TEXT DATA AND TEXT MINING 1

ADVANCED CRIME ANALYSIS UCL

BENNETT KLEINBERG

28 JAN 2019

Text data 1

BRIEFLY ABOUT THE MODULE

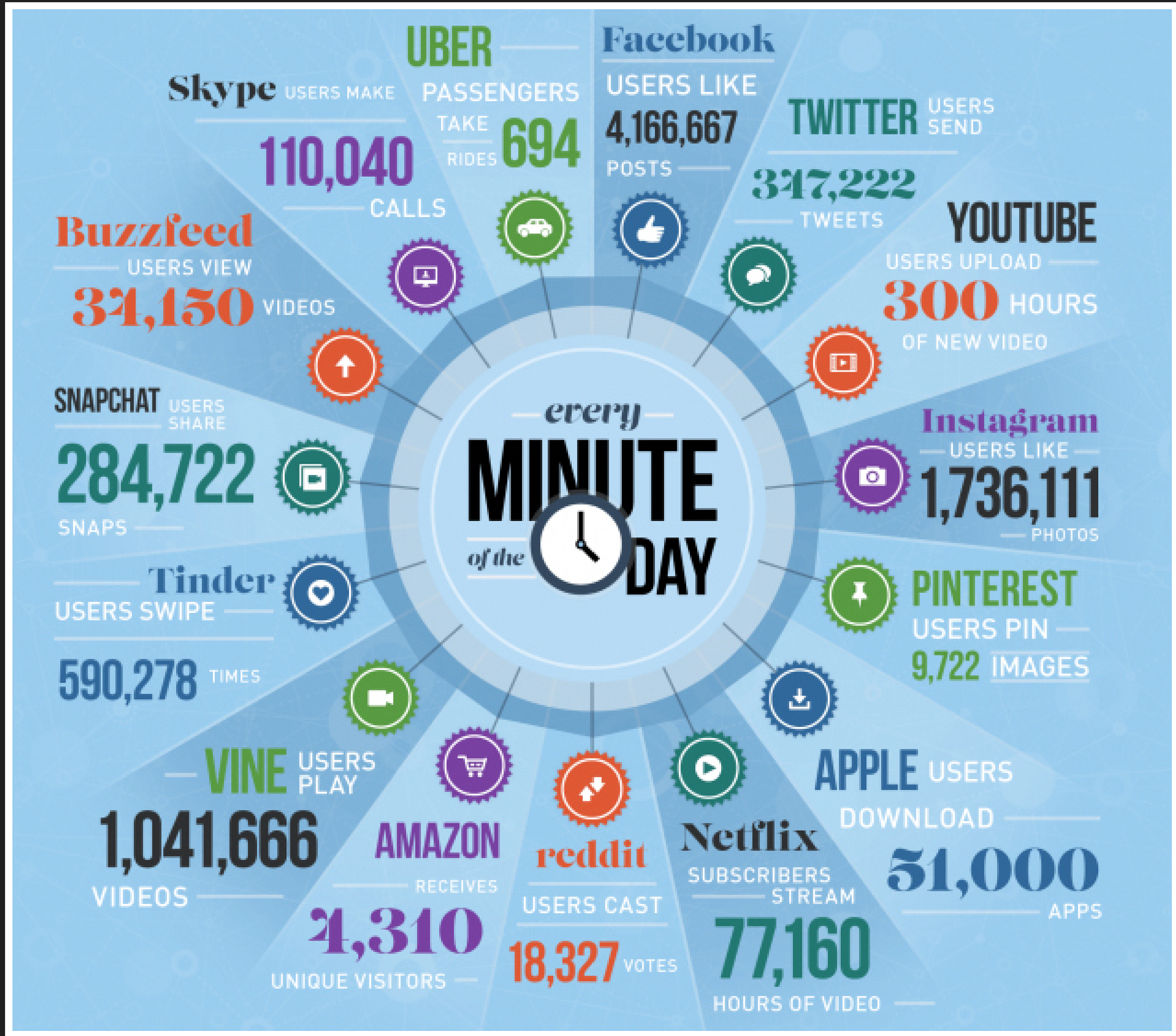
- 0.5 UCL credits = 7.5 ECTS
- 150 learning hours
- 11 weeks with 14 hours/week
- 3 contact hours per week
- **leaves 11 hours of self-study per week**

