TEXT DATA AND TEXT MINING 2

ADVANCED CRIME ANALYSIS

UCL

BENNETT KLEINBERG

4 FEB 2019

Text data 2

TODAY

- Recap TF-IDF
- n-grams
- psycholinguistics
- sentiment analysis

RECAP: TFIDF

Why do we need the TF-IDF?

And what is it?

TF-IDF EXAMPLE

Fakenews corpus: 1000 fake, 1000 real (data)

Step 1: Term-frequencies

```
## 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
```

```
corpus_dfm = dfm(corpus_tokenised
                 , stem = T
                 , remove = stopwords())
dfm_trimmed = dfm_trim(corpus_dfm, sparsity = 0.95)
dfm_trimmed_tf = round(dfm_weight(dfm_trimmed, scheme='prop'), 4)
dfm_trimmed_tf
```

```
## Document-feature matrix of: 2,000 documents, 916 features (87.8% s
```

TF (PROPORTIONS)

document	secretari	ask	govern	document	includ	one	contain
text1	0.0238	0.0238	0.0238	0.0476	0.0238	0.0238	0.0238
text2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
text3	0.0065	0.0000	0.0000	0.0000	0.0000	0.0065	0.0000
text4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
text5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

STEP 2: DF

	x
secretari	174
ask	432
govern	582
document	124
includ	668
one	1204
contain	128
inform	296

STEP 2: DF

Inverse DF with log transform

	x
secretari	1.0604807
ask	0.6655462
govern	0.5361070
document	1.2076083
includ	0.4762535
one	0.2204035
contain	1.1938200
inform	0.8297383

STEP 3: TF-IDF

```
dfm_tfidf(dfm_trimmed
          , scheme_tf = 'prop'
          , scheme_df = 'inverse') ##!!! <-- correction to L4
```

document	secretari	ask	govern	document	includ	one	contain
text1	0.0252	0.0158	0.0128	0.0575	0.0113	0.0052	0.0284
text2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
text3	0.0068	0.0000	0.0000	0.0000	0.0000	0.0014	0.0000
text4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
text5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

TF-IDF FORMAL

- Term frequency
- INVERSE document frequency

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

$$IDF = 1/DF$$

Note: types of DF

TF-IDF AIMS

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

EXTENSION: *N*-GRAMS

PROBLEM?

```
## [1] "It is a great time to be alive"
```

SO FAR…

- we used tokens as unit of analysis
- but: sometimes mutlple tokens might reveal more
- n-grams –> sequences of *n* tokens
  - unigrams: n = 1
  - bigrams: n = 2
  - trigrams: n = 3

UNIGRAMS

```
## [1] "It is a great time to be alive"
```

UNIGRAMS

```
## Document-feature matrix of: 1 document, 8 features (0% sparse).
## 1 x 8 sparse Matrix of class "dfm"
##        features
## docs    it is a great time to be alive
##   text1  1  1 1     1    1  1  1     1
```

BEYOND UNIGRAMS: BIGRAMS

Bigrams = all sequenceis of 2 tokens

```
## [1] "It is a great time to be alive"
```

BIGRAMS

```
## Document-feature matrix of: 1 document, 7 features (0% sparse).
## 1 x 7 sparse Matrix of class "dfm"
##        features
## docs    it_is is_a a_great great_time time_to to_be be_alive
##   text1     1    1       1          1       1     1        1
```

EVEN MORE ...

Trigrams

```
## Document-feature matrix of: 1 document, 6 features (0% sparse).
## 1 x 6 sparse Matrix of class "dfm"
##        features
## docs    it_is_a is_a_great a_great_time great_time_to time_to_be
##   text1       1          1            1             1          1
##        features
## docs    to_be_alive
##   text1           1
```

N-GRAMS IN GENERAL

What happens when we increase *n* in a corpus?

