Text

Week 11

AEM 2850: R for Business Analytics

Cornell Dyson

Spring 2022

Acknowledgements: Andrew Heiss

Announcements

I hope you had a nice spring break!

Reminder: Cornell still requires masks in classrooms as of now

Mini project 1 grades will be posted soon

- Peer-review survey forthcoming

Mini project 2 details will be released in the next 0-1 weeks

Questions before we get started?

Plan for today

Course progress

Prologue

Working with strings in R

Text mining with R

Course progress

Course objectives reminder

1. Develop basic proficiency in R programming
2. Understand data structures and manipulation
3. Describe effective techniques for data visualization and communication
4. Construct effective data visualizations
5. Utilize course concepts and tools for business applications

Where we've been (weeks 1-4)

1. **Develop basic proficiency in R programming**
2. **Understand data structures and manipulation**
3. Describe effective techniques for data visualization and communication
4. Construct effective data visualizations
5. Utilize course concepts and tools for business applications

Where we've been (weeks 5-10)

1. Develop basic proficiency in R programming
2. Understand data structures and manipulation
3. **Describe effective techniques for data visualization and communication**
4. **Construct effective data visualizations**
5. **Utilize course concepts and tools for business applications**

Where we're going next (weeks 11+)

1. Develop basic proficiency in R programming
2. Understand data structures and manipulation
3. Describe effective techniques for data visualization and communication
4. Construct effective data visualizations
5. Utilize course concepts and tools for business applications

All of the above, plus special topics! Tentative plan:

- Week 11: Text
- Week 12: Functions and iteration
- Week 13: Prediction
- Week 14: Causal inference
- Week 15: Wrap up

Schedule overview

Weeks 1-4: Programming Foundations

Weeks 5-10: Data Visualization Foundations

Weeks 11+: Special Topics (mix of programming and dataviz)

See aem2850.toddgerarden.com/schedule for details

Prologue

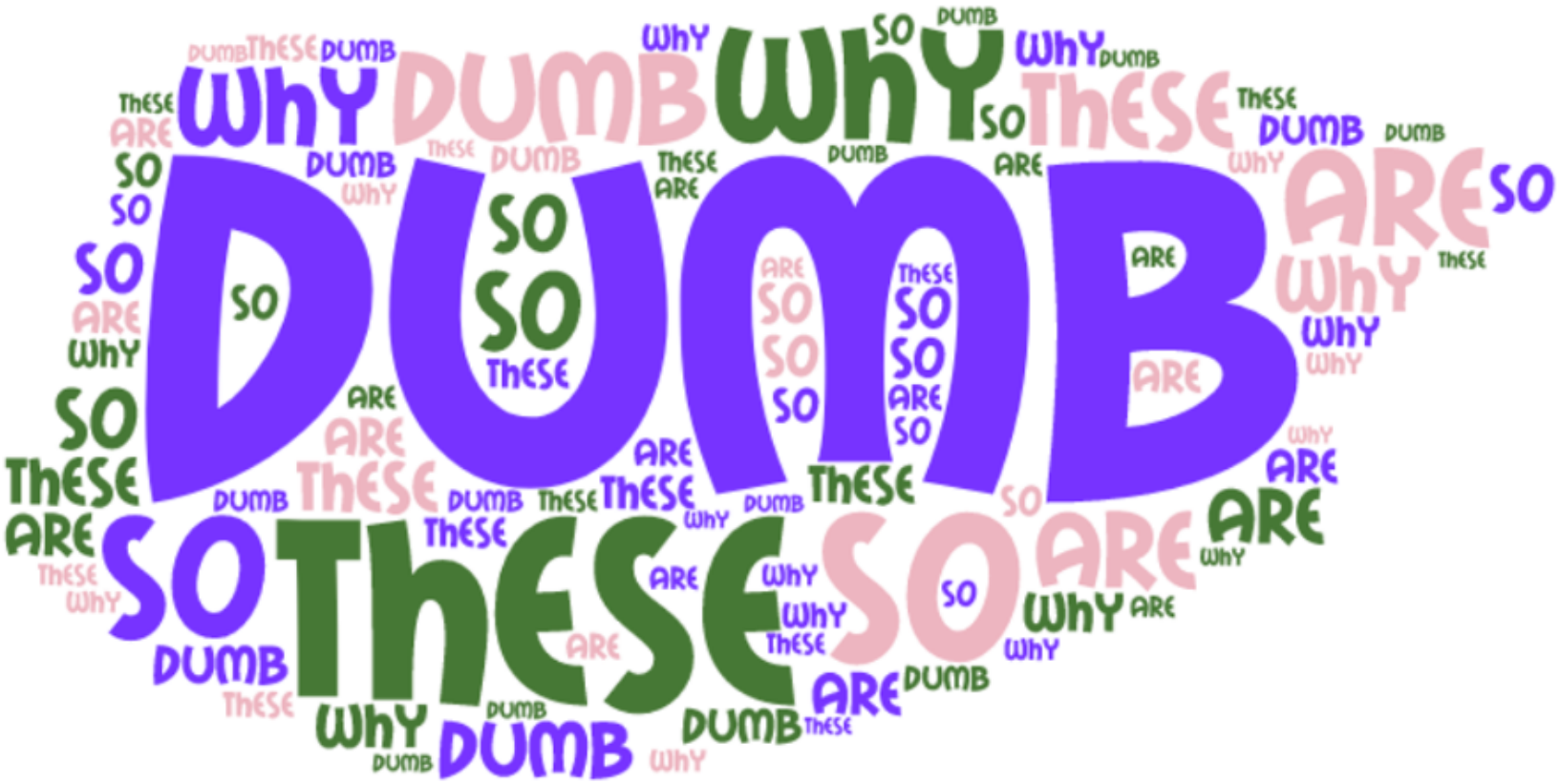
Text comes in many forms

The schrute package contains transcripts of all episodes of The Office (US)

```
library(schrute)
theoffice # this is an object from the schrute package
```

```
## # A tibble: 55,130 × 12
##    index season episode episode_name director   writer    character text  text_w_direction
##    <int>  <int>   <int> <chr>        <chr>      <chr>     <chr>     <chr> <chr>
##  1     1      1       1 Pilot        Ken Kwapis Ricky Ge… Michael   All … All right Jim. …
##  2     2      1       1 Pilot        Ken Kwapis Ricky Ge… Jim       Oh, … Oh, I told you.…
##  3     3      1       1 Pilot        Ken Kwapis Ricky Ge… Michael   So y… So you've come …
##  4     4      1       1 Pilot        Ken Kwapis Ricky Ge… Jim       Actu… Actually, you c…
##  5     5      1       1 Pilot        Ken Kwapis Ricky Ge… Michael   All … All right. Well…
##  6     6      1       1 Pilot        Ken Kwapis Ricky Ge… Michael   Yes,… [on the phone] …
##  7     7      1       1 Pilot        Ken Kwapis Ricky Ge… Michael   I've… I've, uh, I've …
##  8     8      1       1 Pilot        Ken Kwapis Ricky Ge… Pam       Well… Well. I don't k…
##  9     9      1       1 Pilot        Ken Kwapis Ricky Ge… Michael   If y… If you think sh…
## 10    10      1       1 Pilot        Ken Kwapis Ricky Ge… Pam       What? What?
## # … with 55,120 more rows, and 3 more variables: imdb_rating <dbl>, total_votes <int>,
## #   air_date <fct>
```