EXPECTED SELF-STUDY

- Revise the lecture (your responsibility)
- Replicate the code/examples
- Read the required literature (read, annotate, summarise)
- Read additional literature if necessary
- Design own code examples to understand the concept
- Find tutorials/guides online
- If still unclear: attend the code clinics: **Weds 10-11 am**
- or: post it on Moodle or ask us

TODAY

- Why text data?
- Applications to crime and security problems
- Levels of text data
- Quantifying text data
- Considerations in text cleaning


every MINUTE of the DAY
UBER PASSENGERS TAKE 694 RIDES
Facebook USERS LIKE 4,166,667 POSTS
Skype USERS MAKE 110,040 CALLS
TWITTER USERS SEND 347,222 TWEETS
YOUTUBE USERS UPLOAD 300 HOURS OF NEW VIDEO
Buzzfeed USERS VIEW 34,150 VIDEOS
SNAPCHAT USERS SHARE 284,722 SNAPS
Instagram USERS LIKE 1,736,111 PHOTOS
Tinder USERS SWIPE 590,278 TIMES
PINTEREST USERS PIN 9,722 IMAGES
VINE USERS PLAY 1,041,666 VIDEOS
AMAZON RECEIVES 4,310 UNIQUE VISITORS
reddit USERS CAST 18,327 VOTES
Netflix SUBSCRIBERS STREAM 77,160 HOURS OF VIDEO
APPLE USERS DOWNLOAD 51,000 APPS

TEXT IS EVERYWHERE …

- Practically all websites
- Emails
- Messaging
- Government reports
- Laws
- Police reports
- Uni coursework
- Newspapers

… AND EVERYTHING IS TEXT

- videos –> transcripts
- music –> lyrics
- conversations –> transcripts
- speeches –> transcripts

CORE IDEA

Text is a unique documentation of human activity.

We are obsessed with documenting.

TEXT & CRIME SCIENCE

TEXT & CRIME SCIENCE

- hate speech
- police reports
- crimestoppers
- fake reviews
- fear of crime
- cryptofraud

OBTAINING TEXT DATA

QUANTIFYING TEXT DATA

CHALLENGE OF QUANTIFICATION

- a text is not a numerical representation
- compare this to trading data
- a text is just that, “a text”
- but: for quantitative analyses, we need numbers

Text –> numerical representation?

EXAMPLE

All I ever wanted was to love women, and in turn to be loved by them back. Their behavior towards me has only earned my hatred, and rightfully so! I am the true victim in all of this. I am the good guy. Humanity struck at me first by condemning me to experience so much suffering. I didn’t ask for this. I didn’t want this. I didn’t start this war. I wasn’t the one who struck first. But I will finish it by striking back. I will punish everyone. And it will be beautiful. Finally, at long last, I can show the world my true worth.

HOW WOULD YOU QUANTIFY THE EXAMPLE?

FEATURES OF TEXT DATA

- meta dimension
  - no. of words
  - no. of sentences
- syntactic dimension
  - word frequencies
  - verbs, nouns, persons, locations, ..
  - structure of a sentence
- semantic dimension
  - sentiment
  - psycholinguistic features
- text metrics
  - readability
  - lexical diversity

APPROACHES TO TEXT DATA

1. Modelling text data
2. Comparing text data
3. Text data for predictive models

THE `QUANTEDA` PACKAGE

```r
library(quanteda)
```

```
## Package version: 1.2.0
```

```
## Parallel computing: 2 of 4 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