N-GRAMS WITH QUANTEDA

```
unigrams = dfm(x = fakenews_corpus
               , ngrams = 1
               )
```

document	as	secretary	of	routinely	asked	her	maid	to
text1	2	1	2	1	1	5	1	9
text2	0	0	0	0	0	0	0	3
text3	1	1	13	0	0	1	0	5
text4	4	0	10	0	0	0	0	6
text5	4	0	17	0	0	0	0	22

DFM WITH ADDITIONAL CONTROLS

```
unigrams_cleaned = dfm(x = fakenews_corpus
                     , ngrams = 1
                     , stem = T
                     , remove = stopwords()
                     )
```

document	secretari	routin	ask	maid	print	sensit
text1	1	1	1	1	2	2
text2	0	0	0	0	0	0
text3	1	0	0	0	0	0
text4	0	0	0	0	0	0
text5	0	0	0	0	0	0

BIGRAMS

```
bigrams_cleaned = dfm(x = fakenews_corpus
                 , ngrams = 2 ## <---!!!
                 , stem = T
                 )
```

document	as_secretari	secretari_of	of_routin	routin_ask
text1	1	1	1	1
text2	0	0	0	0
text3	0	0	0	0
text4	0	0	0	0
text5	0	0	0	0

What happens when we increase *n* in a corpus?

```
dim(unigrams_cleaned)
```

```
## [1]  2000 15630
```

```
dim(bigrams_cleaned)
```

```
## [1]   2000 357458
```

WEIGHTING N-GRAMS

```
dfm_tfidf(bigrams_cleaned
          , scheme_tf = 'prop'
          , scheme_df = 'inverse')
```

document	as_secretari	secretari_of	of_routin	routin_ask
text1	0.0135	0.0093	0.0224	0.0246
text2	0.0000	0.0000	0.0000	0.0000
text3	0.0000	0.0000	0.0000	0.0000
text4	0.0000	0.0000	0.0000	0.0000
text5	0.0000	0.0000	0.0000	0.0000

N-GRAMS

- generalisation of “single” tokens
- often used in bag-of-word models
- common in predictive modelling

SENTIMENT ANALYSIS

SENTIMENT ANALYSIS: AIM

- measure positive/negative tone
- “emotionality” of a text
- builds on the “language -> behavior” and “cognition -> language” nexus

BASICS OF SENTIMENT ANALYSIS

1. tokenise text
2. construct a lexicon of sentiment words
3. judge the sentiment words
4. match tokens with sentiment lexicon

1. TOKENISE TEXT

From:

```
## [1] "Your hyperbole makes you sound more like a Trump supporter th
```

… to

```
## tokens from 1 document.
## text1 :
##  [1] "Your"       "hyperbole" "makes"      "you"        "sound"
##  [6] "more"       "like"      "a"          "Trump"      "supporter"
## [11] "than"       "a"         "Stein"      "but"        "both"
## [16] "groups"     "of"        "losers"     "will"       "continue"
## [21] "to"         "whine"     "in"         "unity"
```

2. LEXICON OF SENTIMENT WORDS

- do all words have a potential sentiment?
- maybe focus on adjectives/adverbs, maybe verbs?

Luckily: many sentiment lexicons exists

2. LEXICON OF SENTIMENT WORDS

The `lexicon` package

```
lexicon::hash_sentiment_nrc[, 1]
```

```
##                 x
##    1:     abandon
##    2:   abandoned
##    3: abandonment
##    4:        abba
##    5:   abduction
##   ---
## 5464:       youth
## 5465:        zeal
## 5466:     zealous
## 5467:        zest
## 5468:         zip
```

2. LEXICON OF SENTIMENT WORDS

```
lexicon::hash_sentiment_slangsd[, 1]
```

```
##                                x
##      1:                      a a
##      2:    a bad amount of money
##      3:             a bad mother
##      4: a bad taste in my mouth
##      5:         a bag o' beagles
##    ---
## 48273:                   smegma
## 48274:          smegma popsicle
## 48275:              smegma slap
## 48276:            smegma stache
## 48277:              smegma team
```

