

Textual data and text mining (1)

Daniele Rotolo

Introductory Data Science for Innovation (995N1) – Week 7, 15 November 2021

Outline

Outline

- Defining text mining
- Text mining first steps
- Tokenisation
- Lemmatisation
- Stemming
- tf-idf

Defining text mining

What is text mining?

“[...] text mining seeks to extract useful information from data sources through the identification and exploration of interesting patterns [...]”

“[...] data sources are document collections, and interesting patterns are found not among formalized database records but in the unstructured textual data in the documents in these collections” (Feldman and Sanger 2006)

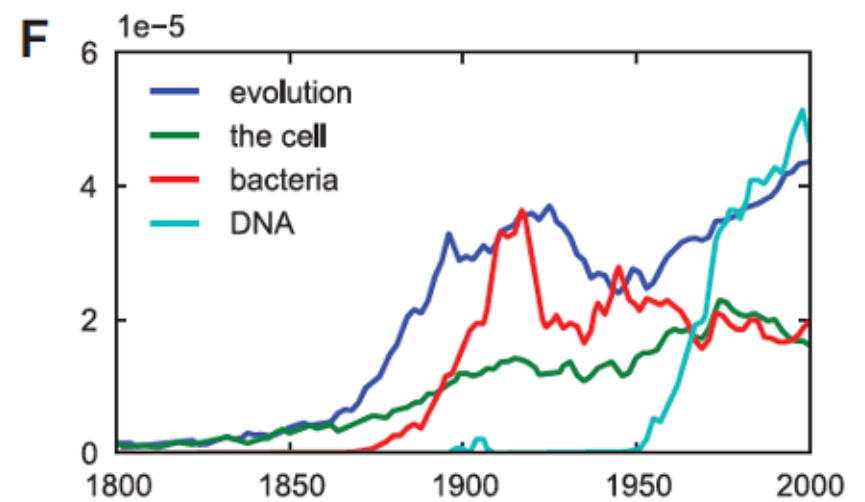
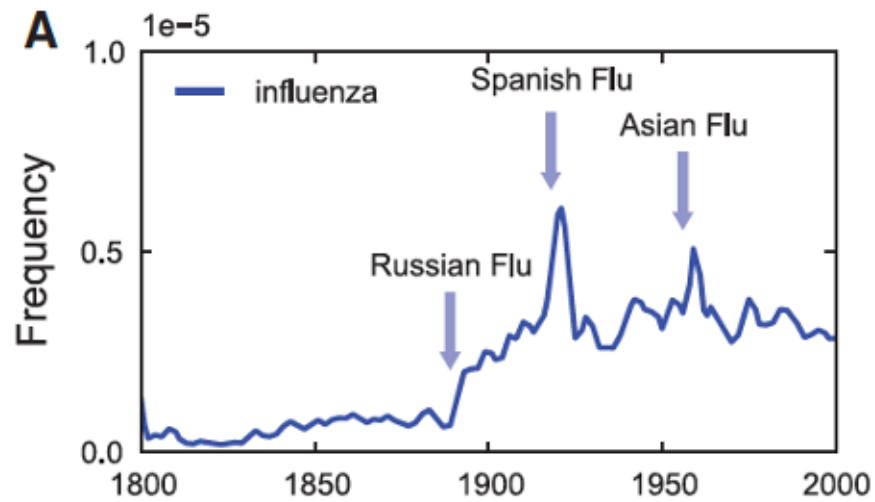
“Text mining represents the ability to take large amounts of unstructured language and quickly extract useful and novel insights that can affect stakeholder decision-making” (Kwartler 2017)

Why text mining?

- Increasing access to data in the form of text
 - Newspapers
 - Bibliometric data (full text of publications and patents)
 - Social media (e.g. Twitter)
 - Parliamentary debates (e.g. <https://hansard.parliament.uk>)
 - ...
- A phenomenon of considerable magnitude (<https://www.webfx.com/internet-real-time/>)
- Textual data are unstructured

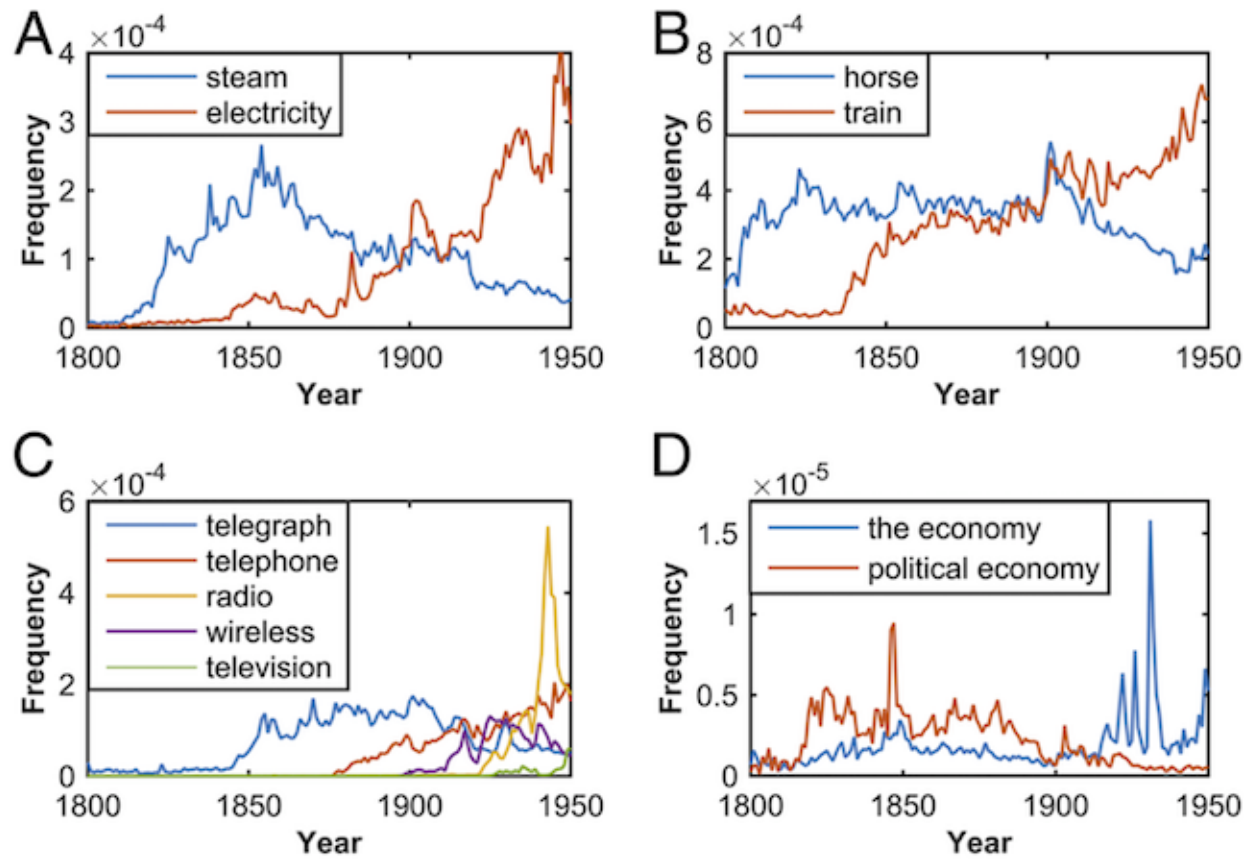
How can we identify patterns in these large amount of unstructured data?

Example 1



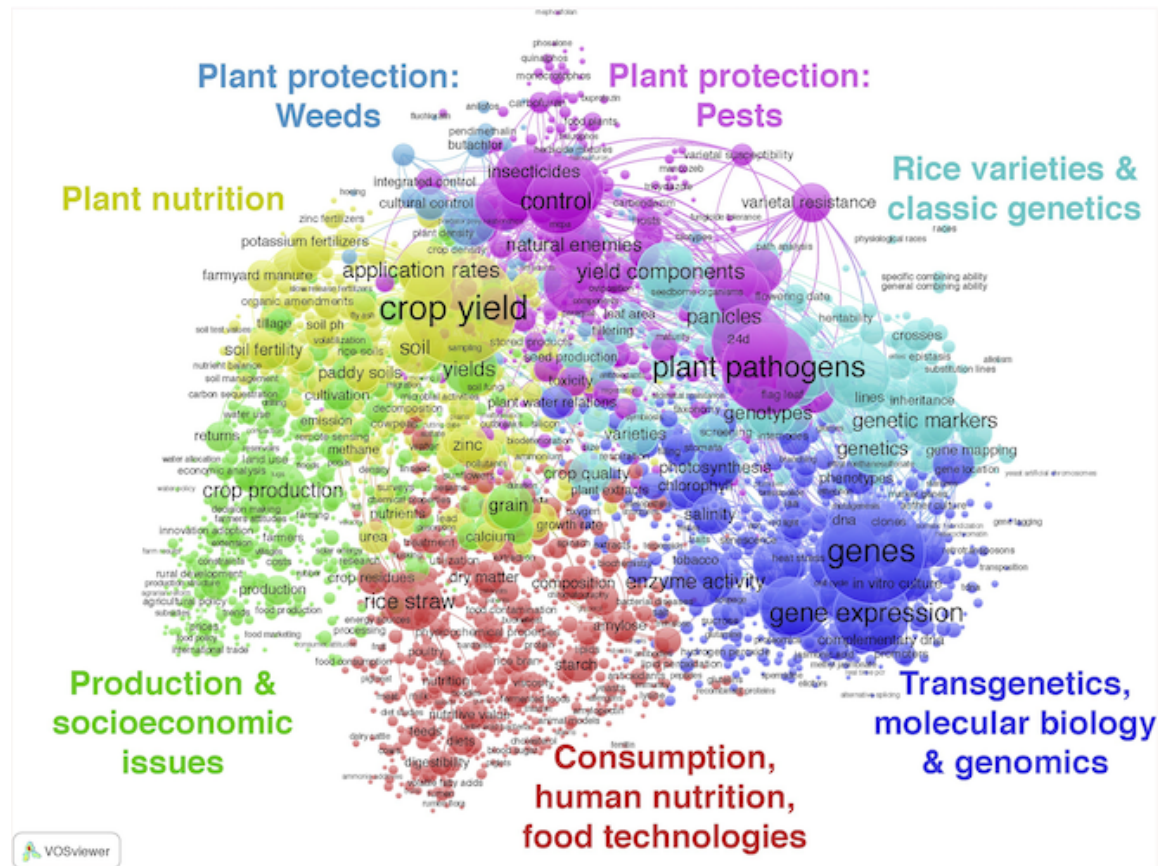
Source: Examining linguistic and cultural phenomena (1800-2000) - sample of 5 millions books (Michel et al. 2011)

Example 2



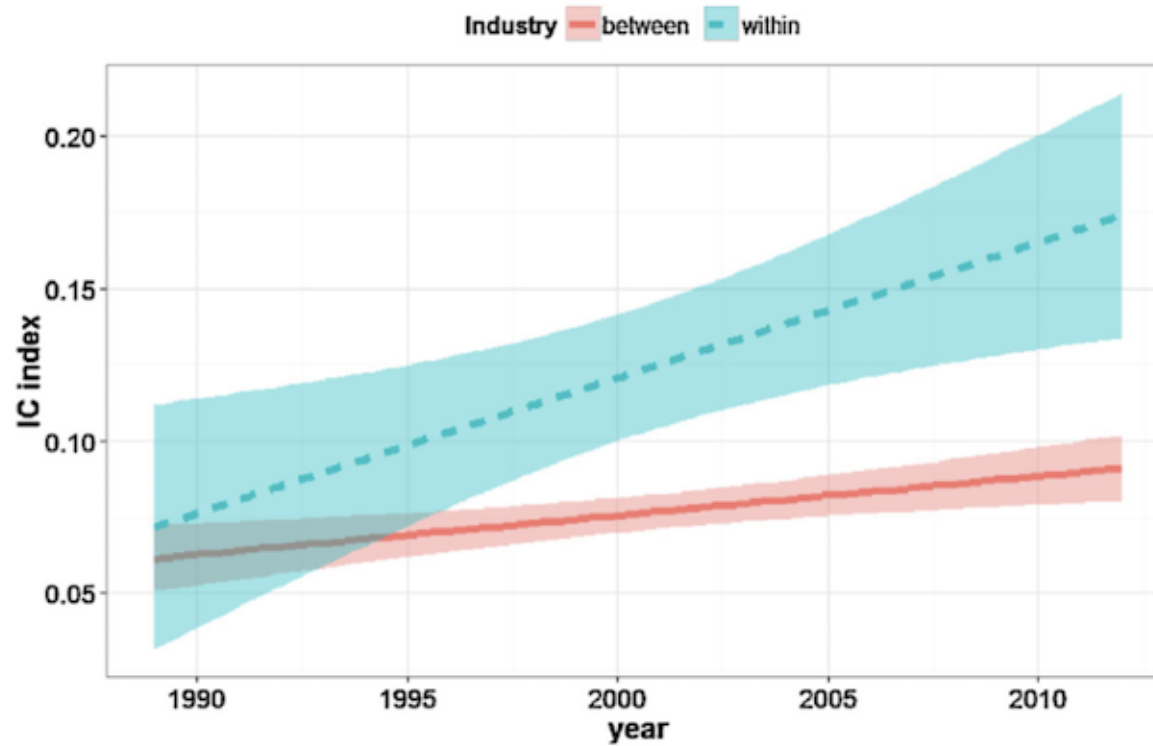
Source: A sample of 150 year of articles published in British periodicals (Lansdall-Welfare et al. 2017)

Example 3



Source: Research priorities and societal demand(Ciarli and Ràfols 2019)