Text can be analyzed in many ways



Take word clouds, the pie chart of text

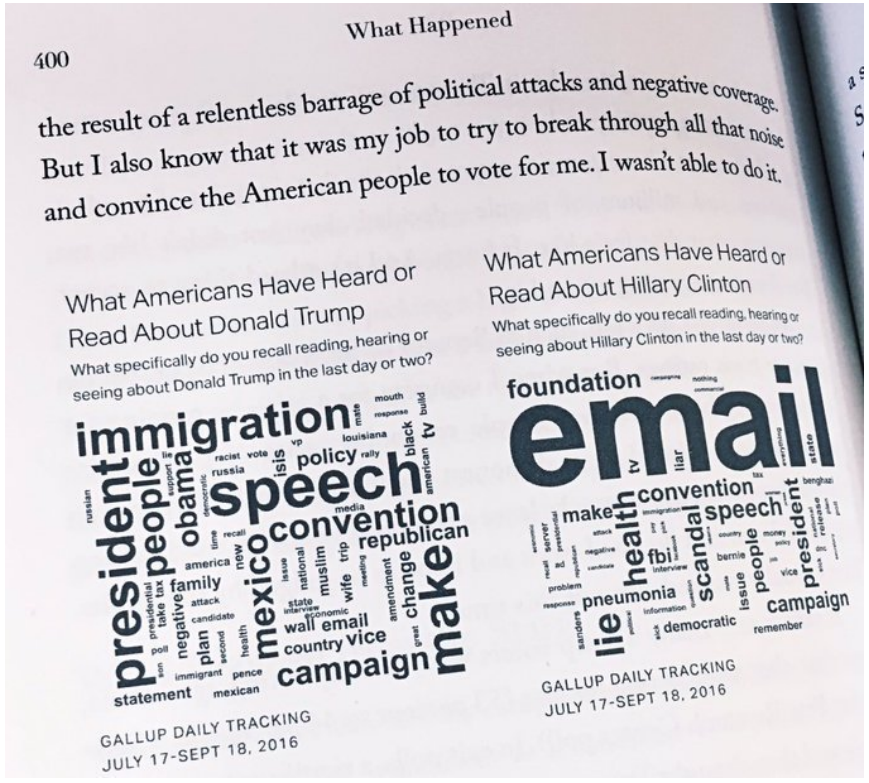
Why are word clouds bad?

- Poor grammar (of graphics)
  - Usually the only aesthetic is size
  - Color, position, etc. contain no content
- Raw word frequency is not always informative

Why are word clouds good?

- Can visualize one-word descriptions
- Can highlight a single dominant word or phrase
- Can make before/after comparisons

Some cases are okay

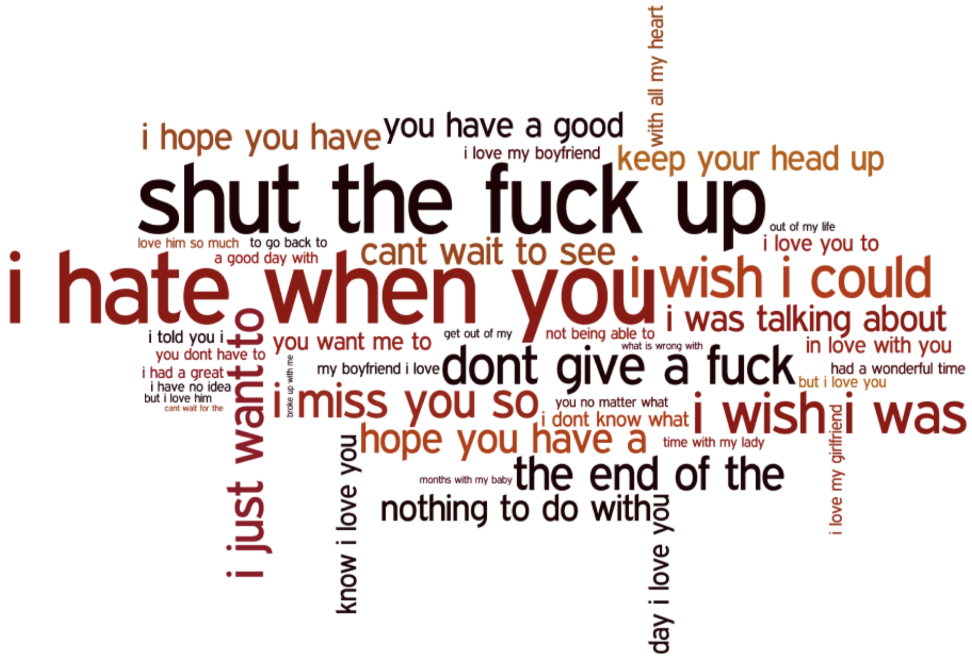


Trump word cloud is uninformative

Clinton word cloud is okay

- Highlights email as the **single** dominant narrative about Hillary Clinton prior to the 2016 election

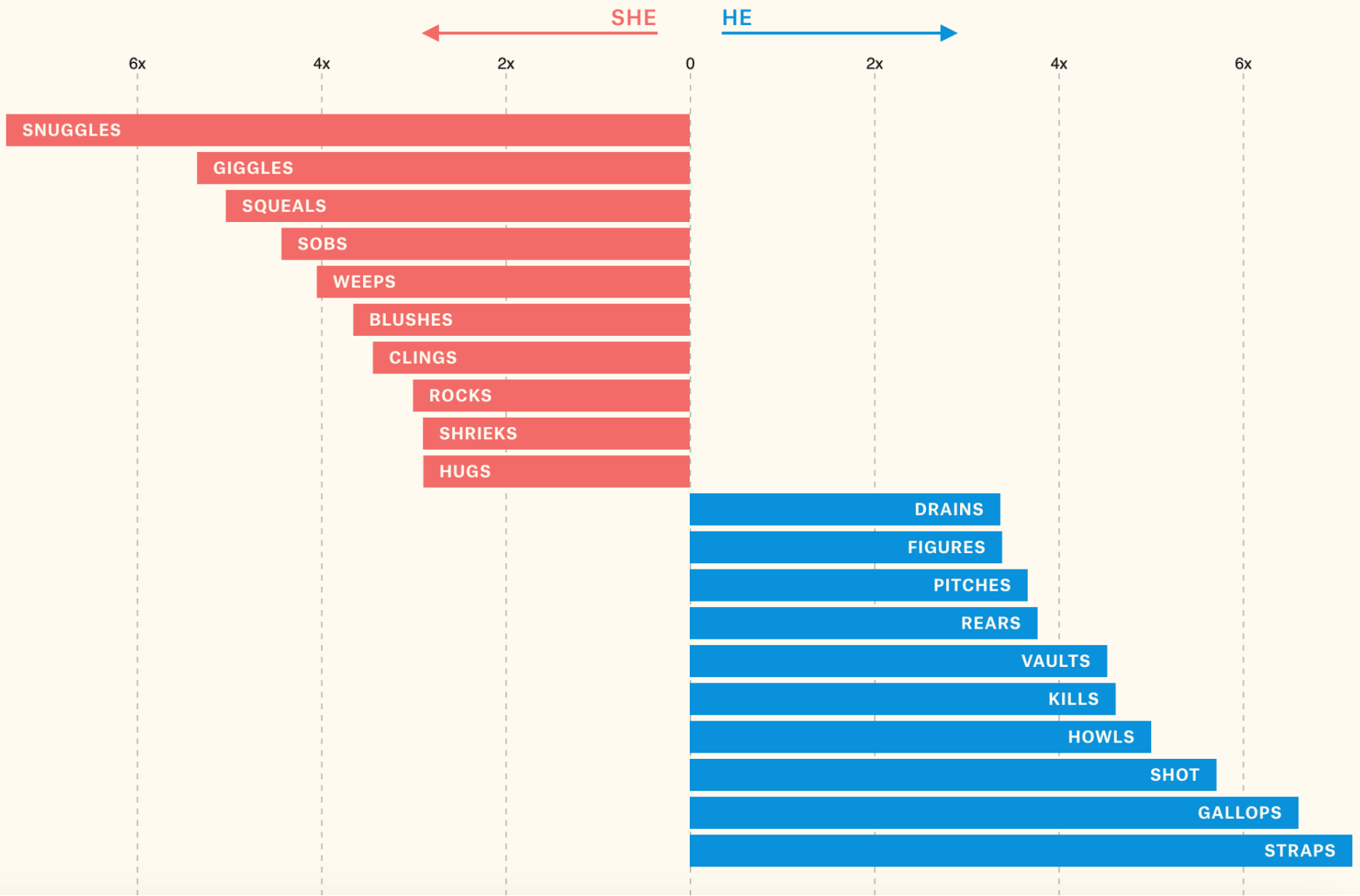
Twitter before and after breakups (4-grams)



