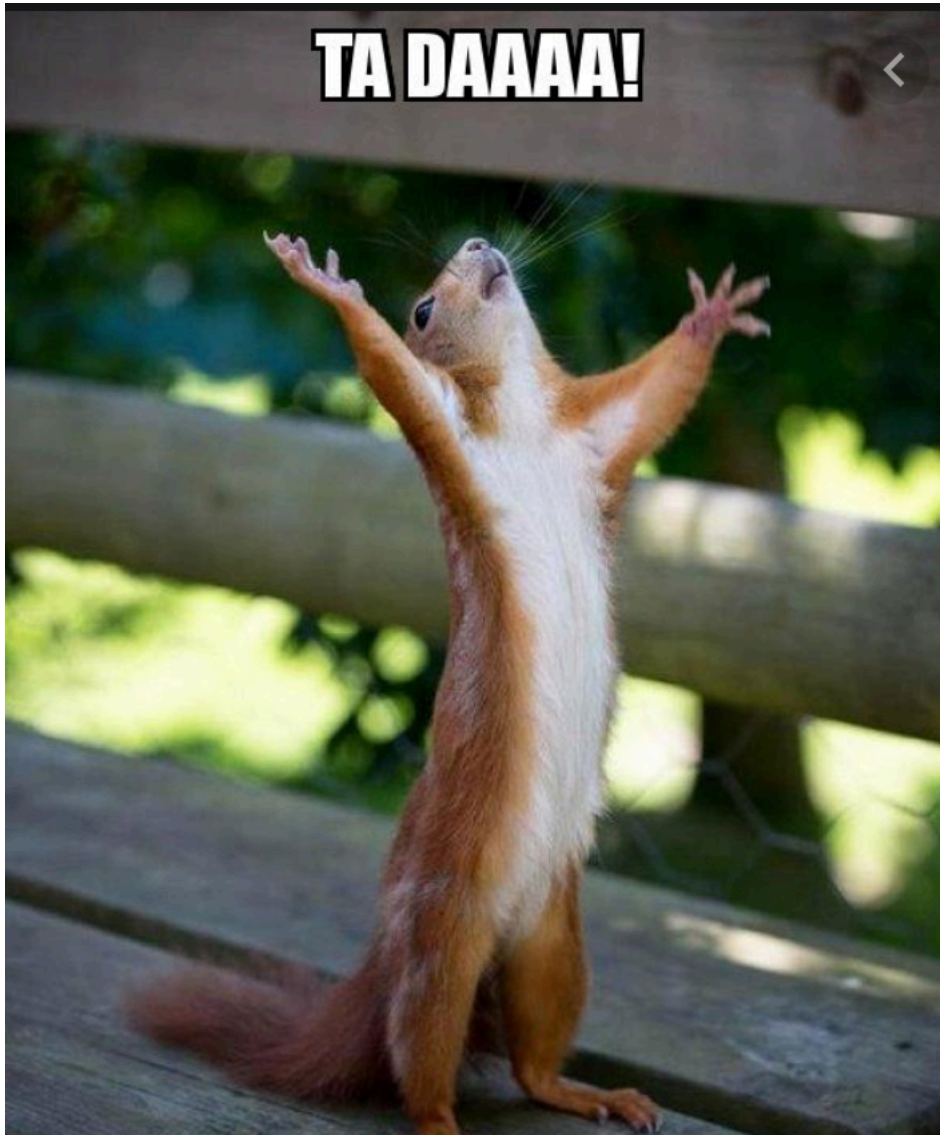
```
get_sentiments("nrc")
```

```
## # 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
```

So how do we do sentiment analysis

```
tidy_book %>%
  inner_join(get_sentiments("bing")) %>%
  count(sentiment, sort = TRUE)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 positive   5052
## 2 negative   3652
```