```
##
## Attaching package: 'quanteda'
```

```
## The following object is masked from 'package:utils':
##
##     View
```

- quanteda: Quantitative Analysis of Textual Data
  - documentation
  - tutorials
  - examples

LEVELS OF TEXT DATA

- characters `c('h', 'a', 't', 'r', 'e', 'd')`
- words `hatred`
- sentences `I didn’t ask for this.`
- documents: individual text files
- corpora: collection of documents

COUNTING META FEATURES IN R

text level	R function
characters	`nchar()`
words	`quanteda::ntoken()`
sentences	`quanteda::nsentence()`

Homework: read about the type/token distinction here and here.

R EXAMPLES

```
#sentences
no_of_sentences = nsentence(er)
no_of_sentences
```

```
## text1
##    13
```

```
#words 1
no_of_words_1 = ntoken(er)
no_of_words_1
```

```
## text1
##   123
```

```
#words 2
no_of_words_2 = ntype(er)
no_of_words_2
```

```
## text1
##    72
```

TYPE-TOKE RATIO

Note: often used metric for “lexical diversity” is the TTR (type-token ratio).

```
string_a = "I didn’t ask for this. I didn’t want this."
string_b = "But I will finish it by striking back."
```

What are the type-token ratios of each string?

TYPE-TOKEN RATIO

```r
ntype(string_a)/ntoken(string_a)
```

```
##     text1
## 0.6363636
```

```r
ntype(string_b)/ntoken(string_b)
```

```
## text1
##     1
```

NUANCED META FEATURES

- Characters per word

```
nchar(er)/ntoken(er)
```

```
##     text1
## 4.317073
```

- Words per sentence

```
ntoken(er)/nsentence(er)
```

```
##     text1
## 9.461538
```

TEXT REPRESENTATIONS

TEXT REPRESENTATIONS

- represent a text by its tokens (terms)
- each text consists of a frequency of its tokens

```
"I think I believe him"
```

- create a column for each token
- count the frequency

text_id	I	think	believe	her
text1	2	1	1	1

TERM FREQUENCY

- frequency of tokens in each document
- represented in a table (matrix)
- tokens are features of a document
- voilá: fancy name –> **Document Feature Matrix (= DFM)**

```
example_string_tok = tokens("I think I believe him")
```

DFM

- from ‘tokens’ object, create a DFM table

```
dfm(example_string_tok)
```

```
## Document-feature matrix of: 1 document, 4 features (0% sparse).
## 1 x 4 sparse Matrix of class "dfm"
##          features
## docs     i think believe him
##   text1  2     1       1   1
```

- Sparsity: % of zero-cells
  - why is sparsity = 0% here?
  - what would you expect if we take additional documents

DFM WITH MULTIPLE DOCUMENTS

Document-term frequency matrix

```
multiple_docs_tok = tokens(c("I think I believe him", "This is a cool
```

```
dfm(multiple_docs_tok)
```

```
## Document-feature matrix of: 2 documents, 9 features (50% sparse).
## 2 x 9 sparse Matrix of class "dfm"
##        features
## docs    i think believe him this is a cool function
##   text1 2     1       1   1    0  0 0    0        0
##   text2 0     0       0   0    1  1 1    1        1
```

DFM WITH TWO LONE-ACTORS

“All I ever wanted was to love women, and in turn to be loved by them back. Their behavior towards me has only earned my hatred, and rightfully so! I am the true victim in all of this. I am the good guy. Humanity struck at me first by condemning me to experience so much suffering. I didn’t ask for this. I didn’t want this. I didn’t start this war. I wasn’t the one who struck first. But I will finish it by striking back. I will punish everyone. And it will be beautiful. Finally, at long last, I can show the world my true worth.”

DFM WITH TWO TEXTS

The Industrial Revolution and its consequences have been a disaster for the human race. They have greatly increased the life-expectancy of those of us who live in “advanced” countries, but they have destabilized society, have made life unfulfilling, have subjected human beings to indignities, have led to widespread psychological suffering (in the Third World to physical suffering as well) and have inflicted severe damage on the natural world. The continued development of technology will worsen the situation.

DFM REPRESENTATION

- Create a “mini corpus” for convenience
- makes using the quanteda pipeline easier