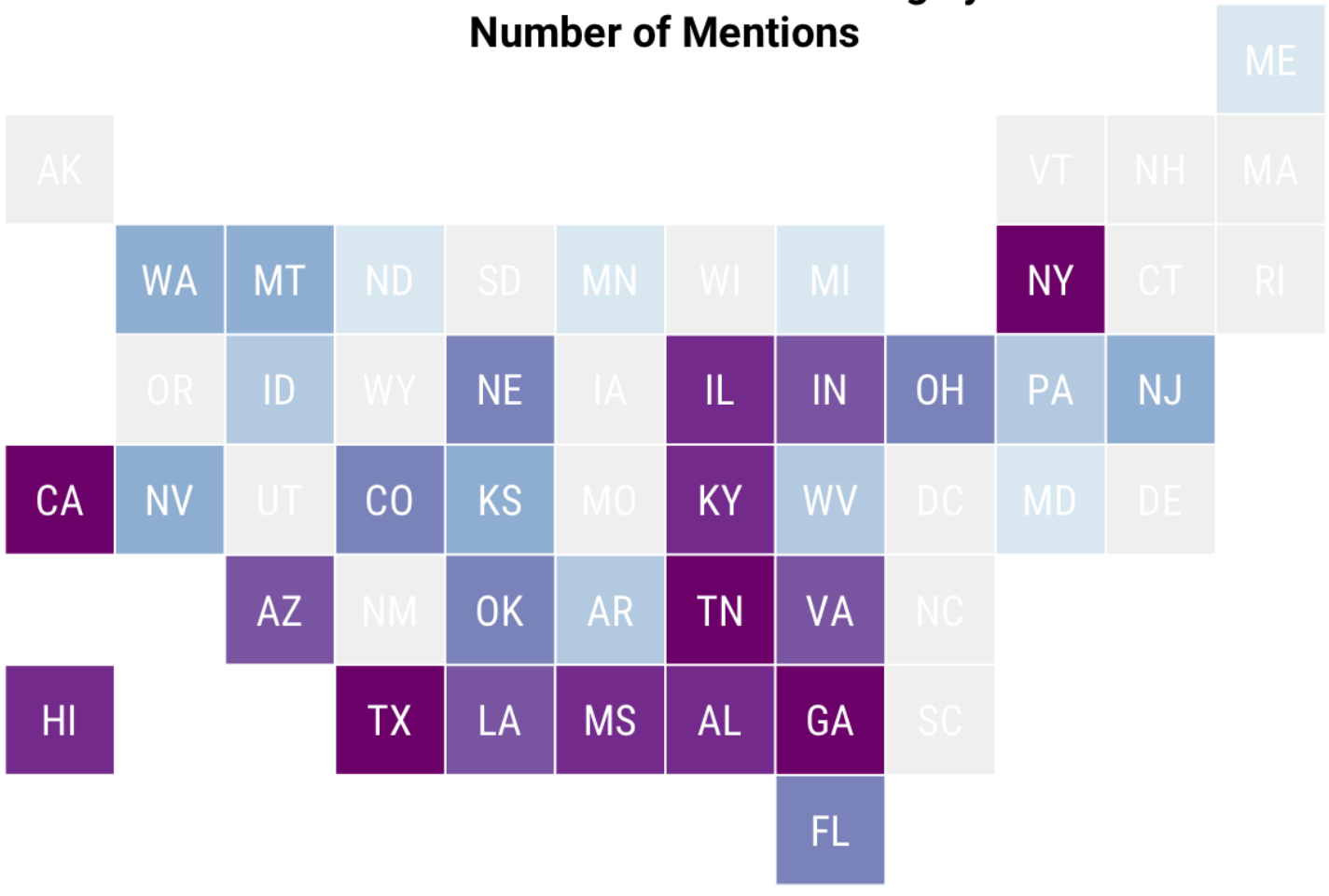
Better yet: use other methods to analyze and visualize text

The most used words for women vs. men

Likelihood that certain words appear after “she” vs. “he” in screen direction.



What States Are Mentioned in Song Lyrics?
Number of Mentions



Working with strings in R

Strings are nothing new

```
nycflights13::flights %>%
  select(carrier, tailnum, origin, dest)
```

```
## # A tibble: 336,776 × 4
##    carrier tailnum origin dest
##    <chr>   <chr>   <chr>  <chr>
##  1 UA      N14228  EWR    IAH
##  2 UA      N24211  LGA    IAH
##  3 AA      N619AA  JFK    MIA
##  4 B6      N804JB  JFK    BQN
##  5 DL      N668DN  LGA    ATL
##  6 UA      N39463  EWR    ORD
##  7 B6      N516JB  EWR    FLL
##  8 EV      N829AS  LGA    IAD
##  9 B6      N593JB  JFK    MCO
## 10 AA      N3ALAA  LGA    ORD
## # … with 336,766 more rows
```

```
read_csv("data/our-companies.csv") %>%
  select(name)
```

```
## # A tibble: 22 × 1
##    name
##    <chr>
##  1 Allbirds Inc
##  2 Alphabet Inc. Class A
##  3 Anheuser Busch Inbev SA
##  4 Apple Inc.
##  5 Berkshire Hathaway Inc. Class B
##  6 Bumble Inc
##  7 Capri Holdings Ltd
##  8 Costco Wholesale Corporation
##  9 Electronic Arts Inc.
## 10 Levi Strauss & Co.
## # … with 12 more rows
```

Strings in R

Strings are also referred to as "characters" (abbreviated `chr`)

Strings can be stored in many ways:

- Vectors
- Data frame columns
- Elements in a list

So far we have used them as we would any other data

But sometimes we'll want to filter on, modify, or analyze strings

The stringr package

`stringr` is loaded as part of the core tidyverse

All `stringr` functions have intuitive names that start with `str_`

We will cover a few handy functions:

1. `str_length`
2. `str_trim`
3. `str_detect`

See `vignette("stringr")` for more

1) stringr::str_length

`str_length` tells you the number of characters in a string

```
str_length("supercalifragilisticexpialidocious")
```

```
## [1] 34
```

```
theoffice %>%
  distinct(character) %>%
  slice_head(n = 5) %>%
  mutate(name_length = str_length(character))
```

```
## # A tibble: 5 × 2
##   character name_length
##   <chr>           <int>
## 1 Michael             7
## 2 Jim                 3
## 3 Pam                 3
## 4 Dwight              6
## 5 Jan                 3
```

2) stringr::str_trim

`str_trim` removes leading/trailing whitespace

```
str_trim("I went to Cornell, you ever heard of it?            ")
```

```
## [1] "I went to Cornell, you ever heard of it?"
```

```
str_trim("            I went to Cornell, you ever heard of it?  ")
```

```
## [1] "I went to Cornell, you ever heard of it?"
```

3) stringr::str_detect

`str_detect` can be used to match patterns

```
first_4_characters <- theoffice %>%
  distinct(character) %>%
  slice_head(n = 4) %>%
  pull(character)
first_4_characters
```

```
## [1] "Michael" "Jim"     "Pam"     "Dwight"
```

```
str_detect(first_4_characters, "Dwight")
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```
str_detect(first_4_characters, "a")
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

3) stringr::str_detect

`str_detect` is a powerful way to `filter` a data frame

