Introduction to Text Analaysis

SGSSS Workshop Series

Theresa Gessler

University of Zurich | http://theresagessler.eu/ | @th_ges

11 November 2020

Program

What is text analysis

How to

- Corpus
- Tokens
- Document Feature Matrix

Preparing Text Data (pre-processing)

Wrangling Text Data for Analysis (transforming)

Introduction

About me

- 2015–2019: PhD in Political Science (European University Institute)
- since 2019: Postdoctoral Researcher (University of Zurich, Digital Democracy Lab and Department of Political Science)
- co-organizer of the Zurich Summer School for Women in Political Methodology

Research Interests

- Immigration, Democracy, Digitalization
- Political Behavior, Party Politics, Political Communication
- Computational Social Science, Quantitative Text Analysis

Text Analysis Tools

package for the **qu**antitative **an**alysis of **te**xtual **da**ta (https://quanteda.io/)

```
install.packages("quanteda")
library(quanteda)
library(tidyverse)
```

Cheat sheets & Co

- RStudio Cheat Sheets: https://rstudio.com/resources/cheatsheets/
- Stefan's quanteda cheat sheet: https://muellerstefan.net/files/quanteda-cheatsheet.pdf
- Quanteda Tutorials by Kohei Watanabe & Stefan: https://tutorials.quanteda.io/

Text Analysis

The spectrum

manual / hermeneutic analysis of content ↔ automated analysis of content

→ we focus on **automated analysis**, with varying degrees of human input

Definitions

- Systematic, objective, quantitative analysis of message characteristics (Neuendorf 2002, *The Content Analysis Guidebook*, 1)
- A variant of content analysis that is expressly quantititative, not just in terms of representing textual content numerically but also in analyzing it, typically using computation and statistical methods. (text analysis course by Ken Benoit)
- many related methods: content analysis, text analysis, text mining, natural language processing, text as data, ...

Why text as data

- traditional empirical work in political and social science
  - **quantitative methods** with limited understanding of (unstructured) text
  - **qualitative methods** with close analysis of small text collections

↑

↓

- masses of available text
  - e.g. by governments, media, organizations, laws, court decisions, speeches, ...
  - digitalization of existing text collections

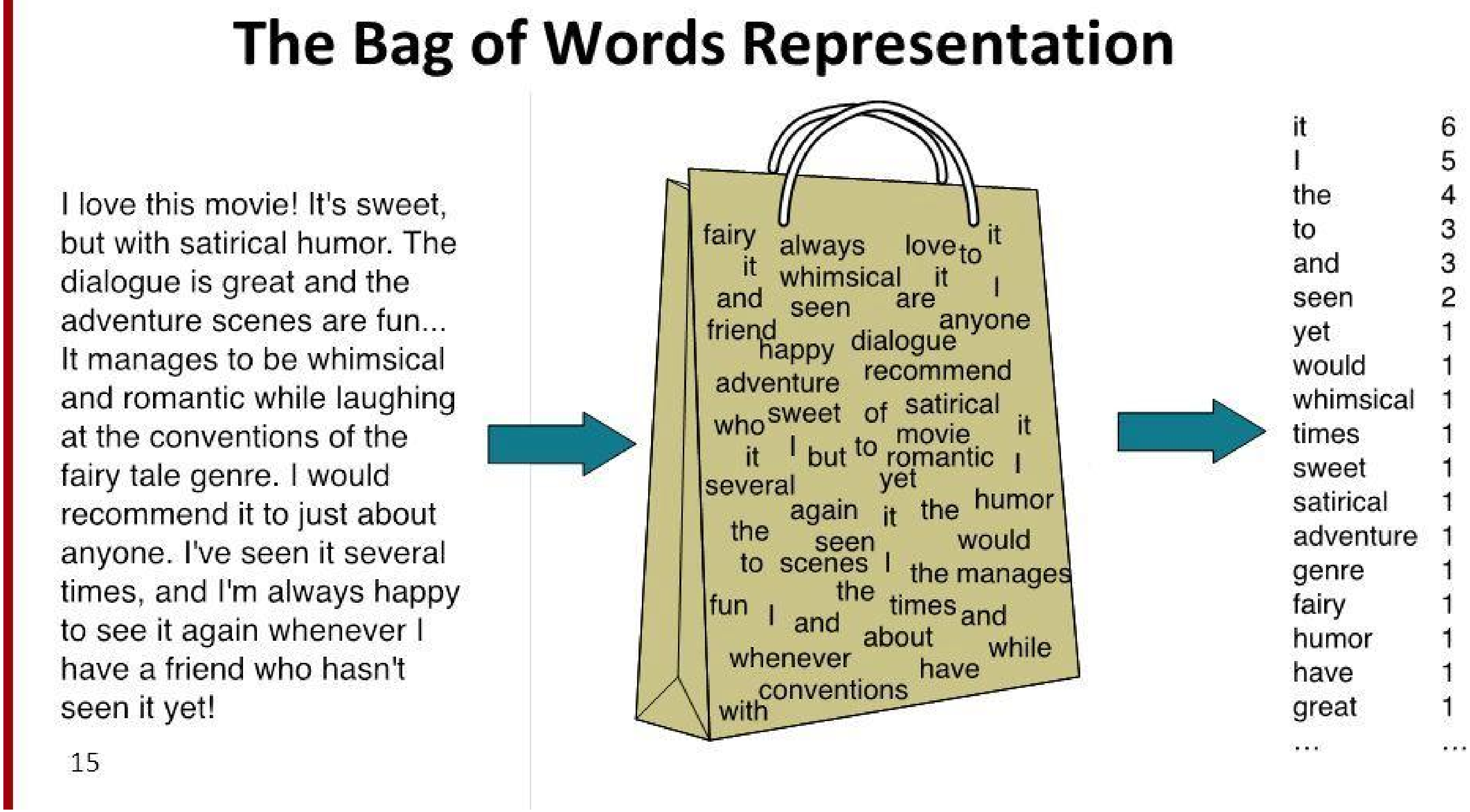
→ **untapped potential of interesting (new) data!**