2. LEXICON OF SENTIMENT WORDS

```
lexicon::hash_sentiment_socal_google[, 1]
```

```
##                    x
##    1:        a pillar
##    2:         ab liva
##    3:            able
##    4:   above average
##    5: above mentioned
##   ---
## 3286:   you know what
## 3287:           young
## 3288:         younger
## 3289:        youthful
## 3290:      zero entry
```

3. JUDGE THE SENTIMENT WORDS

Normal strategy

- crowdsourced annotation
- decide on judgment scale
- multiple judgments per word
- assess inter-rater reliability

3. JUDGE THE SENTIMENT WORDS

Again: mostly already done for you

```
lexicon::hash_sentiment_nrc
```

```
##                  x  y
##    1:      abandon -1
##    2:    abandoned -1
##    3:  abandonment -1
##    4:         abba  1
##    5:    abduction -1
##   ---
## 5464:        youth  1
## 5465:         zeal  1
## 5466:      zealous  1
## 5467:         zest  1
## 5468:          zip -1
```

Binary judgment: -1 or 1

3. JUDGE THE SENTIMENT WORDS

```
lexicon::hash_sentiment_slangsd
```

```
##                                x    y
##      1:                      a a -0.5
##      2:   a bad amount of money -0.5
##      3:             a bad mother -0.5
##      4: a bad taste in my mouth -0.5
##      5:        a bag o' beagles -0.5
##    ---
## 48273:                   smegma -0.5
## 48274:          smegma popsicle -0.5
## 48275:              smegma slap -0.5
## 48276:            smegma stache -0.5
## 48277:              smegma team -0.5
```

Finer judgment: -1.00, -0.50, 0.50, 1.00

3. JUDGE THE SENTIMENT WORDS

```
lexicon::hash_sentiment_socal_google
```

```
##                        x          y
##    1:           a pillar  2.9045431
##    2:            ab liva -0.9578700
##    3:               able  2.6393740
##    4:      above average  3.2150018
##    5:    above mentioned  2.5815803
##   ---
## 3286:      you know what -0.3177500
## 3287:              young -0.5298408
## 3288:            younger -0.9971062
## 3289:           youthful -0.1287262
## 3290:         zero entry  1.8376710
```

Continuous scale: -30 to +30

4. MATCH TOKENS WITH SENTIMENT LEXICON

- Classic approach: one sentiment score (`syuzhet` package)

> *[…] was devastated when she found out that she had colon cancer. She entered the third phase of her cancer. It was especially terrifying because her husband passed away from lung cancer after receiving chemotherapy. […]*

4. MATCH TOKENS WITH SENTIMENT LEXICON

Classic approach: one sentiment score

```
syuzhet::get_sentiment(example_text)
```

```
## [1] -1.55
```

4. MATCH TOKENS WITH SENTIMENT LEXICON

- Newer approach: sentiment for each sentence

The `sentimentr` (Rinker) package

```
sentimentr::sentiment(example_text)
```

```
##    element_id sentence_id word_count  sentiment
## 1:          1           1         16 -0.1625000
## 2:          1           2          8 -0.2651650
## 3:          1           3         15 -0.5034878
## 4:          1           4         12 -0.1443376
## 5:          1           5         16 -0.4650000
```

4. MATCH TOKENS WITH SENTIMENT LEXICON

- Newer approach: sentiment for each sentence
  - needs punctuated data
  - and good sentence disambiguation
  - without punctuation: whole string = 1 sentence
- What about valence shifters?

This is not ideal.