```
theoffice %>% select(season, episode, character, text) %>%
  filter(str_detect(text,    # where to match a pattern
                    "sale")) # what pattern to match
```

```
## # A tibble: 369 × 4
##    season episode character text
##     <int>   <int> <chr>     <chr>
##  1      1       2 Jim       This is my biggest sale of the year. They love me over there …
##  2      1       2 Jim       Mr. Decker, we didn't lose your sale today, did we? Excellent…
##  3      1       3 Jim       That is a great offer. Thank you. I really think I should be …
##  4      1       3 Jan       From sales?
##  5      1       4 Michael   Look, look, look. I talked to corporate, about protecting the…
##  6      1       5 Michael   All right, time, time out. Come on, sales, over here. Bring i…
##  7      1       6 Jan       Alan and I have created an incentive program to increase sale…
##  8      1       6 Jan       We've created an incentive program to increase sales.
##  9      1       6 Jim       Plus you have so much more to talk to this girl about, You're…
## 10      1       6 Stanley   I thought that was the incentive prize for the top salesperso…
## # … with 359 more rows
```

3) stringr::str_detect

`str_detect` is case-sensitive by default

```
theoffice %>% select(season, episode, character, text) %>%
  filter(str_detect(text,
                    "Sale")) # sale and Sale produce different output
```

```
## # A tibble: 28 × 4
##    season episode character          text
##     <int>   <int> <chr>              <chr>
##  1      2      11 Michael            No, no. Salesmen and profit centers.
##  2      2      14 Michael            Old fashioned raid. Sales on Accounting. Yeah. Follo…
##  3      2      14 Michael and Dwight Ahhhh! Whoo hoo! Come on, come on, come on, come on!…
##  4      2      14 Michael            Oh, and I'm not? Why would you say that? Because I'm…
##  5      2      17 Jim                Dwight was the top salesman of the year at our compa…
##  6      2      17 Michael            Speaker at the Sales Convention. Been there, done th…
##  7      2      17 Dwight             Saleswoman has a v*g1n*.
##  8      2      17 Speaker            Next, I'd like to introduce the Dunder Mifflin Sales…
##  9      2      17 Dwight             Salesman of Northeastern Pennsylvania, I ask you onc…
## 10      3       5 Angela             Sales take a long time.
## # … with 18 more rows
```

3) stringr::str_detect

You could use multiple calls to `str_detect`, or pass multiple arguments:

```
theoffice %>% select(season, episode, character, text) %>%
  filter(str_detect(text,
                    "sale|Sale")) # look for sale OR Sale
```

```
## # A tibble: 391 × 4
##    season episode character text
##     <int>   <int> <chr>     <chr>
##  1      1       2 Jim       This is my biggest sale of the year. They love me over there …
##  2      1       2 Jim       Mr. Decker, we didn't lose your sale today, did we? Excellent…
##  3      1       3 Jim       That is a great offer. Thank you. I really think I should be …
##  4      1       3 Jan       From sales?
##  5      1       4 Michael   Look, look, look. I talked to corporate, about protecting the…
##  6      1       5 Michael   All right, time, time out. Come on, sales, over here. Bring i…
##  7      1       6 Jan       Alan and I have created an incentive program to increase sale…
##  8      1       6 Jan       We've created an incentive program to increase sales.
##  9      1       6 Jim       Plus you have so much more to talk to this girl about, You're…
## 10      1       6 Stanley   I thought that was the incentive prize for the top salesperso…
## # … with 381 more rows
```

3) stringr::str_detect

You could consolidate this -- regex parentheses are like in math

```
theoffice %>% select(season, episode, character, text) %>%
  filter(str_detect(text,
                    "(s|S)ale")) # look for sale OR Sale
```

```
## # A tibble: 391 × 4
##    season episode character text
##     <int>   <int> <chr>     <chr>
##  1      1       2 Jim       This is my biggest sale of the year. They love me over there …
##  2      1       2 Jim       Mr. Decker, we didn't lose your sale today, did we? Excellent…
##  3      1       3 Jim       That is a great offer. Thank you. I really think I should be …
##  4      1       3 Jan       From sales?
##  5      1       4 Michael   Look, look, look. I talked to corporate, about protecting the…
##  6      1       5 Michael   All right, time, time out. Come on, sales, over here. Bring i…
##  7      1       6 Jan       Alan and I have created an incentive program to increase sale…
##  8      1       6 Jan       We've created an incentive program to increase sales.
##  9      1       6 Jim       Plus you have so much more to talk to this girl about, You're…
## 10      1       6 Stanley   I thought that was the incentive prize for the top salesperso…
## # … with 381 more rows
```

3) stringr::str_detect

You could `regex` to ignore all cases and control other pattern matching details

```
theoffice %>% select(season, episode, character, text) %>%
  filter(str_detect(text,
                    regex("Sale", ignore_case = TRUE))) # ignore case
```

```
## # A tibble: 392 × 4
##    season episode character text
##     <int>   <int> <chr>     <chr>
##  1      1       2 Jim       This is my biggest sale of the year. They love me over there …
##  2      1       2 Jim       Mr. Decker, we didn't lose your sale today, did we? Excellent…
##  3      1       3 Jim       That is a great offer. Thank you. I really think I should be …
##  4      1       3 Jan       From sales?
##  5      1       4 Michael   Look, look, look. I talked to corporate, about protecting the…
##  6      1       5 Michael   All right, time, time out. Come on, sales, over here. Bring i…
##  7      1       6 Jan       Alan and I have created an incentive program to increase sale…
##  8      1       6 Jan       We've created an incentive program to increase sales.
##  9      1       6 Jim       Plus you have so much more to talk to this girl about, You're…
## 10      1       6 Stanley   I thought that was the incentive prize for the top salesperso…
## # … with 382 more rows
```

3) stringr::str_detect

When I say ignore all cases, I mean IGNORE ALL CASES!

```
theoffice %>% select(season, episode, character, text) %>%
  filter(str_detect(text,
                    regex("sale", ignore_case = TRUE))) %>%
  filter(!str_detect(text, "(s|S)ale")) # find non-standard form(s)
```

```
## # A tibble: 1 × 4
##   season episode character text
##    <int>   <int> <chr>     <chr>
## 1      3       3 Dwight    I HAVE EXCELLENT SALES NUMBERS!
```

3) stringr::str_detect

`str_detect` can be combined with familiar functions to summarize data

```
theoffice %>%
  filter(str_detect(text, regex("Sale", ignore_case = TRUE))) %>%
  count(character, sort = TRUE)
```

```
## # A tibble: 46 × 2
##    character     n
##    <chr>     <int>
##  1 Michael      91
##  2 Dwight       80
##  3 Jim          51
##  4 Andy         31
##  5 Pam          26
##  6 Ryan         10
##  7 Clark         8
##  8 Gabe          7
##  9 David         6
## 10 Angela        5
## # … with 36 more rows
```

3) stringr::str_detect

`str_detect` can be combined with familiar functions to summarize data

```
theoffice %>%
  filter(str_detect(text,
                    regex("that's what she said", ignore_case = TRUE))) %>%
  count(character, sort = TRUE)
```

```
## # A tibble: 8 × 2
##   character     n
##   <chr>     <int>
## 1 Michael      23
## 2 Dwight        3
## 3 Jim           2
## 4 Creed         1
## 5 David         1
## 6 Holly         1
## 7 Jan           1
## 8 Pam           1
```

3) stringr::str_detect

`str_detect` with regular expressions can be very powerful

```
theoffice %>% select(character, text) %>%
  filter(str_detect(text, "assistant.*manager")) %>%
  slice_head(n = 10)
```