Basic assumptions

When doing quantitative text analysis, we assume...

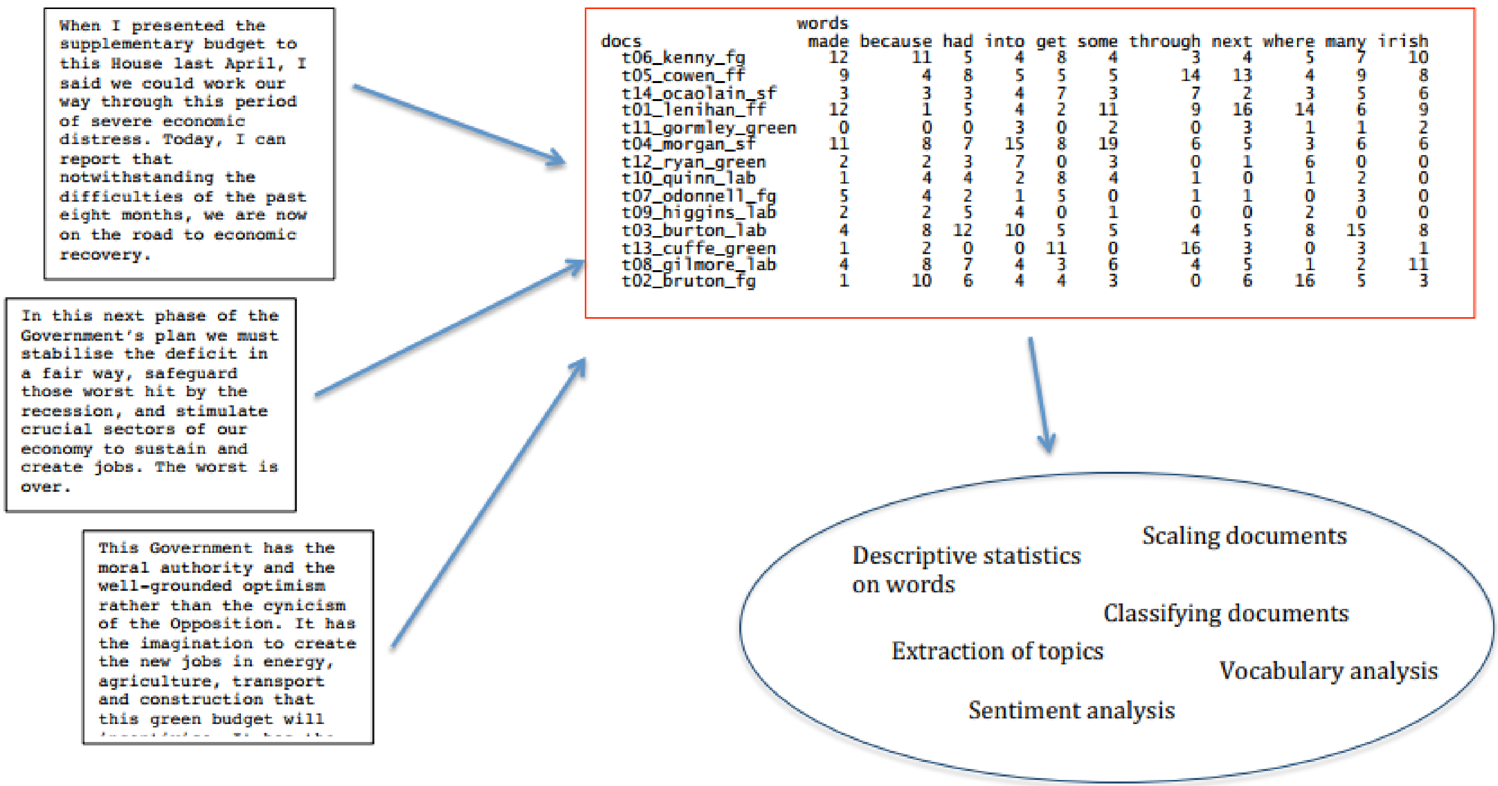
- ...That texts represent an **observable implication** of some **underlying characteristic** of interest (usually an attribute of the author)
- ...That texts can be represented through extracting their **features**, e.g. words
- ...That we can analyze a **document-feature matrix** with quantitative methods to measure these underlying characteristics