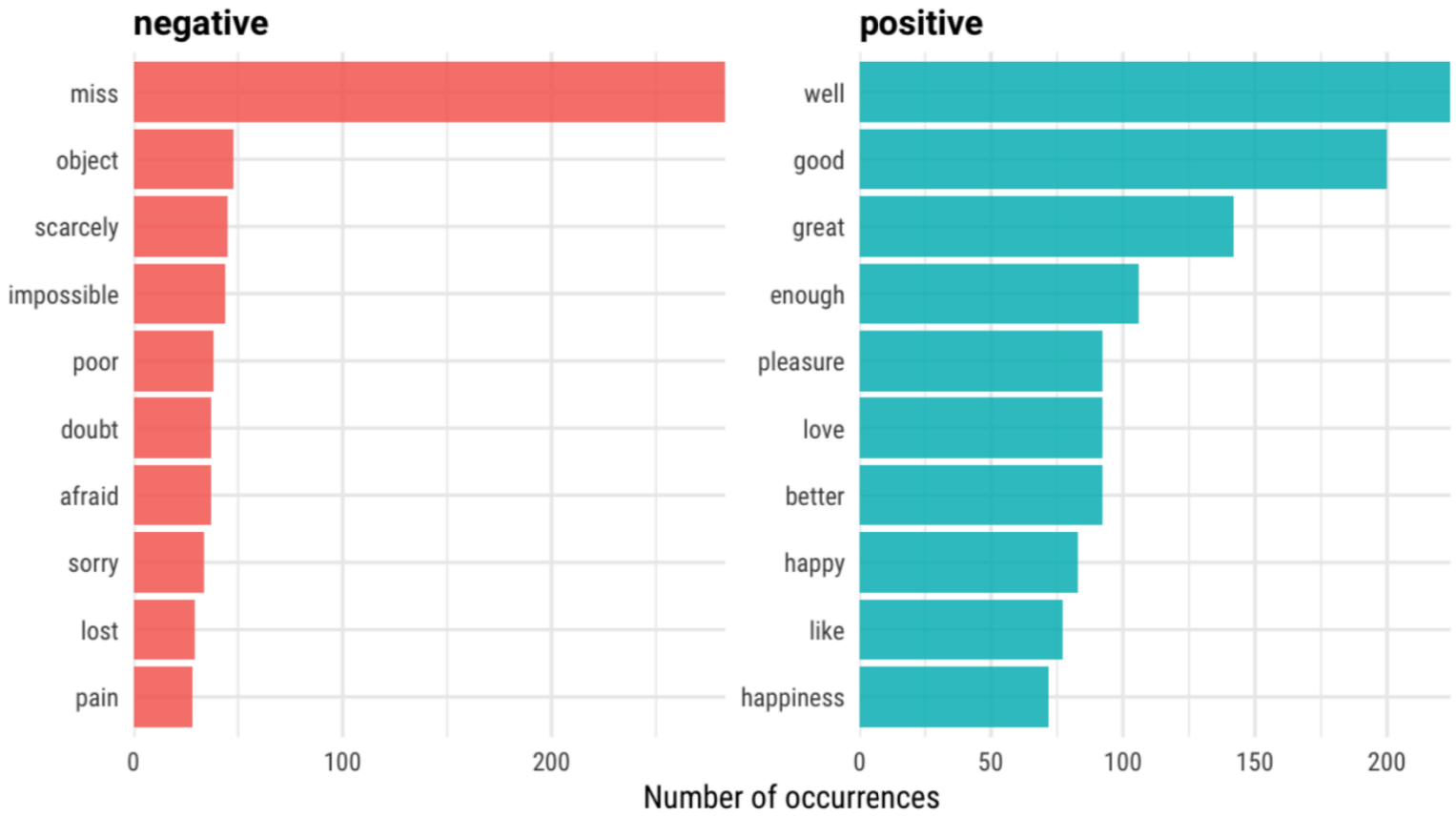
So how do we do sentiment analysis

```
tidy_book %>%
  inner_join(get_sentiments("bing")) %>%
  count(sentiment, sort = TRUE)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 positive   5052
## 2 negative   3652
```

```
tidy_book %>%
  inner_join(get_sentiments("bing")) %>%
  count(sentiment, word, sort = TRUE)
```

```
## # A tibble: 1,430 x 3
##    sentiment word          n
##    <chr>     <chr>     <int>
##  1 negative  miss        283
##  2 positive  well        224
##  3 positive  good        200
##  4 positive  great       142
##  5 positive  enough      106
##  6 positive  better       92
##  7 positive  love         92
##  8 positive  pleasure     92
##  9 positive  happy        83
## 10 positive  like         77
## # … with 1,420 more rows
```




```
tidy_book %>%
  inner_join(get_sentiments("bing")) %>%
  count(sentiment, word, sort = TRUE) %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup %>%
  ggplot(aes(fct_reorder(word, n),
             n,
             fill = sentiment)) +
  geom_col() +
  coord_flip() +
  facet_wrap(~ sentiment, scales = "free")
```