```
## # A tibble: 10 × 2
##    character text
##    <chr>     <chr>
##  1 Dwight    I, but if there were, I'd be protected as assistant regional manager?
##  2 Dwight    And that's why you have an assistant regional manager.
##  3 Michael   No, I am the team manager. You can be assistant to the team manager.
##  4 Dwight    Hey, Pam, I'm assistant regional manager, and I can take care of him. Part o…
##  5 Michael   All right. Well then, you are now acting manager of Dunder Mifflin Scranton …
##  6 Dwight    Uh,... my first sale, my promotion to assistant regional manager, our basket…
##  7 Jim       Oh, that's because at first it was a made up position for Dwight, just to ma…
##  8 Charles   So you're the assistant to the regional manager?
##  9 Darryl    Since Andy promoted me to assistant regional manager, I've been trying to st…
## 10 Andy      You now, Darryl, this is textbook assistant regional manager stuff here, and…
```

Text mining with R

Core concepts and techniques

Tokens, lemmas, and parts of speech

Sentiment analysis

tf-idf

Topics and LDA

Fingerprinting

Core concepts and techniques

Tokens, lemmas, and parts of speech

Sentiment analysis

tf-idf

Topics and LDA

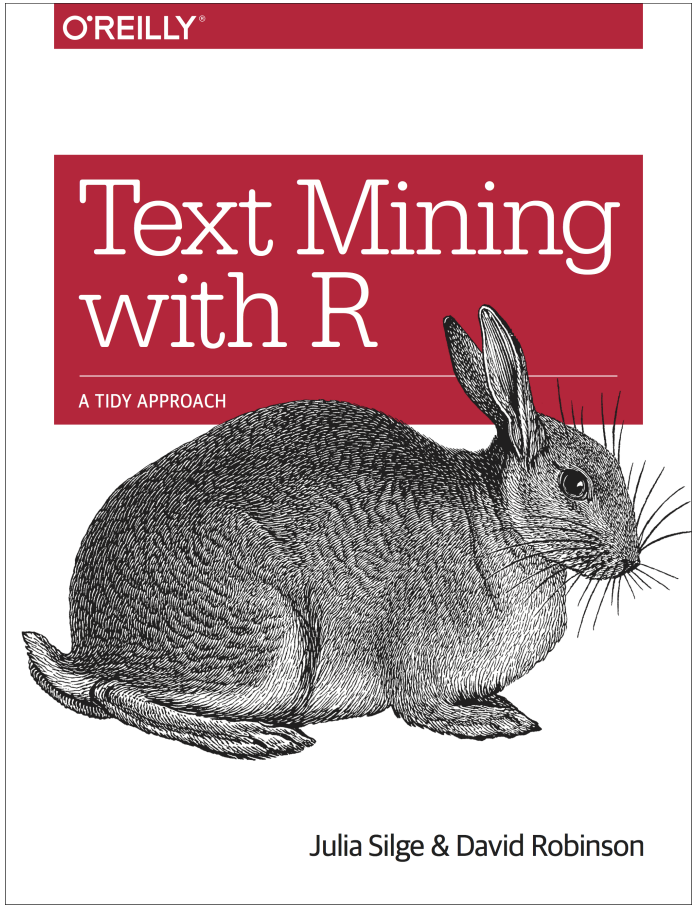
Fingerprinting

We will cover tokens, sentiment analysis, and tf-idf (time permitting)

The tidytext package

We will use the `tidytext` package

`tidytext` brings tidy data concepts and `tidyverse` tools to text analysis



Regular text

THE BOY WHO LIVED Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Mr. Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large mustache. Mrs. Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbors. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere. The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs. Potter was Mrs. Dursley's sister, but they hadn't met for several years; in fact, Mrs. Dursley pretended she didn't have a sister, because her sister and her good-for-nothing husband were as unDursleyish as it was possible to be. The Dursleys shuddered to think what the neighbors would say if the Potters arrived in the street. The Dursleys knew that the Potters had a small son, too, but they had never even seen him. This boy was another good r...

Text as data

Text can be stored in data frames as character strings

Here, each row corresponds to a chapter

```
head(hp1_data)
```

```
## # A tibble: 6 × 2
##   chapter text
##     <int> <chr>
## 1       1 "THE BOY WHO LIVED   Mr. and Mrs. Dursley, of number four, Privet Drive, were …
## 2       2 "THE VANISHING GLASS   Nearly ten years had passed since the Dursleys had woke…
## 3       3 "THE LETTERS FROM NO ONE   The escape of the Brazilian boa constrictor earned …
## 4       4 "THE KEEPER OF THE KEYS   BOOM. They knocked again. Dudley jerked awake. \"Whe…
## 5       5 "DIAGON ALLEY   Harry woke early the next morning. Although he could tell it w…
## 6       6 "THE JOURNEY FROM PLATFORM NINE AND THREE-QUARTERS   Harry's last month with t…
```

Tidy text and tokens

Tidy text format: a table with one **token** per row

What is a token?

- Any meaningful unit of text used for analysis
- Often words, but also letters, n-grams, sentences, paragraphs, chapters, etc.

The relevant token depends on the analysis you are doing

So the definition of tidy text depends on what you are doing!

Tokenization is the process of splitting text into tokens

Tokenization: words

`tidytext::unnest_tokens` tokenizes **words** by default

- Optionally: characters, ngrams, sentences, lines, paragraphs, etc.

```
hp1_data %>%
  unnest_tokens(   # convert data to tokens
    input = text, # split text column
    output = word,# make new word column
    ) %>%
  relocate(word)   # move new column to front
```

Note: `unnest_tokens()` expects **output** before **input** if you don't name arguments

```
## # A tibble: 77,875 × 2
##    word    chapter
##    <chr>     <int>
##  1 the           1
##  2 boy           1
##  3 who           1
##  4 lived         1
##  5 mr            1
##  6 and           1
##  7 mrs           1
##  8 dursley       1
##  9 of            1
## 10 number        1
## # … with 77,865 more rows
```

We can treat tokens like any other data

For example, we can count them:

```
hp1_data %>%
  unnest_tokens(
    input = text,
    output = word,
    ) %>%
  count(word, sort = TRUE)
```

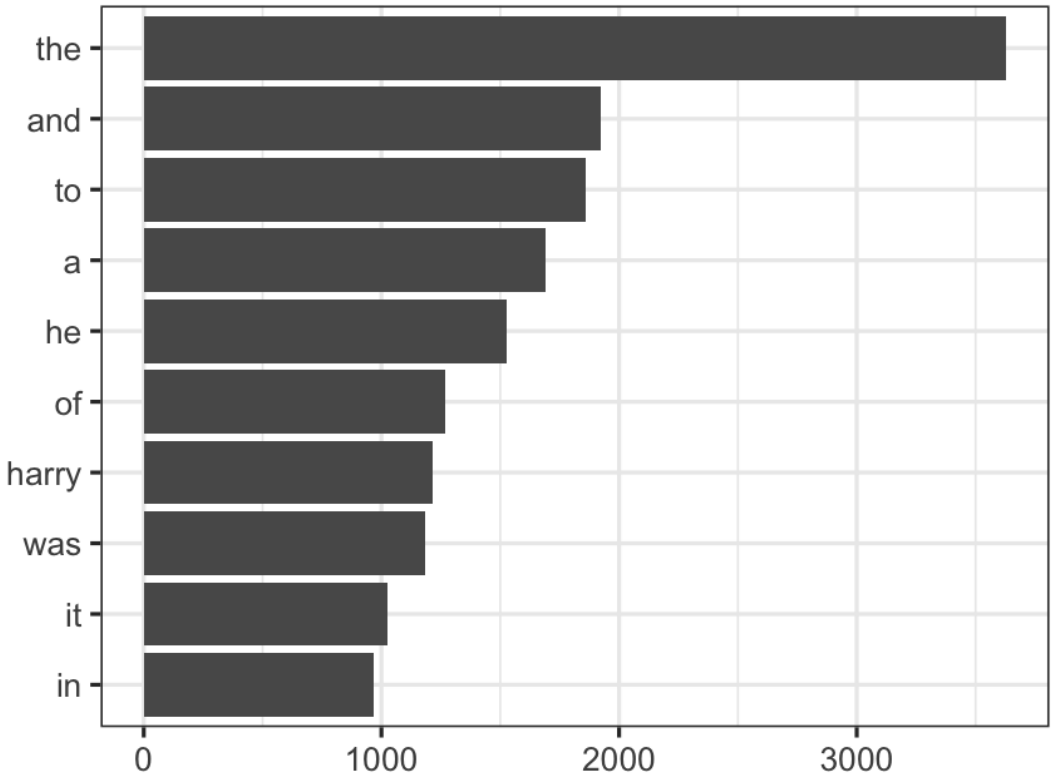
```
## # A tibble: 5,978 × 2
##    word      n
##    <chr> <int>
##  1 the    3629
##  2 and    1923
##  3 to     1861
##  4 a      1691
##  5 he     1528
##  6 of     1267
##  7 harry  1213
##  8 was    1186
##  9 it     1027
## 10 in      968
## # … with 5,968 more rows
```

What do you notice about the most common words?

Raw word frequency is not always informative