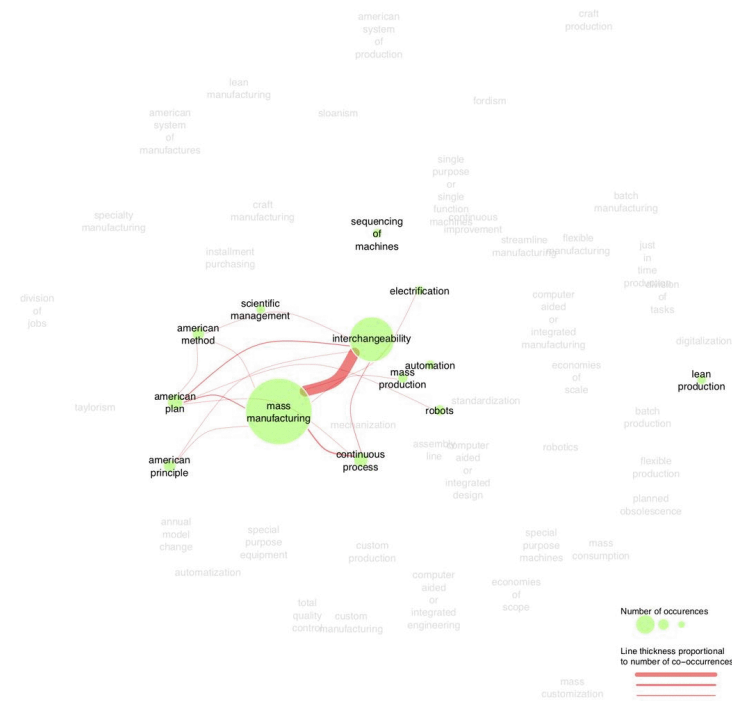
Example 4



Source: Convergence of industries using 2 million newspaper articles from 1989 to 2012 (IC = Industry convergence index, which is based on co-occurrence of industry in a sentence) (Kim et al. 2015)

Example 5

Co-occurrence of keywords in Scientific American issues: [1845,1870)



Source: Emergence of mass production in the text of Scientific American (1845-1995) (Bone and Rotolo 2020)

Types of text mining

- Bag of words
 - Words or groups of words are considered to be a feature of documents
 - The **order of words and the grammar** are not considered
 - **Computationally inexpensive** and data ready for machine learning (Document-Term Matrices)
- Syntactic parsing
 - Syntactic rules used to build the sentence are defined
 - Words or groups of words are tagged (e.g. adjectives, nouns, verbs)
 - **Computationally expensive** and complex language-dependent models
 - **In-depth analysis** of the relationships between the elements of a corpus

Text mining first steps

The `tidytext` package in R



Source: <https://www.tidytextmining.com>

How can we “tidy” unstructured data?

Support the Guardian
Available for everyone, funded by readers
[Subscribe](#) → [Contribute](#) →


Search jobs Sign in Search

The Guardian
For 200 years
News website of the year

UK edition ▾

[News](#) [Opinion](#) [Sport](#) [Culture](#) [Lifestyle](#) [More ▾](#)

[UK](#) [World](#) [Coronavirus](#) **Climate crisis** [Football](#) [Business](#) [Environment](#) [UK politics](#) [Education](#) [Society](#) [Science](#) [Tech](#) [Global development](#)



Cop26

Libby Brooks
Fri 12 Nov 2021 15:00 GMT
[f](#) [t](#) [e](#)

The good, the bad and the Irn-Bru: how Cop26 played out

The climate summit is winding down after 12 days of talks, protests, deals, frustration and a fizzy drink

[Cop26 - latest updates](#)



📷 The declarations at Cop26 left lingering questions of credibility given all were agreed outside the main UN framework. Photograph: Ben Stansall/AFP/Getty Images

On Friday, youth campaigners, indigenous leaders and Extinction Rebellion members raised a cacophony of chants and drum beats outside **Cop26**, as civil society groups inside the conference complex staged a walkout to join them.

Within the UN-controlled blue zone, delegates darted through the endless meeting halls, or hunched around laptops, as the clock counted down tense minutes to the end of the 12-day conference that is widely understood to be crucial to the future of humanity.

The deals already reached

The ragged final hours of Cop26 are a distinct contrast to the carefully choreographed first days, when world leaders arrived with bustling

Source: The Guardian, 12 November 2021

How can we “tidy” unstructured data?

“On Friday, youth campaigners, indigenous leaders and Extinction Rebellion members raised a cacophony of chants and drum beats outside Cop26, as civil society groups inside the conference complex staged a walkout to join them. Within the UN-controlled blue zone, delegates darted through the endless meeting halls, or hunched around laptops, as the clock counted down tense minutes to the end of the 12-day conference that is widely understood to be crucial to the future of humanity. The deals already reached The ragged final hours of Cop26 are a distinct contrast to the carefully choreographed first days, when world leaders arrived with bustling entourages to deliver a flourish of eye-catching pledges and, in the case of Boris Johnson, eye-watering metaphors, as the host nation’s prime minister proffered a string of clumsy analogies, likening the climate crisis to a football game and then a James Bond movie in his welcome address....”

Source: The Guardian, 12 November 2021

Tidy data

Wickham and Grolemund (2017) describe tidy data as data were

- **Observations** are in rows
- **Variables** are in columns
- Each **value** is in a cell

| country | year | cases | population |
|-------------|------|--------|------------|
| Afghanistan | 1999 | 745 | 19987071 |
| Afghanistan | 2000 | 9693 | 20595360 |
| Brazil | 1999 | 37737 | 172106362 |
| Brazil | 2000 | 80488 | 174504898 |
| China | 1999 | 218258 | 1272015272 |
| China | 2000 | 218766 | 1280425583 |

variables

| country | year | cases | population |
|-------------|------|--------|------------|
| Afghanistan | 1999 | 745 | 19987071 |
| Afghanistan | 2000 | 9693 | 20595360 |
| Brazil | 1999 | 37737 | 172106362 |
| Brazil | 2000 | 80488 | 174504898 |
| China | 1999 | 218258 | 1272015272 |
| China | 2000 | 218766 | 1280425583 |

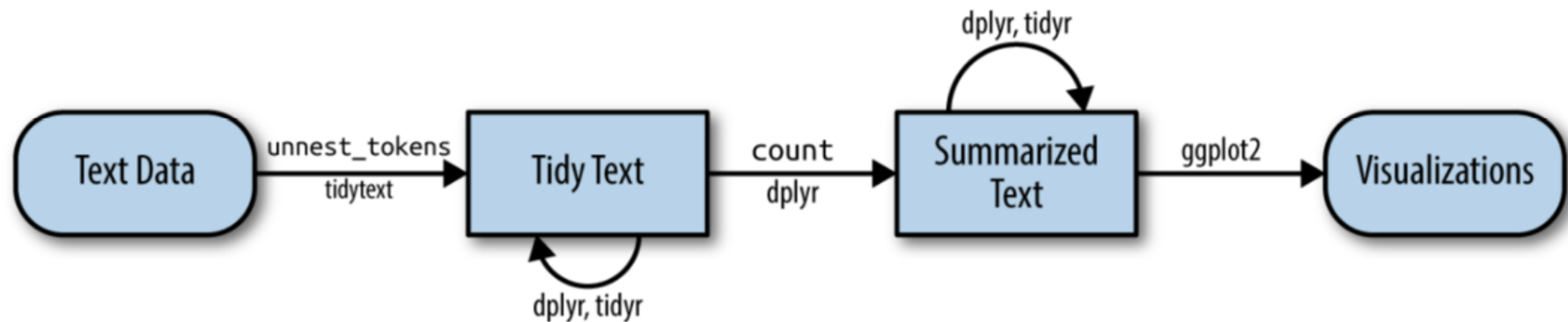
observations

| country | year | cases | population |
|-------------|------|--------|------------|
| Afghanistan | 1999 | 745 | 19987071 |
| Afghanistan | 2000 | 9693 | 20595360 |
| Brazil | 1999 | 37737 | 172106362 |
| Brazil | 2000 | 80488 | 174504898 |
| China | 1999 | 218258 | 1272015272 |
| China | 2000 | 218766 | 1280425583 |

values

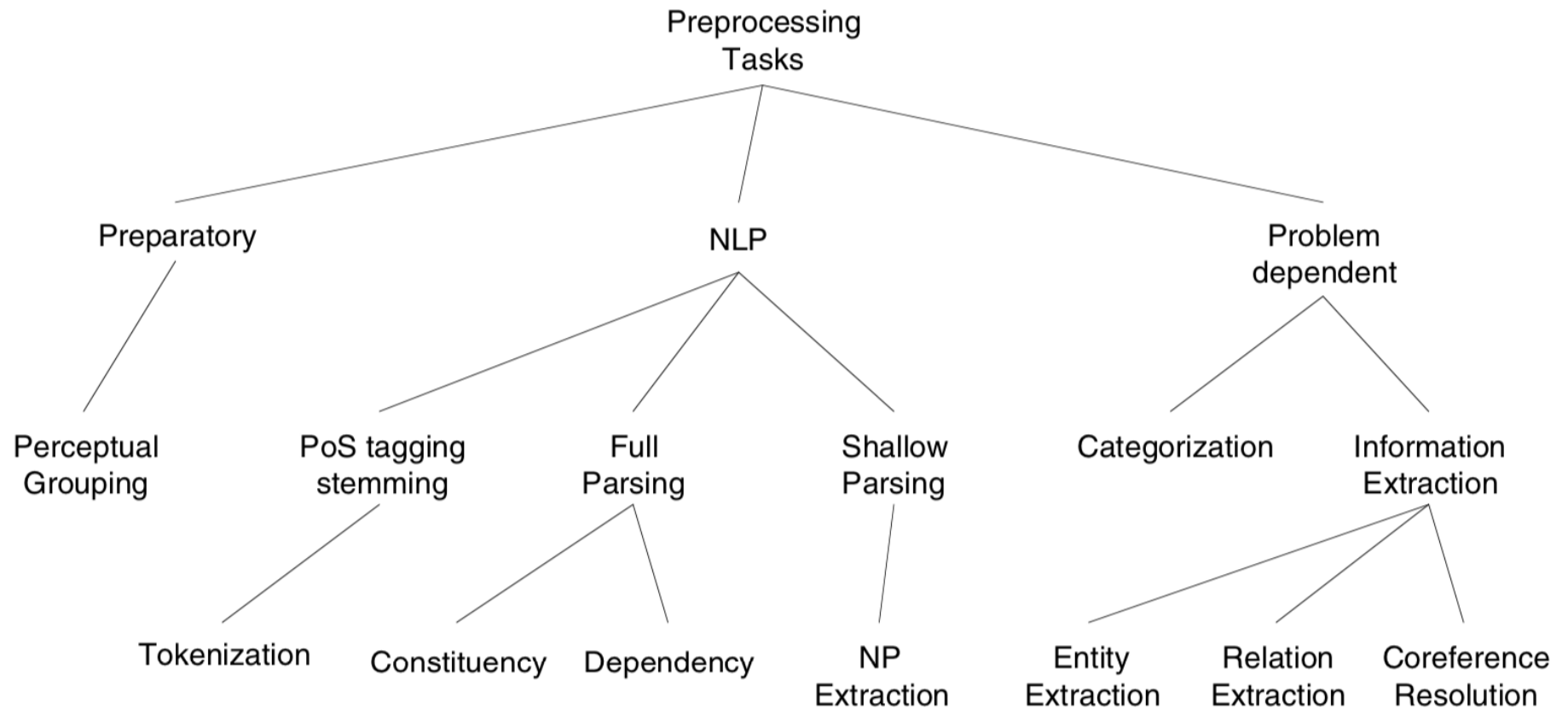
Source: Wickham and Grolemund (2017)

An overview of the process



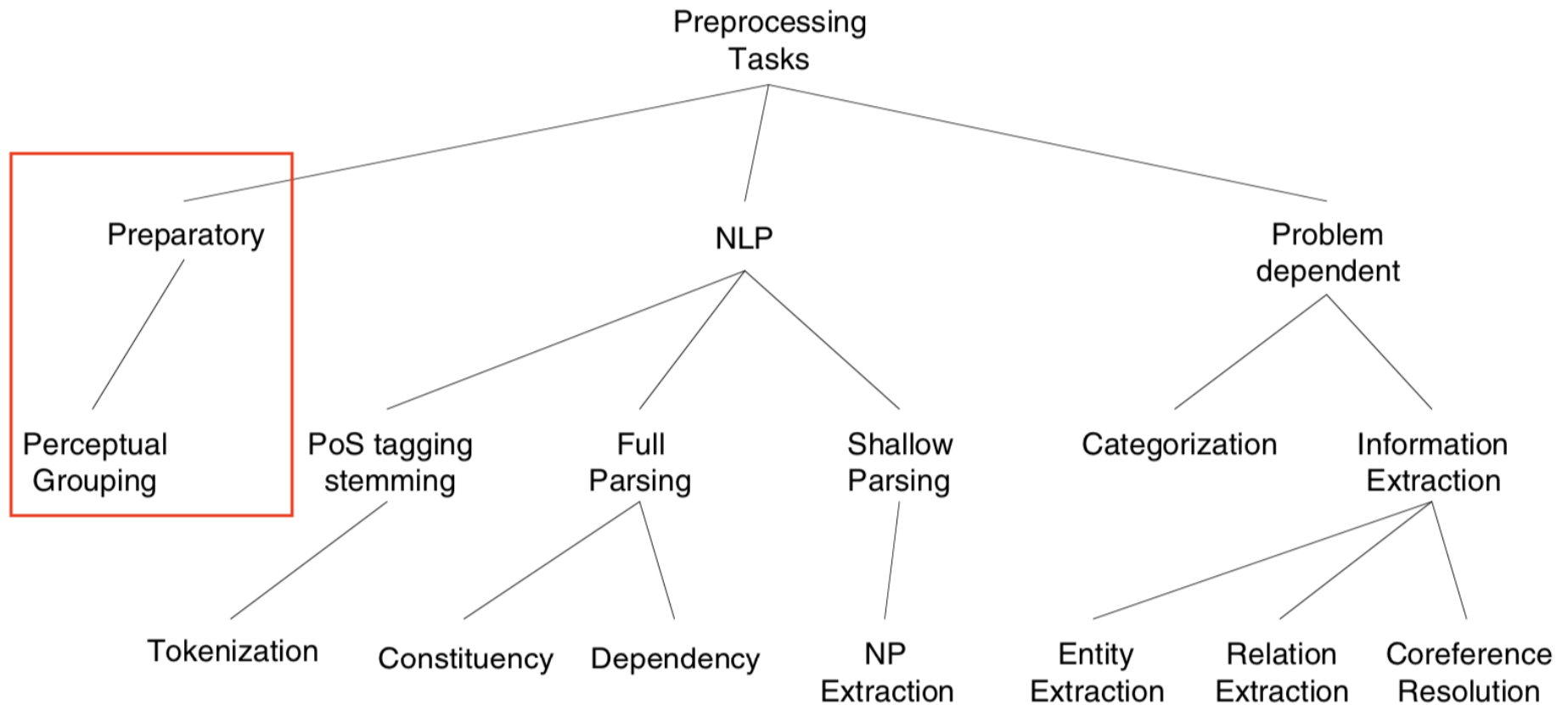
Source: Silge and Robinson (2017)

