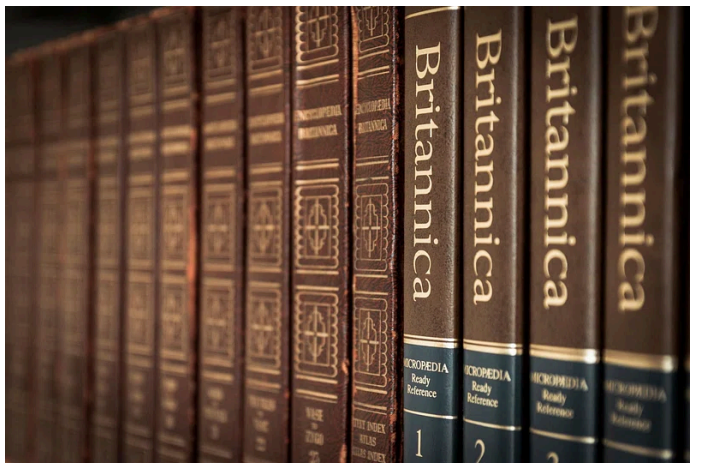
Basics of Automated Text Analysis and Dictionary Approaches

Week 2: Wrangling Text and Classification with Wordlists

prof. dr. Wouter van Atteveldt & dr. Philipp K. Masur

LARGEST ENCYCLOPEDIAS IN THE WORLD

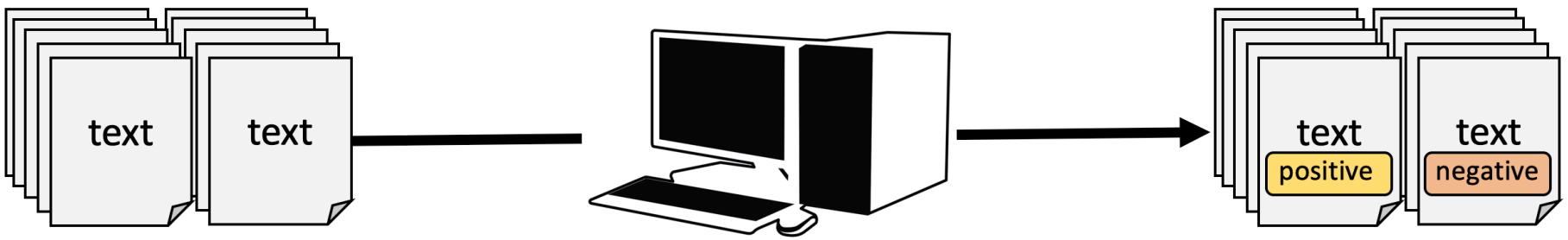
- Encyclopedia Britannica
  - maintained by about 100 full-time editors and more than 4,000 contributors
  - 2010 version of the 15th edition, which spans 32 volumes and 32,640 pages, was the last printed edition
  - Today, 40 million words on half a million topics



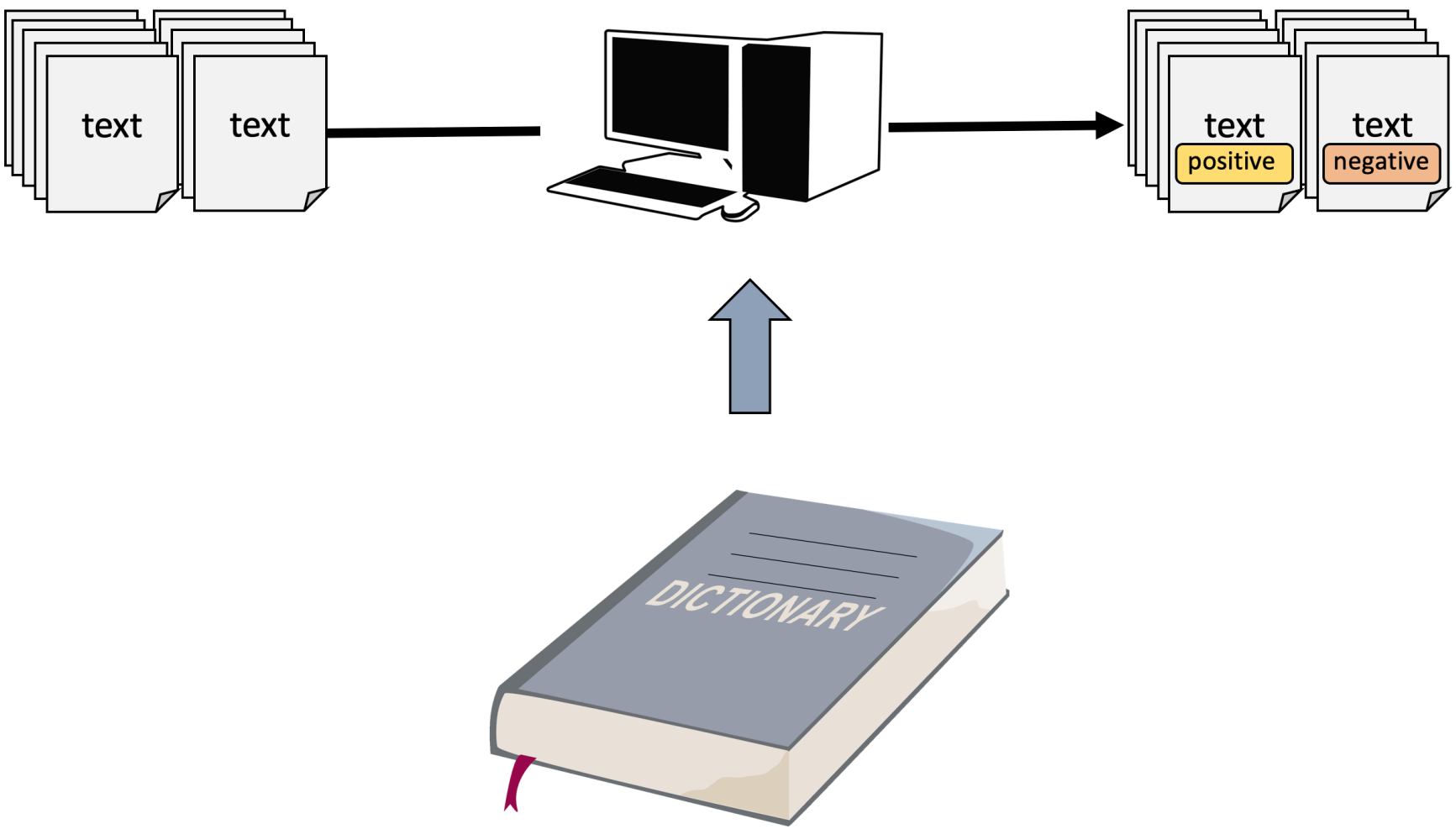
Encyclopedia Britannica

- Wikipedia
  - Biggest encyclopedia worldwide
  - more than 280,000 active editors and 110,461,483 registered users
  - ~61 million articles

AUTOMATED TEXT ANALYSIS…



...WITH DICTIONARIES!



CONTENT OF THIS LECTURE

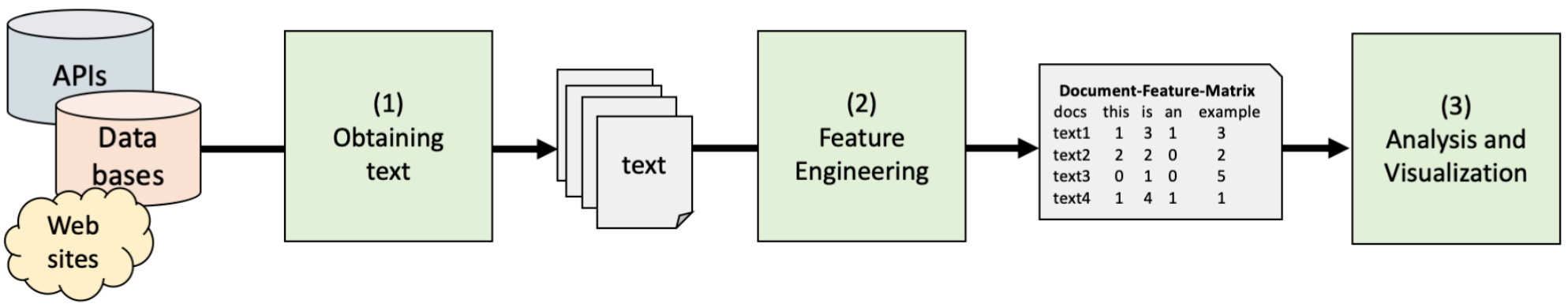
1. Basics of Automated Text Analysis
   - 1.1. Text as Data
   - 1.2. Tokenization
   - 1.3. Ways to Organize Text(s) Numerically
   - 1.4. Text cleaning, stopwords, stemming, lemmatization
   - 1.5. Descriptive Text Analyses
   - 1.6. Keyword-in-Context Searches
2. Dictionary Approaches
   - 3.1. Text Classification Pipeline Using Dictionaries
   - 3.3. Feature Engineering
   - 3.4. Text Classification
   - 3.5. Validation
4. Examples from the literature
   - 4.1. Sourcing Pandemic News (Mellado et al., 2021)
   - 4.2. Political discourses on social media (Heidenreich et al., 2019)
5. Summary and Conclusion

Basics of Text Analysis

From raw text to numerical presentations of texts.

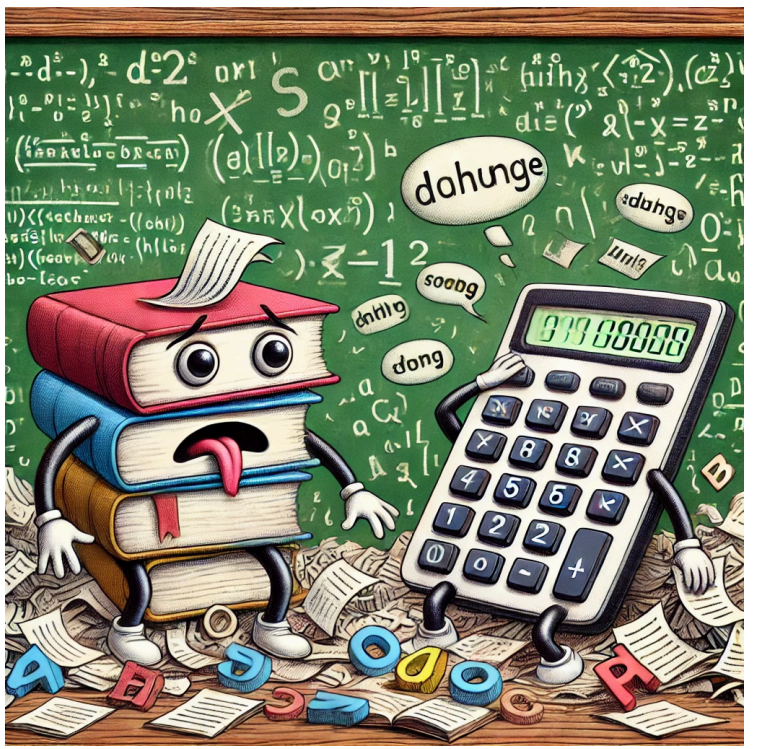
BASIC TEXT ANALYSIS

- Before we start to classify unlabeled text, we often engage in what we call “text wrangling”
- The first goal is find ways to represent texts in numerical frameworks
- Then, we often want to conduct simple descriptive analyses on the texts (e.g., which words are used most often)



A CHALLENGE AT THE BEGINNING

- Texts are strings composed of words, spaces, numbers, and punctation.
- Handling this type of complex, at times messy, data requires some thought and effort.
- Problem: Algorithms process numbers, they do not read text!
- Consider the following string, representing a (very) short text:




```
text <- "To be, or not to be: that is the question."
print(text)
```

```
[1] "To be, or not to be: that is the question."
```

FROM TEXT TO TOKENS

- The elements in this string do not contain any metadata or information to tell the computer which characters are words and which are not (of course it is easy for us to tell, but not for a computer).
- Identifying these kinds of boundaries between words is where the process of **tokenization** comes in.
- For example, we can use the simple function `str_split()` to tell R to split the string into separate strings, whenever there is an "white space", resulting in somewhat meaningful tokens:

```
library(tidyverse)
str_split(text,
          pattern = " ")      # <-- separate words by white space between them
```

```
[[1]]
 [1] "To"        "be,"       "or"        "not"       "to"        "be:"
 [7] "that"      "is"        "the"       "question."
```