```
hp1_data %>%
  unnest_tokens(word, text) %>%
  count(word, sort = TRUE) %>%
  slice_max(order_by = n, # order rows by count
            n = 10) %>%   # slice top 10 rows
  ggplot(aes(x = n,
             y = fct_reorder(word, n))) +
  geom_col() +
  labs(x = NULL, y = NULL) +
  theme_bw()
```

How can we make this better?



Stop words

We can filter out common **stop words** that we want to ignore

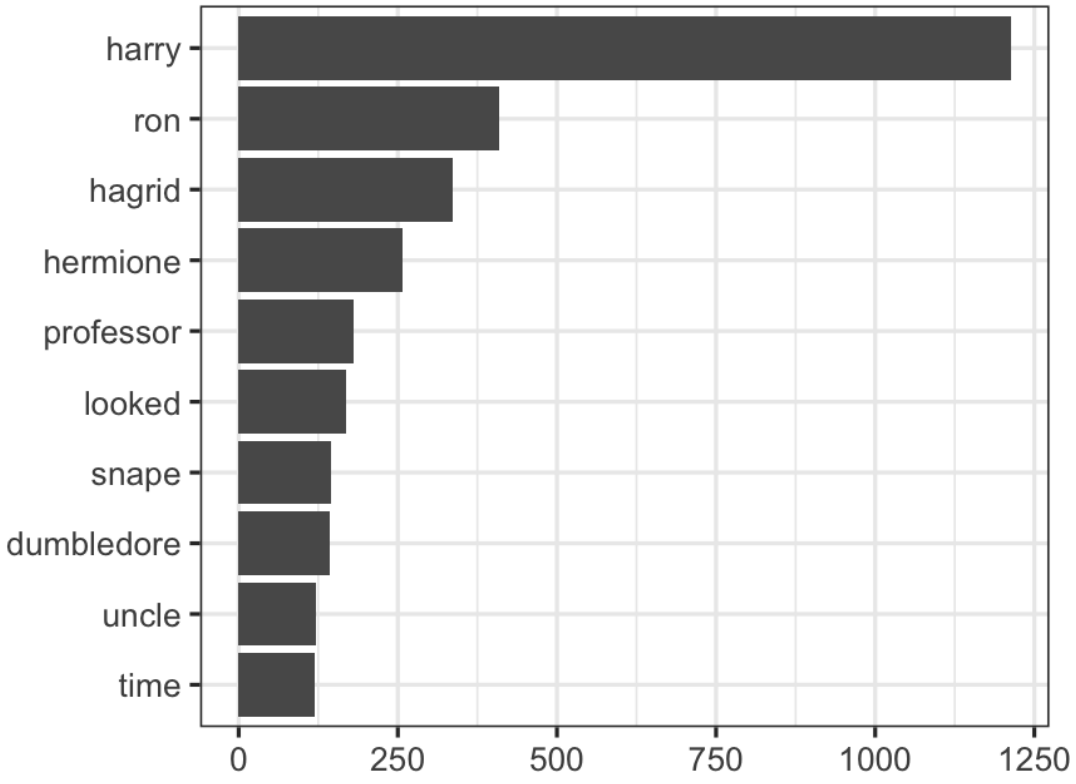
```
stop_words # this is an object from the tidytext package
```

```
## # A tibble: 1,149 × 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 1,139 more rows
```

Token frequency: words

