Computational Social Science

Introduction to Natural Language Processing I

Dr. Thomas Davidson

Rutgers University

October 13, 2025

Plan

1. Course updates
2. What is NLP?
3. Preprocessing texts and the bag-of-words representation

Course updates

- Homework 2 due Friday, end of day
  - Submit homework via Github
  - Add link to Canvas once submitted
- Project proposals due next Thursday
  - Identify a suitable topic, data, and plan for analysis
  - Submit via Canvas

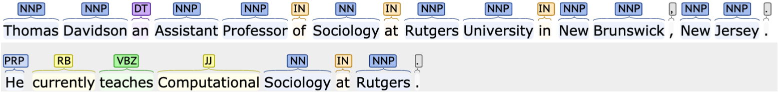
Introduction to NLP

What is natural language processing?

- Three components of NLP*:
  - Natural language / “text as data”
    - A corpus of text (e.g. books, reviews, tweets, e-mails)
  - (Computational) linguistics
    - Linguistic theory to guide analysis and computational approaches to handle data
  - Statistics
    - Statistical methods to make inferences

*Not *that* NLP: https://en.wikipedia.org/wiki/Neuro-linguistic_programming

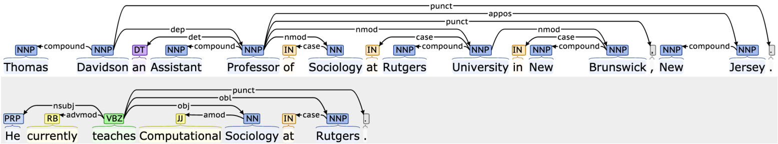
NLP tasks: Part-of-speech tagging



Examples created using https://corenlp.run/

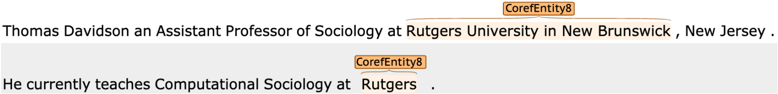
Introduction to NLP

NLP tasks: Dependency-parsing



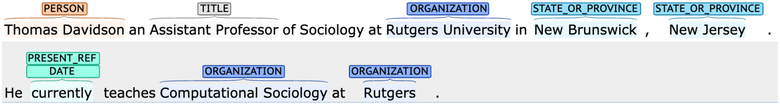
Examples created using https://corenlp.run/

NLP tasks: Co-reference resolution



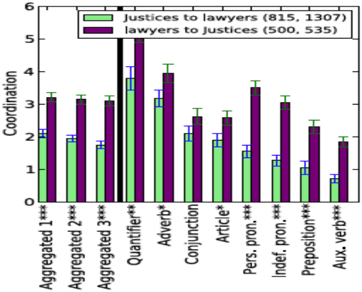
Examples created using https://corenlp.run/

NLP tasks: Named-entity recognition



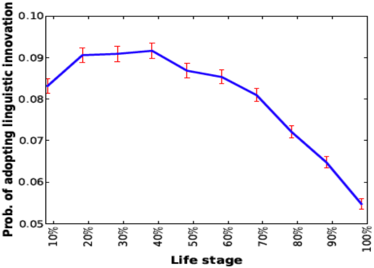
Examples created using https://corenlp.run/

Applications: Power dynamics



Danescu-Niculescu-Mizil, Cristian, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. "Echoes of Power: Language Effects and Power Differences in Social Interaction." In Proceedings of the 21st International Conference on World Wide Web, 699–708. ACM. http://dl.acm.org/citation.cfm?id=2187931.

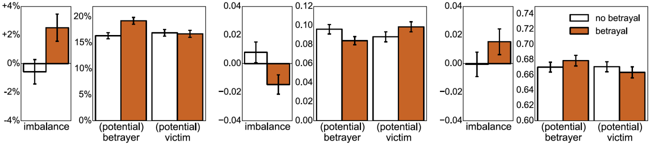
Applications: Identity and group membership



(c) Adoption of lexical innovations

Danescu-Niculescu-Mizil, Cristian, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. "No Country for Old Members: User Lifecycle and Linguistic Change in Online Communities." In Proceedings of the 22nd International Conference on World Wide Web, 307–18. ACM. http://dl.acm.org/citation.cfm?id=2488416.