FROM TEXT TO TOKENS

- The elements in this string do not contain any metadata or information to tell the computer which characters are words and which are not (of course it is easy for us to tell, but not for a computer).
- Identifying these kinds of boundaries between words is where the process of **tokenization** comes in.
- For example, we can use the simple function `str_split()` to tell R to split the string into separate strings, whenever there is an “white space”, resulting in somewhat meaningful tokens:

```
library(tidyverse)
str_split(text,
          pattern = ":")          # <-- separate words by identifying the colon
```

```
[[1]]
[1] "To be, or not to be"    " that is the question."
```

FROM TEXT TO TOKENS

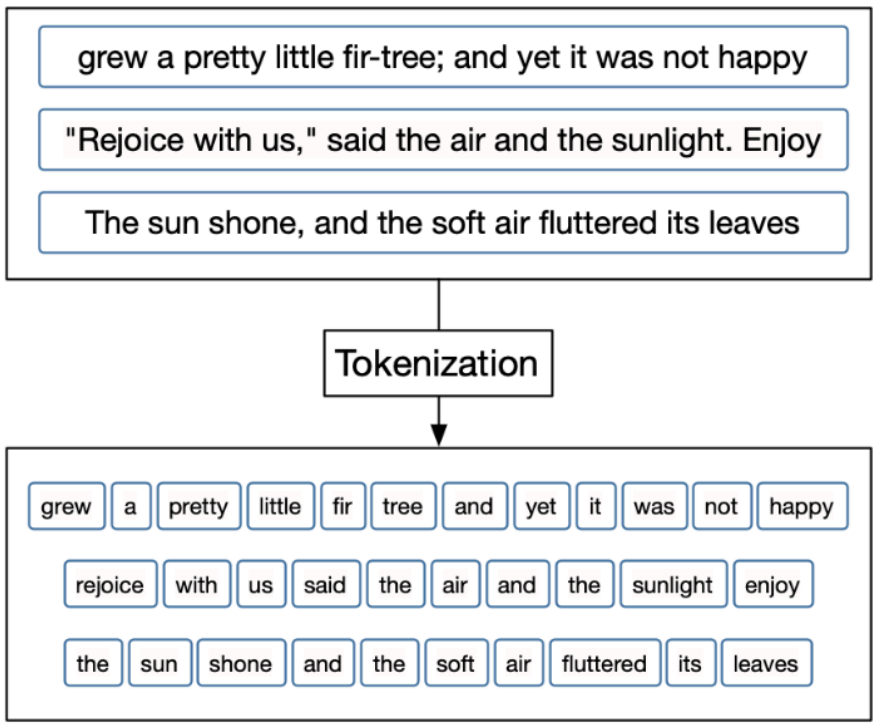
- The elements in this string do not contain any metadata or information to tell the computer which characters are words and which are not (of course it is easy for us to tell, but not for a computer).
- Identifying these kinds of boundaries between words is where the process of **tokenization** comes in.
- For example, we can use the simple function `str_split()` to tell R to split the string into separate strings, whenever there is an “white space”, resulting in somewhat meaningful tokens:

```
library(tidyverse)
str_split(text,
          pattern = "[[:punct:]]")  # <-- separate words by identifying punctuation symbols
```

```
[[1]]
[1] "To be"               " or not to be"        " that is the question"
[4] ""
```

TOKENIZATION

- Separating based on white space or specific “regular expressions” sounds correct intuitively, but is not a very principled approach
- More generally, tokenization means:
  - taking an input (e.g., a string of characters that represent natural language) and
  - a token type (a meaningful unit of text, such as a character, word, or sentence)
  - and splitting the input (string) into pieces (tokens) that correspond to the type (e.g., word)



UNDERSTANDING TOKENIZATION

There are many different R packages for implementing tokenization, e.g., tokenizers, quanteda, tidytext…

```
library(tokenizers)
text_data <- tibble(doc_id = paste0("text", 1:2),
                    author = c("William Shakespeare", "William Wordsworth"),
                    work = c("Hamlet", "The Tables Turned"),
                    text = c("To be, or not to be: that is the question.",
                             "Come forth into the light of things, Let Nature be your teacher."))
text_data
```

```
# A tibble: 2 × 4
  doc_id author             work              text
  <chr>  <chr>              <chr>             <chr>
1 text1  William Shakespeare Hamlet            To be, or not to be: that is the question.
2 text2  William Wordsworth  The Tables Turned Come forth into the light of things, Let Nature be your teacher.
```

```
text_data$text |>
  tokenize_words()   # <-- tokenize into words using the "tokenizers" package
```

```
[[1]]
 [1] "to"       "be"       "or"       "not"      "to"       "be"       "that"     "is"       "the"      "question"

[[2]]
 [1] "come"    "forth"   "into"    "the"     "light"   "of"      "things"  "let"     "nature"  "be"      "your"    "teacher"
```

TYPES OF TOKENS

- Thinking of a token as a **word** is a useful start.
- Tokenizing at the word level is perhaps the most common and widely used tokenization.
- However, we can generalize the idea of a token beyond only a single word to other units of text.

Type of token	Example
characters	“I”, “l”, “o”, “v”, “e”, “y”, “o”, “u”, “v”, “e”, “r” ,“y”,…
subword tokens	“I”, “love”, “d”, “you”, “enorm”, “ous”, “ly”
words	“I”, “love”, “you”, “very”, “much”
sentences	“I love you very much”
lines	“He went to her and said ’I love”, “you very much.’ She responded quickly”
bi-grams	“I love”, “love you”, “you very,”very much”
n-grams	“I love you”, “love you very”, “you very much”

MORE THAN ONE WORD: N-GRAMS

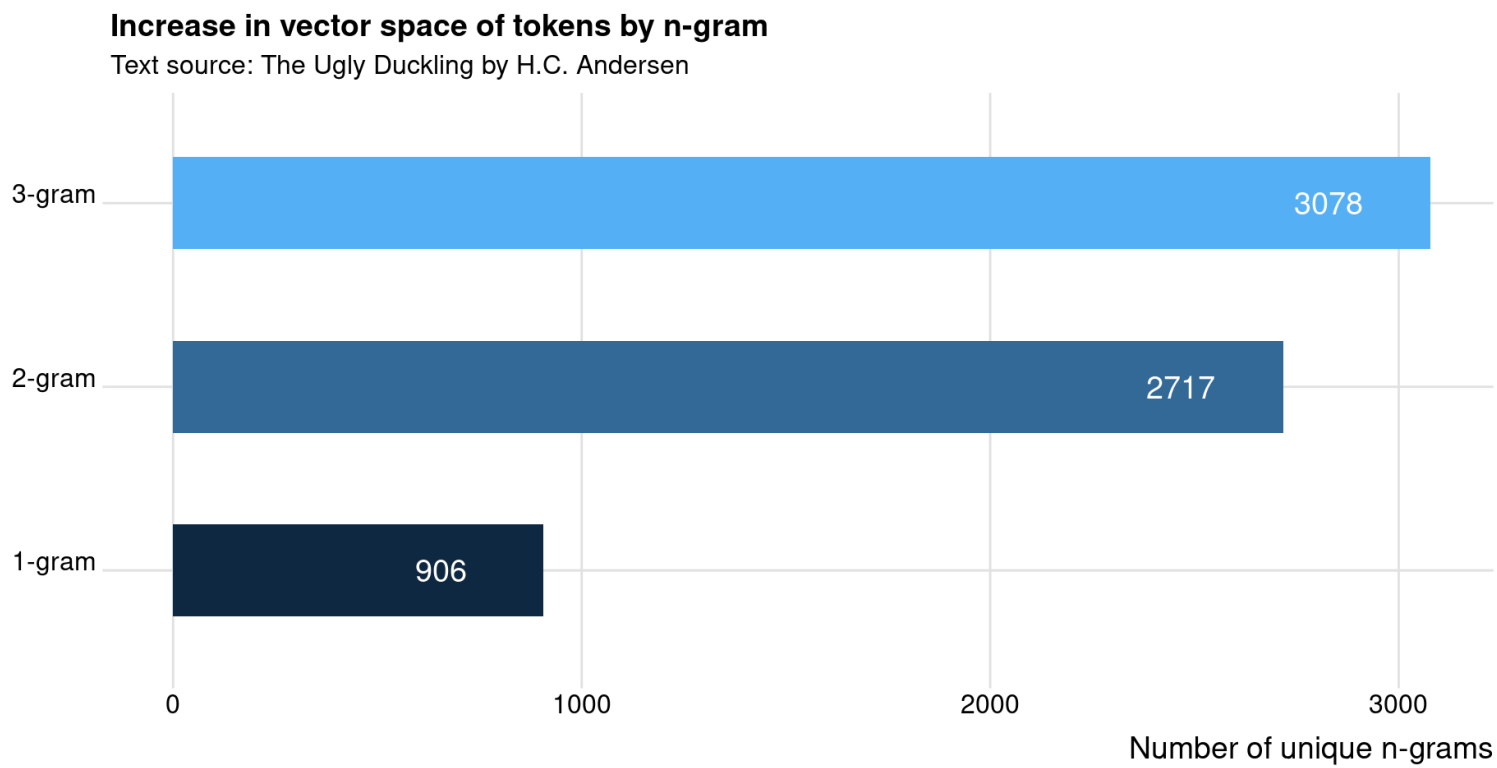
- The simplest case is a bigram (or 2-gram), where each feature is a pair of adjacent words.
- Notice how the words in the bigrams overlap so that the word “be” appears at the end of the first bigram and beginning of the second bigram
- In other words, n-gram tokenization slides along the text to create overlapping sets of tokens

```
text_data$text |>
  tokenize_ngrams(n = 2)
```

```
[[1]]
[1] "to be"        "be or"        "or not"       "not to"       "to be"        "be that"      "that is"
[8] "is the"       "the question"

[[2]]
 [1] "come forth"   "forth into"   "into the"     "the light"    "light of"     "of things"    "things let"
 [8] "let nature"   "nature be"    "be your"      "your teacher"
```

LOGIC OF N-GRAMS

- Using a higher value for n keeps more information, but the vector space of tokens increases dramatically, corresponding to a reduction in token counts.
- The longer the n-gram, the larger the list of unique combinations, which leads to a worse data scarcity problems, so even more attention must be paid to feature selection and/or trimming



A TIDY FORMAT FOR TEXT ANALYSIS

docid	*author*	*work*	*text*
text1	William Shakespeare	Hamlet	To be, or not to be: that is the question.
text2	William Wordsworth	The Tables Turned	Come forth into the light of things, Let Nature be your teacher.

1. Tokenization
2. Wide to long transformation

docid	*author*	*work*	*word*
text1	William Shakespeare	Hamlet	To
text1	William Shakespeare	Hamlet	be
text1	William Shakespeare	Hamlet	or
text1	William Shakespeare	Hamlet	not
text1	William Shakespeare	Hamlet	to
text1	William Shakespeare	Hamlet	be
text1	William Shakespeare	Hamlet	that
text1	William Shakespeare	Hamlet	is
text1	William Shakespeare	Hamlet	the
text1	William Shakespeare	Hamlet	questions
text2	William Wordsworth	The Tables Turned	Come
text2	William Wordsworth	The Tables Turned	into
…	…	…	…