A DIFFERENT APPROACH

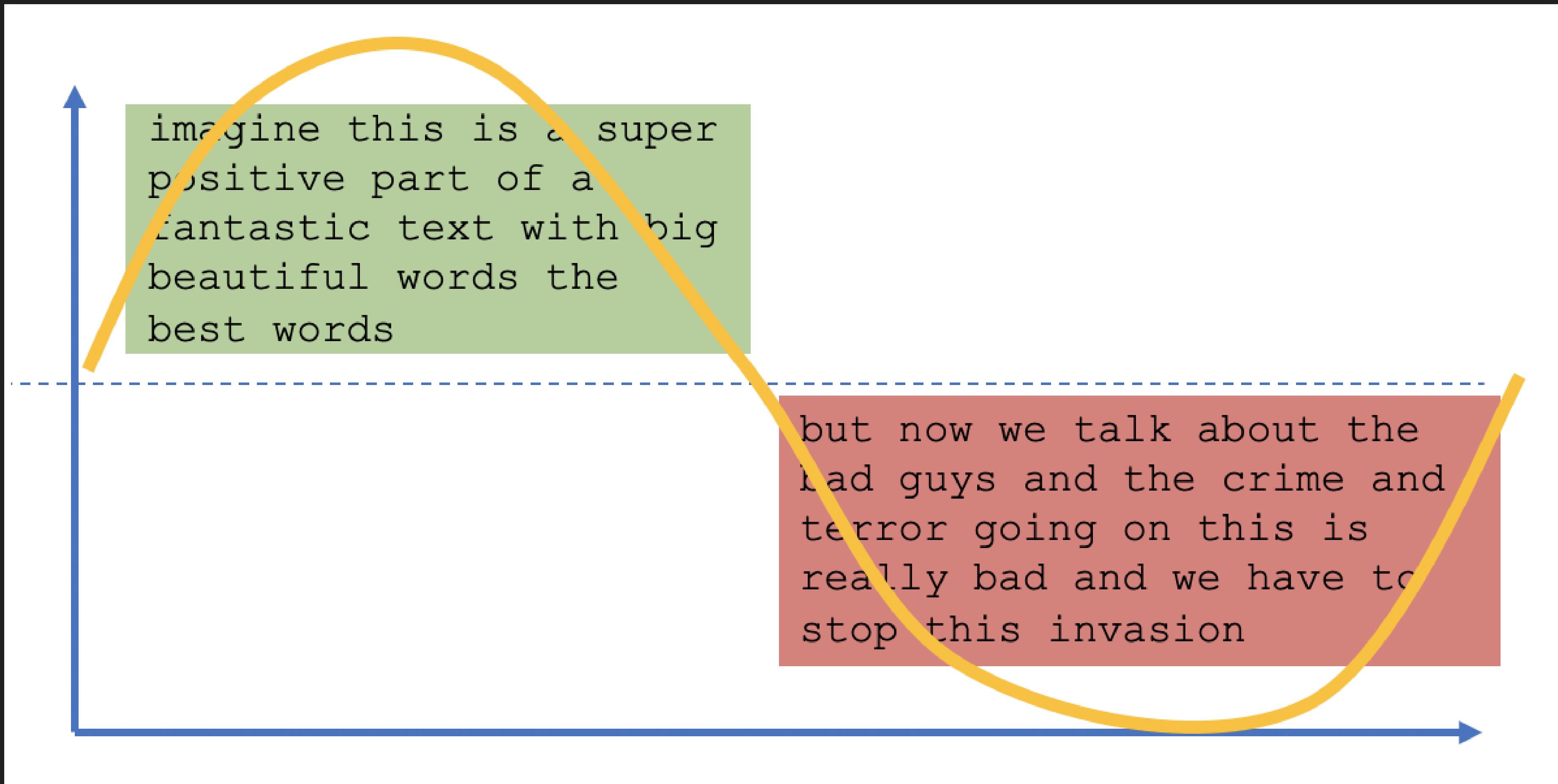
Dynamic sentiment analysis

- Inspired by: Matthew Jockers' work

Assumption:

- sentiment is dynamic within texts
- static approaches mask sentiment dynamics
- worst case: sentiment completely off

imagine this is a super positive part of a fantastic text with big beautiful words the best words	+
but now we talk about the bad guys and the crime and terror going on this is really bad and we have to stop this invasion	-


imagine this is a super
positive part of a
fantastic text with big
beautiful words the
best words
but now we talk about the
bad guys and the crime and
terror going on this is
really bad and we have to
stop this invasion

SENTIMENT TRAJECTORIES

From our EMNLP work

1. Parse text input into words
2. Match sentiment lexicon to each word
   - Match valence shifters to each context
     - Apply valence shifter weights
     - Build a naïve context around the sentiment
     - Return modified sentiment
3. Length-standardise sentiment vector

SENTIMENT TRAJECTORIES

Parse input

```r
source('./r_deps/naive_context_sentiment/ncs.R')
```

text	index
it	25
was	26
especially	27
terrifying	28
because	29
her	30
husband	31
passed	32
away	33
from	34

text	index
lung	35

SENTIMENT TRAJECTORIES

Match sentiment

text	index	y
it	25	NA
was	26	NA
especially	27	NA
terrifying	28	-1
because	29	NA
her	30	NA
husband	31	NA
passed	32	NA
away	33	NA
from	34	NA
lung	35	NA

SENTIMENT TRAJECTORIES

Match valence shifters

text	index	y.x	y.y
it	25	NA	NA
was	26	NA	NA
especially	27	NA	2
terrifying	28	-1	NA
because	29	NA	NA
her	30	NA	NA
husband	31	NA	NA
passed	32	NA	NA
away	33	NA	NA
from	34	NA	NA
lung	35	NA	NA

SENTIMENT TRAJECTORIES

Valence shifters

- 1 = negator (not, never, …):-1.00
- 2 = amplifier (very, totally, …): 1.50
- 3 = deamplifier (hardly, barely, …): 0.50
- 4 = adversative conjunction (but, however, …): 0.25

SENTIMENT TRAJECTORIES

Apply valence shifter weights

text	index	sentiment	valence	weights
it	25	NA	NA	1.0
was	26	NA	NA	1.0
especially	27	NA	2	1.5
terrifying	28	-1	NA	1.0
because	29	NA	NA	1.0
her	30	NA	NA	1.0
husband	31	NA	NA	1.0
passed	32	NA	NA	1.0
away	33	NA	NA	1.0
from	34	NA	NA	1.0
lung	35	NA	NA	1.0

SENTIMENT TRAJECTORIES

Build ‘naive’ context around sentiment

- 2 words around sentiment word

text	index	sentiment	valence	weights
was	26	NA	NA	1.0
especially	27	NA	2	1.5
terrifying	28	-1	NA	1.0
because	29	NA	NA	1.0
her	30	NA	NA	1.0

SENTIMENT TRAJECTORIES

Calculate modified sentiment

```
## [1] "sentiment change for \"devastated\": -0.5 --> -0.5"
## [1] "sentiment change for \"found\": 0.6 --> 0.6"
## [1] "sentiment change for \"cancer\": -0.75 --> -0.75"
## [1] "sentiment change for \"cancer\": -0.75 --> -0.75"
## [1] "sentiment change for \"terrifying\": -1 --> -1.5"
## [1] "sentiment change for \"cancer\": -0.75 --> -0.75"
## [1] "sentiment change for \"receiving\": 0.6 --> 0.6"
## [1] "sentiment change for \"exposed\": -0.5 --> -0.125"
## [1] "sentiment change for \"cancer\": -0.75 --> -0.75"
```

text	sentiment	valence	weights	sentiment_score_mod
was	NA	NA	1.0	0.0
especially	NA	2	1.5	0.0
terrifying	-1	NA	1.0	-1.5
because	NA	NA	1.0	0.0
her	NA	NA	1.0	0.0