Applications: Trust and betrayal



(a) Positive sentiment (percentage of sentences)

(b) Planning discourse markers (avg. number per sentence)

(c) Politeness (avg. message score)

Figure 3: Friendships that will end in betrayal are imbalanced. The eventual betrayer is more positive, more polite, but plans less than the victim. The white bars correspond to matched lasting friendships, where the roles of potential betrayer and victim are arbitrarily assigned; in these cases, the imbalances disappear. Error bars mark bootstrapped standard errors (Efron, 1979).

Niculae, Vlad, Srijan Kumar, Jordan Boyd-Graber, and Cristian Danescu-Niculescu-Mizil. 2015. "Linguistic Harbingers of Betrayal: A Case Study on an Online Strategy Game." In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. Beijing, China: ACL. http://arxiv.org/abs/1506.04744.

Text as data

Table 1 Four principles of quantitative text analysis

(1)	All quantitative models of language are wrong—but some are useful.
(2)	Quantitative methods for text amplify resources and augment humans.
(3)	There is no globally best method for automated text analysis.
(4)	Validate, Validate, Validate.

- Justin Grimmer and Brandon Stewart, 2013

Introduction to NLP

"Computational approaches are sometimes less subtle and deep than the reading of a skillful analyst, who interprets text in context. Nevertheless, ... recent advances in NLP and ML are being used to enhance qualitative analysis in two ways. First, supervised ML prediction tools can "learn" and reliably extend many sociologically interesting textual classifications to massive text samples far beyond human capacity to read, curate, and code. Second, unsupervised ML approaches can "discover" unnoticed, surprising regularities in these massive samples of text that may merit sociological consideration and theorization." - James Evans and Pedro Aceves, 2016

Introduction to NLP

NLP class timeline

- Week 7
  - Pre-processing, bag-of-words, and the vector-space model
- Week 8
  - Word embeddings
- Week 9
  - Topic models
- Week 11
  - Supervised text classification
- Week 14
  - Large language models and generative AI

Working with text

Pre-processing

- There are several steps we need to take to “clean” or “pre-process” texts for analysis
  - Tokenization
  - Stemming/lemmatization
  - Stop-word removal
  - Handling punctuation and special characters

Working with text

Tokenization

- Tokenization is the process of splitting a document into words
  - e.g. "Cognito, ergo sum" $\Rightarrow$ ("Cognito,", "ergo", "sum")
- In English this is pretty trivial, we just split using white-space
- Tokenization is more difficult in languages like Mandarin
  - It requires more complex parsing to understand grammatical structures

Working with text

Stemming/lemmatization

- We often want to reduce sparsity by reducing words to a common root
  - e.g. (“school”, schools”, “schooling”, “schooled”) $\Rightarrow$ “school”
- Stemming is a simple, heuristic-based approach
- Lemmatization is a more rigorous approach based on morphology, but is more computationally-intensive and often unnecessary

Working with text

Stop-word removal

- Stop-words are frequently occurring words that are often removed
- The intuition is that they add little meaning and do not help us to distinguish between documents
    - e.g. Virtually all texts in English will contain the words "and", "the", "of", etc.
- Most NLP packages have lists of stop-words to easily facilitate removal.

Working with text

Handling punctution and special characters

- In many cases we may want to remove punctuation and other special characters (e.g. HTML, unicode)
  - This is often done using regular expressions
  - Words are typically set to lowercase

Working with text

Pre-process with caution!

- Researchers often apply these techniques before starting an analysis, but it may affect our results*
  - There is no one-size-fits-all solution, so think carefully before removing anything
  - It's often useful to experiment to see if pre-processing steps affect results

Pre-process with caution!

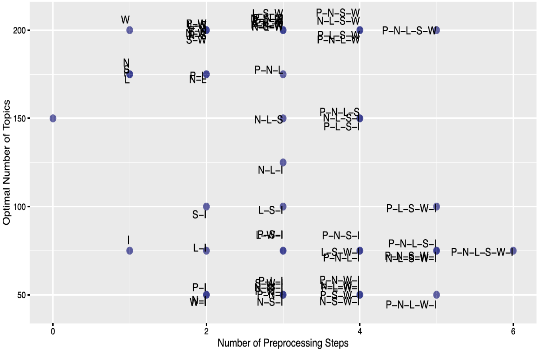


Figure 2. Plot depicting the optimal number of topics (as selected via perplexity) for each of 64 preprocessing specifications not including trigrams. On the x-axis is the number of preprocessing steps, and the y-axis is the number of topics. Each point is labeled according to its specification.