The tidy format

1. Each variable must have its own column
2. Each observation must have its own row
3. Each value must have its own cell

THE "TIDYTEXT" PACKAGE

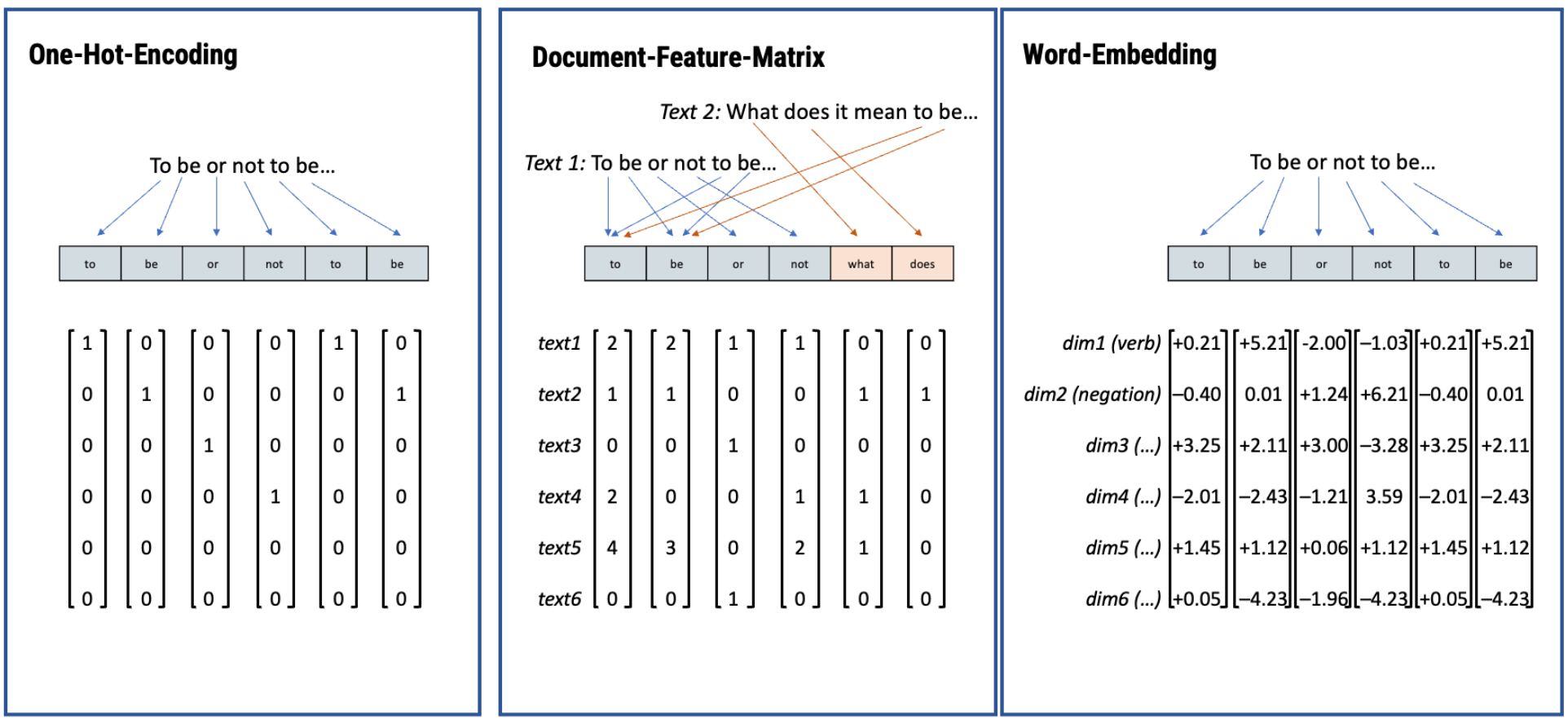
- In this course, we will use the package `tidytext`, whose core data format is a regular data frame where each row is a word
- The function `unnest_tokens()` also tokenizes, but the resulting data structure differs (a tidy data set)
- It thereby determines how we move forward in our analysis

```
library(tidytext)
tokens <- text_data|>
  unnest_tokens(word, text)
print(tokens, n = 20)
```

```
# A tibble: 22 × 4
   doc_id author              work              word
   <chr>  <chr>               <chr>             <chr>
 1 text1  William Shakespeare Hamlet            to
 2 text1  William Shakespeare Hamlet            be
 3 text1  William Shakespeare Hamlet            or
 4 text1  William Shakespeare Hamlet            not
 5 text1  William Shakespeare Hamlet            to
 6 text1  William Shakespeare Hamlet            be
 7 text1  William Shakespeare Hamlet            that
 8 text1  William Shakespeare Hamlet            is
 9 text1  William Shakespeare Hamlet            the
10 text1  William Shakespeare Hamlet            question
11 text2  William Wordsworth  The Tables Turned come
12 text2  William Wordsworth  The Tables Turned forth
13 text2  William Wordsworth  The Tables Turned into
14 text2  William Wordsworth  The Tables Turned the
15 text2  William Wordsworth  The Tables Turned light
16 text2  William Wordsworth  The Tables Turned of
17 text2  William Wordsworth  The Tables Turned things
18 text2  William Wordsworth  The Tables Turned let
19 text2  William Wordsworth  The Tables Turned nature
20 text2  William Wordsworth  The Tables Turned be
# ℹ 2 more rows
```

FROM TOKENS TO NUMBERS

Reminder: In automated text classification, we need to break text into tokens and then represent these tokens as numbers, so that a computer can read them:



HOW TO PRESENT TEXT NUMERICALLY USING “TIDYTEXT”?

- The simplest approach is to count how many times each unique token is included in each text
- By summarizing our tokens across documents and words, we gain a first numerical presentation of our data
- The resulting tidy data frame contains three columns
  - **doc_id:** A unique identifier for each document in the corpus
  - **word:** the list of words included in each document
  - **n:** The number of times the word appears in each document

```
tokens |>
  summarise(n = n()) |>
  arrange(-n)
```

```
# A tibble: 1 × 1
      n
  <int>
1    22
```

THE DOCUMENT-TERM MATRIX (DTM)

- Based on this summarized table, we can quickly create one of the most common text as data representations: the document-term matrix (also called the term-document matrix or document-feature matrix)
- It represents a set of documents as a matrix, where each row represents a document, each column represents a term (word), and the numbers in each cell show how often that word occurs in that document.
- This is very similar to a standard two-dimensional data set (e.g., resulting from a survey) and can thus be treated similarly

```
tokens |>
  group_by(doc_id, word) |>          # <-- group by document id and word
  summarise(n = n()) |>             # <-- summarize number of words per document
  pivot_wider(names_from = word,    # <-- transform from long to wide
              values_from = n) %>%
  replace(is.na(.), 0)              # <-- replace NAs with zero
```

```
# A tibble: 2 × 19
# Groups:   doc_id [2]
  doc_id    be    is   not    or question  that   the    to  come forth  into   let light nature    of teacher things
  <chr>  <int> <int> <int> <int>    <int> <int> <int> <int> <int> <int> <int> <int> <int>  <int> <int>   <int>  <int>
1 text1      2     1     1     1        1     1     1     2     0     0     0     0     0      0     0       0      0
2 text2      1     0     0     0        0     0     1     0     1     1     1     1     1      1     1       1      1
# ℹ 1 more variable: your <int>
```

TIDY FORMAT VS. DTM

Classic Document-Feature Matrix

docid	*be*	*is*	*not*	*or*	*question*	*that*	*to*	...
text1	2	1	1	1	1	1	2	...
text2	1	0	0	0	0	0	0	...

Long to wide transformation

Tidy representation

docid	*word*	*n*
text1	be	2
text1	to	2
text1	is	1
text1	not	1
text1	or	1
text1	question	1
text1	that	1
text1	the	1
text2	be	1
text2	come	1
text2	forth	1
text2	into	1
...	...	...

THE STATE OF THE UNION SPEECHES CORPUS

THE DATA SET

```r
library(sotu)
sotu_text <- bind_cols(sotu_meta, text = sotu_text) |>  # <-- bind texts and meta data
  rename(doc_id = X) |>                                  # <-- rename unique identifier
  as_tibble()                                           # <-- transform to tidy format
print(sotu_text, n = 25)
```

```
# A tibble: 240 × 7
   doc_id president          year years_active party                sotu_type text
    <int> <chr>             <int> <chr>        <chr>                <chr>     <chr>
 1      1 George Washington  1790 1789-1793    Nonpartisan          speech    "Fellow-Citizens of the Senate and House…
 2      2 George Washington  1790 1789-1793    Nonpartisan          speech    "\n\n Fellow-Citizens of the Senate and …
 3      3 George Washington  1791 1789-1793    Nonpartisan          speech    "\n\n Fellow-Citizens of the Senate and …
 4      4 George Washington  1792 1789-1793    Nonpartisan          speech    "Fellow-Citizens of the Senate and House…
 5      5 George Washington  1793 1793-1797    Nonpartisan          speech    "\n\n Fellow-Citizens of the Senate and …
 6      6 George Washington  1794 1793-1797    Nonpartisan          speech    "\n\n Fellow-Citizens of the Senate and …
 7      7 George Washington  1795 1793-1797    Nonpartisan          speech    "\n\nFellow-Citizens of the Senate and H…
 8      8 George Washington  1796 1793-1797    Nonpartisan          speech    "\n\n Fellow-Citizens of the Senate and …
 9      9 John Adams         1797 1797-1801    Federalist           speech    "\n\n Gentlemen of the Senate and Gentle…
10     10 John Adams         1798 1797-1801    Federalist           speech    "\n\n Gentlemen of the Senate and Gentle…
11     11 John Adams         1799 1797-1801    Federalist           speech    "\n\n Gentlemen of the Senate and Gentle…
12     12 John Adams         1800 1797-1801    Federalist           speech    "\n\n Gentlemen of the Senate and Gentle…
13     13 Thomas Jefferson   1801 1801-1805    Democratic-Republican written  "\n\n Fellow Citizens of the Senate and …
14     14 Thomas Jefferson   1802 1801-1805    Democratic-Republican written  "\n\n To the Senate and House of Represe…
15     15 Thomas Jefferson   1803 1801-1805    Democratic-Republican written  "\n\n To The Senate and House of Represe…
16     16 Thomas Jefferson   1804 1801-1805    Democratic-Republican written  "\n\n The Senate and House of Representa…
17     17 Thomas Jefferson   1805 1805-1809    Democratic-Republican written  "\n\n The Senate and House of Representa…
18     18 Thomas Jefferson   1806 1805-1809    Democratic-Republican written  "\n\n The Senate and House of Representa…
19     19 Thomas Jefferson   1807 1805-1809    Democratic-Republican written  "\n\n The Senate and House of Representa…
20     20 Thomas Jefferson   1808 1805-1809    Democratic-Republican written  "\n\n The Senate and House of Representa…
21     21 James Madison      1809 1809-1813    Democratic-Republican written  "\n\n Fellow-Citizens of the Senate and …
22     22 James Madison      1810 1809-1813    Democratic-Republican written  "\n\n Fellow-Citizens of the Senate and …
23     23 James Madison      1811 1809-1813    Democratic-Republican written  "\n\n Fellow-Citizens of the Senate and …
24     24 James Madison      1812 1809-1813    Democratic-Republican written  "\n\n Fellow-Citizens of the Senate and …
25     25 James Madison      1813 1813-1817    Democratic-Republican written  "\n\n Fellow-Citizens of the Senate and …
# ℹ 215 more rows
```

A SPEECH BY BARACK OBAMA IN 2009

```
sotu_text |>
  filter(president == "Barack Obama" & year == "2009") |> # <-- filter based on name and year
  select(text) |>                                         # <-- select only text
  as.character() |>                                       # <-- transform to characters
  writeLines()                                            # <-- print all lines
```

```
Madam Speaker, Mr. Vice President, Members of Congress, the First Lady of the United States--she's arc

I know that for many Americans watching right now, the state of our economy is a concern that rises ab

But while our economy may be weakened and our confidence shaken, though we are living through difficul

The weight of this crisis will not determine the destiny of this Nation. The answers to our problems c

Now, if we're honest with ourselves, we'll admit that for too long, we have not always met these respc

The fact is, our economy did not fall into decline overnight, nor did all of our problems begin when t

In other words, we have lived through an era where too often short-term gains were prized over long-te

Now is the time to act boldly and wisely to not only revive this economy, but to build a new foundatic

As soon as I took office, I asked this Congress to send me a recovery plan by President's Day that wou

Over the next 2 years, this plan will save or create 3.5 million jobs. More than 90 percent of these j

Because of this plan, there are teachers who can now keep their jobs and educate our kids, health care

Now, I know there are some in this Chamber and watching at home who are skeptical of whether this plar

And that's why I've asked Vice President Biden to lead a tough, unprecedented oversight effort; becaus

So the recovery plan we passed is the first step in getting our economy back on track. But it is just

I want to speak plainly and candidly about this issue tonight, because every American should know that