```
mini_corpus = corpus(c(er, ub))
summary(mini_corpus)
```

```
## Corpus consisting of 2 documents:
##
##   Text Types Tokens Sentences
##  text1    72    123        13
##  text2    63     88         3
##
## Source: /Users/bennettkleinberg/GitHub/ucl_aca_20182019/slides/* c
## Created: Sun Feb  3 18:37:47 2019
## Notes:
```

DFM REPRESENTATION

```
corpus_tokenised = tokens(mini_corpus)
corpus_dfm = dfm(corpus_tokenised)
```

```
knitr::kable(corpus_dfm[, 1:8])
```

document	all	i	ever	wanted	was	to	love	women
text1	2	10	1	1	1	3	1	1
text2	0	0	0	0	0	3	0	0

…

```
knitr::kable(corpus_dfm[, 31:38])
```

document	am	the	true	victim	of	this	good	guy
text1	2	4	2	1	1	4	1	1
text2	0	7	0	0	3	0	0	0

Is this ideal?

WHAT ARE THE MOST FREQUENT "TERMS"?

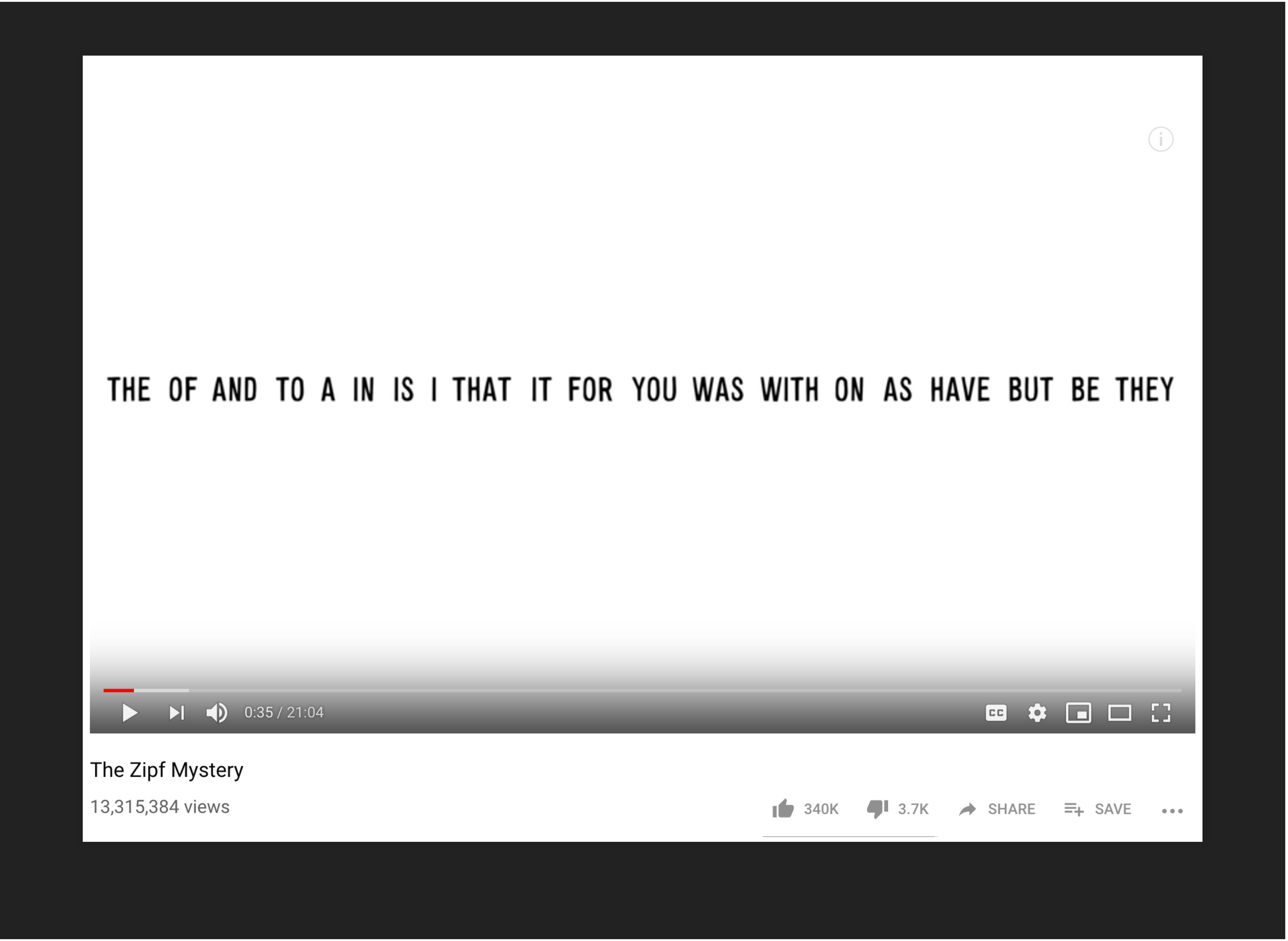
```
topfeatures(corpus_dfm[1])
```