Denny, Matthew J., and Arthur Spirling. 2018. "Text Preprocessing For Unsupervised Learning" Political Analysis 26 (02): 168–89. https://doi.org/10.1017/pan.2017.44.

Working with text

Word counts

- Now we have done some pre-processing, one of the most basic ways we can start to analyze tests is by counting the frequency of words.
  - e.g. "I think, therefore I am" ⇒

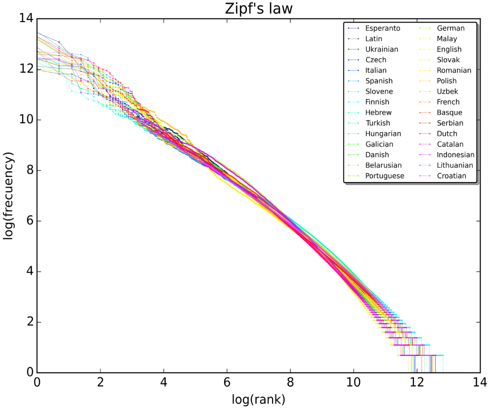
Word	Count
I	2
think	1
therefore	1
am	1

Working with text

Frequency distributions

- **Zipf's law**: *A word's frequency is inversely proportional to its rank order in the frequency distribution*.
  - "the" is the most common word in the English language, accounting for 7% of all words in the *Brown Corpus of American English*
  - "and" and "of" compete for second place, each accounting for ~3.5% of words in the corpus
  - The most frequent 135 words account for approximately half the 1 million words in the corpus
  - Around 50,000 words, representing half the total unique words in the corpus, are *hapax legomena*, words which only occur once

Working with text: Zipf's law



A plot of the rank versus frequency for the first 10 million words in 30 Wikipedias (dumps from October 2015) in a log-log scale (Source: Wikipedia).

Working with text

Bag-of-words

- Documents are often treated as “bags of words”, i.e. we treat a document as a collection of words without retaining information about the order
  - e.g. “This is a document” $\Rightarrow$ (“document”, “This”, “a”, “is”)

Working with text

Example: Loading data

```
library(tidyverse)
library(tidytext)
library(gutenbergr)

ef <- gutenberg_download(41360) # Download Elementary Forms
cm <- gutenberg_download(61) # Download Communist Manifesto

ef$title <- "Elementary Forms"
cm$title <- "Communist Manifesto"

texts <- bind_rows(ef, cm)
write_csv(texts, "../data/marxdurkheim.csv")
```

Working with text

In this example, each text is represented as a table, where the first column is the ID in the Project Gutenberg database and the `text` field contains each sentence as a new row.

```r
print(tail(texts$text))
```

```
## [1] "chains. They have a world to win."
## [2] "chains. They have a world to win."
## [3] ""
## [4] ""
## [5] "WORKING MEN OF ALL COUNTRIES, UNITE!"
## [6] "WORKING MEN OF ALL COUNTRIES, UNITE!"
```

Working with text

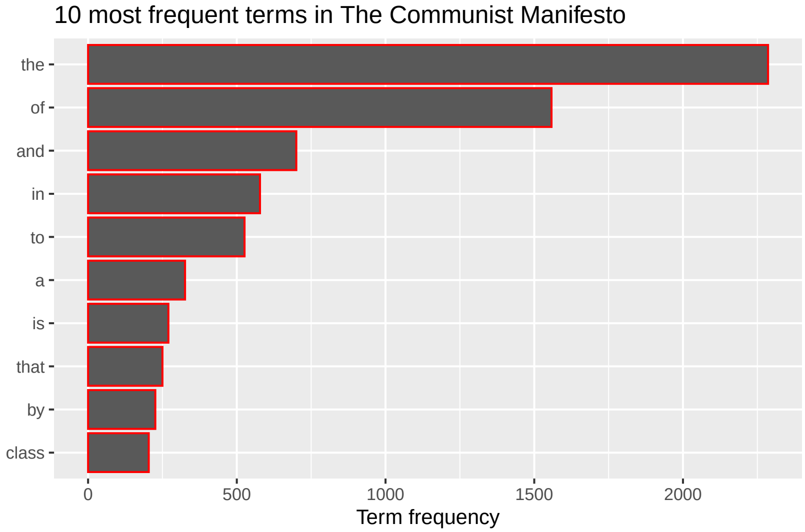
Tokenizing using `tidytext`

We are going to be using the `tidytext` package to conduct our analyses. The `unnest_tokens` function is used to tokenize the text, resulting in a table containing the original book ID and each token as a separate row.

