Basics of Quantitative Text Analysis

Bamberg Summer Institute in Computational Social Science

Carsten Schwemmer, University of Bamberg

2019-08-01

Before we start

These are the new packages that we will need today:

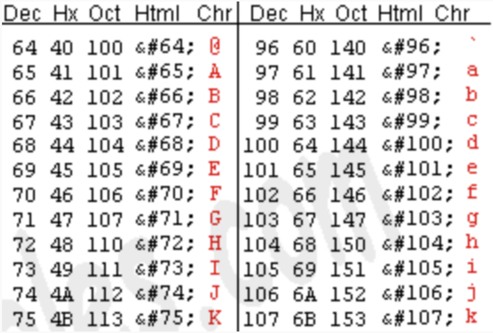
```r
install.packages(c('stm', 'stminsights', 'textdata',
                   'quanteda', 'caret'))
```

Here is the location for lecture materials (including datasets):
https://github.com/compsocialscience/summer-institute/tree/master/2019/bamberg/materials

Basics of quantitative text analysis

Character encoding

Computers store text in form of digits. For each character, e.g. a, there is a corresponding sequence of numbers used for character encoding.



Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
64	40	100	&#64;	@	96	60	140	&#96;	`
65	41	101	&#65;	A	97	61	141	&#97;	a
66	42	102	&#66;	B	98	62	142	&#98;	b
67	43	103	&#67;	C	99	63	143	&#99;	c
68	44	104	&#68;	D	100	64	144	&#100;	d
69	45	105	&#69;	E	101	65	145	&#101;	e
70	46	106	&#70;	F	102	66	146	&#102;	f
71	47	107	&#71;	G	103	67	147	&#103;	g
72	48	110	&#72;	H	104	68	150	&#104;	h
73	49	111	&#73;	I	105	69	151	&#105;	i
74	4A	112	&#74;	J	106	6A	152	&#106;	j
75	4B	113	&#75;	K	107	6B	153	&#107;	k

https://www.asciitable.com/

Character encoding

Today, many different standards for text encoding exist (e.g. ASCII, UTF-8, UTF-16, Latin-1, ..). If you read textual data and declare the wrong encoding, some characters will not be parsed correctly:

```
all_good <- readLines('encoding_issues_utf8.txt',
                     encoding = 'UTF-8')
all_good
```

```
## [1] "NLP rocks" "Éncôdíng_cäuses_headaçhe$"
```

```
suffering <- readLines('encoding_issues_utf8.txt',
                       encoding = 'latin1')
suffering
```

```
## [1] "NLP rocks" "Ã<e2><80><b0>ncÃ´dÃng_cÃ¤uses_headaÃ§he$"
```

Character encoding - advice

- store your data in one consistent encoding to avoid headaches
- `utf-8` is commonly used and the default encoding for many R packages (e.g. `tidyverse`).
- if you don't know the encoding of a text, try several encodings and qualitatively inspect results.

Text is complex

- "Time flies like an arrow, fruit flies like a banana."
- "Make peace, not war; make war, not peace."
- quantitative models (need to) simplify textual data and will fail to capture some complexity

How can we deal with this complexity?

”We destroy language to turn it into data” - Ken Benoit, IC2S2 2019

Tokens

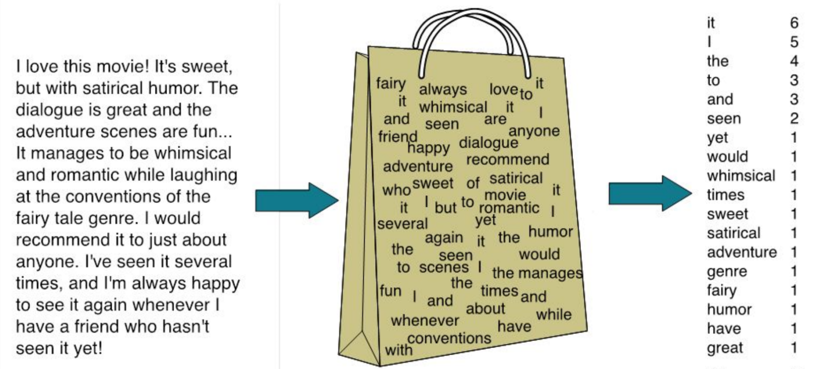
“Make love, not war.” -> `c('make', 'love', 'not', 'war')`

- tokens define semantic elements of strings; important for many applications of text analysis (e.g. counting)
- predominantly separated by white space and punctuation marks
- converting text to tokens raises several questions:
  - what unit of analysis? words? sentences?
  - which rules (algorithms) shall be used to define tokens?
  - which tokens are not useful for the analysis?