The 'bag of words' assumption



Source: programmersought.com

The 'bag of words' assumption



docs	words made	because	had	into	get	some	through	next	where	many	irish
t06_kenny_fg	12	11	5	4	8	4	3	4	5	7	10
t05_cowen_ff	9	4	8	5	5	5	14	13	4	9	8
t14_ocaolain_sf	3	3	3	4	7	3	7	2	3	5	6
t01_lenihan_ff	12	1	5	4	2	11	9	16	14	6	9
t11_gormley_green	0	0	0	3	0	2	0	3	1	1	2
t04_morgan_sf	11	8	7	15	8	19	6	5	3	6	6
t12_ryan_green	2	2	3	7	0	3	0	1	6	0	0
t10_quinn_lab	1	4	4	2	8	4	1	0	1	2	0
t07_odonnell_fg	5	4	2	1	5	0	1	1	0	3	0
t09_higgins_lab	2	2	5	4	0	1	0	0	2	0	0
t03_burton_lab	4	8	12	10	5	5	4	5	8	15	8
t13_cuffe_green	1	2	0	0	11	0	16	3	0	3	1
t08_gilmore_lab	4	8	7	4	3	6	4	5	1	2	11
t02_bruton_fg	1	10	6	4	4	3	0	6	16	5	3

Source: Slapin, J. B., and S.-O. Proksch (2008). A Scaling Model for Estimating Time-Series Party Positions from Texts." American Journal of Political Science 52 (3): 705-22.

Methods of automated text analysis

Grimmer, J. and B. M. Stewart (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. Political Analysis 21, 267-297.