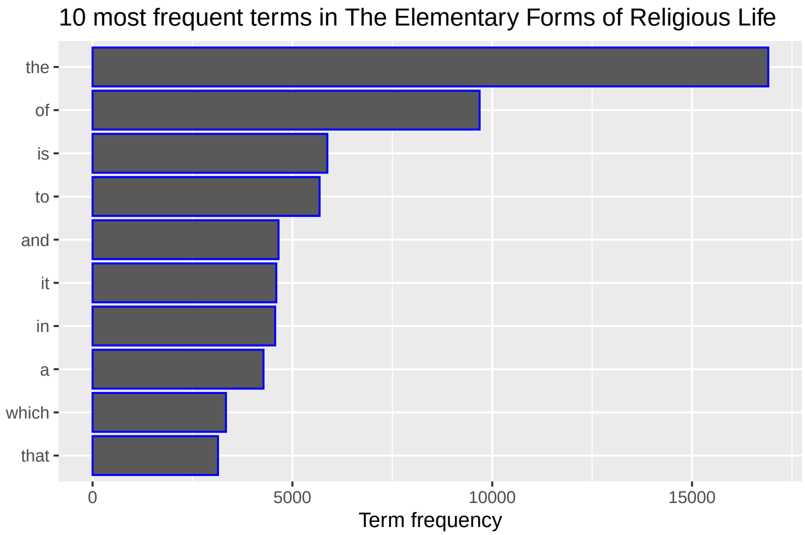
```
tidy.text <- texts %>% unnest_tokens(word, text)
tail(tidy.text$word)
```

```
## [1] "working"   "men"       "of"        "all"       "countries" "uni
```

Working with text



Working with text



Working with text

Removing stopwords

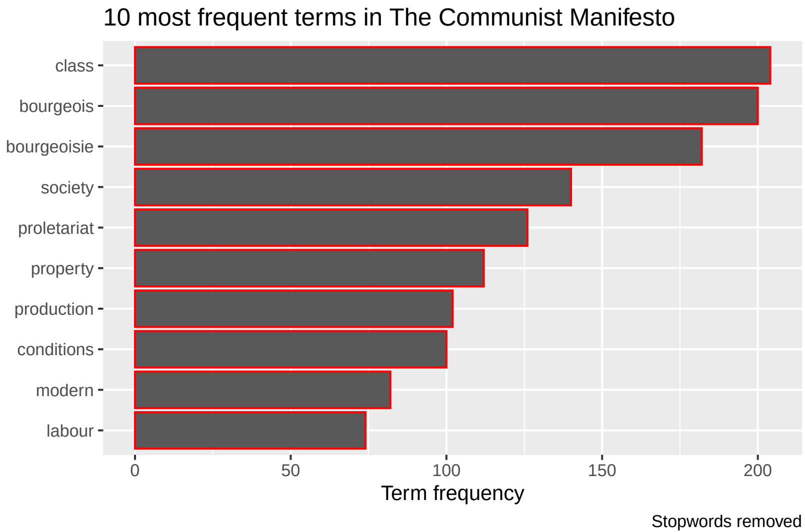
We can load a corpus of stop words contained in `tidytext` and use `anti_join` to filter our texts. This retains all records without a match in stopwords.

```
data(stop_words)
head(stop_words)
```