N-grams

unigram

bigram

trigram

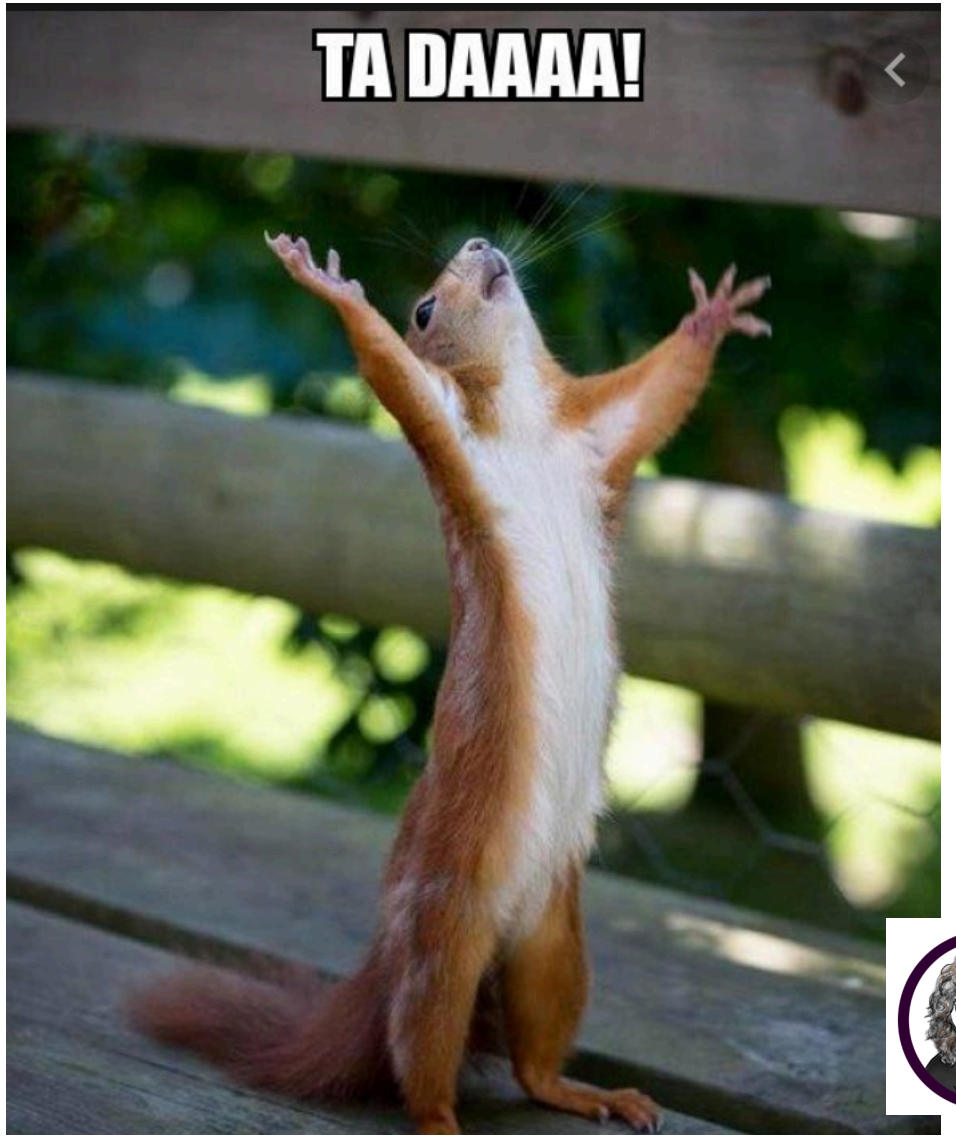
n-gram (n = 4)

So how do we do n-grams

```
tidy_ngram <- full_text %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

tidy_ngram
```

```
## # A tibble: 122,203 x 2
##    gutenberg_id bigram
##           <int> <chr>
##  1         1342 pride and
##  2         1342 and prejudice
##  3         1342 prejudice by
##  4         1342 by jane
##  5         1342 jane austen
##  6         1342 austen chapter
##  7         1342 chapter 1
##  8         1342 1 it
##  9         1342 it is
## 10         1342 is a
## # … with 122,193 more rows
```



So how do we do n-grams

```
tidy_ngram <- full_text %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

tidy_ngram
```