Preprocessing tasks



Source: Feldman and Sanger (2006)

Preparatory processing

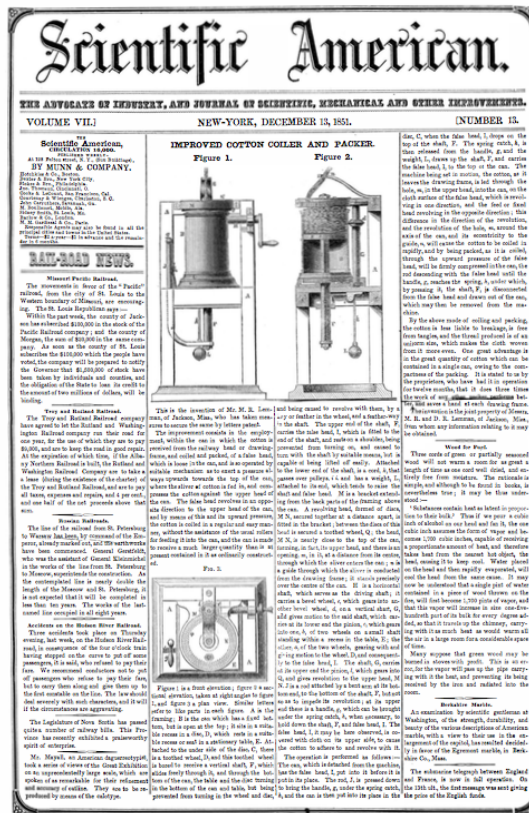


Source: Feldman and Sanger (2006)

Preparatory processing

- Text data may be in formats that are not ready for text mining
- These include:
 - PDF files
 - XML files
 - scanned images
 - recorded audio (e.g. speeches)
 - www
 - handwritten text
 - ...

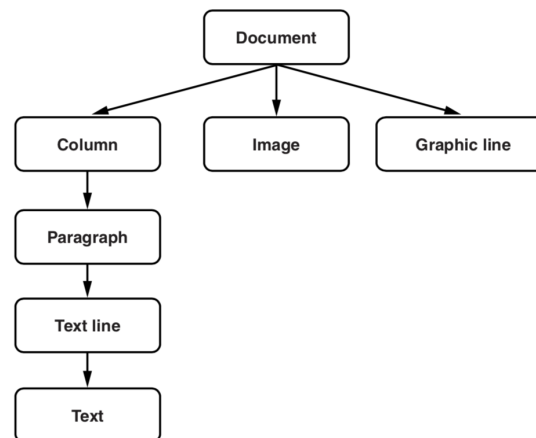
Preparatory processing



Source: Scientific American, December 1851

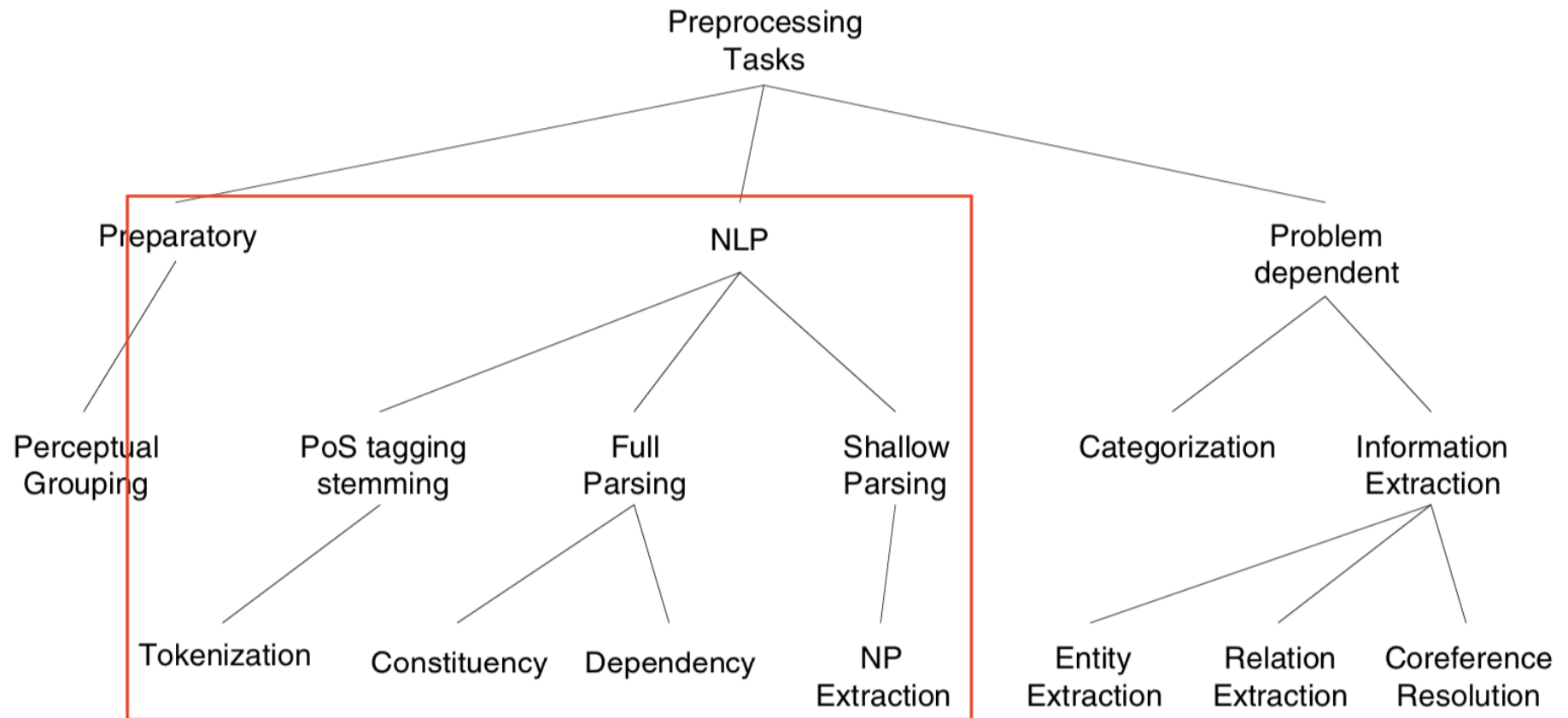
Preparatory processing

- A key preparatory task is **perceptual grouping**
- The aim is to group the primitive elements of the documents into objects of higher levels, i.e. to generate an **O-Tree**
- For example, some OCR software packages can recognise objects such as columns from scanned images



Source: O-Tree (Feldman and Sanger 2006)

Natural Language Processing (NLP)



Source: Feldman and Sanger (2006)

Natural Language Processing (NLP)

- Natural Language Processing (NLP) is an important area of an interdisciplinary research domain called **computational linguistic**
- NLP provides techniques to **transform and process text data**, so to identify patterns in these data
- NLP is particularly important for **“syntactic parsing”** than for **“bag of words”** text mining
- Three main approaches:
 1. Part-of-Speech (POS) tagging
 2. Full parsing
 3. Shallow parsing

Natural Language Processing (NLP)

Part-of-Speech (POS) tagging

- Words are categorised according to the role they play in the sentence: article, noun, verb, adjective, preposition, number, proper noun, etc.
- List of [POS tags](#)
- The tokenisation of text into words or groups of words (*we will see this in practice later*)

Firms contributing to scientific publications are likely to achieve higher financial performance

Source: <https://corenlp.run>

Natural Language Processing (NLP)

Full parsing

- The objective is to perform a full syntactical analysis of sentence identifying two elements
 - **Constituency grammars:** short phrases that convey a meaning
 - **Noun phrase (NP):** subjects or objects to a verb
 - **Verb phrase (VP):** verbs
 - **Adjective phrase (ADJP):** adjectives to qualify nouns and pronouns
 - **Adverb phrase (ADVP):** adverbs to modify nouns, verbs, or adverbs
 - **Prepositional phrase (PP):** prepositions to describe words or phrases
 - **Dependency grammars:** relationships between words (e.g. a subject and an object depend on a verb)

Natural Language Processing (NLP)

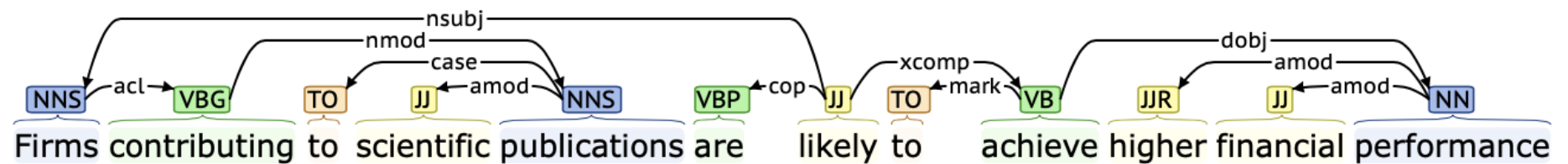
Constituency grammars

```
(ROOT
  (S
    (NP
      (NP (NNS Firms))
      (VP (VBG contributing)
        (PP (TO to)
          (NP (JJ scientific) (NNS publications))))))
    (VP (VBP are)
      (ADJP (JJ likely)
        (S
          (VP (TO to)
            (VP (VB achieve)
              (NP (JJR higher) (JJ financial) (NN performance))))))))))
```

Source: <https://corenlp.run>

Natural Language Processing (NLP)

Dependency grammars



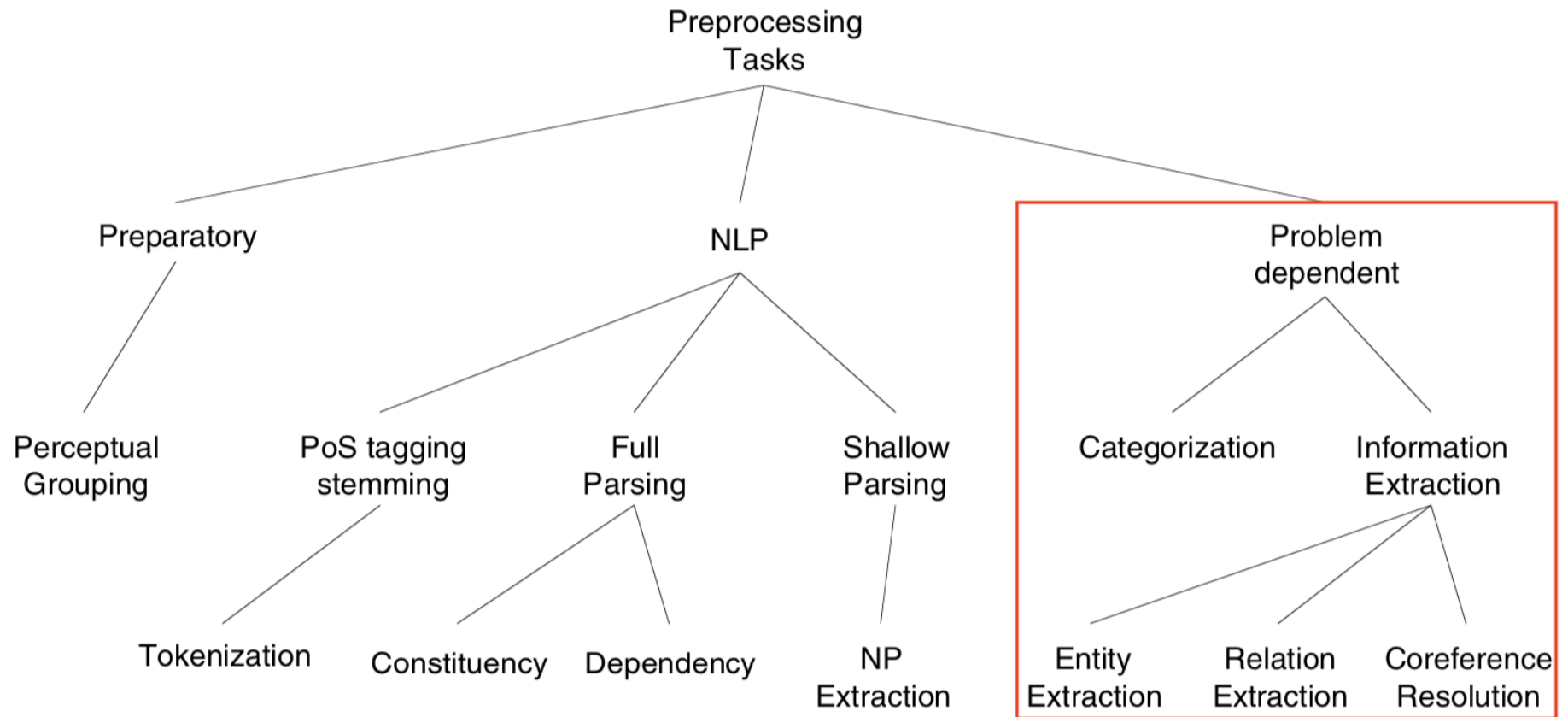
Source: <https://corenlp.run>

Natural Language Processing (NLP)

Shallow parsing

- Performing a full parsing for large text corpora could be **computationally expensive**
- Shallow parsing reduces the **depth** of the parsing analysis for the sake of **speed**
 - **Simple and short phrases** are identified
 - **Unclear and ambiguous dependency** are left unresolved
- The results of the shallow parsing are sufficient to characterise a corpus

Problem dependent processing



Source: Feldman and Sanger (2006)

Problem dependent processing

- Text categorization or classification
 - Assigning **tags** representing concepts or keywords to documents
 - Tags may have a **hierarchical structure**
 - Examples: [Medical Subject Headings](#), [JEL Classification System](#)
- Information extraction
 - Extracting information and presenting this in a **structured format** (e.g. tables, charts), beyond the simple information retrieval (e.g. identification of keywords in text)
 - **Entity extraction**: entities in the text (e.g. individuals)
 - **Relation extraction**: relationships between entities as in the text
 - **Coreference resolution**: expressions referring to the same entity (e.g. Daniel, him, his)

Our focus

- We will focus on text mining based on **bags of words**
- Techniques to **process** and **analyse** the text
 - Tokenisation
 - Lemmatisation
 - Stemming
 - tf-idf
 - Visualisation
 - Sentiment analysis
 - Topic modelling

Tokenisation

What is tokenisation?