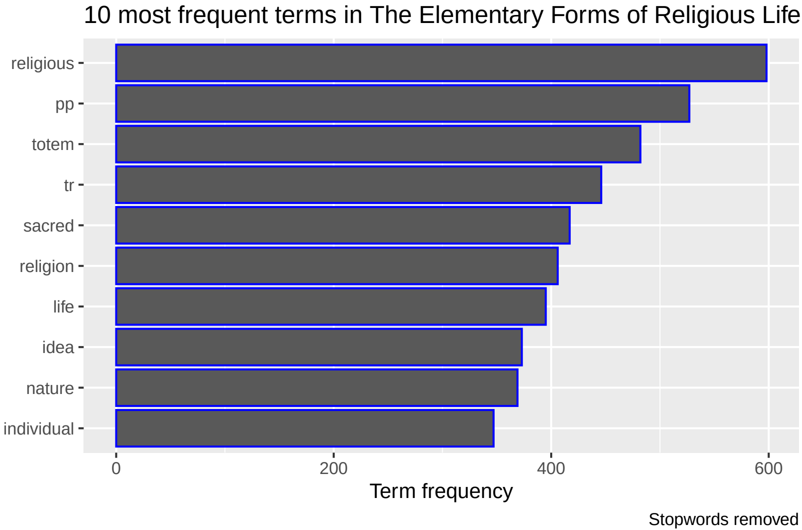
```
## # A tibble: 6 x 2
##   word      lexicon
##   <chr>     <chr>
## 1 a         SMART
## 2 a's       SMART
## 3 able      SMART
## 4 about     SMART
## 5 above     SMART
## 6 according SMART
```

```
tidy.text <- tidy.text %>%
  anti_join(stop_words)
```

Working with text



Working with text



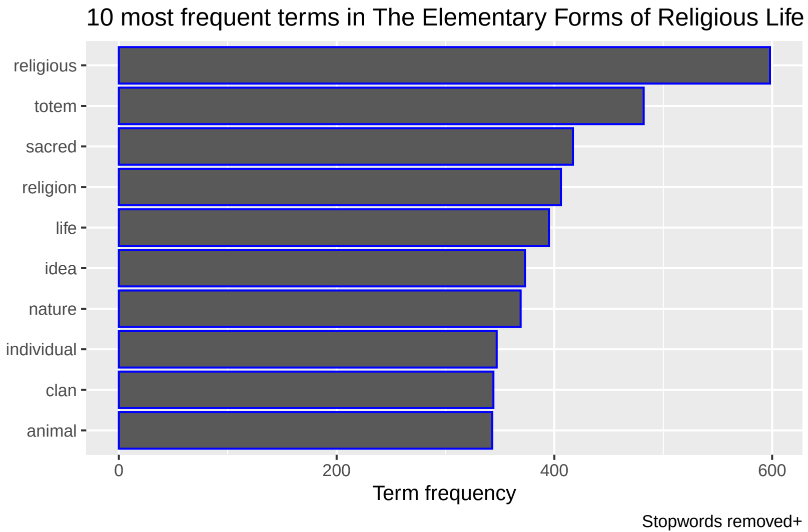
Working with text

The last example shows how there is still some "junk" in the Durkheim text. We can try to remove this by adding to our stopword list.

```
to.remove <- tibble(text=c("pp", "tr")) %>% unnest_tokens(word, text)

tidy.text <- tidy.text %>%
  anti_join(to.remove)
```

Working with text



Working with text

Stemming

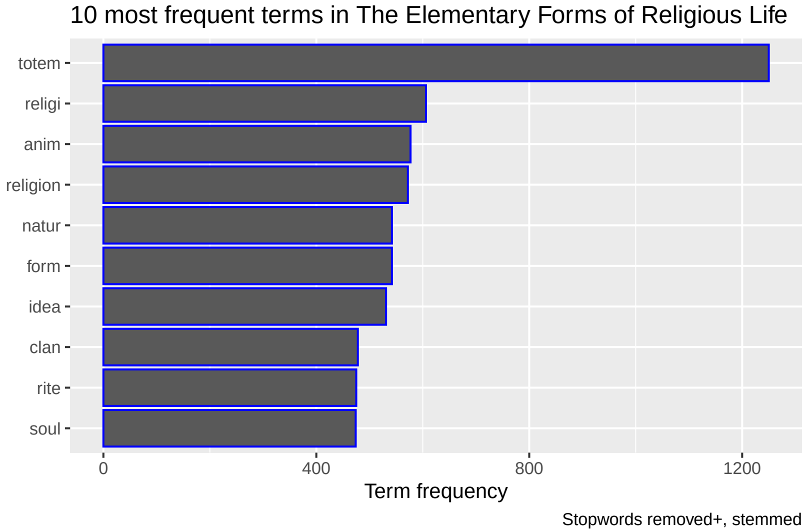
We can stem the terms using a function from the package `SnowballC`, which is a wrapper for a commonly used stemmer called the Porter Stemmer, written in C.

```
library(SnowballC)

tidy.text <- tidy.text %>% mutate_at("word", funs(wordStem((.), languag
```

Stemmer solution from https://cbail.github.io/SICSS_Basic_Text_Analysis.html. See for more info on stemming and lemmatizing in R.