```

THE DTM AS A MATRIX

- We can `pivot` the long "one-term-per-document-per-row" into a wide document-term matrix:

```
tidy_text <- sotu_text |>
  unnest_tokens(word, text) |>                    # <-- tokenize into words and tidy format
  group_by(doc_id, word) |>                        # <-- group by document id and words
  summarise(n = n()) |>                           # <-- summarize number of words per document
  ungroup()                                       # <-- ungroup the data set (to be sure)

dfm <- tidy_text |>
  pivot_wider(names_from=word, values_from=n, values_fill=0)
dfm
```

```
# A tibble: 240 × 30,586
   doc_id     a abroad accession according   add adequate administration admitted adopted advancement affairs afford
    <int> <int>  <int>     <int>     <int> <int>    <int>          <int>    <int>   <int>       <int>   <int>  <int>
 1      1    21      1         1         1     1        1              1        1       1           1       3      2
 2      2    21      1         0         0     2        0              1        0       0           0       1      1
 3      3    42      0         0         0     0        2              1        0       1           1       1      2
 4      4    32      2         0         2     3        1              0        0       0           0       1      1
 5      5    34      0         0         1     0        0              0        0       0           0       1      1
 6      6    48      0         0         1     0        1              0        0       1           0       1      0
 7      7    33      0         0         0     2        2              0        0       0           0       2      0
 8      8    57      0         0         0     1        1              1        0       2           0       0      2
 9      9    21      0         0         0     1        1              0        0       0           0       1      0
10     10    41      0         0         0     1        0              0        0       2           0       0      2
# ℹ 230 more rows
# ℹ 30,573 more variables: affording <int>, against <int>, aggressors <int>, agree <int>, agriculture <int>,
#   aids <int>, all <int>, allowed <int>, already <int>, also <int>, am <int>, among <int>, an <int>, and <int>,
#   answered <int>, any <int>, appointments <int>, arduous <int>, are <int>, armed <int>, arrangements <int>, as <int>,
#   ascertained <int>, at <int>, attended <int>, attention <int>, auspicious <int>, authority <int>, avoiding <int>,
#   basis <int>, be <int>, been <int>, before <int>, best <int>, better <int>, between <int>, blessed <int>,
#   blessings <int>, branch <int>, burthens <int>, but <int>, by <int>, call <int>, can <int>, cares <int>, …
```

```
#dfm_text <- tidy_text |>
#  cast_dfm(document = doc_id, term = word,  value = n)  # <-- create document-term matrix
```

SPARSITY

- DTMs are called **sparse** because they contain a lot of zeros
- More technically, a set of numbers (e.g. vector, matrix, etc.), is considered **sparse** when a high percentage of the values are assigned a constant default value (e.g. zero)
- Representing all the numbers including zeroes is a 'dense' representation, which is very memory inefficient
- In general, sparsity creates some problems:
  - Increase in model complexity (leads to more coefficients/features being included in the model)
  - Risk for overfitting: With too many features, models tend to fit the noise in the data and not generalize well
  - Size of the data set: Leads to computationally expensive procedures

TIDY TEXT FORMAT IS A 'SPARSE FORMAT'

- The tidy text format is considerably a 'sparse representation' as it doesn't include any zeros (there are no rows for words that are not included in the document!)
- Can speed up computations, and allows us to stay within the framework established by the `tidyverse`

```
tidy_text |>
  arrange(-n)
```

```
# A tibble: 431,846 × 3
   doc_id word      n
    <int> <chr> <int>
 1    122 the    2887
 2    124 the    2516
 3    119 the    2432
 4    201 the    2376
 5    123 the    2347
 6    111 the    2338
 7     60 the    2182
 8    110 the    2154
 9    199 the    2143
10    158 the    2142
# ℹ 431,836 more rows
```

```
# Cells in the DTM format
ncol(dfm) * nrow(dfm)
```

```
[1] 7340640
```

```
# Cells in the tidy format
length(tidy_text) * nrow(tidy_text)
```

```
[1] 1295538
```

"BAG-OF-WORDS" APPROACH

- Both representations follow the so-called "bag-of-word" model
- Such a model disregards grammar and word order and only focus on word frequencies

To be, or not to be: that is the question. Come forth into the light of things, Let Nature be your teacher.

docid	*word*	*n*
text1	be	2
text1	to	2
text1	is	1
text1	not	1
text1	or	1
text1	question	1
text1	that	1
text1	the	1
text2	be	1
text2	come	1
text2	forth	1
text2	into	1
…	…	…

DESCRIPTIVE ANALYSES OF THE “BAG-OF-WORD” MODEL

- Since the tidy token list is a regular data frame, we can simply group by word and summarize to get frequency information
- This way, we can compute a variety of descriptive frequencies such as word frequency overall, occurence in number of speeches, occurrence in all speeches, etc.

```
sotu_tokens <- sotu_text |>
  unnest_tokens(word, text)                              # <-- tokenize into words
frequencies <- sotu_tokens |>
  group_by(word) |>                                      # <-- group by word
  summarize(termfreq=n(),                                # <-- computer overall occurrences of a word
            docfreq=length(unique(doc_id)),               # <-- number of speeches in which it occurs
            relfreq=(docfreq/nrow(sotu_text))*100) |>    # <-- relative occurrence in all speeches
  arrange(-termfreq, -docfreq)                           # <-- sort results
head(frequencies, n = 8)
```

```
# A tibble: 8 × 4
  word  termfreq docfreq relfreq
  <chr>    <int>   <int>   <dbl>
1 the     165601     240     100
2 of      106402     240     100
3 and      68063     240     100
4 to       68037     240     100
5 in       43429     240     100
6 a        31342     240     100
7 that     24113     240     100
8 for      21701     240     100
```

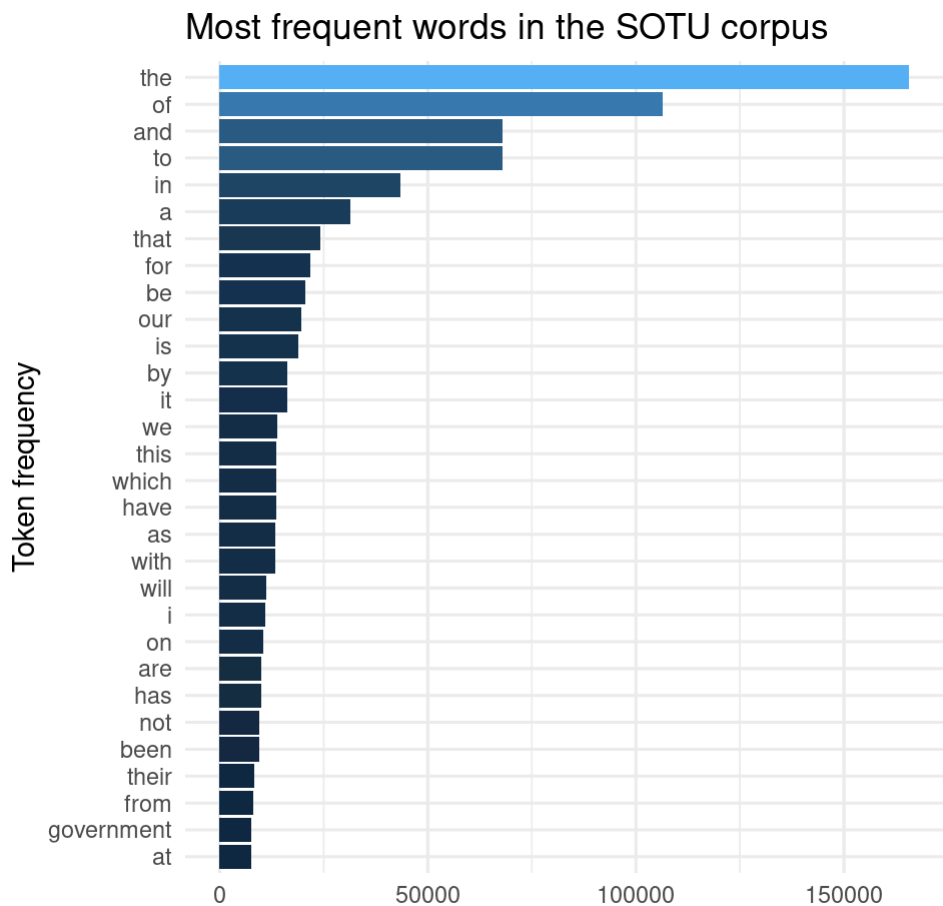
THE (UN-)AVOIDABLE WORDCLOUD

- One of the most famous text visualizations is the word cloud, an image where each word is displayed in a size that is representative of its frequency.
- Word clouds are often criticized since they are (sometimes) pretty but mostly not very informative.
- Only a single aspect of the words is visualized (frequency), which is often not that informative: the most frequent words are generally uninformative “stop words” like “the” and “I”.

```
library(ggwordcloud)
frequencies |>
  slice_max(termfreq, n = 100) |>
  ggplot() +
  geom_text_wordcloud(aes(label = word,
                          size = termfreq,
                          color = termfreq),
                      rm_outside = T) +
  theme_void(base_size = 15)
```

ALTERNATIVE (BETTER?) VISUALIZATION

```r
frequencies |>
  slice_max(termfreq, n = 30) |>
  ggplot(aes(x = fct_reorder(word, termfreq),
             y = termfreq,
             fill = termfreq)) +
  geom_col() +
  coord_flip() +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(x = "Token frequency",
       y = "",
       title = "Most frequent words in the SOTU corpus")
```



WEIGHTING AND SELECTING DOCUMENTS AND TERMS

- So far, the text-as-numbers representations simply show the count of each word in each document.
- Many words, however, are not informative for many questions.
- This was especially apparent if you look at the last word cloud, in which the most frequent words were “the”, “of”, “and”, “to”, etc.
- An unfiltered document-term matrix contains a lot of relevant information: For example, if a president uses the word “terrorism” more often than the word “economy”, that could be an indication of their policy priorities.
- However, there is also a lot of noise crowding out this signal, which is why we engage in a form of weighting and selecting (also known as text preprocessing or cleaning)

TEXT CLEANING, STEMMING, LEMMATIZING

- Text contains a lot of noise
  - Very uncommon words
  - Spelling, scraping mistakes (HTML code, boilerplate, etc)
  - Stop words (e.g., a, the, I, will)
  - Conjugations of the same word (want, wants)
  - Near synonyms (wants, loves)
- Cleaning steps needed to reduce noise:
  - Removing unnecessary symbols (e.g., punctuations, numbers…)
  - Removing stopwords (e.g., 'a', 'the'…)
  - Frequency trimming (removing rare words)
  - Normalization: Stemming (wants -> want) *OR* lemmatizing (ran -> run)
  - Frequency transformation: tf-idf

REMOVING STOPWORDS

- Common words that carry little (or perhaps no) meaningful information are called **stop words** (e.g., 'a', 'the', 'didn't', 'of'…)
- It is common advice and practice to remove stop words for various text analysis tasks, but stop word removal is more nuanced than many resources may lead you to believe

```
sotu_tokens |> pull(word) |> head(n = 8)
```

```
[1] "fellow"   "citizens" "of"       "the"      "senate"   "and"      "house"
[8] "of"
```

```
sotu_tokens |> anti_join(stop_words, by = "word") |> pull(word) |> head(n = 8)
```

```
[1] "fellow"          "citizens"        "senate"          "house"
[5] "representatives" "embrace"         "satisfaction"    "opportunity"
```

TRIMMING OVERALLY INFREQUENT WORDS

- A second useful step can be to trim (or prune) the most infrequent words.
- These words are often a relatively large part of the total vocabulary, but play only a very minor role in most analyses.

```
sotu_tokens |> pull(word) |> unique() |> length()
```

```
[1] 30585
```

```
sotu_tokens |> group_by(word) |> filter(n() >= 100) |> pull(word) |> unique() |> length()
```

```
[1] 2108
```

NORMALIZATION: STEMMING

- What if we aren't interested in the difference between e.g., "trees" and "tree" and we want to treat both together?
- Stemming refers to the process of identifying the base word (or stem) for a data set of words and is thus concerned with the linguistics subfield of morphology (i.e., how words are formed).

```
sotu_tokens |>
  mutate(stem = SnowballC::wordStem(word)) |>
  select(word, stem) |>
  print(n = 12)
```

```
# A tibble: 1,988,203 × 2
   word            stem
   <chr>           <chr>
 1 fellow          fellow
 2 citizens        citizen
 3 of              of
 4 the             the
 5 senate          senat
 6 and             and
 7 house           hous
 8 of              of
 9 representatives repres
10 i               i
11 embrace         embrac
12 with            with
# ℹ 1,988,191 more rows
```

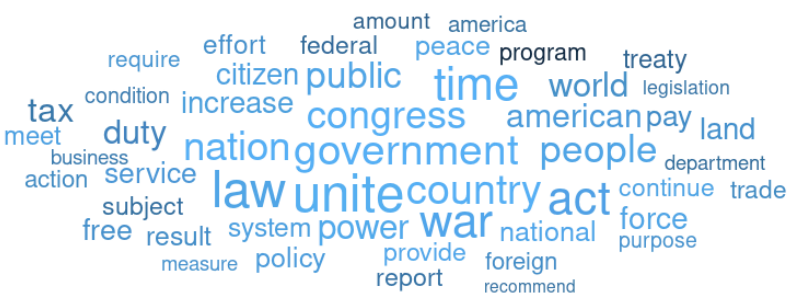
NORMALIZATION: LEMMATIZATION

- Instead of using set rules to cut words down to their stems, lemmatization uses knowledge about a language's structure to reduce words down to their lemmas, the canonical or dictionary forms of words
- Lemmatizers use a rich lexical database like 'WordNet' to look up word meanings for a given part-of-speech use (Miller 1995)
- We can use the package `textstem` to lemmatize our words

```
sotu_tokens |>
  mutate(lemmata = textstem::lemmatize_words(word)) |>
  select(word, lemmata) |>
  print(n = 12)
```

```
# A tibble: 1,988,203 × 2
   word            lemmata
   <chr>           <chr>
 1 fellow          fellow
 2 citizens        citizen
 3 of              of
 4 the             the
 5 senate          senate
 6 and             and
 7 house           house
 8 of              of
 9 representatives representative
10 i               i
11 embrace         embrace
12 with            with
# ℹ 1,988,191 more rows
```

THE EFFECT OF SUCH TEXT CLEANING



TF-IDF TRANSFORMATION (WEIGHTING)

- So far, we looked at how frequent a word occurs in a document (speech).
- Another approach is to look at a term's *inverse document frequency* (idf), which decreases the weight for commonly used words and increases the weight for words that are not used very much in a collection of documents.
- This can be combined with term frequency to calculate a term's tf-idf (the two quantities multiplied together), the frequency of a term adjusted for how rarely it is used.