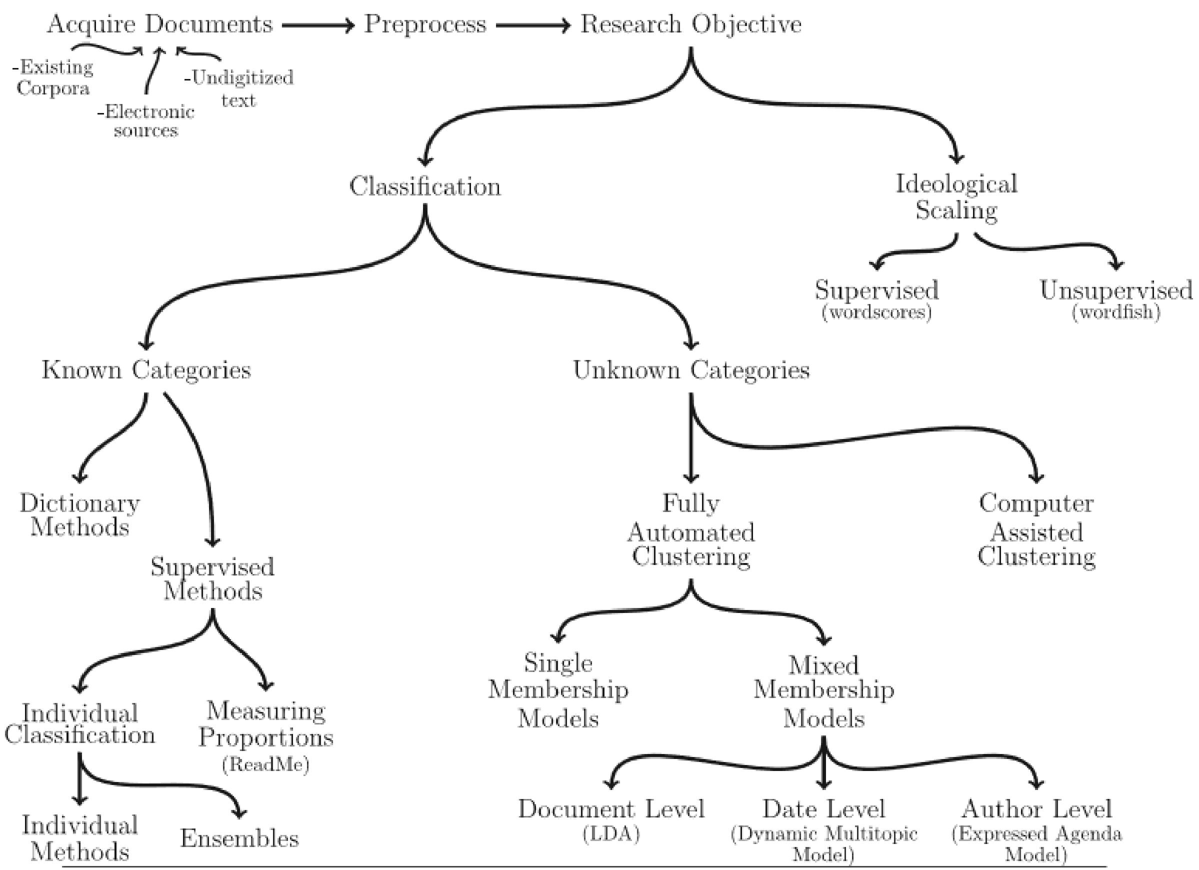


Fig. 1 An overview of text as data methods.

How To

Analysis process

Getting the texts

(existing records, scanning, scraping, ...)

↓

convert the text to data

(cleaning, 'pre-processing')

↓

quantitative text analysis

↓

Validation and interpretation

Workflow

Three types of objects in quanteda:

- **corpus**
  - texts as strings with metadata in data frame
- **tokens**
  - separated individual features in list of vectors
  - more efficient but maintains the word order
- **document-feature matrix (dfm)**
  - Frequency of features per document in matrix / table format
  - most efficient structure, but no information about positions of the words ('bag of words')

Corpus

Exercises: US Presidential Debate

- First presidential debate between Donald Trump & Joe Biden, moderated by Chris Wallace
- debate transcript with speakers and time stamps

Transcript obtained from Kaggle: https://www.kaggle.com/headsortails/us-election-2020-presidential-debates

Exercise Corpus

- **load** 'us_election_2020_1st_presidential_debate.csv'
- **inspect** the dataset: content, structure, variables
  - *bonus*: **wrangle**: generate a shorter speaker variable
- **corpus**: use `corpus()` to create a `quanteda` corpus
  - *bonus*: specify useful names for each text in the corpus

Solution Corpus

```r
first_debate <- read.csv("us_election_2020_1st_presidential_debate.csv",
  stringsAsFactors = F,encoding="UTF-8")
head(first_debate)
```

```
##                      speaker minute                                   text
## 1              Chris Wallace  01:20 Good evening from the Health E
## 2              Chris Wallace  02:10 This debate is being conducted
## 3  Vice President Joe Biden  02:49            How you doing, man?
## 4 President Donald J. Trump  02:51             How are you doing?
## 5  Vice President Joe Biden  02:51                      I'm well.
## 6              Chris Wallace  03:11 Gentlemen, a lot of people bee
```

Solution Corpus

```
# optional : speaker
first_debate ← first_debate %>%
  mutate(speaker=str_extract(speaker,"[A-z]*$"))

debate_corp ← corpus(first_debate)

# optional : renaming
docnames(debate_corp) ← paste0(1:nrow(first_debate),"_",
                                first_debate$speaker)
```

Corpus

- **corpus**: Structured collection of texts
  - Documents: Texts
  - Document variables / docvars: variables obtained from data set

```
debate_corp[1:4]
```

```
## Corpus consisting of 4 documents and 2 docvars.
## 1_Wallace :
## "Good evening from the Health Education Campus of Case Wester..."
##
## 2_Wallace :
## "This debate is being conducted under health and safety proto..."
##
## 3_Biden :
## "How you doing, man?"
##
## 4_Trump :
## "How are you doing?"
```

Corpus

Summary of the corpus

```
summary(debate_corp) %>% head()
```

```
##        Text Types Tokens Sentences speaker minute
## 1 1_Wallace    88    135         8 Wallace  01:20
## 2 2_Wallace    83    116         5 Wallace  02:10
## 3   3_Biden     6      6         1   Biden  02:49
## 4   4_Trump     5      5         1   Trump  02:51
## 5   5_Biden     3      3         1   Biden  02:51
## 6 6_Wallace    89    149         9 Wallace  03:11
```

Important terms

- **Text**: each document of the corpus
- **Tokens**: total number of words in a text (or corpus), independent of repetitions
- **Types**: Number of different words in a text (or corpus)

Corpus

Analysis: Keywords in context

In which context are terms used in the text corpus?