Working with text



Working with text: Zipf's law

Let's get counts of words across both texts to analyze their distribution.

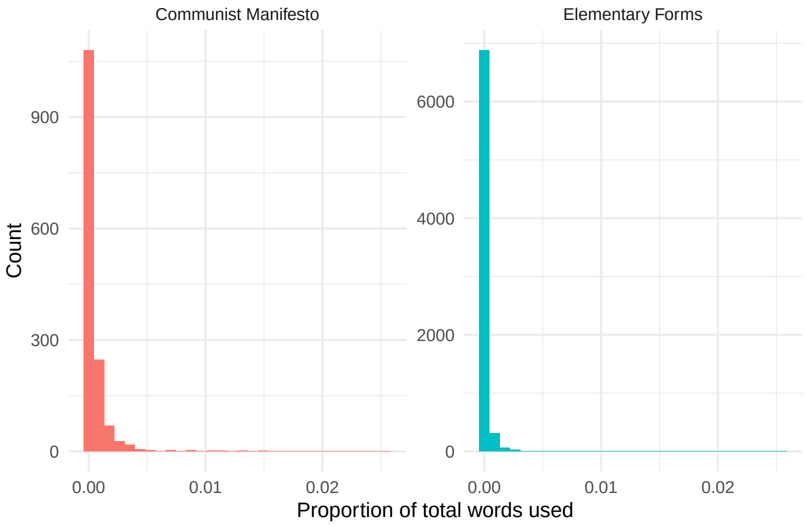
```
# Count words by text
text.words <- tidy.text %>% count(title, word, sort = TRUE)

# Get total number of words in each text
total.words <- text.words %>% group_by(title) %>%
  summarize(total = sum(n))

# Merge
words <- left_join(text.words, total.words)
head(words)
```

```
## # A tibble: 6 x 4
##   title             word       n total
##   <chr>             <chr>  <int> <int>
## 1 Elementary Forms totem    1250 78807
## 2 Elementary Forms religi    606 78807
## 3 Elementary Forms anim      577 78807
```

Working with text: Zipf's law

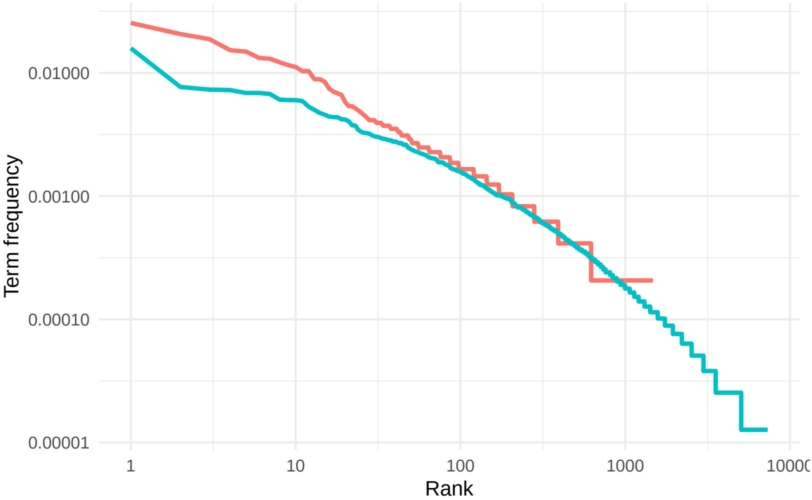


Working with text: Zipf's law

Calculating rank and frequency for each word in each text.

```
freq_by_rank <- words %>%
  group_by(title) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total) %>%
  ungroup()
```

Working with text: Zipf's law



Log−log axes

Working with text

N-grams

- So far we have just considered treating a text as a "bag-of-words"
- One way to maintain some information about word order (and hence syntax) is to use N-grams
- An *N-gram** is a sequence of N words
- We often split texts into N-grams to capture basic syntactic units like phrases
  - N is usually small.
    - $N = 2$ is called a "bigram"; $N = 3$ is a "trigram"
  - e.g. "I like peanut butter" contains the following bigrams: "I like", "like peanut", & "peanut butter".