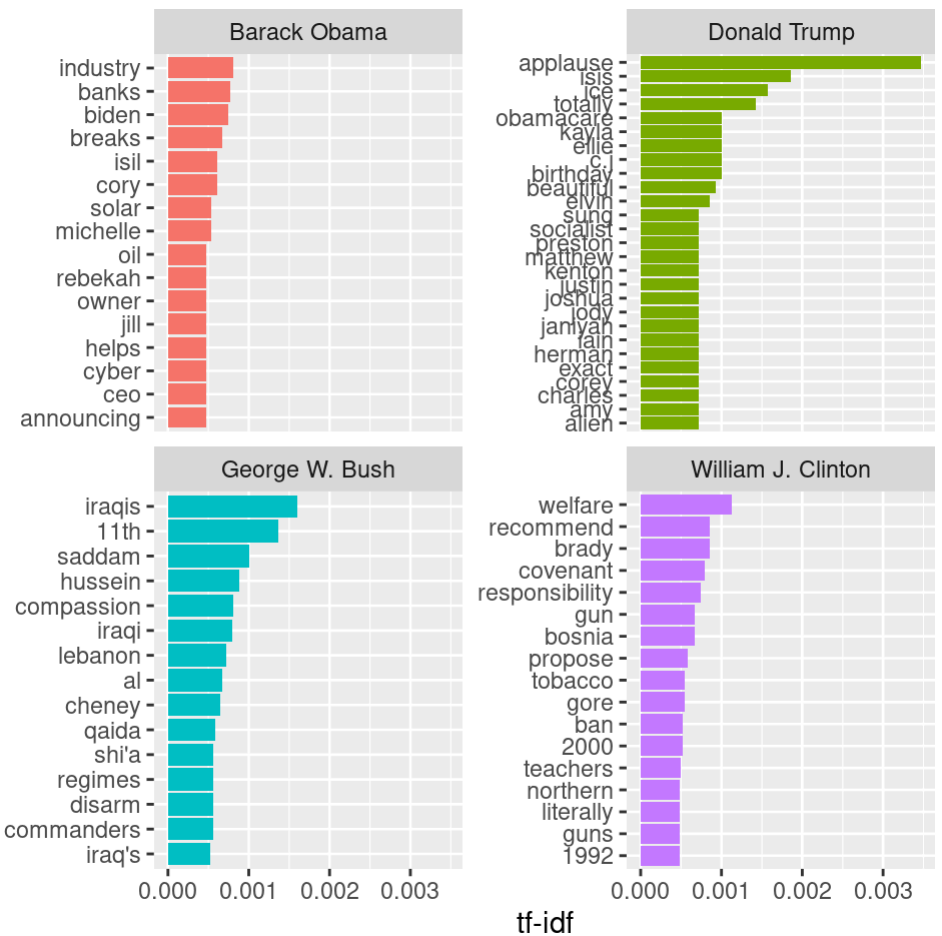
```
sotu_if_idf <- sotu_tokens |>
  filter(president %in% c("Barack Obama",
                          "Donald Trump",
                          "George W. Bush",
                          "William J. Clinton")) |>
  anti_join(stop_words, by = "word") |>
  group_by(president, word) |>
  summarize(n = n()) |>
  bind_tf_idf(word, president, n) |>
  group_by(president) |>
  arrange(-tf_idf)
sotu_if_idf
```

```
# A tibble: 16,700 × 6
# Groups:   president [4]
   president          word         n       tf   idf  tf_idf
   <chr>              <chr>    <int>    <dbl> <dbl>   <dbl>
 1 Donald Trump       applause   117 0.0121   0.288 0.00347
 2 Donald Trump       isis        13 0.00134  1.39  0.00186
 3 George W. Bush     iraqis      20 0.00116  1.39  0.00161
 4 Donald Trump       ice         11 0.00114  1.39  0.00157
 5 Donald Trump       totally     10 0.00103  1.39  0.00143
 6 George W. Bush     11th        17 0.000985 1.39  0.00137
 7 William J. Clinton welfare     89 0.00392  0.288 0.00113
 8 George W. Bush     saddam      25 0.00145  0.693 0.00100
 9 Donald Trump       birthday     7 0.000722 1.39  0.00100
10 Donald Trump       c.j          7 0.000722 1.39  0.00100
# ℹ 16,690 more rows
```

MOST “IMPORTANT” WORDS IN PRESIDENTS’ SPEECHES

The statistic **tf-idf** is intended to measure how important a word is to a document in a collection of documents, for example, to one president in a collection of speeches:

```
sotu_if_idf |>
  slice_max(tf_idf, n = 15) %>%
  ungroup() %>%
  ggplot(aes(x = tf_idf,
             y = fct_reorder(word, tf_idf),
             fill = president)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~president, ncol = 2,
             scales = "free_y") +
  labs(x = "tf-idf", y = NULL)
```



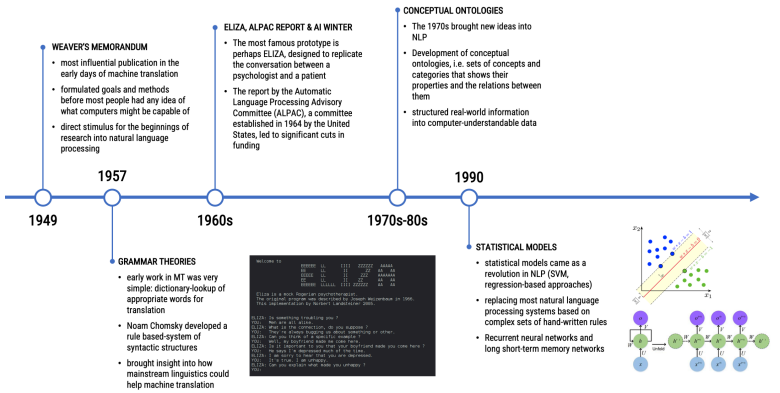
Break (5 Minutes)

Deductive Approaches: Dictionary Analysis

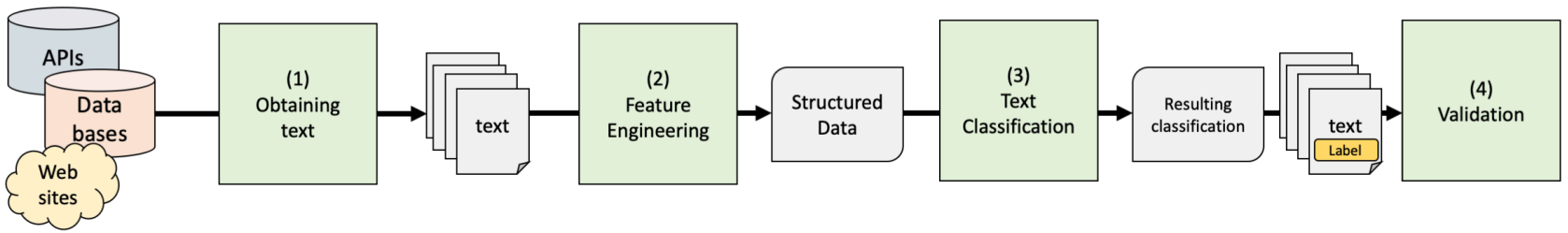
Using fixed set of terms per concept to automatically code texts.

WHAT ARE DEDUCTIVE APPROACHES?

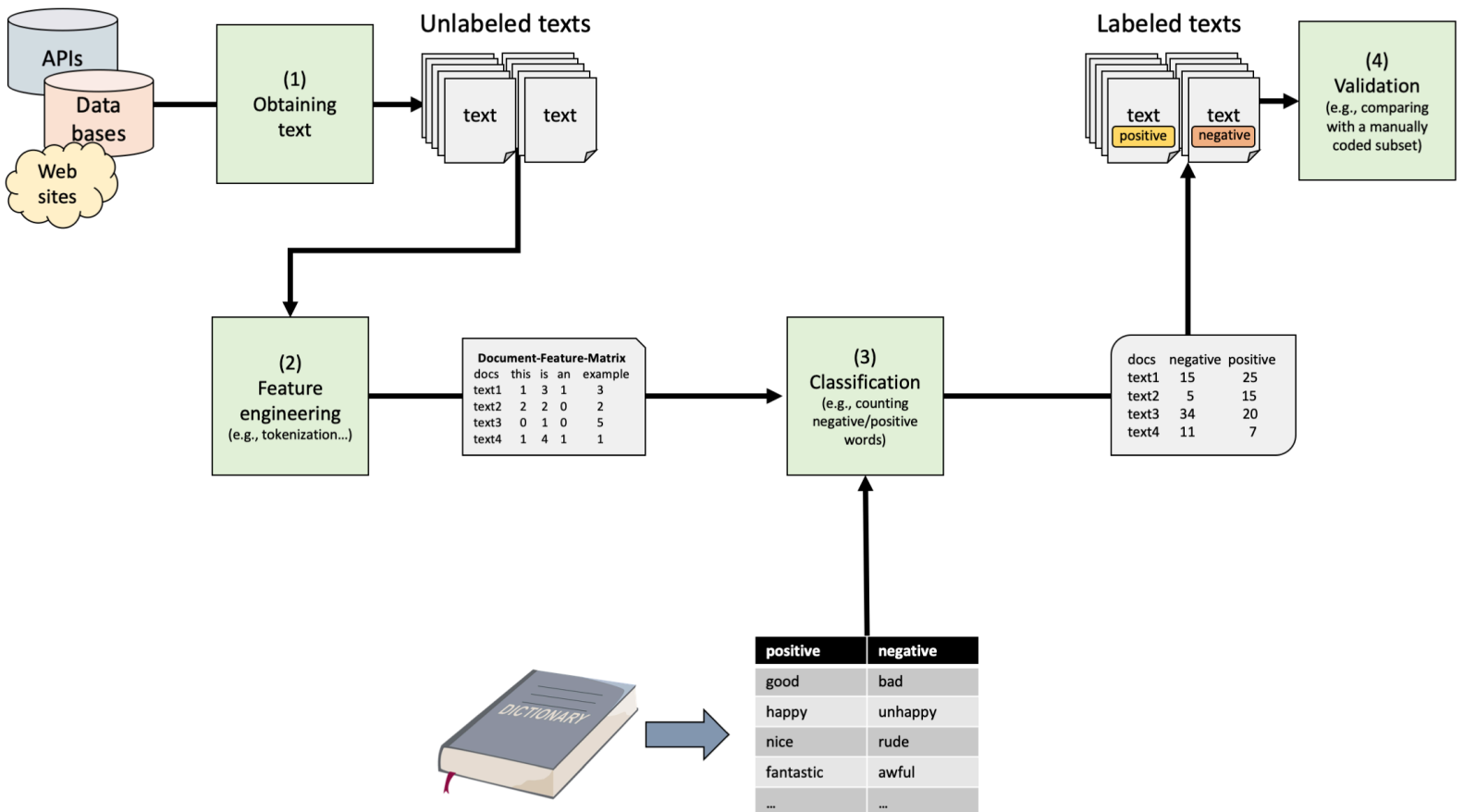
- Quite old technique of content analysis (since 60s)
- Coding rules are set *a priori* by researcher based on a predefined “text theory”
- Computer then uses these rules to decode text in a deterministic way
- Rules can differ substantially:
  - based on individual words or group of words (e.g., articles that contain “government” are coded as “politics”)
  - based on patterns (e.g., the sender of a mail can be identified by looking for “FROM:”)
  - combinations of both