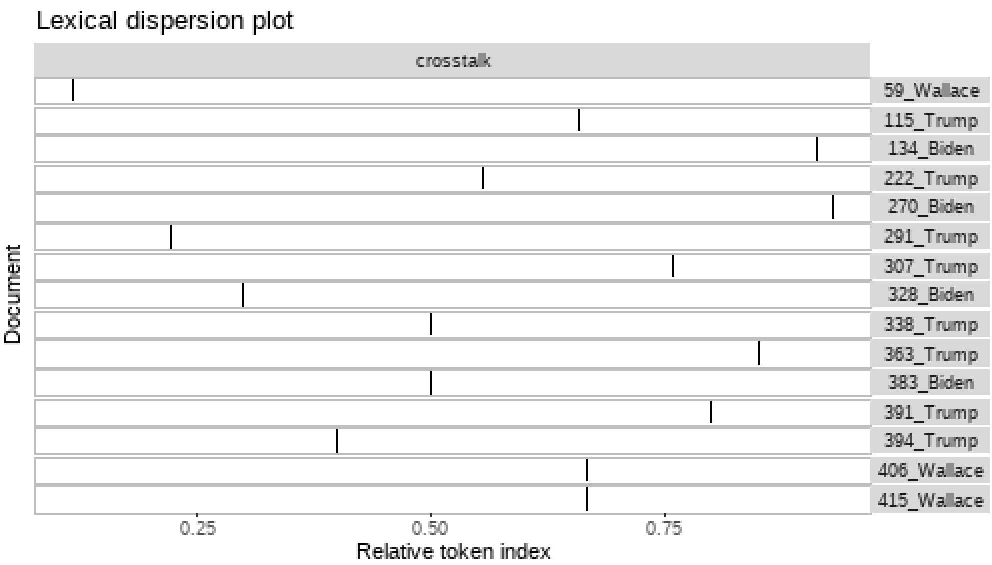
```
kwic(debate_corp, "country",window=4) %>%
  head()
```

```
##
##    [167_Trump, 9]            to you, the | country | would have been left
##  [167_Trump, 150] should have closed our | country | . Wait a minute
##    [169_Trump, 9] should have closed our | country | because you thought it
##   [215_Trump, 36]     the history of our | country | . And by the
##    [226_Trump, 9]      to shut down this | country | and I want to
##   [228_Trump, 29]       to shut down the | country | . We just went
```

Analysis: Use

Where are terms used in the text corpus?

```
kwic(debate_corp, "crosstalk") %>% head(15) %>%
  textplot_xray()
```



Text statistics

How are the texts written in general?

e.g. readability statistics at text level

```
textstat_readability(debate_corp) %>% head()
```

```
##    document    Flesch
## 1 1_Wallace  62.15573
## 2 2_Wallace  50.10547
## 3   3_Biden  97.02500
## 4   4_Trump  97.02500
## 5   5_Biden 120.20500
## 6 6_Wallace  70.34232
```

e.g. frequent word combinations: `textstat_collocations()`

Exercise Readability

- **calculate** readability score
  - *bonus*: check the documentation for different metrics and look at the differences
- **merge** it back to the original dataset
- **wrangle:** who is on average most readable?

Solution Readability

```
# calculate readability
readability_stats <- textstat_readability(debate_corp)

# merging
first_debate$document <- paste0(1:nrow(first_debate),"_",
                                first_debate$speaker)
readability_stats <- left_join(first_debate,readability_stats)

# analyze
readability_stats %>%
  group_by(speaker) %>%
  select(Flesch) %>%
  summarize_all(mean)
```

```
## # A tibble: 3 x 2
##   speaker Flesch
##   <chr>    <dbl>
## 1 Biden     86.0
## 2 Trump     88.4
## 3 Wallace   73.0
```

Tokens

Tokens

- **individual features**, stored in list of vectors
- more efficient format than corpus but retains the word order
  - *'chop' the sentences without 'shaking' the bag*