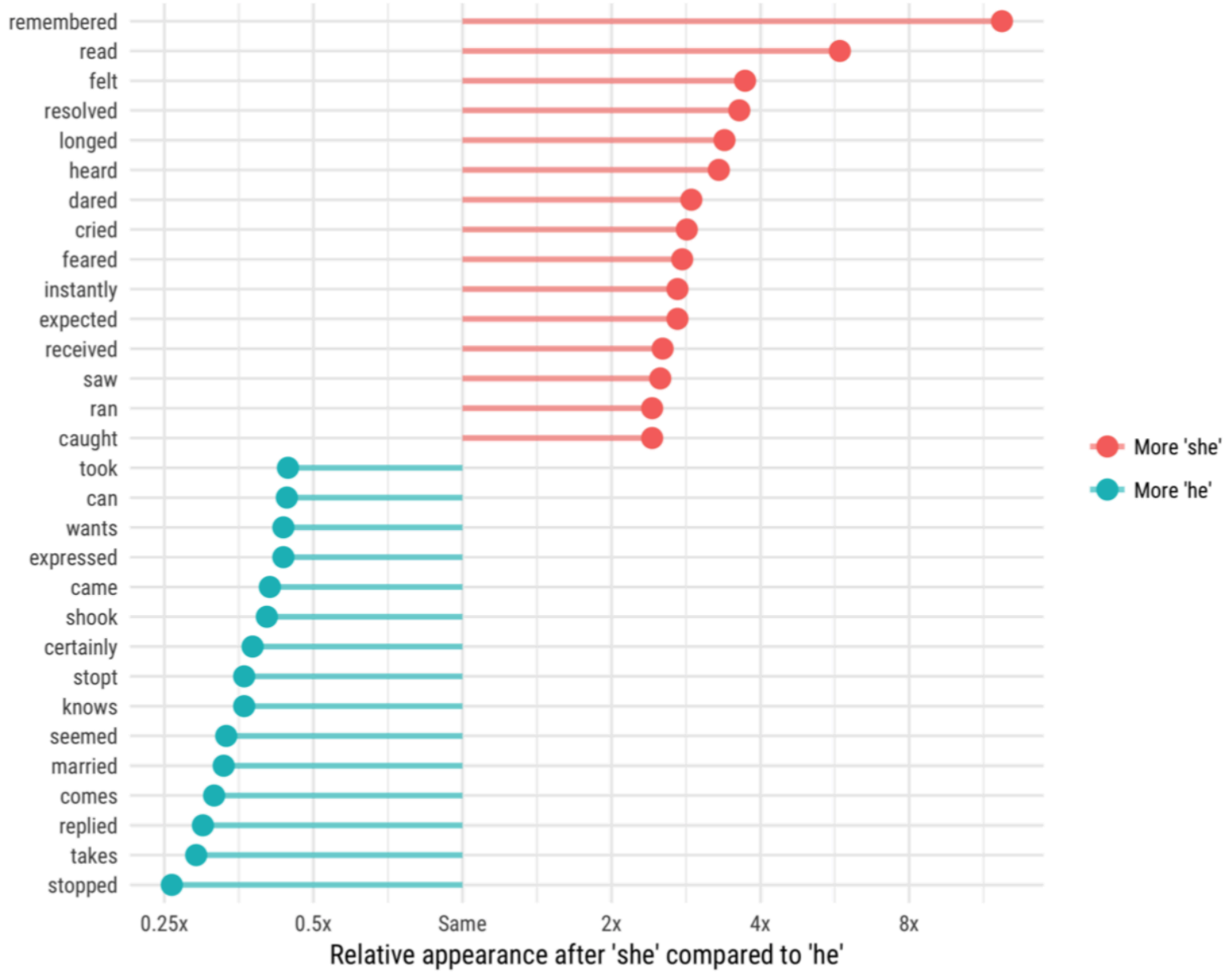
```
## # A tibble: 122,203 x 2
##    gutenberg_id bigram
##           <int> <chr>
##  1         1342 pride and
##  2         1342 and prejudice
##  3         1342 prejudice by
##  4         1342 by jane
##  5         1342 jane austen
##  6         1342 austen chapter
##  7         1342 chapter 1
##  8         1342 1 it
##  9         1342 it is
## 10         1342 is a
## # … with 122,193 more rows
```

```
tidy_ngram %>%
  count(bigram, sort = TRUE)
```

```
## # A tibble: 54,998 x 2
##    bigram       n
##    <chr>    <int>
##  1 of the     464
##  2 to be      443
##  3 in the     382
##  4 i am       302
##  5 of her     260
##  6 to the     252
##  7 it was     251
##  8 mr darcy   243
##  9 of his     234
## 10 she was    209
## # … with 54,988 more rows
```

Words paired with 'he' and 'she' in Jane Austen's novels

Women remember, read, and feel while men stop, take, and reply



1) Twitter and sentiment analysis

Some help if needed:

a) Scrape and analyse tweets on a topic of your choice (or a tweeter) using the format in:

https://cfss.uchicago.edu/notes/twitter-api-practice/

b) Perform some sentiment analysis on those tweets with the help of:
https://tabvizexplorer.com/sentiment-analysis-using-r-and-twitter/

(another fun example here: https://hackernoon.com/text-processing-and-sentiment-analysis-of-twitter-data-22ff5e51e14c)

2) Some fun text analysis if it interests you …

- Some fun stuff here - https://juliasilge.com/blog/gender-pronouns/
- Go through this tutorial: https://juliasilge.com/blog/tidy-text-classification/