```
hp1_data %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by = "word") %>%
  count(word, sort = TRUE) %>%
  slice_max(order_by = n,
            n = 10) %>%
  ggplot(aes(x = n,
             y = fct_reorder(word, n))) +
  geom_col() +
  labs(x = NULL, y = NULL) +
  theme_bw()
```

That's better!



Tokenization: n-grams

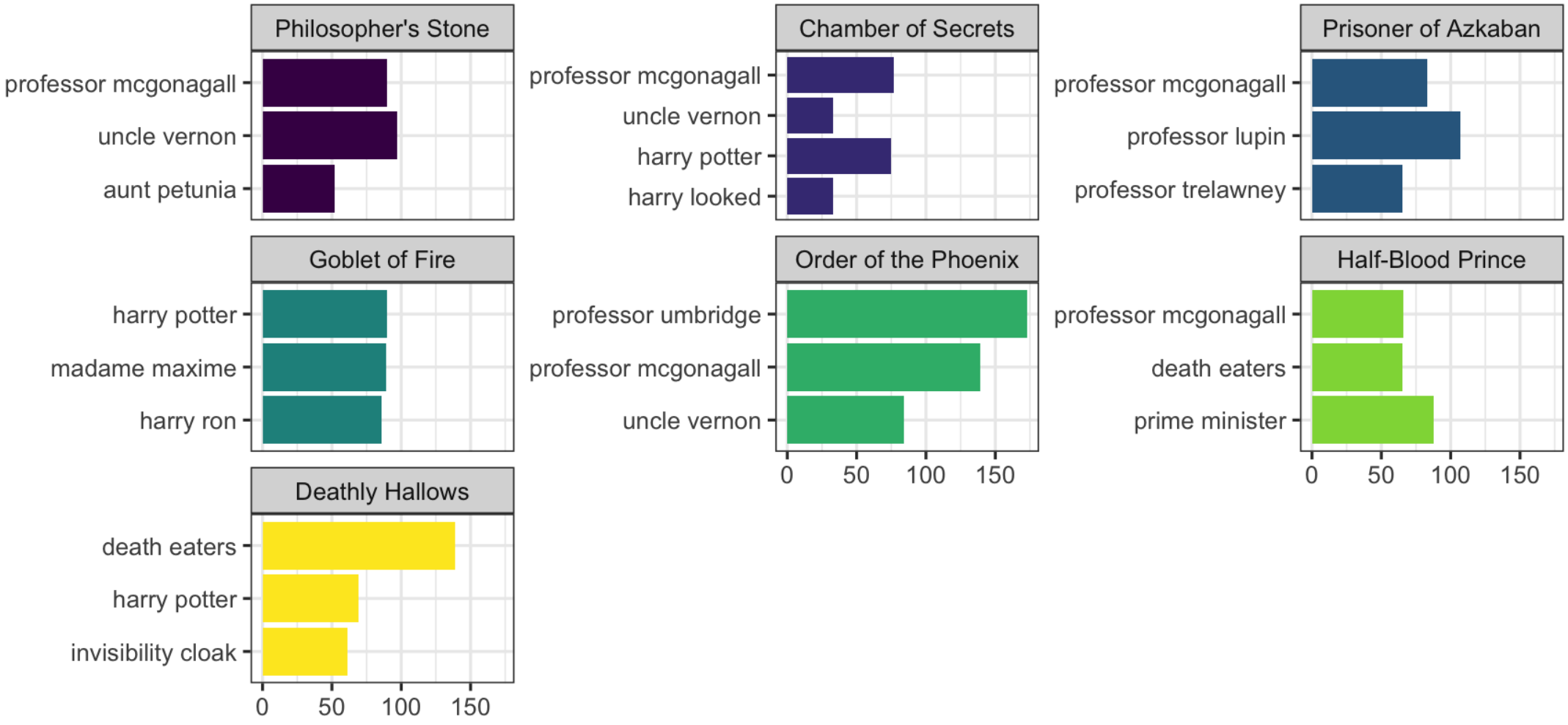
n contiguous tokens are an **n-gram**

Example: two consecutive words are a bigram

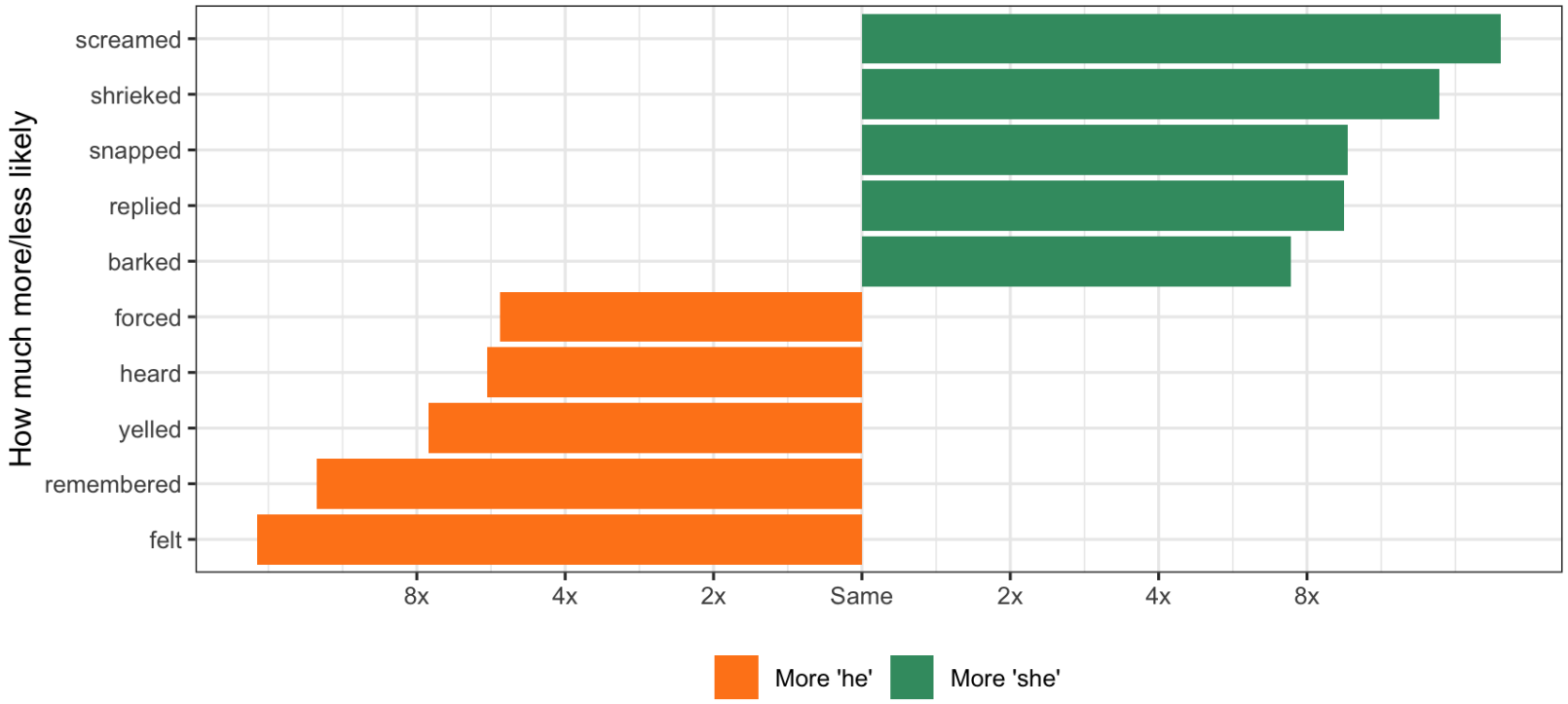
```
hp1_data %>%
  unnest_tokens(
    input = text,
    output = bigram,   # new column bigram
    token = "ngrams", # we want ngrams
    n = 2) %>%         # we want bigrams
  relocate(bigram)
```

```
## # A tibble: 77,858 × 2
##    bigram      chapter
##    <chr>         <int>
##  1 the boy           1
##  2 boy who           1
##  3 who lived         1
##  4 lived mr          1
##  5 mr and            1
##  6 and mrs           1
##  7 mrs dursley       1
##  8 dursley of        1
##  9 of number         1
## 10 number four       1
## # … with 77,848 more rows
```

Token frequency: n-grams



Token frequency: n-gram ratios



How would you make this?

This is not an endorsement of J. K. Rowling's views on sex and gender!

Token frequency: n-gram ratios

1. Identify and count words associated with "he", "she"

```
hp_pronouns <- c("he", "she") # not apologizin

bigram_he_she_counts <- hp %>%
  unnest_tokens(bigram, text, token = "ngrams"
  count(bigram, sort = TRUE) %>%
  # split the bigram column into two columns
  separate(bigram, c("word1", "word2"), sep = '
  # only choose rows where the first word is h
  filter(word1 %in% hp_pronouns) %>%
  # count unique pairs of words
  count(word1, word2, wt = n, sort = TRUE) %>%
  rename(total = n)
```

```
## # A tibble: 2,096 × 3
##    word1 word2  total
##    <chr> <chr>  <int>
##  1 he    was     2490
##  2 he    had     2138
##  3 he    said    1270
##  4 he    could    925
##  5 she   said     615
##  6 she   was      603
##  7 he    looked   428
##  8 he    did      386
##  9 she   had      385
## 10 he    would    349
## # … with 2,086 more rows
```

Token frequency: n-gram ratios

1. Compute ratios for the most common bigrams

```
word_ratios <- bigram_he_she_counts %>%
  # look at each of the second words
  group_by(word2) %>%
  # keep rows where the second word appears 10
  filter(sum(total) > 10) %>%
  ungroup() %>%
  # pivot out the word1 column so that there's
  pivot_wider(names_from = word1, values_from =
  # filter out missing he/she cases (subjectiv
  filter(!is.na(he) & !is.na(she)) %>%
  # convert word2 counts to frequencies by he/
  mutate_if(is.numeric, ~ (.) / sum(.)) %>%
  # compute logged ratio of the she counts to
  mutate(logratio = log2(she / he)) %>%
  # sort by that ratio
  arrange(desc(logratio))