From tokens to “bag of words” (bow)

- disassembling texts into tokens is the foundation for the bag of words model (bow)
- bow is a very simplified representation of text where only token frequencies are considered
- advantages: easy to implement, scales well, often good enough
- disadvantages: discards language properties like polisemy and word order

Bag of words - example



I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1

First dataset for today

- we will be using a sample from a Kaggle Data Science for Good challenge
- DonorsChoose.org provided the data and hosts an online platform where teachers can post requests for resources and people can make donations to these projects
- the goal of the original challenge was to match previous donors with campaigns that would most likely inspire additional donations
- the dataset includes request texts and context information. A description of variables is available here

What could we learn from this data?

Examples of questions we might ask:

- how has classroom technology use changed over time? How does it differ by geographic location and the age of students?
- how do the requests of schools in urban areas compare to those in rural areas?
- what predicts whether a project will be funded?
- how do the predictors of funding success vary by geographic location? Or by economic status of the students?

Loading packages and data

```r
library(tidyverse)
df <- read_csv('data/donors_choose_sample.csv')
df %>% pull(project_title) %>%  head(5)
```

```
## [1] "Books for Brains"
## [2] "Softball Teams Need Gloves and Ball Part 3"
## [3] "Reading and Math Games for Sneaky Learning (They'll
Never Know!)"
## [4] "Hands-On, Visual Math Inspires Learning!"
## [5] "Working on Working Memory"
```

Text example for one donation request

```r
cat(df$project_essay[1])
```

```
## A typical day in our room consists of a lot of questions coming from
interested minds. Our students' brains are like sponges at this age, soaking up
every bit of information they can. However some times there are underlying
factors that students struggle with on a daily basis that clouds their
learning.
##
## We have really amazing students at our school and I couldn't be more proud
of their thirst for learning!
##
## Our school is located in intercity urban San Antonio. Due to the low
socioeconomic status of our school's community, our students often come in with
more on their mind than learning fractions. Domestic violence, homelessness,
and hunger are many things that our students face when they go home.
##
## At our school, we take pride in our students and truly care about how each
student is feeling. We go out of our way to provide different services such as:
counselors, snacks, plenty of hugs, etc., to handle each situation so that our
students' can come right back and perform at their best.
<!--DONOTREMOVEESSAYDIVIDER-->I am hoping to re-incorporate rich literature
into my students daily school life. The students at my school are so plugged
in, consumed by their devices, they sometimes miss out on broadening their
imagination.
##
## We live in a world being consumed by technology.
##
## The books I am requesting coincide with a genre study, a study on a
fantastic children's author, and so much more. I want to fill my library with
```

Preparing texts

We use a regular expression to clean up the donation texts:

```r
# remove noise
df$project_essay <- str_replace_all(df$project_essay,
        pattern = '<!--DONOTREMOVEESSAYDIVIDER-->',
        replacement = '\n\n')
# validate
str_detect(df$project_essay[1],
           '<!--DONOTREMOVEESSAYDIVIDER-->')
```

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

Text analysis using quanteda

- a variety of R packages support quantitative text analyses. We will focus on quanteda, which is created and maintained by social scientists behind the Quanteda Initiative
- other packages that might be interesting for you: `koRpus`, `tidytext`, `tm`, `polmineR`

Quanteda corpus object

You can create a quanteda corpus from (1) a character vector or (2) a data frame, which automatically includes meta data as document variables:

```r
library(quanteda)
donor_corp <- corpus(df, text_field = 'project_essay',
                 docid_field = 'project_id')
docvars(donor_corp)$text <- df$project_essay # store unprocessed text
ndoc(donor_corp) # no. of documents
```

```
## [1] 10000
```