```
##      .      i      ,    the   this     to    and     by     me dic
##     12     10      4      4      4      3      3      3      3
```

```
topfeatures(corpus_dfm[2])
```

```
##       the      have         ,        to         .        of
##         7         7         4         3         3         3
##        in suffering     world
##         2         2         2
```

Highly recommended: Vsauce on Zipf's Law


THE OF AND TO A IN IS I THAT IT FOR YOU WAS WITH ON AS HAVE BUT BE THEY
0:35 / 21:04
The Zipf Mystery
13,315,384 views
340K
3.7K
SHARE
SAVE

WORD HIERARCHIES

- some words at more meaning than others
- stopwords = meaningless (?)
- in any case: too frequent words, don’t tell much about the documents
- ideally: we want to get an “importance score” for each word

BUT HOW TO GET THE IMPORTANT WORDS?

WORD IMPORTANCE

document	and	in	turn	be	loved	by
text1	3	2	1	2	1	3
text2	2	2	0	0	0	0

Ideally, we want to “reward” words that are:

- important locally
- but not ‘inflated’ globally

METRIC FOR WORD IMPORTANCE

- Term frequency: occurence/overall words in document

document	and	in	turn	be	loved	by
text1	0.024	0.016	0.008	0.016	0.008	0.024
text2	0.023	0.023	0.000	0.000	0.000	0.000

```
3/ntoken(mini_corpus[1])
```

```
##      text1
## 0.02439024
```

Term frequency: reward for words that occur often in a document.

METRIC FOR WORD IMPORTANCE

Problem: some words just occur a lot anyway (e.g. “stopwords”).

Correct for global occurrence:

	x
and	2
in	2
turn	1
be	1
loved	1
by	1

Document frequency: number of documents with each token

token.

COMBINING TERM FREQUENCY AND DOCUMENT FREQUENCY

- take the local importance

document	and	in	turn
text1	0.024	0.016	0.008
text2	0.023	0.023	0.000

- correct for global occurrences

x	
and	2
in	2
turn	1

TF/DF

```
#text1: "and"
0.024/2

#text2: "and"
0.022/2
```

```
#text1: "turn"
0.008/1

#text2: "turn"
0.000/1
```

TF-IDF

- Term frequency
- INVERSE document frequency

$TFIDF = TF/DF$ = $TFIDF = TF * IDF$, since

$$IDF = 1/DF$$

IDF often modelled as $IDF = log(\frac{N}{DF})$

TF-IDF

Note: for the exact formula for the inverse DF refer to the quanteda docs.