TEXT CLASSIFICATION PIPELINE USING DICTIONARIES



TEXT CLASSIFICATION PIPELINE USING DICTIONARIES



KEYWORD-IN-CONTEXT SEARCHES

- The basic idea of a dictionary is to look for one or several keywords in the text
- So-called “keyword-in-context” searches can be very helpful to understand how these words have been used in different documents (here: speeches)

```
options(width = 250)
library(ccsamsterdamR)
tidy_kwic(sotu_text, "terror", window = 5) |>
  select(pre:doc_id, president, year) |>
  head(n = 18)
```

```
                       pre s1          target s2                              post doc_id             president year
1                lose their  |      terror       | fortifications in those              29          James Monroe 1817
2                    of its  |     terrors       | not withstanding the                 42         Andrew Jackson 1830
3                 spread its |     terrors       | not with standing                    44         Andrew Jackson 1832
4               of constant  |      terror       | and annoyance to                     62       Millard Fillmore 1850
5                    by the  |      terror       | of confiscation and                  79         Andrew Johnson 1867
6                  to spread |      terror       | among those whose                    86       Ulysses S. Grant 1874
7             amphitrite and |      terror       | have been launched                   95      Chester A. Arthur 1883
8          monitors puritan  |      terror       | and amphitrite contracted            96      Chester A. Arthur 1884
9               of constant  |      terror       | to the settlers                      98       Grover Cleveland 1886
10         defense monitors  |      terror       | puritan amphitrite and              105       Grover Cleveland 1893
11             of tyrannous  |      terror       | the peace of                        116     Theodore Roosevelt 1904
12               the german  |      terror       | and whom we                         130         Woodrow Wilson 1918
13                blood and  |      terror       | is a painful                        131         Woodrow Wilson 1919
14             countries by  |      terror       | and force and                       136        Calvin Coolidge 1924
15                 with the  |      terror       | and despair of                      149 Franklin D. Roosevelt 1938
16                  hope of  |   terrorizing     | our people and                      153 Franklin D. Roosevelt 1942
17                  by the   |      terror       | of nazi domination                  156 Franklin D. Roosevelt 1945
18                reign of   |      terror       | in europe nineteen                  156 Franklin D. Roosevelt 1945
```

GETTING A DICTIONARY FOR EMOTIONS

- For this example, we are going to use the NRC word-emotion association lexicon (Mohammad & Turney, 2013)

```
library(textdata)
nrc <- lexicon_nrc()
head(nrc)
```

```
# A tibble: 6 × 2
  word      sentiment
  <chr>     <chr>
1 abacus    trust
2 abandon   fear
3 abandon   negative
4 abandon   sadness
5 abandoned anger
6 abandoned fear
```

```
nrc %>%
  group_by(sentiment) |> # <-- group by sentiment
  tally()                # <-- count words
```

```
# A tibble: 10 × 2
   sentiment        n
   <chr>        <int>
 1 anger         1245
 2 anticipation   837
 3 disgust       1056
 4 fear          1474
 5 joy            687
 6 negative      3316
 7 positive      2308
 8 sadness       1187
 9 surprise       532
10 trust         1230
```

FEATURE ENGINEERING FOR DICTIONARY APPROACHES

- For dictionary analyses, we need to extract words as dictionaries contain word lists the reflect a particular concept
- But we also want to make sure that the words from the raw texts are transformed in a way that the reflect their true form (i.e., how it is likely to be included in the dictionary)
- Stemming wouldn't work because most dictionaries do not include word stems, yet lemmatization is meaningful as it reduces e.g., plural to singular, etc.

```
# Without lemmatization
sotu_dict <- sotu_text |>
  unnest_tokens(word, text)

# With lemmatization
sotu_dict2 <- sotu_text |>
  unnest_tokens(word, text) |>
  mutate(word = textstem::lemmatize_words(word))
```

THE ACTUAL DICTIONARY ANALYSIS

- We use the text corpus containing all speeches
- We tokenize the speeches (separating them into words)
- We can explore lemmatization as meaningful feature engineering
- We join the dictionary and the tokenized data set (note how `left_join()` preserves unmatched tokens)

```
# Combining raw text tokens and emotion dictionary
sotu_emotions <- left_join(sotu_dict, nrc,
                           relationship = "many-to-many")

# Combining lemmatized text tokens and emotion dictionary
sotu_emotions2 <- left_join(sotu_dict2, nrc,
                            relationship = "many-to-many")

# Comparison
sotu_emotions  |> filter(!is.na(sentiment)) |> nrow()
```

```
[1] 552320
```

```
sotu_emotions2 |> filter(!is.na(sentiment)) |> nrow()
```

```
[1] 698513
```

DESCRIPTIVE ANALYSES WITH THE NEW CREATED DATASET

- The result of our dictionary analysis is a new column called “sentiment”, which contains the codes for particular words in the speeches
- This new data set contains all information needed to do a more substantial analysis

```
head(sotu_emotions2, n = 10)
```

```
# A tibble: 10 × 8
   doc_id president          year years_active party       sotu_type word           sentiment
    <int> <chr>             <int> <chr>        <chr>       <chr>     <chr>          <chr>
 1      1 George Washington  1790 1789-1793    Nonpartisan speech    fellow         positive
 2      1 George Washington  1790 1789-1793    Nonpartisan speech    fellow         trust
 3      1 George Washington  1790 1789-1793    Nonpartisan speech    citizen        positive
 4      1 George Washington  1790 1789-1793    Nonpartisan speech    of             <NA>
 5      1 George Washington  1790 1789-1793    Nonpartisan speech    the            <NA>
 6      1 George Washington  1790 1789-1793    Nonpartisan speech    senate         trust
 7      1 George Washington  1790 1789-1793    Nonpartisan speech    and            <NA>
 8      1 George Washington  1790 1789-1793    Nonpartisan speech    house          <NA>
 9      1 George Washington  1790 1789-1793    Nonpartisan speech    of             <NA>
10      1 George Washington  1790 1789-1793    Nonpartisan speech    representative <NA>
```

EMOTIONALITY PER SPEECH

- For example, we can compute the amount of words coded with a particular emotions per speech

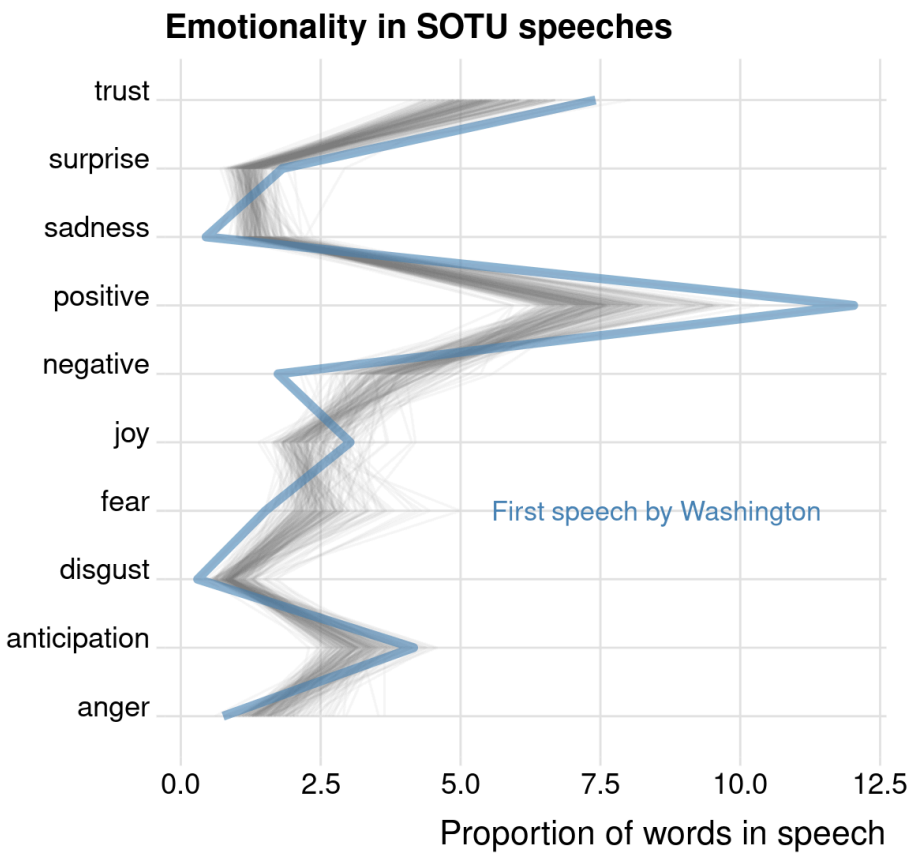
```r
library(ggridges)

# Summarize data
sotu_plot_data <- sotu_emotions2 |>
  group_by(doc_id, sentiment) |>
  summarize(n = n()) |>
  mutate(prop = n/sum(n))
sotu_plot_data
```

```
# A tibble: 2,640 × 4
# Groups:   doc_id [240]
   doc_id sentiment        n    prop
    <int> <chr>        <int>   <dbl>
 1      1 anger           10 0.00756
 2      1 anticipation    55 0.0416
 3      1 disgust          4 0.00303
 4      1 fear            20 0.0151
 5      1 joy             40 0.0303
 6      1 negative        23 0.0174
 7      1 positive       159 0.120
 8      1 sadness          6 0.00454
 9      1 surprise        24 0.0182
10      1 trust           98 0.0741
# ℹ 2,630 more rows
```

EMOTIONALITY PER SPEECH

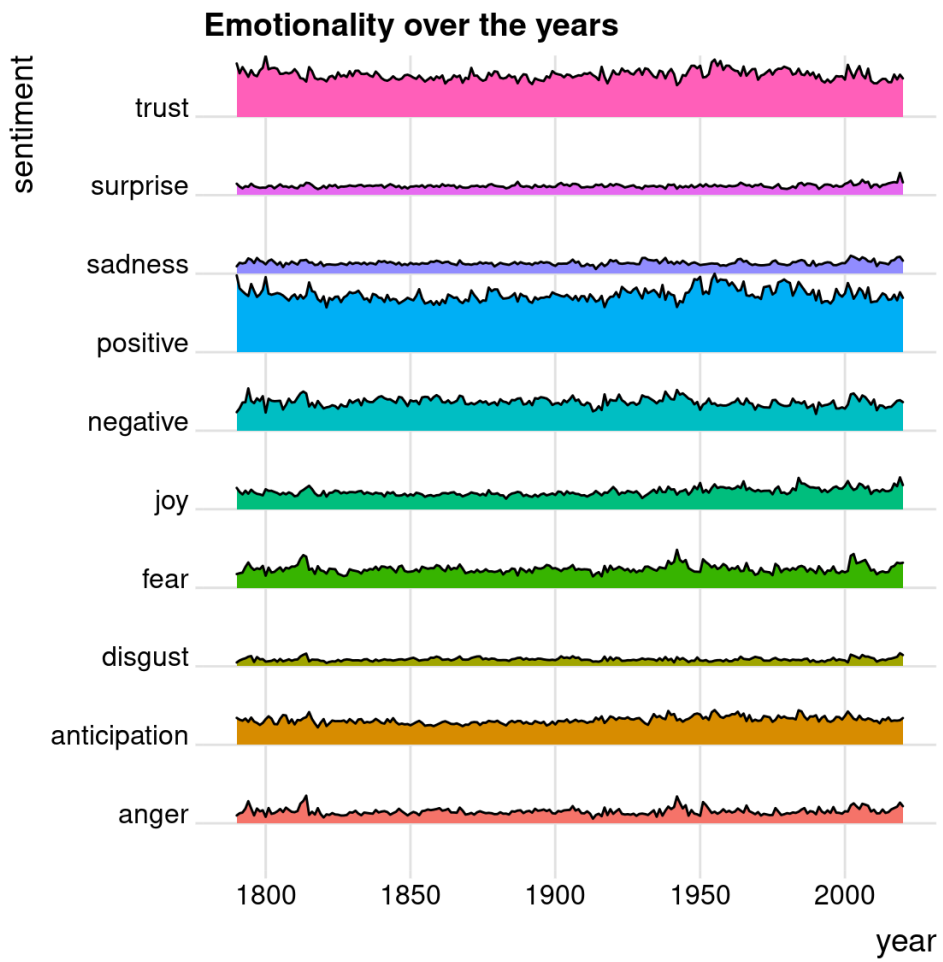
- For example, we can compute the amount of words coded with a particular emotions per speech

```r
library(ggridges)

# Summarize data
sotu_plot_data <- sotu_emotions2 |>
  group_by(doc_id, sentiment) |>
  summarize(n = n()) |>
  mutate(prop = n/sum(n)) |>
  filter(!is.na(sentiment))

# Plotting the data
ggplot() +
 geom_line(data = sotu_plot_data,
           aes(x = sentiment, y = prop*100,
               group = doc_id),
           linewidth = .5,
           alpha = .05, color = "grey50")+
 geom_line(data = sotu_plot_data |> filter(doc_id == 1),
           aes(x = sentiment, y = prop*100, group = 1),
           linewidth = 2,
           color = "steelblue",
           alpha = .6) +
 annotate("text", x = "fear", y = 8.5, size = 4.5,
          label = "First speech by Washington",
          color = "steelblue", ) +
 theme_ridges(font_size = 16) +
 coord_flip() +
 labs(x = "", y = "Proportion of words in speech",
      title = "Emotionality in SOTU speeches")
```



EMOTIONS OVER THE YEARS

```r
# Summarize data
sotu_emo_time <- sotu_emotions2 |>
  group_by(year, sentiment) |>
  summarize(n = n())  |>
  mutate(prop = n / sum(n)) |>
  ungroup() |>
  filter(!is.na(sentiment))

# Plot the summarized data
ggplot(sotu_emo_time) +
  geom_ridgeline(aes(x=year, y=sentiment,
                     height=prop/max(prop),
                     fill=sentiment)) +
  theme_ridges() +
  guides(fill="none") +
  labs(title = "Emotionality over the years")
```



VALIDATION

- To estimate the validity of our dictionary analysis, we manually code a random sample of the documents and compare the coding with the dictionary results
- General procedure
  - Draw a random subsample of the documents
  - Code actual documents with human coders (the ‘gold standard’)
  - Combine the manual coding results with the results from the automated sentiment analysis
  - Produce reliability and validation scores, e.g., correlation between manual and automated coding, precision, recall…
- Agreement with manual coding, which is often regarded as the ‘gold-standard’, is often seen as validity (but this holds only if human beings are inerrant!)