Keywords in context (KWIC)

Corpus objects can be used to discover keywords in context (KWIC):

```
kwic_donor <- kwic(donor_corp, pattern = c("ipad"),
                      window = 5) # context window
head(kwic_donor, 3)
```

```
##
## [54bea65a2cadc0f79f367fb1b76d6cfc, 15] is very important. The |
## [54bea65a2cadc0f79f367fb1b76d6cfc, 29] use advanced technology. An
|
## [54bea65a2cadc0f79f367fb1b76d6cfc, 188] computer, much less an |
##
## iPad | will also help my students
## iPad | would enhance my students'
## iPad | . The use of an
```

Tokenization

Tokens can be created from a corpus or character vector. The documentation (`?tokens()`) illustrates several options, e.g. for the removal of punctuation

```
donor_tokens <- tokens(donor_corp)
donor_tokens[[1]][1:20] # text 1, first 20 tokens
```

```
## [1] "A" "typical" "day" "in" "our"
## [6] "room" "consists" "of" "a" "lot"
## [11] "of" "questions" "coming" "from" "interested"
## [16] "minds" "."  "Our" "students" "'"
```

Basic form of tokens

- after tokenization text, some terms with similar semantic meaning might be regarded as different features (e.g. `love`, `loving`)
- one solution is the application of stemming, which tries to reduce words to their basic form:

```r
words <- c("love", "loving", "lovingly",
           "loved", "lover", "lovely")
char_wordstem(words, 'english')
```

```
## [1] "love"  "love"  "love"  "love"  "lover" "love"
```

To stem or not to stem?

- whether stemming generates useful features or not varies by use case
- in the context of topic modeling, a recent study suggests that stemmers produce no meaningful improvement (for English language)
- an alternative is lemmatization, available via packages likes spacyr and udpipe

Stopwords

Multiple preprocessing steps can be chained via the pipe operator, e.g normalizing to lowercase and removing common English stopwords:

```r
donor_tokens <- donor_tokens %>%
tokens_tolower() %>%
tokens_remove(stopwords('english'),
              padding = TRUE) # keep empty strings

donor_tokens[[1]][1:10]
```

```
## [1] "" "typical" "day" "" "" "room"
## [7] "consists" "" "" "lot"
```

Detecting collocations (phrases)

- collocations are sequences of tokens which symbolize shared semantic meaning, e.g. `United States`
- Quanteda can detect collocations with log-linear models. An important parameter is the minimum collocation frequency, which can be used to fine-tune results (see also `textstat_select()`)

```
colls <- textstat_collocations(donor_tokens,
         min_count = 200) # minimum frequency
donor_tokens_c <- tokens_compound(donor_tokens, colls) %>%
                  tokens_remove('') # remove empty strings
donor_tokens_c[[1]][1:5] # first five tokens of first text
```

```
## [1] "typical_day" "room" "consists" "lot" "questions"
```

Document-Feature Matrix (DFM)

- most models for automated text analysis require matrices as an input format
- a common variant which directrly translates to the bag of words format is the document term matrix (in quanteda: document-feature matrix):

doc_id	I	like	hate	currywurst
1	1	1	0	1
2	1	0	1	1

Creating a Document-Feature Matrix (dfm)

- problem: textual data is highly dimensional -> dfms's potentially grow to millions of rows & columns -> matrices for large text corpora don't fit in memory
- features are not uniformly distributed (see e.g. Zipf's law), most cells contain zeroes
- solution: sparse data format, which does not include zero counts. Quanteda natively implements DFM's as sparse matrices

DFM’s in quanteda

- Quanteda can create DFM’s from character vectors, corpora and token objects
- preprocessing steps unaffected by for word order can also be done during or after the creation of DFM’s (see documentation for `tokens()`)

```r
dfm_donor <- dfm(donor_tokens_c, remove_numbers = TRUE)
dim(dfm_donor)
```

```
## [1] 10000 27505
```