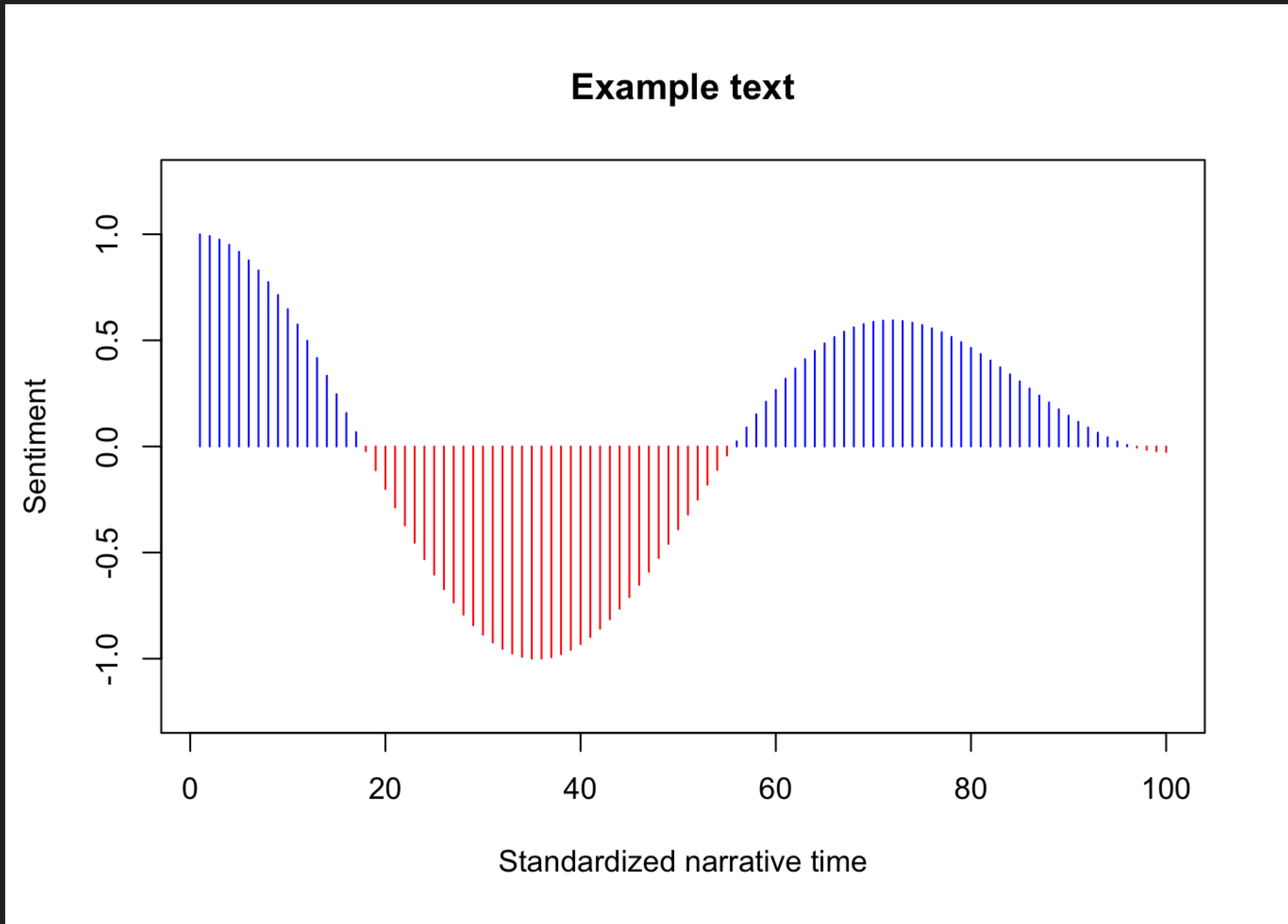
SENTIMENT TRAJECTORIES

Length-standardisation

- aim: transform all sentiments (modified + valence-shifter weighted) to a vector
- standard vector length for comparisons
- here: 100 values with Discrete Cosine Transformation