- The objective is to break text into **meaningful elements**
- For example, we can break a document into
 - chapters
 - sections
 - paragraphs
 - sentences
 - words
 - syllables

What is tokenisation?

- A commonly used approach is to break the text into words, i.e. **tokens**
- A token could be
 - a single word (unigram)
 - two subsequent words (i.e. bigram)
 - ...
 - n subsequent words (n-gram)
- We can use token to build a **tidy dataset**, where each row reports a **token**

Tokenisation in **tidytext**

- In `tidytext`, the function `unnest_tokens` enables us to tokenise the text
- This function
 - removes punctuation
 - converts tokens into lower case
- For more options see `??unnest_tokens`

Example: (1) Data on news articles

- Text from 692 news English articles on COP26 published by newspapers on 12 November 2021

The screenshot shows the Nexis database search results for 'COP26'. The interface includes a search bar at the top with 'COP26' entered. Below the search bar, the results are displayed as a list of news articles. The first article is titled 'The unreal spectacle of COP26: COP26 has had a prepackaged feel to it, and more than a whiff of self-indulgence' and is dated 12 Nov 2021. The second article is titled 'Climate activists send a message for COP26' and is also dated 12 Nov 2021. The interface includes various filters on the left side, such as 'Narrow by' (English, 12 Nov, 2021 to 12 Nov, 2021, Newspapers) and 'Search within results' (Include, Exclude). There is also a 'Timeline' section showing a bar chart for the year 2021. The right side of the interface has a 'Sort by' dropdown set to 'Relevance' and a 'Disclaimer' link.

Source: Nexis database (available at the University of Sussex Library)

Example: (2) Data on news articles

- Each news article has a numeric id
- Text is include in the fields
 - "Title"
 - "Headline"
 - "Hlead"

```
news_articles_example.csv x
1 id,Title,Published date,Publication location,Publication,Publication type,Length,Section,Word
  count,Countries,Byline,Agg-copyright,Cite,Company,Headline,Hlead,Publication,Publication type,Pub-copyright,Show,Term,Ticker
2 1,"The unreal spectacle of COP26;COP26 has had a prepackaged feel to it, and more than a whiff of self-indulgence","November 12, 2021
Friday",Non-jurisdictional,Breaking News from globeandmail.com,News,825,,825,,Konrad Yakabuski,,REGIE NATIONALE DES USINES RENAULT SA,REGIE
NATIONALE DES USINES RENAULT SAGRAHAM HOLDINGS CO,"The unreal spectacle of COP26COP26 has had a prepackaged feel to it, and more than a whiff of
self-indulgence",LeadAs the United Nations COP26 climate conference hurtled toward its climax this week, it was hard not to get the impression that
the 40,000 official participants gathered in Glasgow for what many billed as the last chance to save the planet were playing a game of virtual
reality.Over the course of two weeks, the world bore witness to a flurry of declarations followed by prolonged sessions of mutual backslapping among
the signatories, and admonishments from climate activists about the utter inadequacy of said commitments.Texte/TextIn Glasgow, more than 40
countries, including Canada, agreed to ""accelerate a transition away from unabated coal power generation,"" But China, the United States and India
– which together account for about 70 per cent of the coal burned to generate electricity – were conspicuously absent from the deal. As was
Australia, whose economy depends more than most on coal exports.Another non-binding agreement involving 130 countries – Canada included – committed
signatories to ending deforestation by 2030. The emptiness of the undertaking was apparent as Indonesian officials specified that it would be
""obviously inappropriate and unfair"" to adhere to the pledge at the expense of economic development.Another caveat-filled agreement to which
Canada signed on involved a pledge to ""work toward all sales of new cars and vans being zero emission globally by 2040, and by no later than 2035
in leading markets,"" But four of the world's largest automakers – Volkswagen, Toyota, Renault-Nissan and Hyundai-Kia – were not among the
signatories. Neither was China.Speaking of which, Chinese President Xi Jinping skipped the Glasgow gabfest altogether rather than face a public
scolding about his country's ""dangerous lack of urgency,"" as former U.S. president Barack Obama described it. Mr. Xi refused to be lectured to by
leaders from the developed countries that have a century-long emissions head start over China – now the world's biggest emitter of greenhouse
gases.John Kerry, U.S. President Joe Biden's climate envoy, nevertheless handed Mr. Xi a propaganda victory with a joint declaration in which the
United States and China agreed to co-operate on climate-related efforts. But the agreement – which according to The Washington Post, was ""the
product of nearly three dozen negotiating sessions"" between U.S. and Chinese diplomats over the course of the past year – was thin gruel even by UN
standards.By week's end, British Prime Minister Boris Johnson, seeking to bask in the glow of the Glasgow summit to distract from the domestic
political scandal now engulfing his government, was nudging the parties toward a face-saving final agreement that would preserve the notion that
COP26 had been worth its weight in CO2 emissions despite appearances to the contrary.The whole event had a prepackaged feel to it, and more than a
whiff of self-indulgence among the participants who live for such gatherings and the networking opportunity they represent.It was as if, inside the
Glasgow bubble, no one had even heard of geopolitics or technology, even though they remain the biggest obstacles to tackling climate change. In
reality, the energy transition is akin to a high-stakes game of chess that threatens to reconfigure power structures across the globe.Even countries
dependent on imported fossil fuels that theoretically stand to benefit from the phasing out of oil, natural gas and coal are unwilling to risk the
```

Example: (3) Reading the data

- We load the `readr`, `tidyverse`, `tidytext`, and `ggplot` packages
- We store our data into R as a dataframe/tibble
- We focus on the text reported in the field “Title”

```
library(readr)
library(ggplot2)
library(tidyverse)
library(tidytext)

my_text <- read_csv("news_articles_example.csv") %>%
  select(id, Title)
```


Example: (3) Reading the data

```
print(my_text, n = 6)
```

```
## # A tibble: 692 × 2
##       id Title
##   <dbl> <chr>
## 1     1 The unreal spectacle of COP26;COP26 has had a prepackaged feel to it, a...
## 2     2 Climate activists send a message for COP26
## 3     3 COP26: Nicola Sturgeon urges Boris Johnson to return and use position t...
## 4     4 COP26: Nicola Sturgeon urges Boris Johnson to return and use position t...
## 5     5 COP26: Police Scotland arrested eight people on penultimate day of Glas...
## 6     6 COP26: Top 10 bizzare moments of the Glasgow climate talks ranked
## # ... with 686 more rows
```

Example: (4) Tokenisation (unigrams)

We tokenise each sentence into unigrams

```
my_text_uni <- my_text %>%  
  unnest_tokens(output = word, input = Title)
```

```
print(my_text_uni, n = 6)
```

```
## # A tibble: 7,925 × 2  
##       id word  
##   <dbl> <chr>  
## 1     1 the  
## 2     1 unreal  
## 3     1 spectacle  
## 4     1 of  
## 5     1 cop26  
## 6     1 cop26  
## # ... with 7,919 more rows
```

Example: (4) Tokenisation (bigrams)

We could also tokenise into n-grams, for example bigrams

```
my_text_bi <- my_text %>%  
  unnest_tokens(output = bigram, input = Title,  
                token = "ngrams", n = 2)  
  
print(my_text_bi, n = 6)
```

```
## # A tibble: 7,238 × 2  
##       id bigram  
##   <dbl> <chr>  
## 1     1 1 the unreal  
## 2     1 1 unreal spectacle  
## 3     1 1 spectacle of  
## 4     1 1 of cop26  
## 5     1 1 cop26 cop26  
## 6     1 1 cop26 has  
## # ... with 7,232 more rows
```

Example: (4) Tokenisation (trigrams)

... or trigrams

```
my_text_tri <- my_text %>%  
  unnest_tokens(output = trigram, input = Title,  
                token = "ngrams", n = 3)
```

```
print(my_text_tri, n = 6)
```

```
## # A tibble: 6,557 × 2  
##       id trigram  
##   <dbl> <chr>  
## 1     1 1 the unreal spectacle  
## 2     1 1 unreal spectacle of  
## 3     1 1 spectacle of cop26  
## 4     1 1 of cop26 cop26  
## 5     1 1 cop26 cop26 has  
## 6     1 1 cop26 has had  
## # ... with 6,551 more rows
```

Example: (4) Tokenisation (characters)

... or sequences of characters (five in the case below)

```
my_text_char <- my_text %>%  
  unnest_tokens(output = character_shingles, input = Title,  
                token = "character_shingles", n = 5)
```

```
print(my_text_char, n = 6)
```

```
## # A tibble: 36,769 × 2  
##       id character_shingles  
##   <dbl> <chr>  
## 1     1 1 theun  
## 2     1 1 heunr  
## 3     1 1 eunre  
## 4     1 1 unrea  
## 5     1 1 nreal  
## 6     1 1 reals  
## # ... with 36,763 more rows
```

Example: (5) POS tagging

```
my_text_uni_pos <- my_text_uni %>%  
  left_join(parts_of_speech, by = "word")
```

```
print(my_text_uni_pos, n = 6)
```

```
## # A tibble: 15,047 × 3  
##       id word      pos  
##   <dbl> <chr>    <chr>  
## 1     1 the      Definite Article  
## 2     1 the      Adverb  
## 3     1 unreal   Adjective  
## 4     1 spectacle Noun  
## 5     1 of        Noun  
## 6     1 of        Preposition  
## # ... with 15,041 more rows
```


Example: (5) POS tagging

Show entries

Search:

| pos | n |
|----------------------------------|----------------------------------|
| <input type="text" value="All"/> | <input type="text" value="All"/> |
| Noun | 4953 |
| | 1521 |
| Adverb | 1456 |
| Verb (usu participle) | 1449 |
| Preposition | 1395 |

Showing 1 to 5 of 13 entries

Previous 2 3 Next

Example: (6) Counting words

Let's focus on unigrams and count the number of words

```
my_text_uni_count <- my_text_uni %>%  
  count(word, sort = T)
```

```
print(my_text_uni_count, n = 6)
```

```
## # A tibble: 2,436 × 2  
##   word      n  
##   <chr>   <int>  
## 1 to      300  
## 2 the     188  
## 3 of      161  
## 4 cop26   160  
## 5 on      148  
## 6 climate 135  
## # ... with 2,430 more rows
```

Example: (6) Counting words

Show entries

Search:

| word | n |
|----------------------------------|----------------------------------|
| <input type="text" value="All"/> | <input type="text" value="All"/> |
| to | 300 |
| the | 188 |
| of | 161 |
| cop26 | 160 |
| on | 148 |

Showing 1 to 5 of 2,436 entries

Previous 2 3 4 5 ... 488 Next

Example: (6) Counting words

Basic descriptive statistics

```
summarise(my_text_uni_count,  
          num_words = n(),  
          mean = mean(n),  
          sd     = sd(n),  
          min   = min(n),  
          max   = max(n))
```

```
## # A tibble: 1 × 5  
##   num_words mean    sd  min  max  
##      <int> <dbl> <dbl> <int> <int>  
## 1      2436  3.25  11.1     1   300
```

Example: (6) Counting words

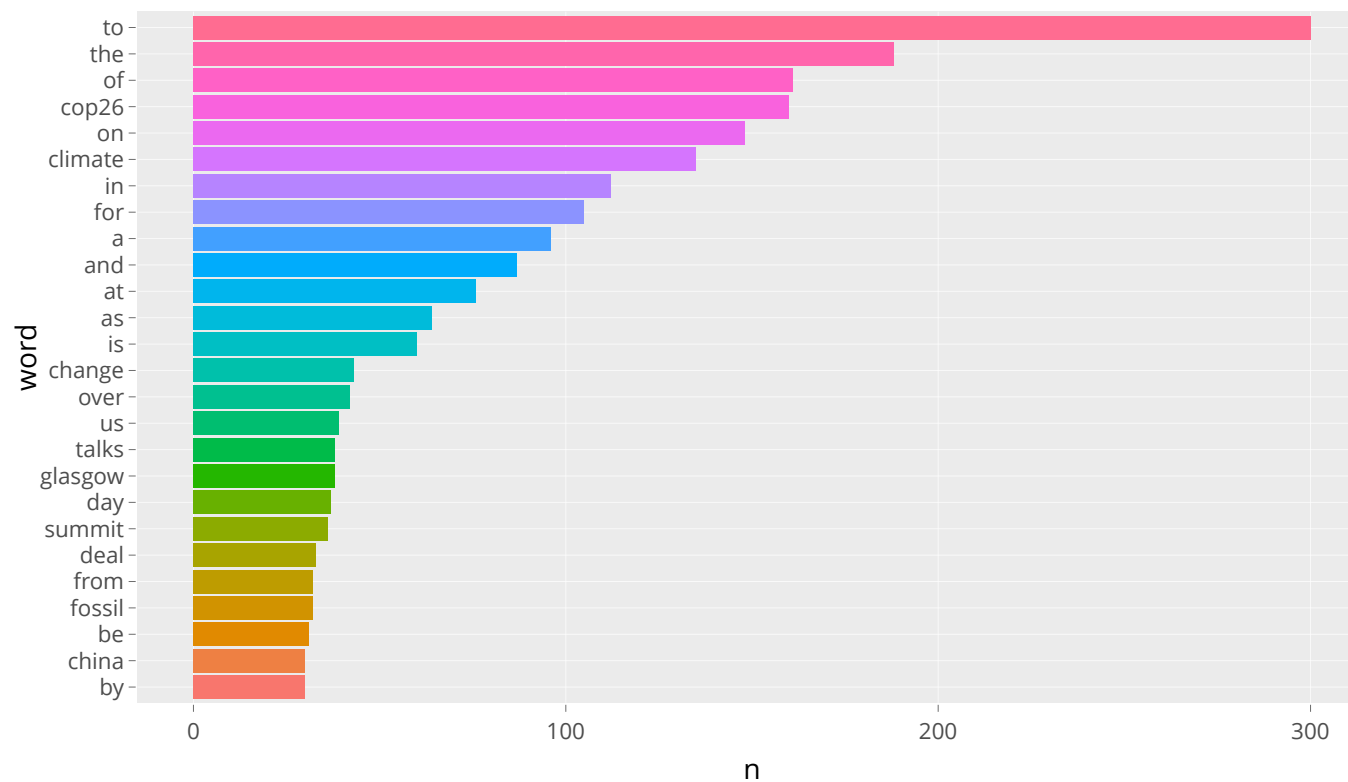
We can plot the most frequent words representing the 1% of the total sample

```
min_occur <- quantile(my_text_uni_count$n, 0.99)
```

```
g <- my_text_uni_count %>%  
  filter(n >= min_occur) %>%  
  mutate(word = reorder(word, n)) %>%  
  ggplot(aes(x = word, y = n, fill = word)) +  
  geom_col() +  
  theme(legend.position = "none") +  
  coord_flip()
```