word_ratios
```

```
## # A tibble: 262 × 4
##    word2           he      she logratio
##    <chr>        <dbl>    <dbl>    <dbl>
##  1 screamed 0.000171 0.00338     4.30
##  2 shrieked 0.000228 0.00338     3.89
##  3 snapped  0.000399 0.00386     3.27
##  4 replied  0.000228 0.00217     3.25
##  5 barked   0.000228 0.00169     2.89
##  6 cried    0.000628 0.00338     2.43
##  7 sounded  0.000685 0.00290     2.08
##  8 say      0.000456 0.00169     1.89
##  9 checked  0.000342 0.00121     1.82
## 10 jerked   0.000342 0.00121     1.82
## # … with 252 more rows
```

Token frequency: n-gram ratios

1. Rearrange the data for plotting

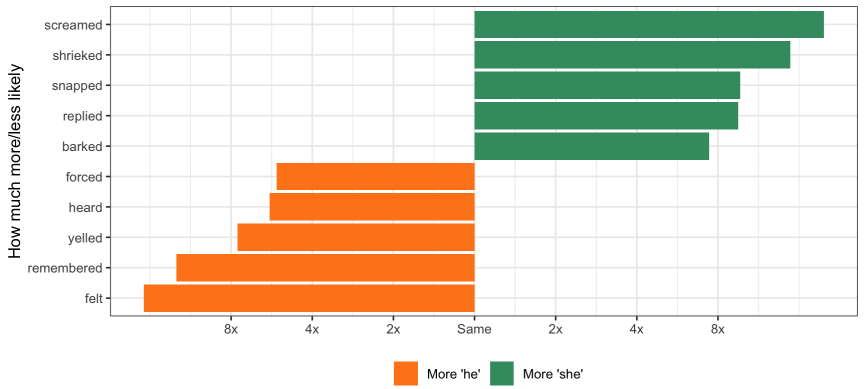
```
plot_word_ratios <- word_ratios %>%
  mutate(abslogratio = abs(logratio)) %>%
  group_by(logratio < 0) %>%
  slice_max(abslogratio, n = 5) %>%
  ungroup()

plot_word_ratios
```

```
## # A tibble: 10 × 6
##    word2            he      she logratio abslograt
##    <chr>         <dbl>    <dbl>    <dbl>      <db
##  1 screamed   0.000171 0.00338      4.30       4.
##  2 shrieked   0.000228 0.00338      3.89       3.
##  3 snapped    0.000399 0.00386      3.27       3.
##  4 replied    0.000228 0.00217      3.25       3.
##  5 barked     0.000228 0.00169      2.89       2.
##  6 felt       0.0163   0.000966    -4.07       4.
##  7 remembered 0.00308  0.000242    -3.67       3.
##  8 yelled     0.00183  0.000242    -2.92       2.
##  9 heard      0.00976  0.00169     -2.53       2.
## 10 forced     0.00131  0.000242    -2.44       2.
```

Token frequency: n-gram ratios

1. Plot the data

```
plot_word_ratios %>%
  ggplot(aes(logratio,
             fct_reorder(word2, logratio),
             fill = logratio < 0)) +
  geom_col() +
  labs(y = "How much more/less likely", x = NULL)
  scale_fill_manual(name = "", labels = c("More 's
                    values = c("#3D9970", "#FF851B
                    guide = guide_legend(reverse =
  scale_x_continuous(breaks = seq(-3, 3),
                     labels = c("8x", "4x", "2x",
                                "Same", "2x", "4x"
  theme_bw() +
  theme(legend.position = "bottom")
```



Sentiment analysis

Simple approach: quantify sentiment of each word in a corpus, then sum them up

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

```
# A tibble: 6,786 × 2
   word        sentiment
   <chr>       <chr>
 1 2-faces     negative
 2 abnormal    negative
 3 abolish     negative
 4 abominable  negative
 5 abominably  negative
 6 abominate   negative
 7 abomination negative
 8 abort       negative
 9 aborted     negative
10 aborts      negative
# … with 6,776 more rows
```

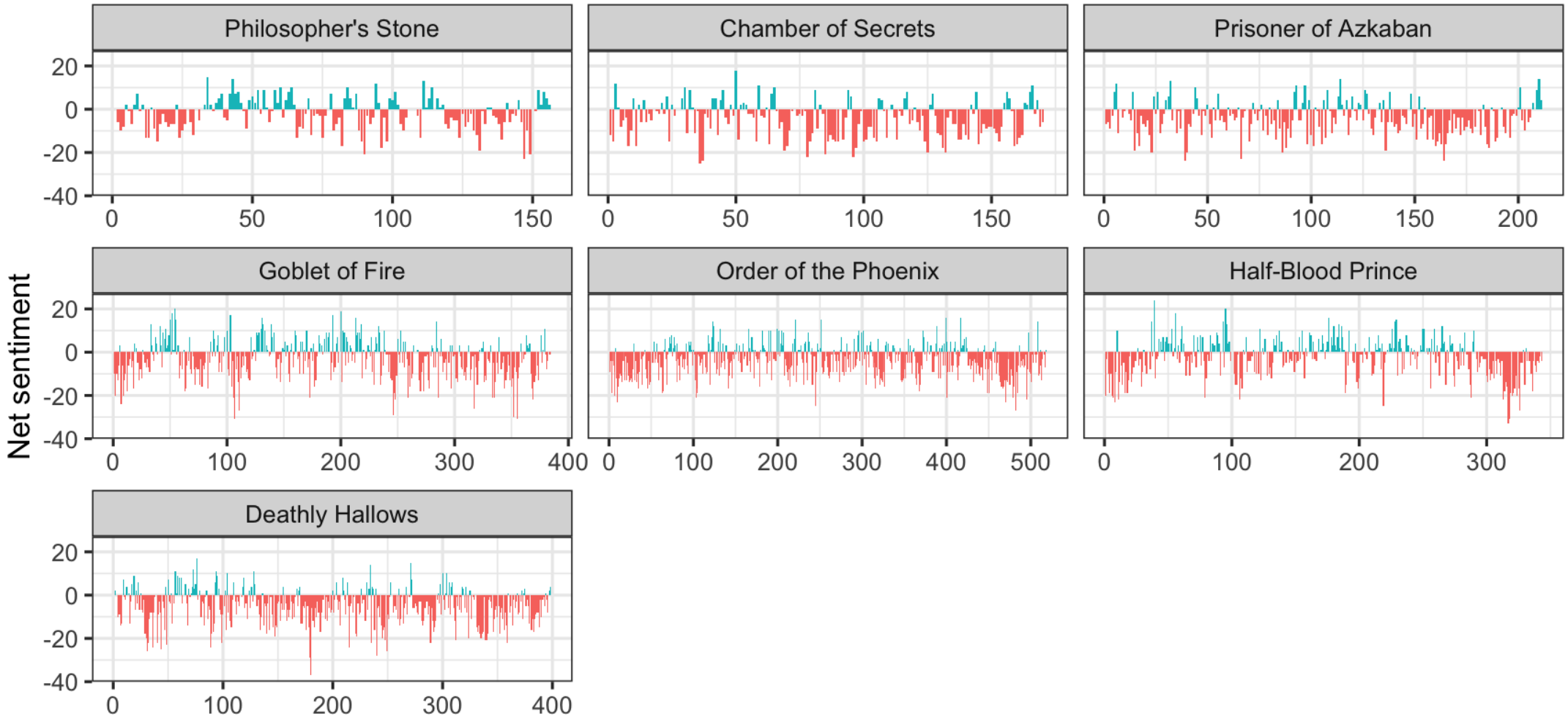
```
get_sentiments("afinn")
```

```
# A tibble: 2,477 × 2
   word       value
   <chr>      <dbl>
 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 2,467 more rows
```

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

```
# A tibble: 13,875 × 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 13,865 more rows
```

Harry Potter sentiment analysis



tf–idf

Term frequency-inverse document frequency

tf-idf is a measure of how important a term is to a document within a broader collection or corpus

$$tf(\text{term}) = \frac{n_{\text{term}}}{n_{\text{terms in document}}}$$

$$idf(\text{term}) = \ln\left(\frac{n_{\text{documents}}}{n_{\text{documents containing term}}}\right)$$

$$tf\text{-}idf(\text{term}) = tf(\text{term}) \times idf(\text{term})$$

Harry Potter tf-idf