SENTIMENT TRAJECTORIES

Length-standardisation



PSYCHOLINGUISTICS

PSYCHOLINGUISTICS

Die-hard assumption: cognition –> language

- assuming that cognition –> language
- we might be interested in knowing about:
  - complex thinking in text
  - tentative language vs certainty
  - focus on past/present/future
  - …

THE LIWC (READ AS “LUKE”)

- developed at UT Austin
- Several papers
- Built with expert focus groups
- Popular in CL community
- dictionary-based approach
- 92 categories

THE LIWC PIPELINE

- read individual files into the LIWC software
- select categories
- retrieve % of words in category

LIWC DEMO

LIWC OUTPUT: META INDICATORS

WC	Analytic	Clout	Authentic	Tone	WPS	Sixltr	Dic	function
135	94.76	86.68	2.21	53.63	135	27.41	80.74	51.85
18	99.00	50.00	23.51	1.00	18	27.78	77.78	33.33
376	81.96	55.28	39.57	5.71	376	23.67	82.45	48.40
274	84.02	92.26	24.13	79.11	274	23.72	82.48	55.47

LIWC OUTPUT: LINGUISTIC PROCESSES

function	pronoun	ppron	i	we	you	shehe	they
51.85	9.63	6.67	0.00	0.00	0.00	6.67	0.00
33.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
48.40	8.78	1.86	0.27	0.27	0.27	0.53	0.53
55.47	13.50	8.39	0.00	6.57	1.46	0.00	0.36

LIWC OUTPUT

prep	auxverb	adverb	conj	negate	verb	adj	compare	interrog	n
16.87	8.58	4.70	6.22	0.97	13.83	5.12	2.63	0.41	
17.12	9.19	3.17	5.07	3.01	17.91	4.75	1.58	1.58	
14.67	9.33	8.00	5.33	1.33	18.67	5.33	2.67	1.33	
15.15	0.00	9.09	9.09	3.03	3.03	6.06	6.06	0.00	

LIWC OUTPUT: PSYCHOLOGICAL PROCESSES

affect	posemo	negemo	anx	anger	sad	social	family
4.29	2.49	1.38	0.00	0.69	0.14	5.81	0.14
4.60	1.11	3.49	0.48	1.27	0.32	14.90	0.48
4.00	4.00	0.00	0.00	0.00	0.00	6.67	1.33
9.09	0.00	9.09	0.00	9.09	0.00	12.12	0.00

LIWC OUTPUT: PSYCHOLOGICAL PROCESSES

male	cogproc	insight	cause	discrep	tentat	certain
0.83	12.17	1.94	1.38	1.66	2.35	2.90
2.06	12.04	2.85	1.11	0.79	3.01	1.58
1.33	10.67	4.00	2.67	0.00	0.00	1.33
0.00	18.18	0.00	0.00	0.00	0.00	9.09

LIWC OUTPUT: PERSONAL CONCERNS

work	leisure	home	money	relig	death
4.98	0.14	0.00	0.55	0	0.14
2.22	0.48	0.79	0.16	0	0.32
5.33	1.33	2.67	4.00	0	0.00
6.06	0.00	0.00	0.00	0	3.03

LIWC OUTPUT: INFORMAL LANGUAGE

death	informal	swear	netspeak	assent	nonflu	filler
0.14	0.55	0.14	0	0.14	0.28	0
0.32	0.00	0.00	0	0.00	0.00	0
0.00	0.00	0.00	0	0.00	0.00	0
3.03	0.00	0.00	0	0.00	0.00	0

LIWC OUTPUT: PUNCTUATION

AllPunc	Period	Comma	Colon	SemiC	QMark	Exclam	Dash	Quote
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

RECAP

- ngrams as generalisation of single-token analyses
- sentiment analysis in general
- sentiment trajectories
- psycholinguistics with the LIWC

OUTLOOK

Tutorial tomorrow

Next week: Reading week

Week 6: Machine learning 1

END