Example: (6) Counting words

- Many words are not particularly relevant (e.g. to, the, in)
- These terms are called stopwords



Example: (7) Removing stopwords

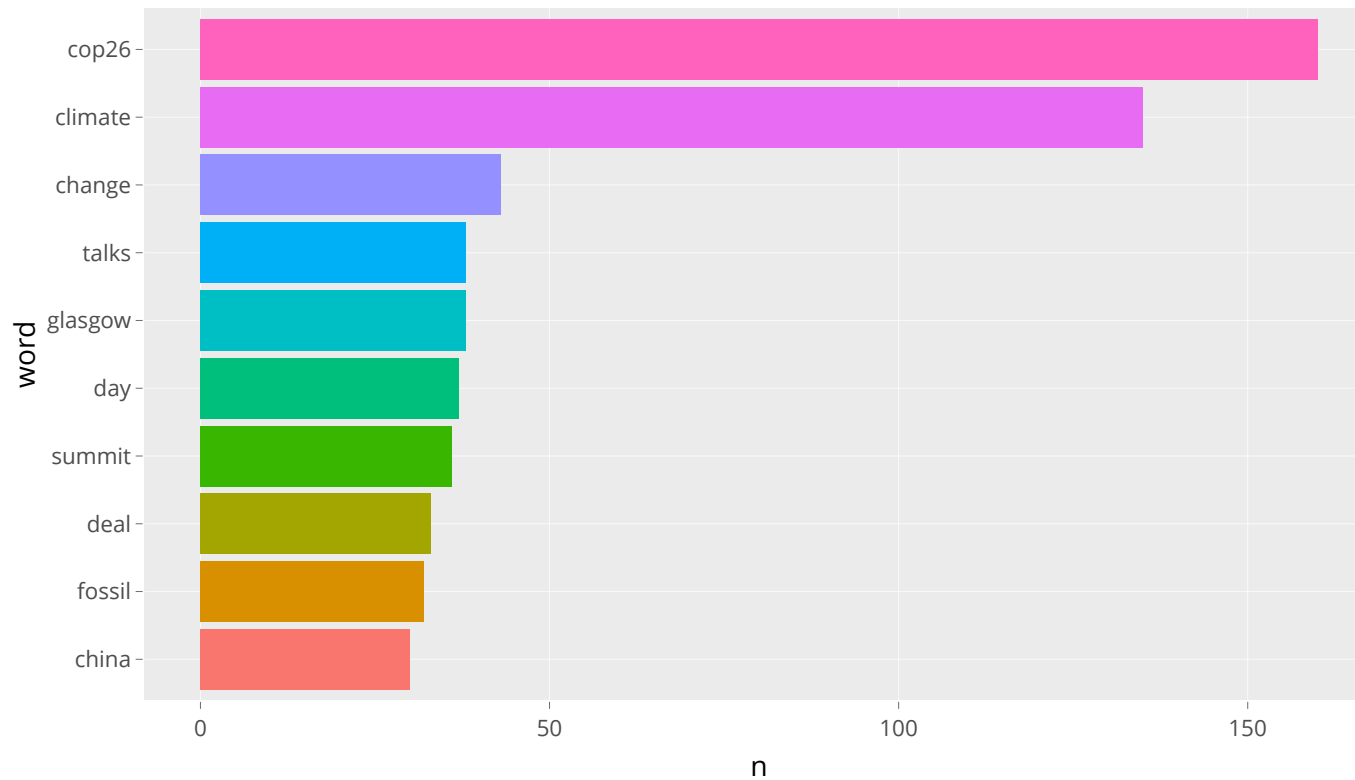
```
data(stop_words)
```

```
my_text_uni <- my_text_uni %>%  
  anti_join(stop_words)
```

```
my_text_uni_count <- my_text_uni %>%  
  count(word, sort = T)
```

```
g <- my_text_uni_count %>%  
  filter(n >= min_occur) %>%  
  mutate(word = reorder(word, n)) %>%  
  ggplot(aes(x = word, y = n, fill = word)) +  
  geom_col() +  
  theme(legend.position = "none") +  
  coord_flip()
```

Example: (7) Removing stopwords



Example: (8) Removing numbers

- Numbers may not be meaningful for your analysis
- We can remove numbers converting text into numeric format

```
my_text_uni <- my_text_uni %>%  
  mutate(word_numeric = as.numeric(word))
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```


Example: (8) Removing numbers

Show entries

Search:

| id | | word | word_numeric |
|----------------------------------|-------------|----------------------------------|---|
| <input type="text" value="All"/> | | <input type="text" value="All"/> | <input type="text" value="All"/> |
| 1 | unreal | | |
| 1 | spectacle | | |
| 1 | cop26 | | |
| 1 | cop26 | | |
| 1 | prepackaged | | |
| Showing 1 to 5 of 4,985 entries | | Previous | <input type="text" value="1"/> 2 3 4 5 ... 997 Next |

Example: (8) Removing numbers

- **NAs** in the table are word, so we can filter those only
- We can reduce the threshold to the most frequent words representing the 2% of the total sample so to explore more words

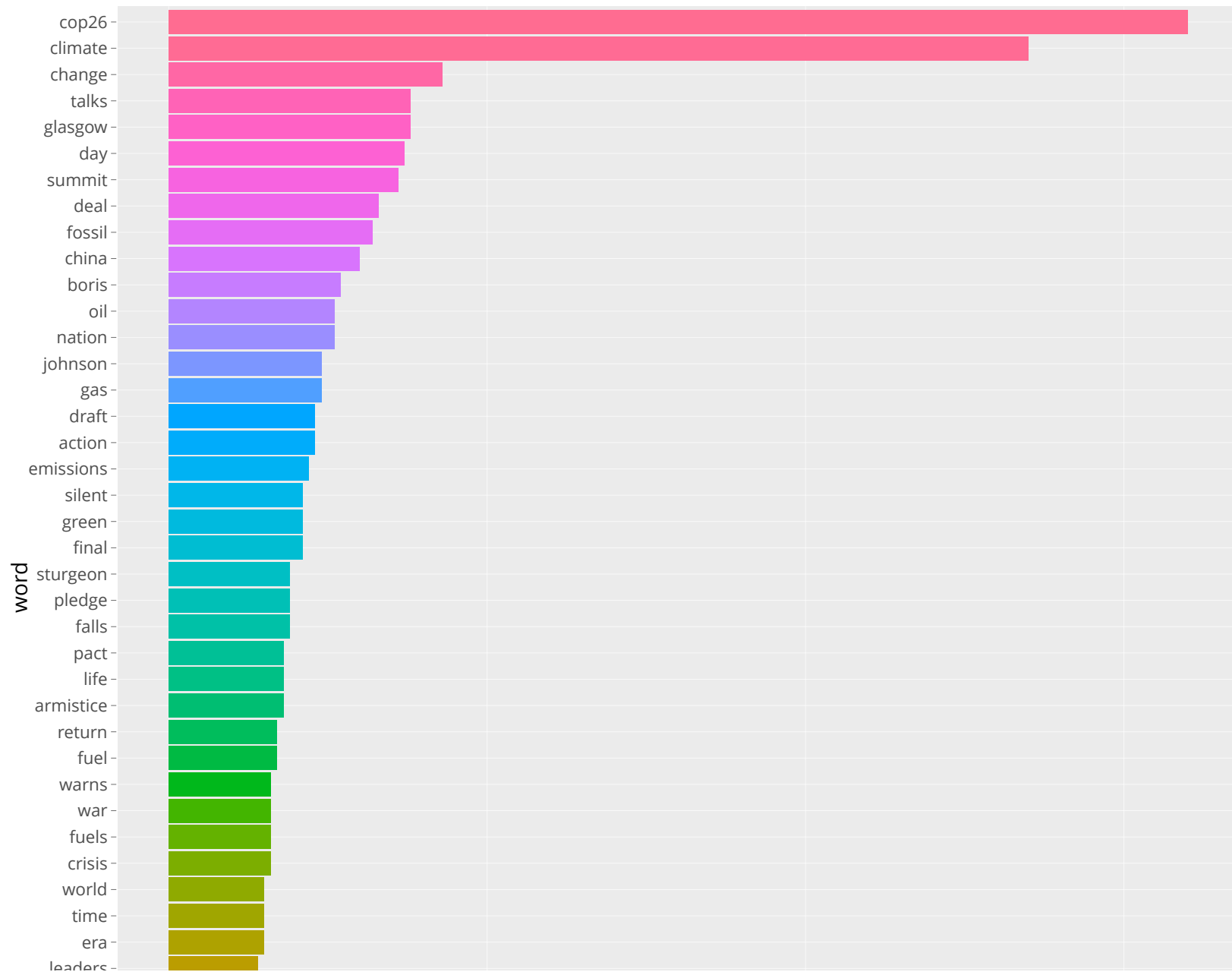
```
my_text_uni <- my_text_uni %>%  
  filter(is.na(word_numeric))
```

```
my_text_uni_count <- my_text_uni %>%  
  count(word, sort = T)
```

```
min_occur <- quantile(my_text_uni_count$n, 0.98)
```

```
g <- my_text_uni_count %>%  
  filter(n >= min_occur) %>%  
  mutate(word = reorder(word, n)) %>%  
  ggplot(aes(x = word, y = n, fill = word)) +  
  geom_col() +  
  theme(legend.position = "none") +  
  coord_flip()
```

Example: (8) Removing numbers



Example: (9) Additional stopwords

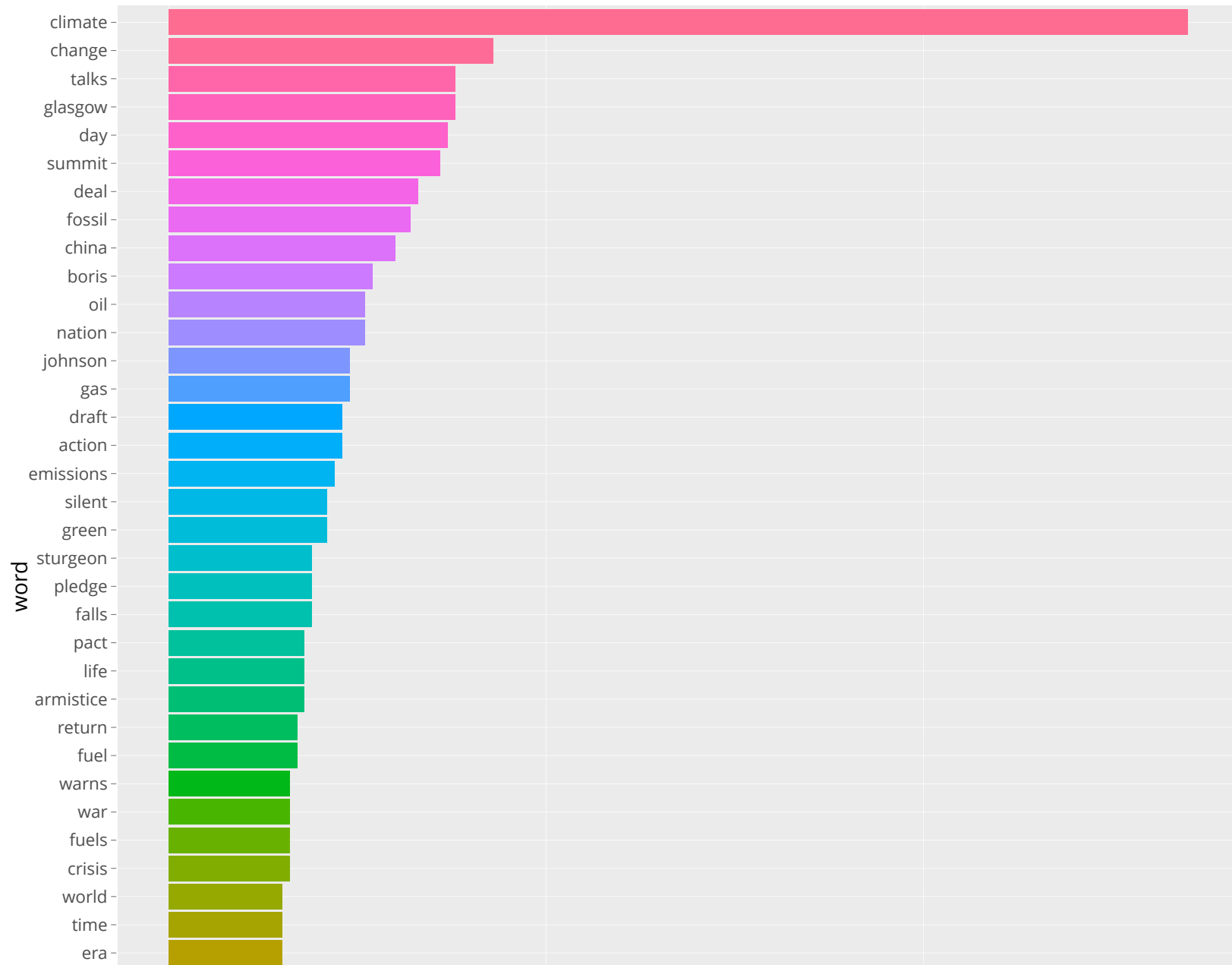
- In some case, you may want to remove words that are not particularly relevant