Use

- **Keywords in Context** (also at corpus-level)
- **pre-processing** (also at dfm-level)
  - removing irrelevant features, manipulation of features
  - *advantage of tokens*: word order provides context
- **Dictionaries** (also at dfm-level)
  - *advantage of tokens*: multi-word expressions, word order as context

→ **What constitutes a feature (word, n-gram, sentence, letter)?**

→ **Which features are relevant data? How do I prepare them?**

Tokenization

- separation into features is called **tokenization** (command: `tokens()`)
- is possible at different levels: word, sentence or character.

```
tokens(debate_corp, what="word")[[1]][1:10]
```

```
##  [1] "Good"      "evening"   "from"      "the"       "Health"    "Education"
##  [7] "Campus"    "of"        "Case"      "Western"
```

```
tokens(debate_corp, what="character")[[1]][1:10]
```

```
##  [1] "G" "o" "o" "d" "e" "v" "e" "n" "i" "n"
```

```
debate_toks <- tokens(debate_corp)
```

→ We return to tokens later for pre-processing and dictionaries

Document feature matrix

Document feature matrix

- **frequency of features per document in matrix format**
- most efficient structure, but no information about positions of the words → 'bag of words'
- origin for most statistical analyses
  - combination of word frequency with document variables

```
debate_dfm <- dfm(debate_toks)
debate_dfm
```

```
## Document-feature matrix of: 789 documents, 2,297 features (99.2% sparse) and 2
docvars.
##            features
## docs         good evening from the health education campus of case western
##   1_Wallace    1       1    2  15      1         1      1  5    1       1
##   2_Wallace    0       0    0  10      2         0      0  1    0       0
##   3_Biden      0       0    0   0      0         0      0  0    0       0
##   4_Trump      0       0    0   0      0         0      0  0    0       0
##   5_Biden      0       0    0   0      0         0      0  0    0       0
##   6_Wallace    0       0    0  10      0         0      0  3    0       0
## [ reached max_ndoc ... 783 more documents, reached max_nfeat ... 2,287 more
features ]
```

Document feature matrix

- can be obtained (indirectly) from corpus or (directly) from tokens object

```
debate_dfm1 <- dfm(debate_toks)
debate_dfm2 <- dfm(debate_corp)
identical(debate_dfm1,debate_dfm2)
```

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

Document feature matrix

Looking inside

```
# Most frequent features
topfeatures(debate_dfm)
```

```
##    .    ,  the   to  you  and    a   of   in that
## 1627 1127  806  562  524  468  391  358  305  299
```

```
# numbers of features / documents
nfeat(debate_dfm)
```

```
## [1] 2297
```

```
ndoc(debate_dfm)
```

```
## [1] 789
```

Document feature matrix

Looking inside

```
# Names of features
featnames(debate_dfm) %>% head()
```

```
## [1] "good"      "evening"   "from"      "the"       "health"    "education"
```

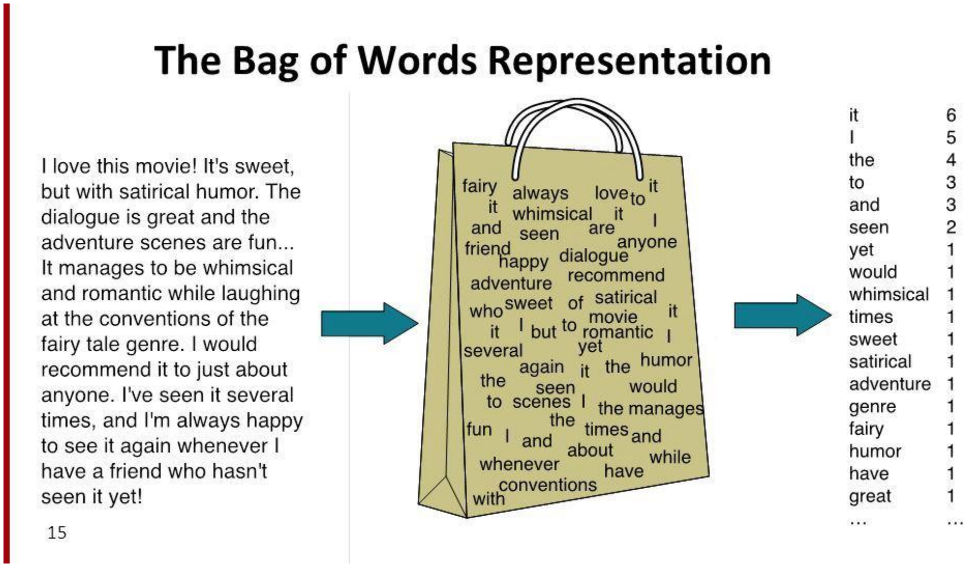
```
# Frequency of features
featfreq(debate_dfm) %>% head()
```

```
##      good   evening      from       the    health education
##        31         1        34       806        11         2
```