VALIDATION PROCEDURE

Automated classification

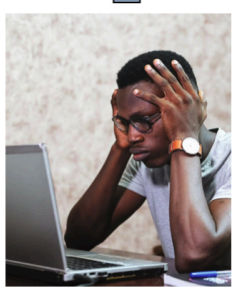
docid	*positive*	*negative*	*classification*
text1	20	2	positive
text2	10	11	negative
text3	32	20	positive
text4	25	4	positive
text5	4	15	negative
text6	12	17	negative
text7	3	6	negative
text8	19	28	negative
text9	45	20	positive
text10	2	1	positive
text11	33	29	positive
text12	7	15	negative
…	…	…	

Gold Standard

docid	*classification*
text1	positive
text2	positive
text3	positive
text4	positive
text5	negative
text6	negative
text7	negative
text8	negative
text9	positive
text10	negative
text11	negative
text12	negative
…	

	Actually Positive	Actually negative
Predicted Positive	4	2
Predicted negative	1	5

positive	negative
good	bad
happy	unhappy
nice	rude
fantastic	awful
…	…



CONFUSION MATRIX

	Actual Positive	Actual Negative
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

- Basis for all validity/performance scores is always the confusion matrix between algorithm and manual coder that assessed the same content
- Disadvantages
  - No correction of random chance agreements
  - only nominal scales
  - no accounting for missings
- Alternative: Cronbach's Alpha (but often seen as too strict)

TYPICAL VALIDITY/PERFORMANCE SCORES

- Whereas we talk about “reliability and validity” in the context of statistical measurement theory, we usually talk about “performance” in the context of text classification
- From the confusion matrix, we can compute all relevant performance scores:
  - **Accuracy:** Share of correct classifications overall (sum of the diagonale / sum of the entire matrix)
  - **Precision:** Probability of a positively coded document is relevant
  - **Recall:** Probability that a relevant document is coded positively
  - **F1-Score:** Mean between precision and recall

PERFORMANCE SCORES

	Actual Positive	Actual Negative
Predicted Positive	300	14
Predicted Negative	25	400

- No clear thresholds: need to be assessed in the research context, but values closer to 1 are desirable
- Can be used to compare the performance of different approaches!

Performance Score	*Formula*	*Example*
Accuracy	(TP + TN)/(TP + TN + FP + FN)	(300 + 400) / (300 + 400 + 25 + 14) = **.947**
Precision	TP / (TP + FP)	300 / (300 + 14) = **.955**
Recall	TP / (TP + FN)	300 / (300 + 25) = **.923**
F-Score	(2 x TP) / (2 x TP + FP + FN)	(2 x 300) / (2 x 300 + 14 + 25) = **.934**

ADVANTAGES AND DISADVANTAGES

- Advantages:
  - Technically easy, many dictionaries exists
  - Transparent and replicable, if dictionary is shared
  - Few resources needed, efficient
- Disadvantages:
  - Low validity for non-trivial concepts (sentiment, frames, specific topics)
    - categories needs be identifiable by simple word lists
    - needs to be tested sufficiently (Validation!!!)
  - May require considerable preprocessing to reduce ambiguity
  - Difficult to create/maintain large dictionaries
  - Can encode biases

HOW CAN WE DEVELOP A DICTIONARY?

- Defining categories/labels
- Inspecting keywords-in-context lists
- Manually coding and comparing word frequencies per category
- Inclusion of different spellings
- Deletion of words the lead to false-positives
- Testing, testing, testing…

Examples in the literature

Studies that used the dictionary approach

EXAMPLE 1: SOURCING PANDEMIC NEWS (MELLADO ET AL)

- The study analyzes the sources and actors present in more than 940,000 posts on COVID-19 published in the 227 Facebook, Instagram, and Twitter accounts of 78 sampled news outlets between January 1 and December 31 of 2020
- The analysis shows the dominance of political sources across countries and platforms, particularly in Latin America
- It demonstrates the strong role of the state in constructing pandemic news and suggesting that mainstream news organizations' social media posts maintain a strong elite orientation
- Health sources were also prominent, while significant diversity of sources, including citizen sources, emerged as the pandemic went on

Mellado et al., 2021

EXAMPLE 1: METHODS

- The authors identified eleven categories (political, business, health, scientific and academic sources, police/security, legal, civil society, citizen, media, sports, and celebrity sources)
- The broke these down into sub-categories that represent formal positions, names of individuals, institutions, organizations and groups, as well as each of their nicknames and acronyms (if any).
- Each national team was responsible for translating the sub-categories into their own language
- Based on this, a manual **dictionary** was created for the seven countries included in the study, which contains over (10,102) entities that belonged to each sub-category at the time of data collection
- This dictionary was used to categorize all posts automatically

EXAMPLE 1: RESULTS - SOURCE DISTRIBUTION PER COUNTRY

Table 2. Sources in social media COVID-19 pandemic news coverage by country.

Sources	Global	Brazil	Chile	Germany	Mexico	Spain	UK	U.S.
Celebrity	2.86%	4.36%	2.91%	3.48%	2.54%	2.89%	2.58%	2.58%
Sport	1.26%	0.81%	1.76%	0.42%	0.65%	1.39%	0.81%	1.47%
Media	3.33%	4.36%	1.86%	6.36%	2.65%	3.63%	4.03%	4.19%
Political	51.17%	51.62%	57.98%	46.93%	50.61%	50.52%	48.40%	45.54%
Police/Security	1.94%	0.39%	2.08%	2.15%	1.09%	2.18%	2.26%	2.40%
Scientific/Educational	6.81%	7.14%	5.61%	8.65%	6.54%	5.55%	10.75%	6.90%
Health	17.53%	18.19%	12.79%	10.91%	21.35%	20.13%	20.30%	17.45%
Business	4.42%	3.42%	5.20%	8.10%	5.50%	3.33%	3.41%	4.23%
Legal	1.01%	2.15%	1.25%	0.86%	0.82%	0.90%	0.14%	1.21%
Citizen	8.05%	5.69%	5.97%	10.64%	6.98%	8.04%	5.82%	13.36%
Civil Society	1.62%	1.88%	2.59%	1.50%	1.27%	1.43%	1.51%	0.67%
Total	100%	100%	100%	100%	100%	100%	100%	100%

EXAMPLE 1: RESULTS - SOURCE DISTRIBUTION OVER TIME

EXAMPLE 2: POLITICAL MIGRATION ON SOCIAL MEDIA

- Heidenreich and colleagues (2019), analyzed migration discourses in the Facebook accounts of political actors (n=1702) across six European countries (Spain, UK, Germany, Austria, Sweden and Poland)
- present new insights into the visibility of migration as a topic
- investigated sentiment about migration, revealing country- and party-specific patterns

Heidenreich et al., 2019

EXAMPLE 2: METHODS

- Downloaded textual data for all migration-related posts (n = 24,578) of members of parliaments (n = 1702) in six countries
- Used automated contend analysis to estimate sentiment towards migration in each post
- Machine translated the whole corpus into English

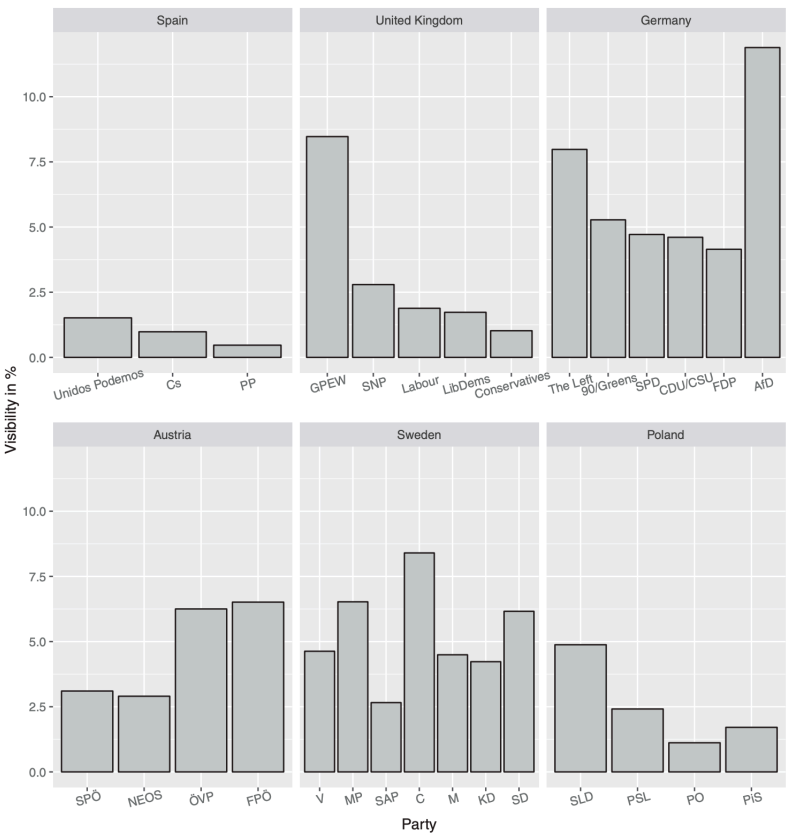
Table 1. Verified Facebook accounts.