*Nothing to do with Scientology https://en.wikipedia.org/wiki/Engram_(Dianetics)

Working with text

N-grams

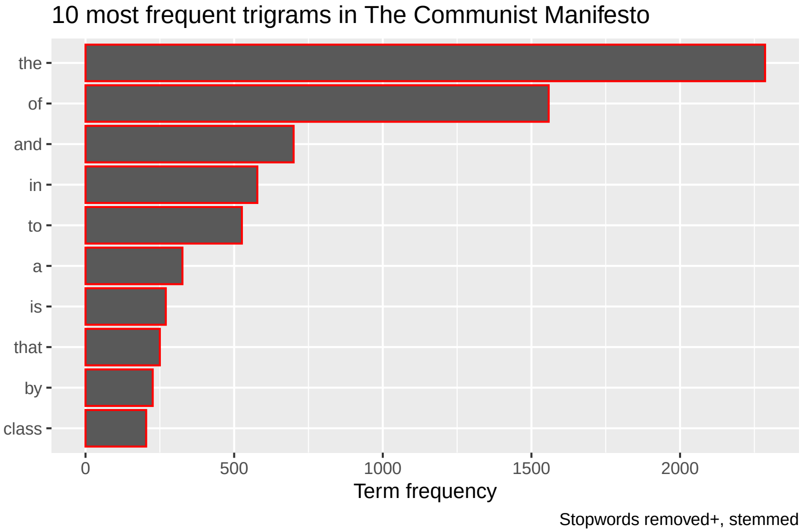
- We can also use *character N-grams* to split documents into sequences of characters
  - e.g. "character" can be split into the following triplets ("cha", "har", "ara", "rac", "act", "cte", "ter")
- Some recent approaches like BERT combine both character and word N-grams into "word pieces".
  - This makes it easy to tokenize new documents since we can always represent them as characters if a word is not in our vocabulary

Trigams

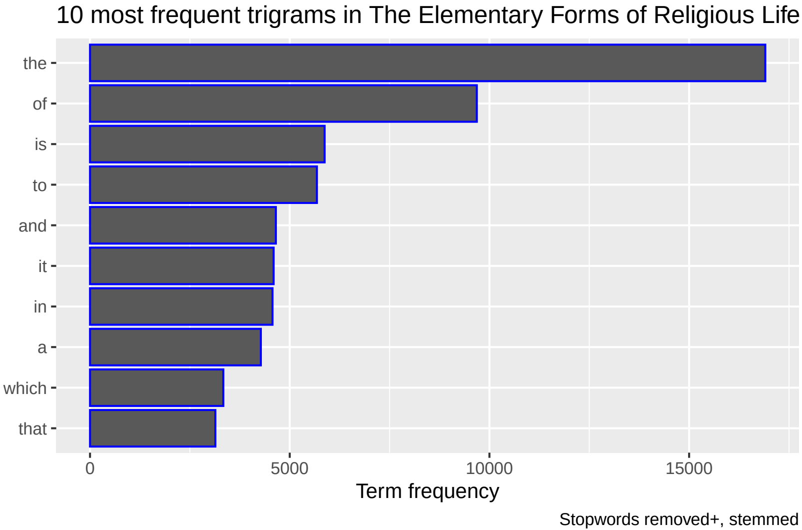
Excercise: Modify `unnest_tokens` to construct trigrams.

```
## # A tibble: 6 x 3
##   gutenberg_id title            word
##          <int> <chr>            <chr>
## 1        41360 Elementary Forms the
## 2        41360 Elementary Forms elementary
## 3        41360 Elementary Forms forms
## 4        41360 Elementary Forms of
## 5        41360 Elementary Forms the
## 6        41360 Elementary Forms religious
```

Trigrams


10 most frequent trigrams in The Communist Manifesto
the
of
and
in
to
a
is
that
by
class
0
500
1000
1500
2000
Term frequency
Stopwords removed+, stemmed

Trigrams



10 most frequent trigrams in The Elementary Forms of Religious Life
the
of
is
to
and
it
in
a
which
that
0
5000
10000
15000
Term frequency
Stopwords removed+, stemmed

Next lecture

- The vector-space model
- Text similarity measures