Pre-processing

Pre-processing

- only certain features provide **relevant** information




```
topfeatures(debate_dfm,20)
```

```
##      .      ,    the     to    you    and      a     of     in   that      i
##   1627   1127    806    562    524    468    391    358    305    299    254
##     it     is   have     he     we people   they      :  going
##    254    230    215    205    183    159    149    149    144
```

Pre-processing

Pre-processing as feature selection

- Examples of features without information
  - Capitalization at the beginning of the sentence
  - 'fillers'
  - singular ↔ plural, cases
- *whether a feature is irrelevant depends on the question*
  - e.g. analysis of 'empty talk', specific concepts, ...

A longer reflection on pre-processing: Denny, M. J. and A. Spirling (2018). Text preprocessing for unsupervised learning: why it matters, when it misleads, and what to do about it. Political Analysis

The impact of pre-processing: preText package

Pre-processing as feature selection

Three broad types of pre-processing:

- **restricting features**:
  - removing punctuation
  - removing numbers
  - removing hashtags and URLs
  - ...
- **removing uninformative features**:
  - 'stopwords'
  - trimming frequent or rare words
- **uniting features**:
  - lowercasing
  - stemming
  - rarely: replacing or compounding of words

Restricting features

- removing punctuation
- removing numbers
- removing hashtags and URLs
- …

Pre-processing: Restricting features

- features beyond words are typically less informative
  - exception: linguistic analysis
- often done during tokenization

From dfm:

```
dfm(debate_corp,remove_punct=T) %>% topfeatures()
```

```
##  the   to  you  and    a   of   in that    i   it
##  806  562  524  468  391  358  305  299  254  254
```

From tokens:

```
debate_toks <- tokens(debate_corp,
  remove_punct=T,
  remove_numbers=T,
  remove_symbols=T)
```

Removing features

- 'stopwords'
- trimming frequent or rare words

Pre-processing: Removing features

- **Common practice**: removing frequent but less meaningful features (so-called 'stopwords')
  - default *stopwords* contains words like 'is', 'the', ...
  - can be defined by researcher, e.g. procedural terms in parliamentary records

From dfm, starting with 2297 features:

```
dfm(debate_corp,remove=stopwords("en")) %>% nfeat()
```

```
## [1] 2144
```

```
debate_dfm <- debate_corp %>% dfm() %>%   dfm_remove(stopwords("en")) %>% nfeat()
```

From tokens:

```
debate_toks <- tokens_remove(debate_toks,stopwords("en"))
```

Pre-processing: Removing features

- **Case-by-case**: removing (frequent or rare) words
  - `dfm_trim()` makes dfm sparser by frequency
    - document-based and corpus-based options
    - can be done in addition or as replacement of stopword removal
  - `dfm_select()` keeps only selected features

From dfm, starting with 2297 features:

```
dfm(debate_corp) %>% dfm_trim(min_termfreq = 5) %>% nfeat()
```

```
## [1] 550
```

```
dfm(debate_corp) %>% dfm_select("*virus*") %>% nfeat()
```

```
## [1] 2
```

From tokens:

```
debate_toks <- tokens_select(debate_toks,"*virus*")
```

Uniting features

- lowercasing
- stemming
- rarely: replacing or compounding of words

Pre-processing: Uniting features

- **By default**: lowercasing
  - automatically with `dfm()` (but can be disabled)
  - optional with tokens

```
dfm(debate_toks,tolower=F)[1:3,1:3]
```

```
## Document-feature matrix of: 3 documents, 3 features (55.6% sparse) and 2 docvars.
##            features
## docs         Good evening Health
##   1_Wallace    1       1      1
##   2_Wallace    0       0      1
##   3_Biden      0       0      0
```

```
tokens_tolower(debate_toks)[[1]][1:20]
```

```
##  [1] "good"         "evening"      "health"       "education"    "campus"
##  [6] "case"         "western"      "reserve"      "university"   "cleveland"
## [11] "clinic"       "chris"        "wallace"      "fox"          "news"
## [16] "welcome"      "first"        "presidential" "debates"      "president"
```

Pre-processing: Uniting features

- **Case-by-case**: altering features
  - stemming or lemmatization

```
dfm(debate_toks,stem=T) %>% topfeatures()
```

```
##        go     peopl    presid      want       say crosstalk      said      know
##       200       160       146        94        85        73        69        67
##      look      well
##        66        58
```