```
my_stop_words <- tibble(word = c("cop26", "final", "news"),  
                        lexicon = "mywords")
```

```
my_text_uni<- my_text_uni %>%  
  anti_join(my_stop_words)
```

```
g <- my_text_uni %>%  
  count(word, sort = T) %>%  
  filter(n >= min_occur) %>%  
  mutate(word = reorder(word, n)) %>%  
  ggplot(aes(x = word, y = n, fill = word)) +  
  geom_col() +  
  theme(legend.position = "none") +  
  coord_flip()
```

Example: (9) Additional stopwords



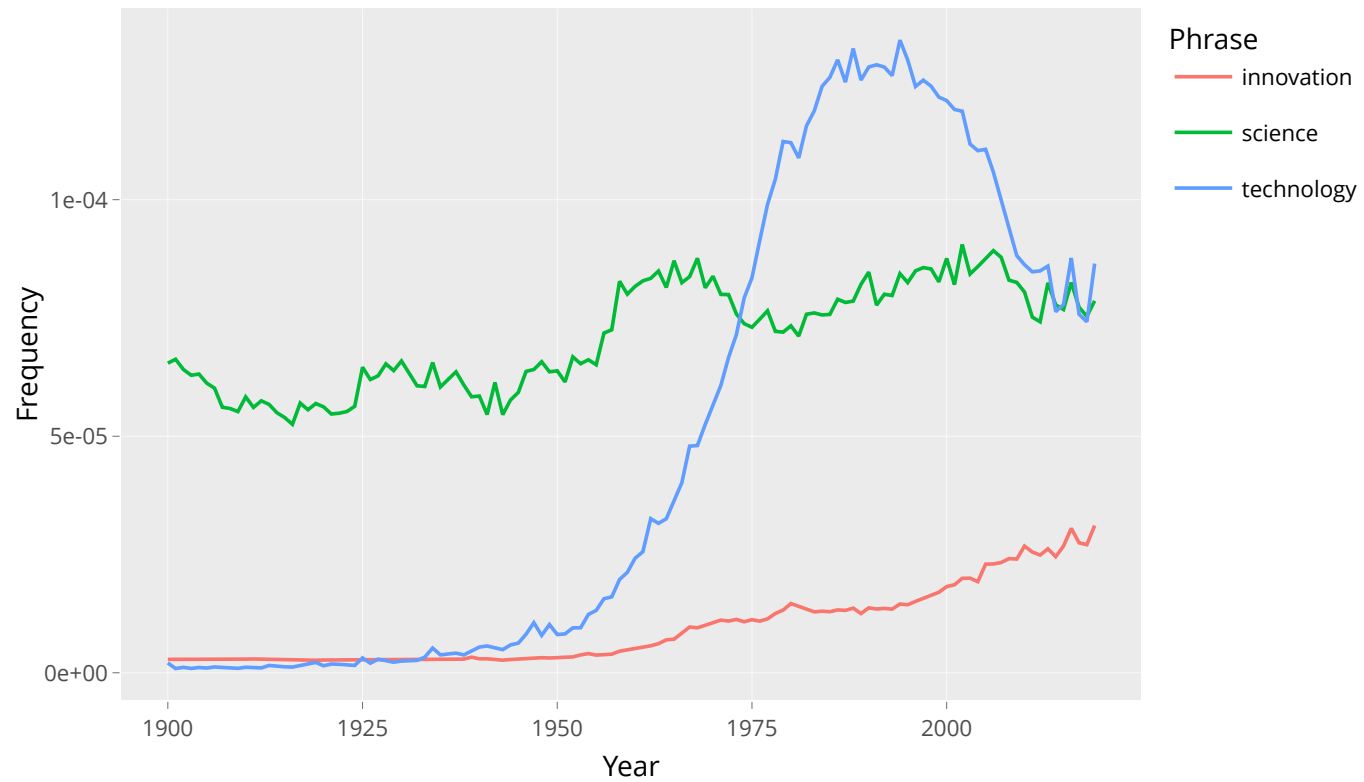
Tokenised data

- [Google Books](#) provides access to tokenised data for several millions of books
- Data structure
 - token (n-gram)
 - number of occurrences (match_count)
 - number of occurrences in distinct books (volume_count)
- **Warning:** The database includes a large amount of data (several gigabytes)
- Google Books [query interface](#)
- We can use the `ngramr` package to gather data

Example: (10) Google Books

```
library(ngramr)
library(ggplot2)
googlebook_data <- ngram(c("science", "technology", "innovation"),
                          smoothing = 0, year_start = 1900)
g <- ggplot(googlebook_data,
            aes(x = Year, y = Frequency, colour= Phrase)) +
  geom_line()
```

Example: (10) Google Books



Lemmatisation

What is lemmatisation?

- Individual tokens are reduced to their base form, called “lemma”
 - “am,” “are,” “is” -> “be”
 - “cars” -> “car”
 - “arrived” -> “arrive”
 - ...
- **Lemmatisation** normalise text (e.g. counting)
- This processing is useful for **sentiment analysis**

Example: (1) Unigrams lemmatisation

- We can use the package `textstem`
- We have already remove stopwords, numbers, and additional stopwords from the list of unigrams

```
library(textstem)
my_text_uni <- my_text_uni %>%
  mutate(word_lemma = textstem::lemmatize_words(word))
```

Example: (1) Unigrams lemmatisation

Show entries

Search:

| id | word | word_numeric | word_lemma |
|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| <input type="text" value="All"/> | <input type="text" value="All"/> | <input type="text" value="All"/> | <input type="text" value="All"/> |
| 1 | unreal | | unreal |
| 1 | spectacle | | spectacle |
| 1 | prepackaged | | prepackaged |
| 1 | feel | | feel |
| 1 | whiff | | whiff |

Showing 1 to 5 of 4,730 entries

Previous 2 3 4 5 ... 946 Next

Example: (2) Counting lemmas

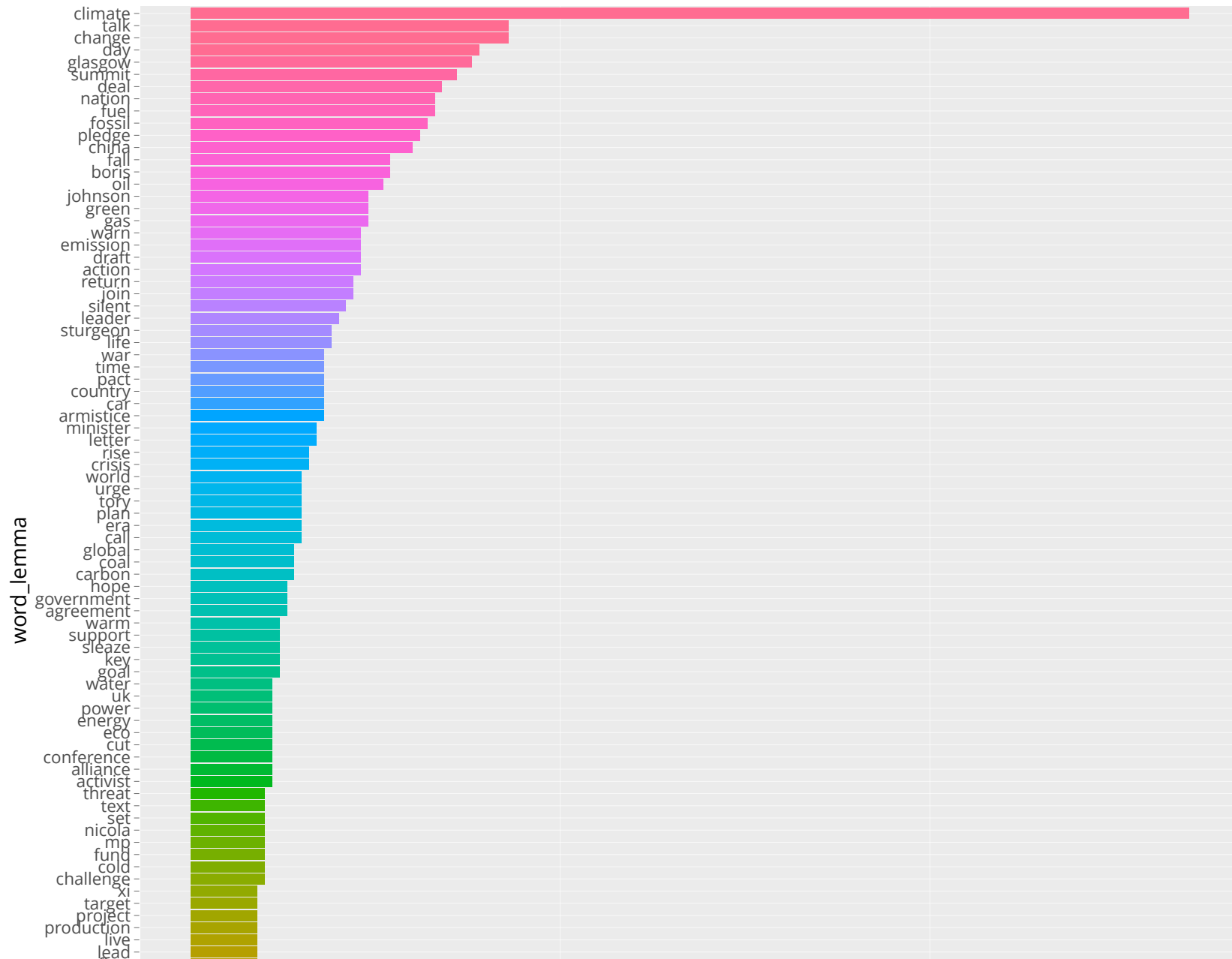
- We can now count lemmas and plot the results

```
my_text_uni_count <- my_text_uni %>%  
  count(word_lemma, sort = T)
```

```
min_occur <- quantile(my_text_uni_count$n, 0.95)
```

```
g <- my_text_uni_count %>%  
  filter(n >= min_occur) %>%  
  mutate(word_lemma = reorder(word_lemma, n)) %>%  
  ggplot(aes(x = word_lemma, y = n, fill = word_lemma)) +  
  geom_col() +  
  theme(legend.position = "none") +  
  coord_flip()
```

Example: (2) Counting lemmas



Stemming

What is stemming?

- Stemming is similar to the lemmatisations
- It removes the suffix of a word
 - “playing” -> “play”
 - “consultant” -> “consult”
 - “magically” -> “magic”
 - ...

Example: (1) Stemming unigrams

- We can use the package `SnowballC`
- We have already removed stopwords, numbers, and additional stopwords from the list of unigrams

```
library(SnowballC)
my_text_uni <- my_text_uni %>%
  mutate(word_stem = wordStem(word))
```

Example: (1) Stemming unigrams

Show entries

Search:

| id | word | word_numeric | word_lemma | word_stem |
|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| <input type="text" value="All"/> | <input type="text" value="All"/> | <input type="text" value="All"/> | <input type="text" value="All"/> | <input type="text" value="All"/> |
| 1 | unreal | | unreal | unreal |
| 1 | spectacle | | spectacle | spectacl |
| 1 | prepackaged | | prepackaged | prepackag |
| 1 | feel | | feel | feel |
| 1 | whiff | | whiff | whiff |

Showing 1 to 5 of 4,730 entries

Previous 2 3 4 5 ... 946 Next

Example: (2) Counting stems

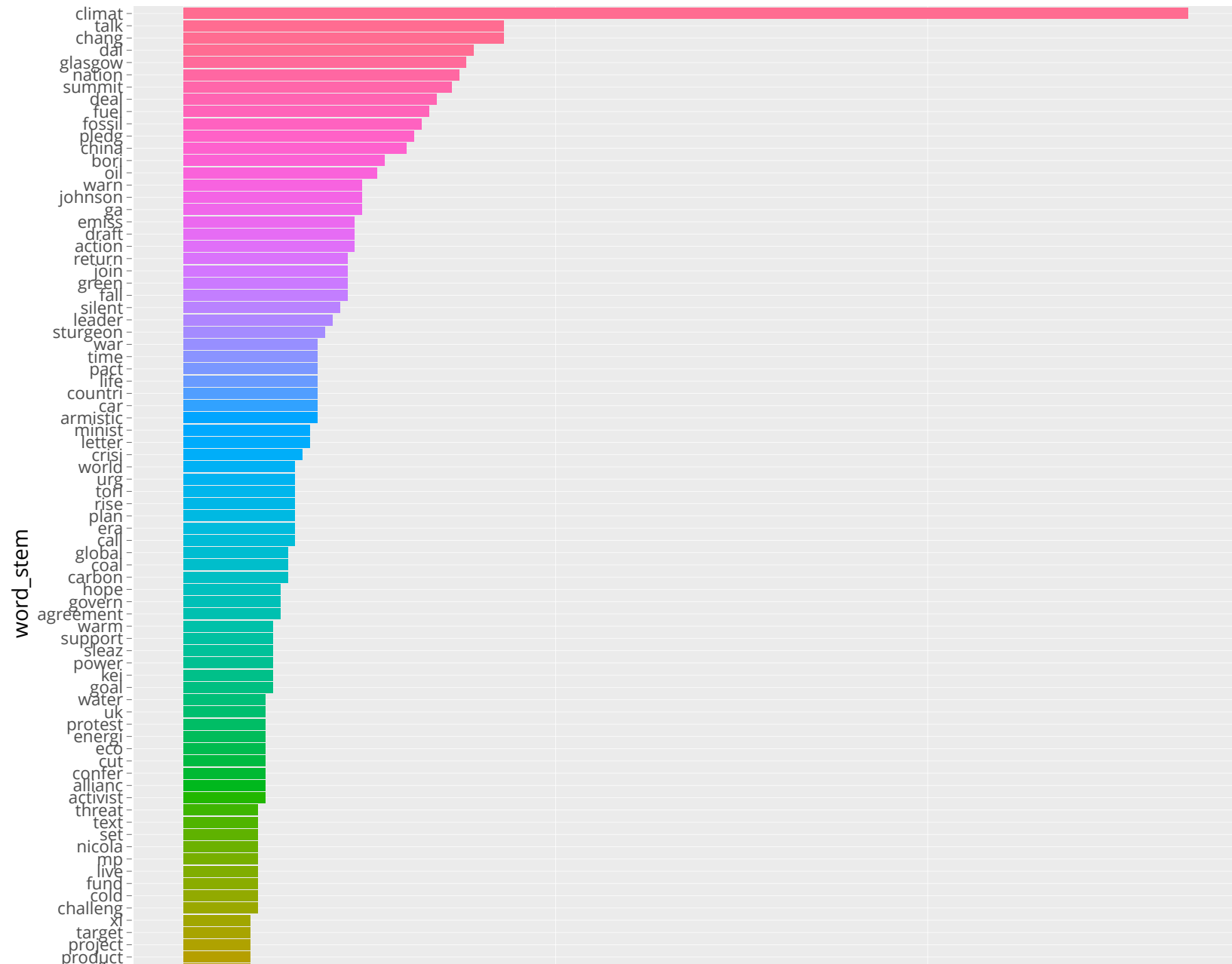
- We can now count words/stems and plot the results