More preprocessing - feature trimming

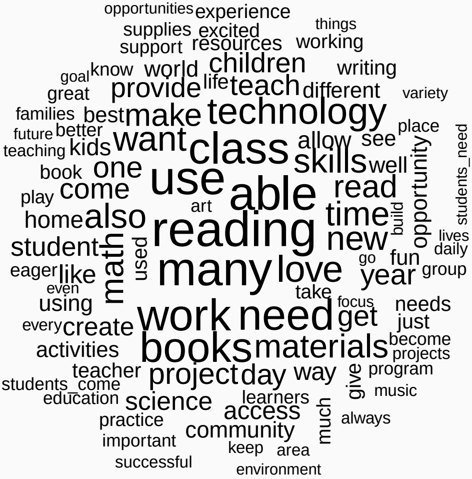
As an alternative (or complement) to manually defining stopwords, terms occuring in either very few or almost all documents can be removed automatically.

```r
dfm_donor <- dfm_donor %>%
  dfm_keep(min_nchar = 2) %>% # remove terms < 2 characters
  dfm_trim(min_docfreq = 0.001,  #2% min
           max_docfreq = 0.50,# 50% max
  docfreq_type = 'prop') # proportions instead of counts
dim(dfm_donor)
```

```
## [1] 10000  6817
```

Inspecting most frequent terms

```
textplot_wordcloud(dfm_donor, max_words = 100, color = 'black')
```



Inspecting most frequent terms

```r
textstat_frequency(dfm_donor) %>% head(10)
```

```
##       feature frequency rank docfreq group
## 1     reading      8279    1    3453   all
## 2        many      7937    2    4918   all
## 3        able      7933    3    4703   all
## 4         use      7805    4    4656   all
## 5       class      7303    5    4337   all
## 6        work      7267    6    4282   all
## 7        need      7179    7    4488   all
## 8       books      6916    8    2379   all
## 9  technology      5983    9    2578   all
## 10       love      5923   10    3774   all
```

Other pre-preprocessing procedures - ngrams

Features can be created from `n` sequences of tokens.

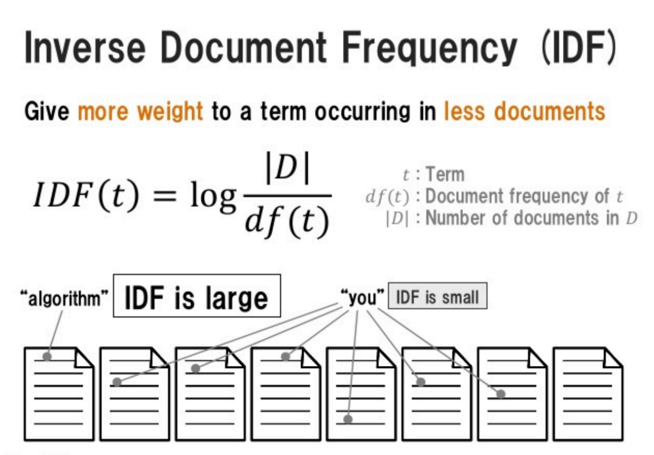
```r
text <- "to be or not to be"
tokens(text, ngrams = 1:2) # unigrams + bigrams
```

```
## tokens from 1 document.
## text1 :
## [1] "to" "be" "or" "not" "to" "be" "to_be"
## [8] "be_or" "or_not" "not_to" "to_be"
```

```r
tokens(text, ngrams = 3) # trigrams only
```

```
## tokens from 1 document.
## text1 :
## [1] "to_be_or"  "be_or_not" "or_not_to" "not_to_be"
```

Other prepreprocessing steps - tfidf



Inverse Document Frequency（IDF)

Give more weight to a term occurring in less documents

$$IDF(t) = \log \frac{|D|}{df(t)}$$

t : Term
$df(t)$: Document frequency of t
$|D|$: Number of documents in D

see `dfm_tfidf()` for the quanteda implementation

Other pre-processing steps we won't cover

- part-of-speech tagging
- named-entity recognition
- both of these approaches are available via the R packages `spacyr` (Python wrapper) and `udpipe`

Questions?