- **stemming**: cutting off the endings of words
  - transform similar words into the same feature, e.g. "immigrants" and "immigrate"
  - many algorithms and language dependent - worth looking at it once
- **lemmatization**: better but more complicated variant: lemmatization

Pre-processing: Uniting features

- **Case-by-case**: altering features
  - splitting hyphenated features
  - replacing features: `dfm_replace()`, `tokens_replace()`
  - compounding features: `tokens_compound()`

From dfm, starting with 2297 features:

```
dfm(debate_corp,split_hyphens=T) %>% nfeat()
```

```
## [1] 2173
```

```
dfm(debate_corp) %>%
dfm_replace(c("say","said"),rep("SPEAK",2)) %>% nfeat()
```

```
## [1] 2296
```

From tokens

```
tokens(debate_toks,split_hyphens=T)
tokens_replace(debate_toks,c("say","said"),rep("say",2))
```

Exercise Pre-processing

- **create a dfm**
- **feature selection**: look through the english stopwords: Is there anything you might not want to remove?
- **feature selection**: experiment with the trimming function
  - remove features occuring less than 10 times
  - remove features occuring in less than 2 documents
  - remove features occuring more than 50 times

Solution Pre-processing

```
debate_dfm <- dfm(debate_corp)

# stopwords
stopwords("en")
```

```
##   [1] "i"          "me"         "my"         "myself"     "we"
##   [6] "our"        "ours"       "ourselves"  "you"        "your"
##  [11] "yours"      "yourself"   "yourselves" "he"         "him"
##  [16] "his"        "himself"    "she"        "her"        "hers"
##  [21] "herself"    "it"         "its"        "itself"     "they"
##  [26] "them"       "their"      "theirs"     "themselves" "what"
##  [31] "which"      "who"        "whom"       "this"       "that"
##  [36] "these"      "those"      "am"         "is"         "are"
##  [41] "was"        "were"       "be"         "been"       "being"
##  [46] "have"       "has"        "had"        "having"     "do"
##  [51] "does"       "did"        "doing"      "would"      "should"
##  [56] "could"      "ought"      "i'm"        "you're"     "he's"
##  [61] "she's"      "it's"       "we're"      "they're"    "i've"
##  [66] "you've"     "we've"      "they've"    "i'd"        "you'd"
##  [71] "he'd"       "she'd"      "we'd"       "they'd"     "i'll"
##  [76] "you'll"     "he'll"      "she'll"     "we'll"      "they'll"
##  [81] "isn't"      "aren't"     "wasn't"     "weren't"    "hasn't"
##  [86] "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
```

Solution Pre-processing

```
# trimming
dfm(debate_corp) %>%
    dfm_trim(min_termfreq = 10,
      min_docfreq = 2,
      max_termfreq = 50,
      verbose=T)
```

```
## Document-feature matrix of: 789 documents, 226 features (97.7% sparse) and 2
docvars.
##            features
## docs         good from health chris first trump vice joe biden minute
##   1_Wallace    1    2      1     1     2     1    1   1     1      2
##   2_Wallace    0    0      2     0     0     1    1   0     1      0
##   3_Biden      0    0      0     0     0     0    0   0     0      0
##   4_Trump      0    0      0     0     0     0    0   0     0      0
##   5_Biden      0    0      0     0     0     0    0   0     0      0
##   6_Wallace    0    0      0     0     4     2    1   0     1      0
## [ reached max_ndoc ... 783 more documents, reached max_nfeat ... 216 more features
]
```

Summary

Three types of objects in quanteda:

- **corpus**
  - Text as strings with metadata in data frame
- **tokens**
  - individual features in list of vectors
  - more efficient but maintains the word order
- **document-feature matrix (dfm)**
  - Frequency of features per document in matrix / table format
  - most efficient structure, but no information about positions of the words ('bag of words')

Transforming dfms

In order to statistically evaluate feature frequencies, we have to **transform the document feature matrix**

→ summarize documents, select features, select documents etc.

- `dfm_subset()` : Selection based on docvars
- `dfm_select()`: Selection of features
  - `dfm_trim()`: Selection of features based on frequency
- `dfm_group()` : grouping of documents based on docvars
- `dfm_weight()` & `dfm_tfidf()` : weighting the feature counts
- `dfm_lookup()` : Looking up dictionaries

→ We'll use some of them next week

→ **Exercises on transforming dfms**

Thank you

gessler@ipz.uzh.ch

http://theresagessler.eu | @th_ges