```
my_text_uni_count <- my_text_uni %>%  
  count(word_stem, sort = T)
```

```
min_occur <- quantile(my_text_uni_count$n, 0.95)
```

```
g <- my_text_uni_count %>%  
  filter(n >= min_occur) %>%  
  mutate(word_stem = reorder(word_stem, n)) %>%  
  ggplot(aes(x = word_stem, y = n, fill = word_stem)) +  
  geom_col() +  
  theme(legend.position = "none") +  
  coord_flip()
```

Example: (2) Counting stems



tf-idf

tf-idf

- Term frequency provides an indication of the most frequent words in a corpus
- Some words may be not very frequent in a corpus, but they may be particular for some documents in the corpus
- We can identify these words using the **term frequency-inverse document frequency** indicator

$$\text{tf-idf}_{i,d} = \text{tf}_{i,d} * \text{idf}_i$$

$$\text{tf-idf}_{i,d} = \frac{\text{n of times term } i \text{ appears in document } d}{\text{n of terms in the document } d} * \ln \left(\frac{\text{n of documents}}{\text{n of documents containing the term } i} \right)$$

Example: (1) Tokenisation, stopwords, numbers

- Text from 651 news articles on COVID-19 published by UK newspapers from 23 November 2020 to 1 December 2020
- We use the lead paragraph (“Hlead”), we tokenize data into unigram, we remove stopwords and numbers
- Our corpus include 242,831 id-unigram pairs

```
my_text_uni<- read_csv("news_articles_example.csv") %>%  
  select(id, Hlead) %>%  
  unnest_tokens(output = word, input = Hlead) %>%  
  anti_join(stop_words) %>%  
  mutate(word_numeric = as.numeric(word)) %>%  
  filter(is.na(word_numeric)) %>%  
  select(-word_numeric)
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

Example: (2) term frequency (tf)

We can calculate the tf (term frequency by document)

```
article_words <- my_text_uni %>%  
  count(id, word, sort = T) %>%  
  ungroup()
```

```
head(article_words)
```

```
## # A tibble: 6 × 3  
##       id word      n  
##   <dbl> <chr>  <int>  
## 1    659 pwa      84  
## 2    659 editor    70  
## 3    659 bar      66  
## 4    170 water     57  
## 5    124 parties   52  
## 6    124 paris    50
```


Example: (3) term frequency (tf)

```
total_words <- article_words %>%  
  group_by(id) %>%  
  summarize(total = sum(n))
```

```
print(total_words, n = 6)
```

```
## # A tibble: 692 × 2  
##       id total  
##   <dbl> <int>  
## 1     1    381  
## 2     2    378  
## 3     3    558  
## 4     4    558  
## 5     5    346  
## 6     6    722  
## # ... with 686 more rows
```

Example: (4) term frequency (tf)

We remove document with only a single word

```
article_words <- left_join(article_words, total_words) %>%  
  filter(total > 1) %>%  
  mutate(word_freq = n/total)
```

```
print(article_words, n = 6)
```

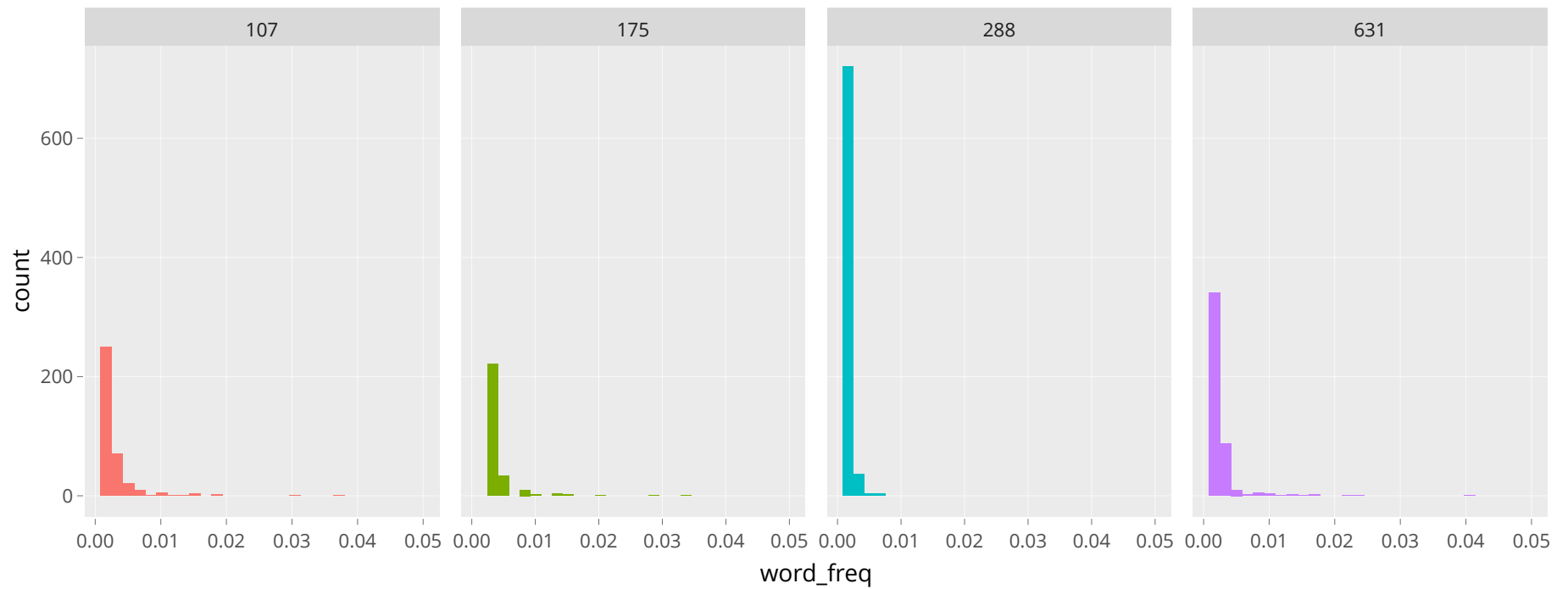
```
## # A tibble: 131,673 × 5  
##       id word      n total word_freq  
##   <dbl> <chr>  <int> <int>    <dbl>  
## 1    659 pwa      84  1168    0.0719  
## 2    659 editor    70  1168    0.0599  
## 3    659 bar      66  1168    0.0565  
## 4    170 water    57  1053    0.0541  
## 5    124 parties  52  2040    0.0255  
## 6    124 paris   50  2040    0.0245  
## # ... with 131,667 more rows
```

Example: (5) term frequency (tf)

Let's look at the term frequency distribution of three random articles

```
g <- article_words %>%  
  filter(id == 107 | id == 175 | id == 288 | id == 631) %>%  
  ggplot(aes(word_freq, fill = as.character(id))) +  
  geom_histogram() +  
  xlim(NA, 0.05) +  
  theme(legend.position = "none") +  
  facet_wrap(~id, ncol = 4)
```

Example: (6) term frequency (tf)



Example: (7) tf-idf

- Some words are very frequent in a number of documents (articles in this case)
- tf-idf helps us to identify words that are able to characterize documents
- In `tidytext` the function `bind_tf_idf` calculate the tf-idf scores starting from a tidy text dataset: one row per token per document including the information about term frequency
- Check the help guide `??bind_tf_idf`

Example: (8) tf-idf

Let's start from reading the data again

```
article_words_tfidf <- read_csv("news_articles_example.csv") %>%  
  select(id, Hlead) %>%  
  unnest_tokens(output = word, input = Hlead) %>%  
  anti_join(stop_words) %>%  
  mutate(word_numeric = as.numeric(word)) %>%  
  filter(!is.na(word_numeric)) %>%  
  select(-word_numeric) %>%  
  count(id, word, sort = T) %>%  
  ungroup() %>%  
  bind_tf_idf(word, id, n)
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

Example: (9) tf-idf

```
print(article_words_tfidf, n = 6)
```

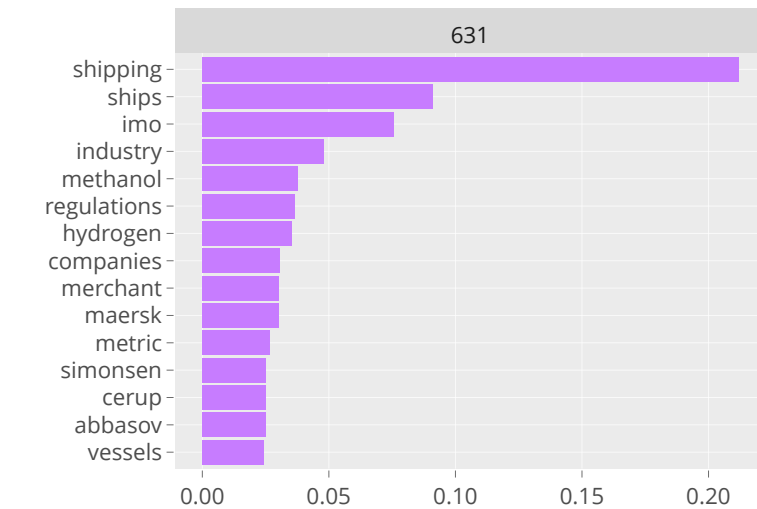
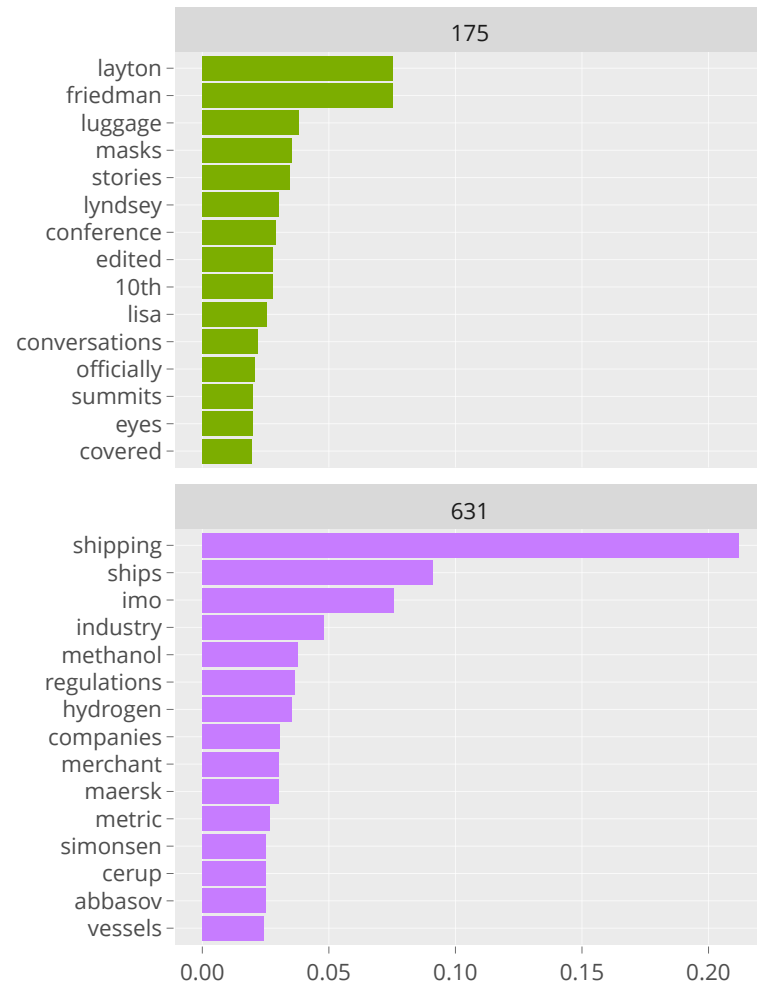
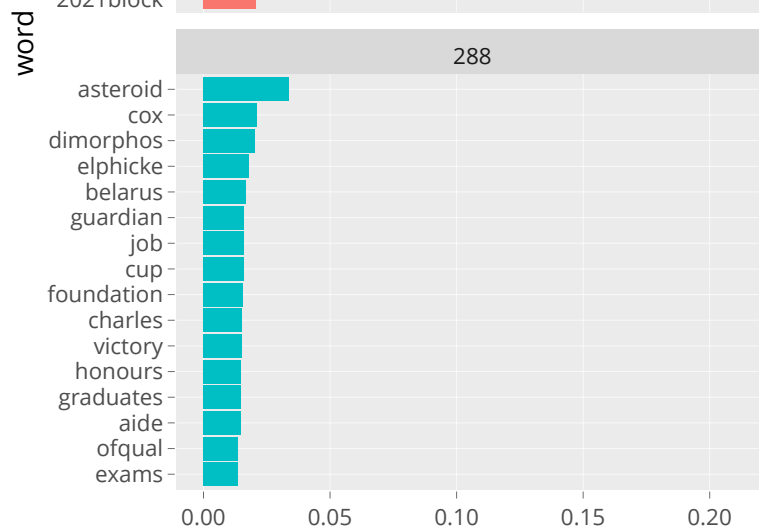
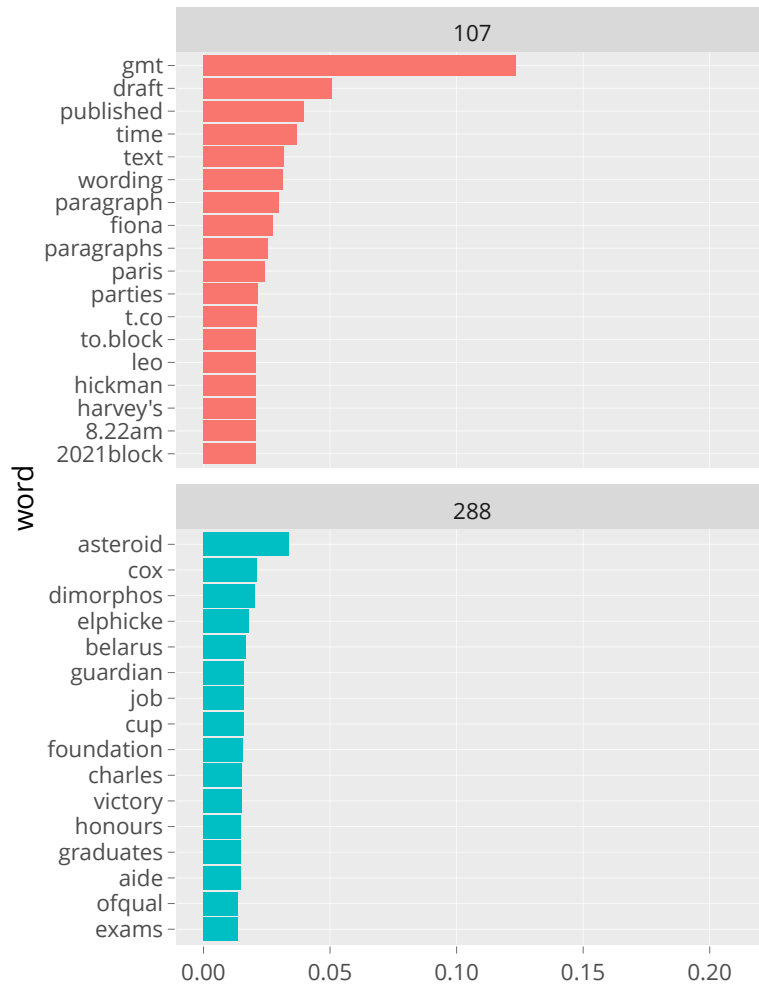
```
## # A tibble: 131,673 × 6
##       id word      n    tf   idf tf_idf
##   <dbl> <chr>  <int> <dbl> <dbl> <dbl>
## 1    659 pwa      84 0.0719  6.54 0.470
## 2    659 editor    70 0.0599  2.93 0.176
## 3    659 bar      66 0.0565  4.14 0.234
## 4    170 water    57 0.0541  2.29 0.124
## 5    124 parties  52 0.0255  2.26 0.0577
## 6    124 paris   50 0.0245  1.56 0.0381
## # ... with 131,667 more rows
```

Example: (10) tf-idf

Let's look at top 15 words by tf-idf in the case of four random articles