Country	Members of parliament	No. with a verified Facebook accounts	Percentage with a verified Facebook account
Spain	349	120	34%
UK	650	559	86%
Germany	709	559	79%
Austria	183	97	53%
Sweden	349	126	36%
Poland	460	241	52%
Total	**2700**	**1702**	**63%**

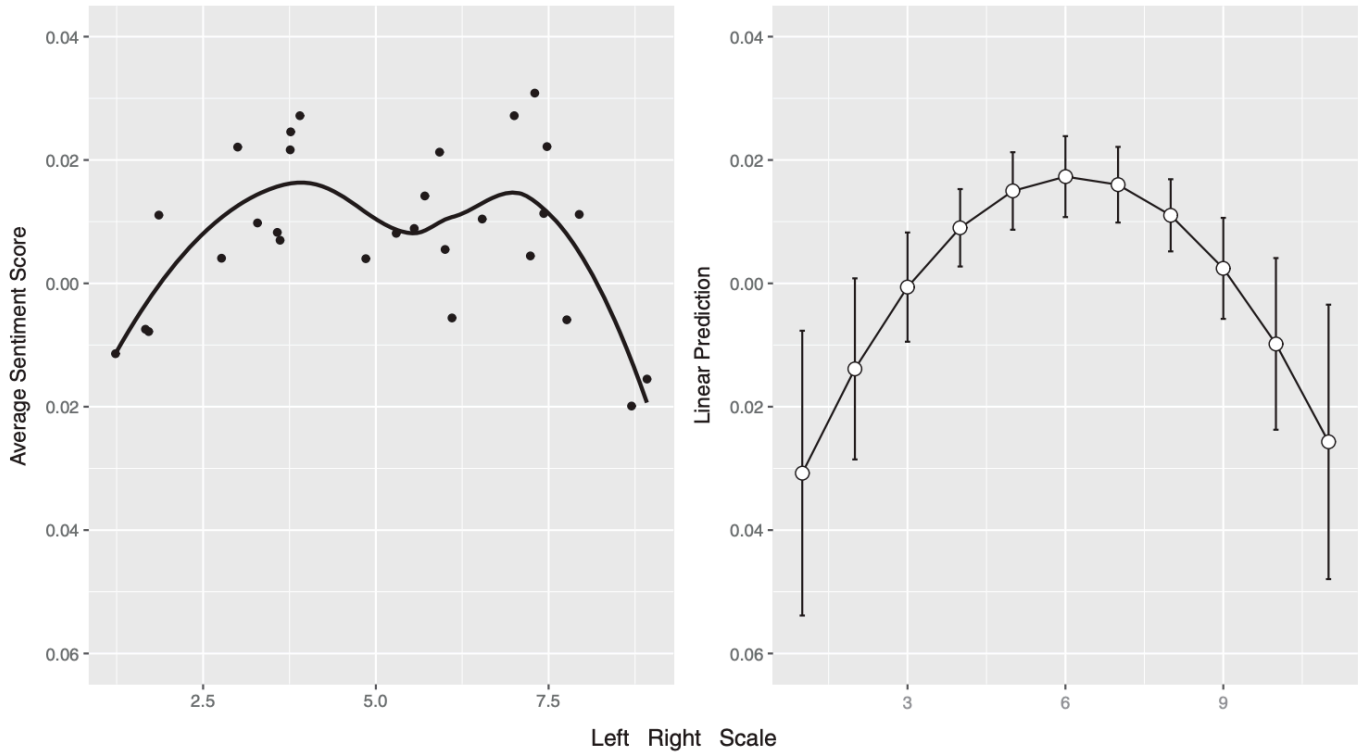
- Used a dictionary-approach (Lexicoder; Young and Soroka, 2012) to cound positive and negative words
- Computed sentiment for each document by calculating as the sum of the scores for all words bearing positive sentiment minus the sum of all scores from negative words, divided by the number of words.

EXAMPLE 2: RESULTS - VISIBILITY

- In Germany and Austria there is indeed descriptive evidence that parties of the right discuss migration more frequently in their Facebook status posts than other parties
- Conversely, in Spain, the UK and Poland the topic tended to be more prominent in the posts of left-wing parties
- At first glance, Sweden seems to be an outlier.
- In sum there is no consistent overall pattern supporting the hypothesis that right wing parties pay more attention to the topic of migration on Facebook than left leaning parties



EXAMPLE 2: RESULTS - SENTIMENT



EXAMPLE 2: RESULTS - PREDICTING SENTIMENT

Table 5. Summary of regression analysis with sentiment as the dependent variable.

	Sentiment (β)			
Predictor variables	Model 1a	Model 1b	Model 2a	Model 2b
Left-right score	−.06	.09	−.01	.01
Squared left-right score		−.63***		−.61***
Country (Reference: Poland)				
Spain	.04	.12		
United Kingdom	.16	.19		
Germany	−.10	.04		
Austria	−.21	−.18		
Sweden	.00	.08		
Receiving Country (0/1)			−.07	−.03
R^2	−.16	.29	−.07	.31

Notes: $n = 29$; *** $p < .001$. Squared left-right score was included in the model via an interaction term (centred left-right score*left-right score). All coefficients are standardised. The reference category for the country covariate was Poland.

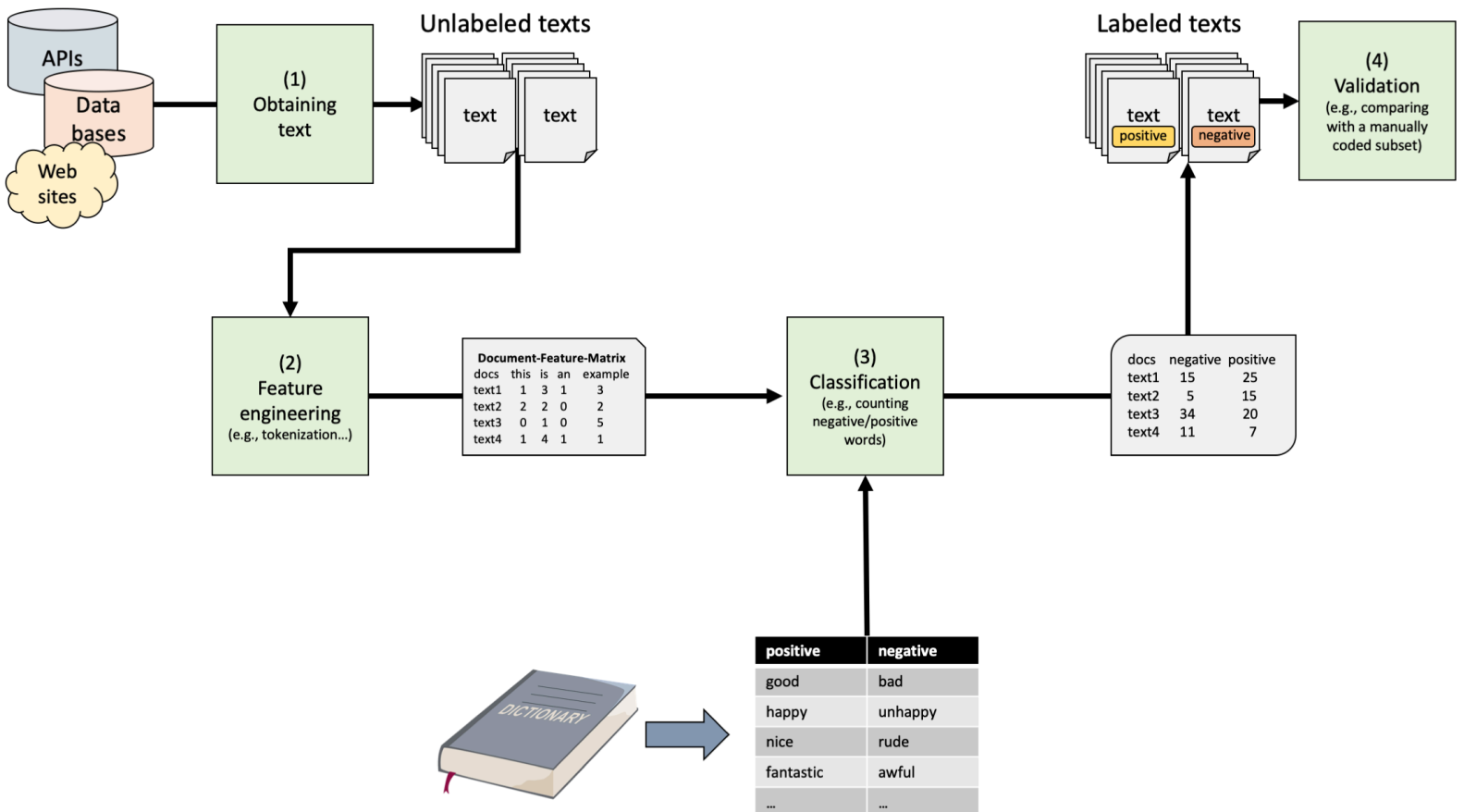
EXAMPLE 2: CONCLUSION

- Migration is a more prominent in countries with positive net migration than in countries where net migration is neutral or negative
- They did not find support for the assumption that right-leaning parties talk more, and more negatively, about migration
- However, political actors from parties of the extreme left and the extreme right of the political spectrum address migration more frequently and more negatively than more moderate political players
- Potential limitations
  - How valid is the automated translation of text into English?
  - How valid is the sentiment approximation based on dictionaries?

Summary and Conclusion

What have we learned?

OVERVIEW


APIs
Data bases
Web sites
(1) Obtaining text
Unlabeled texts
text
text
(2) Feature engineering
(e.g., tokenization...)
Document-Feature-Matrix
docs this is an example
text1 1 3 1 3
text2 2 2 0 2
text3 0 1 0 5
text4 1 4 1 1
(3) Classification
(e.g., counting negative/positive words)
docs negative positive
text1 15 25
text2 5 15
text3 34 20
text4 11 7
Labeled texts
text positive
text negative
(4) Validation
(e.g., comparing with a manually coded subset)
DICTIONARY
positive negative
good bad
happy unhappy
nice rude
fantastic awful
... ...

CONCLUSION

- Automated text analysis is a useful for working with large-scale text corpora
- Dictionaries can be extremely helpful in detecting certain specific terms (e.g., authors, certain words, labels, names…)
- Yet, they are not very good at coding more complex concepts.
- After all, their validity depends on how well a concept can be represented by a simple word list
- We always need to validate our analysis using manually coded material!

NEXT WEEK

- Introduction to supervised machine learning
- Using neural networks for text classification
- The idea of deep learning
- Better word representations using word-embeddings

REQUIRED READING

- Welbers, K., Van Atteveldt, W., & Benoit, K. (2017). Text analysis in R. Communication Methods and Measures, 11(4), 245-265.
- Heidenreich, T., Eberl, J.-M., Lind, F. & Boomgaarden, H. (2020). Political migration discourses on social media: a comparative perspective on visibility and sentiment across political Facebook accounts in Europe. Journal of Ethnic and Migration Studies, (46)7, 1261-1280, https://doi.org/10.1080/1369183X.2019.1665990
- Mellado, C., Hallin, D., Cárcamo, L. Alfaro, R. … & Ramos, A. (2021) Sourcing Pandemic News: A Cross-National Computational Analysis of Mainstream Media Coverage of COVID-19 on Facebook, Twitter, and Instagram. Digital Journalism, 9(9), 1271-1295, https://doi.org/10.1080/21670811.2021.1942114

(available on Canvas)

REFERENCES

- Boumans, J. W., & Trilling, D. (2016). Taking stock of the toolkit: An overview of relevant automated content analysis approaches and techniques for digital journalism scholars. Digital journalism, 4(1), 8-23.
- Heidenreich, T., Eberl, J.-M., Lind, F. & Boomgaarden, H. (2020). Political migration discourses on social media: a comparative perspective on visibility and sentiment across political Facebook accounts in Europe. Journal of Ethnic and Migration Studies, (46)7, 1261-1280, DOI: 10.1080/1369183X.2019.1665990
- Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. Science, 332(6025), 60 –65. https://doi.org/10.1126/science.1200970
- Hvitfeld, E. & Silge, J. (2021). Supervised Machine Learning for Text Analysis in R. CRC Press. https://smltar.com/
- Krippendorff, K. (2004). Content Analysis. Sage.
- Mohammad, S. and Turney, P. (2013). Crowdsourcing a Word-Emotion Association Lexicon. Computational Intelligence, 29(3): 436-465.
- Xue, S. (1982). Chinese Lexicography Past and Present, Dictionaries, 4, 151–169. https://doi.org/10.1353/dic.1982.0009

EXAMPLE EXAM QUESTION (MULTIPLE CHOICE)

When using dictionaries, a very important step refers to “transforming the text to structured data” of feature engineering. What are typical preprocessing steps in this phase?

A. Tokenization, translation, combination, and frequency trimming.

B. Tokenization, stopword removal, normalization, and frequency trimming.

C. Tokenization, inclusion of annotations, interpretation, and frequency trimming.

D. Tokenization, stopword removal, combination, and inclusion of annotations.

EXAMPLE EXAM QUESTION (MULTIPLE CHOICE)

When using dictionaries, a very important step refers to “transforming the text to structured data” of feature engineering. What are typical preprocessing steps in this phase?

A. Tokenization, translation, combination, and frequency trimming.

B. Tokenization, stopword removal, normalization, and frequency trimming.

C. Tokenization, inclusion of annotations, interpretation, and frequency trimming.

D. Tokenization, stopword removal, combination, and inclusion of annotations.

EXAMPLE EXAM QUESTION (OPEN FORMAT)

Name and explain two disadvantages of dictionary approaches.
(4 points, 2 points for correctly naming two disadvantages and 2 points for correctly explaining them)

1. Low validity: Dictionary approaches measure the concepts of interest based on simple word lists. Particularly with non-trivial and complex concepts (e.g., sentiment, frames, topics,…), the assumption that this word in a reliable and valid way may be problematic. For example, true sentiment is more than just counting positive and negative words.
2. May requires considerable text preprocessing: To reduce ambiguity, dictionary approaches require a lot of text preprocessing (e.g., removing stopwords, punctuation, numbers, stemming, lemmatization). For example, the word “like” would be coded as a positive word in most sentiment dictionaries, but in most sentences, it only refers to something being “like” something else.

Thank you for your attention!