```r
knitr::kable(round(dfm_tfidf(corpus_dfm, scheme_tf = 'prop', scheme_c
```

document	and	in	turn	be	loved	by
text1	0	0	0.002	0.005	0.002	0.007
text2	0	0	0.000	0.000	0.000	0.000

TF-IDF

- TF: rewards local importance
- IDF: punishes for global occurrence
- TFIDF value as metric for the importance of words per document

THERE'S MORE TO WORDS

- you can count them [DONE]
- but they also have a function
  - each word has a grammatical function
  - nouns, verbs, pronouns
- called: parts-of-speech

SYNTACTIC DIMENSION

```
library(qdap)
```

x
All
I
ever
wanted
was
to
love
women

PART-OF-SPEECH TAGGING

POS depend on POS framework.

Commonly used: Penn Treebank Project

x	**POS**
All	determiner
I	noun
ever	adverb
wanted	verb
was	verb
to	?
love	verb
women	noun

POS TYPES

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass

POS TAGGING WITH QDAP

```
er_ = "All I ever wanted was to love women"
pos_tagged = pos(er_)
```

```
##
   |
   |                                                                  |
   |
   |==================================================================|
```

```
pos_tagged$POStagged$POStagged
```

```
## [1] all/DT i/FW ever/RB wanted/VBD was/VBD to/TO love/VB women/NNS
## Levels: all/DT i/FW ever/RB wanted/VBD was/VBD to/TO love/VB women
```

POS TAGGING

```
pos(er, percent = F, progress.bar = F)$POSfreq
```

```
##   wrd.cnt CC DT FW IN JJ MD NN NNS PRP PRP$ RB TO VB VBD VBG VBN V
## 1     106  4 10  1 14  8  4 17   1   7    3 10  3  9   6   2   2
##   WP
## 1  1
```

```
pos(ub, percent = F, progress.bar = F)$POSfreq
```

```
##   wrd.cnt CC DT IN JJ MD NN NNS PRP PRP$ RB TO VB VBN VBP WP
## 1      77  3  9  8 11  1 15   4   3    1  2  3  2   7   7  1
```

CONSIDERATIONS IN TEXT CLEANING

RESEARCHER’S DEGREES OF FREEDOM

- stopword removal
- stemming

STOPWORD REMOVAL

- We know many words are “low in meaning”
- So-called stopwords

x
i
me
my
myself
we
hers
herself
it
its
itself
they

You could decide to remove these…

STOPWORD REMOVAL

With stopwords:

document	all	i	ever	wanted	was	to	love	women
text1	2	10	1	1	1	3	1	1
text2	0	0	0	0	0	3	0	0

Without stopwords

document	ever	wanted	love	women	,	turn	loved	back
text1	1	1	1	1	4	1	1	2
text2	0	0	0	0	4	0	0	0

STEMMING

- some words originate from the same “stem”
- e.g. “love”, “loved”, “loving”, “lovely”
- but you might want to reduce all these to the stem

WORD STEMS

```r
love_stem = c("love", "loved", "loving", "lovely")
```

document	love	loved	loving	lovely
text1	1	0	0	0
text2	0	1	0	0
text3	0	0	1	0
text4	0	0	0	1

… AFTER STEMMING

```
knitr::kable(dfm(love_stem_tok, stem = T))
```

document	love
text1	1
text2	1
text3	1
text4	1

OUR MINI CORPUS

From: (incl. stopwords and without stemming)

document	all	i	ever	wanted	was	to	love	women
text1	2	10	1	1	1	3	1	1
text2	0	0	0	0	0	3	0	0

… to (without stopwords and stemmed)

document	ever	want	love	women	,	turn	back	.
text1	1	2	2	1	4	1	2	12
text2	0	0	0	0	4	0	0	3

LIMITATIONS OF TEXT DATA

- a lot of assumptions
- text == behaviour?
- produced text == displayed text?
- linguistic “profiles”
- many decisions in your hand
  - stemming
  - stopwords
  - custom dictionary

RECAP

- levels of text data
- meta features
- syntactic features
- word frequencies
- TFIDF
- parts-pf-speech

OUTLOOK

No tutorial.

Homework: Text data 1 (to come)

Next week: Text data 2

END