```
g <- article_words_tfidf %>%  
  filter(id == 107 | id == 175 | id == 288 | id == 631) %>%  
  group_by(id) %>%  
  top_n(15, tf_idf) %>%  
  ungroup %>%  
  mutate(id = as.factor(id),  
         word = reorder_within(word, tf_idf, id)) %>%  
  ggplot(aes(word, tf_idf, fill = id)) +  
  geom_col() +  
  facet_wrap(~id, nrow = 2, scales = "free_y") +  
  coord_flip() +  
  theme(legend.position = "none") +  
  scale_x_reordered()
```


Example: (11) tf-idf



tf_idf

Example: (12) tf-idf (bigrams)

- In the case of bigrams, to remove stopwords you need first to `separate()`, filter stop words, and then `unite()` bigrams again
- Similarly in the case of n-grams

```
article_words_tfidf <- read_csv("news_articles_example.csv") %>%
  select(id, Hlead) %>%
  unnest_tokens(output = bigram, input = Hlead,
                token = "ngrams", n = 2) %>%
  separate(bigram, c("word_1", "word_2"), sep = " ") %>%
  filter(!word_1 %in% stop_words$word) %>%
  filter(!word_2 %in% stop_words$word) %>%
  unite(bigram, word_1, word_2, sep = " ") %>%
  count(id, bigram, sort = T) %>%
  bind_tf_idf(bigram, id, n)
```

Example: (13) tf-idf (bigrams)

```
print(article_words_tfidf, n = 6)
```

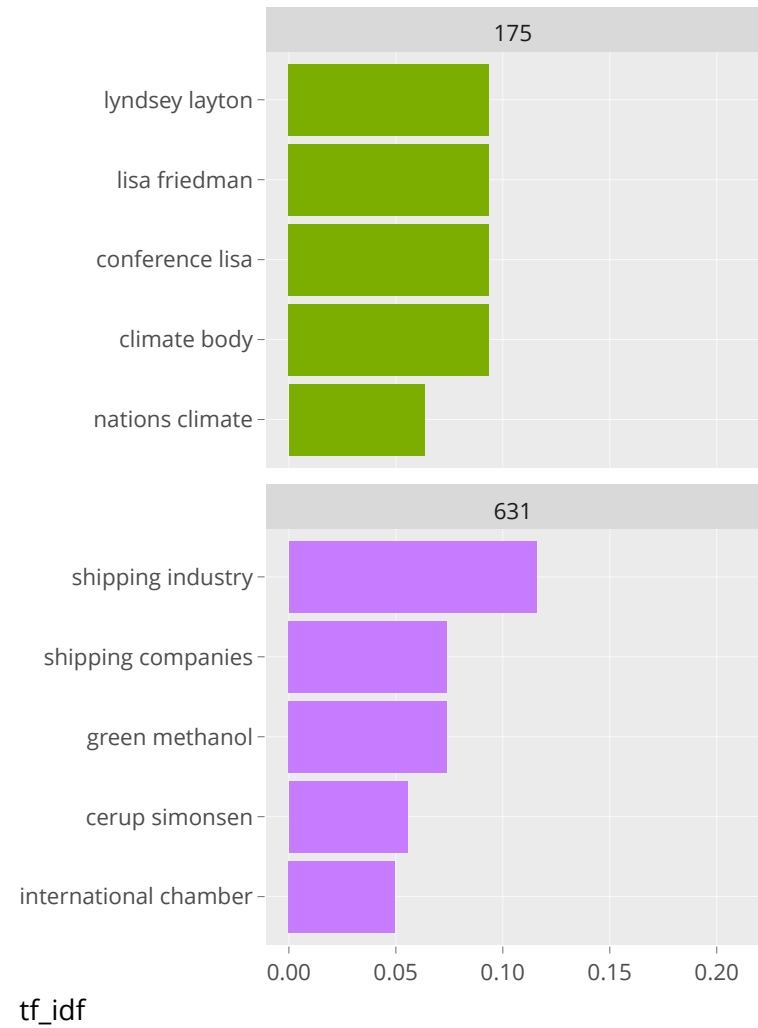
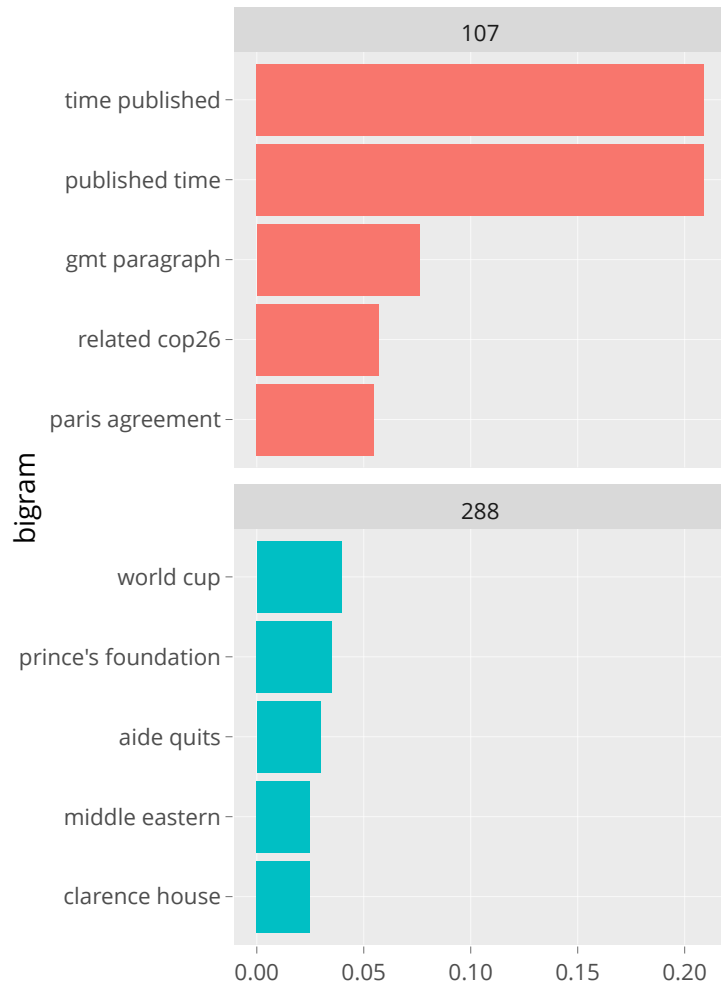
```
## # A tibble: 73,776 × 6
##       id bigram          n      tf   idf tf_idf
##   <dbl> <chr>      <int>  <dbl> <dbl>  <dbl>
## 1   659 editor bar      64 0.0706  6.54 0.461
## 2   124 paris agreement  44 0.0486  1.89 0.0917
## 3   659 pwa editor     34 0.0375  6.54 0.245
## 4   659 data pwa       32 0.0353  6.54 0.231
## 5   659 bar panel      30 0.0331  6.54 0.216
## 6   248 words photos   20 0.0430  6.54 0.281
## # ... with 73,770 more rows
```

Example: (14) tf-idf (bigrams)

Let's look at top 15 bigrams by tf-idf in the case of four random articles

```
g <- article_words_tfidf %>%  
  filter(id == 107 | id == 175 | id == 288 | id == 631) %>%  
  group_by(id) %>%  
  top_n(5, tf_idf) %>%  
  ungroup %>%  
  mutate(id = as.factor(id),  
         bigram = reorder_within(bigram, tf_idf, id)) %>%  
  ggplot(aes(bigram, tf_idf, fill = id)) +  
  geom_col() +  
  facet_wrap(~id, nrow = 2, scales = "free_y") +  
  coord_flip() +  
  theme(legend.position = "none") +  
  scale_x_reordered()
```

Example: (15) tf-idf (bigrams)



Questions

Computer session

Plan

1. Students are grouped in randomly generated groups
2. Groups will select at least one **text corpus** and develop a script in R that undertake a text mining analysis (data are available on Canvas)
3. Groups will upload the R script and the main findings in [Padlet](#) by the end of Week 9 workshop

Groups

Group 1

Adebisi, Jongho, Maria, Keiho

Group 2

Charunan, Poojani, Abdul, Satoshi

Group 3

Oscar, Tsukumo, Jiyoung, Nicholas

Group 4

Alessandro, Shaunna, Jonathan

Anyone missing?

Next time

Next time

- Visualising text data
- Sentiment analysis
- Regular expressions
- Introduction to topic modelling

References

References

- Bone, Frederique, and Daniele Rotolo. 2020. "Text mining historical sources to trace technological change. The case of mass production." Working Paper.
- Ciarli, Tommaso, and Ismael Ràfols. 2019. "The relation between research priorities and societal demands: The case of rice." *Research Policy* 48 (4): 949–67.
<https://doi.org/https://doi.org/10.1016/j.respol.2018.10.027>.
- Feldman, Ronen, and James Sanger. 2006. *The Text Mining Handbook*. Cambridge, United Kingdom: Cambridge University Press. <https://doi.org/10.1017/cbo9780511546914>.
- Kim, Namil, Hyeokseong Lee, Wonjoon Kim, Hyunjong Lee, and Jong Hwan Suh. 2015. "Dynamic patterns of industry convergence: Evidence from a large amount of unstructured data." *Research Policy* 44 (9): 1734–48. <https://doi.org/https://doi.org/10.1016/j.respol.2015.02.001>.
- Kwartler, Ted. 2017. *Text Mining in Practice with R*. Chichester, United Kingdom: John Wiley & Sons Ltd. <https://doi.org/10.1002/9781119282105>.
- Lansdall-Welfare, Thomas, Saatviga Sudhahar, James Thompson, Justin Lewis, FindMyPast Newspaper FindMyPast Newspaper Team, and Nello Cristianini. 2017. "Content analysis of 150 years of British periodicals." *Proceedings of the National Academy of Sciences of the United States of America* 114 (4): E457–65. <https://doi.org/10.1073/pnas.1606380114>.
- Michel, Jean Baptiste, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, Joseph P. Pickett, Dale Hoiberg, et al. 2011. "Quantitative analysis of culture using millions of digitized books." *Science* 331: 176–82. <https://doi.org/10.1126/science.1199644>.
- Silge, Julia, and David Robinson. 2017. *Text mining with R: A tidy approach*. Sebastopol, CA: O'Reilly Media.
- Wickham, Hadley, and Garrett Grolemund. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1st ed. O'Reilly Media, Inc.