Textual Data
MY472 Week 8

November 19, 2019

Plan for today

- Text formats and encoding
- “Text as data” - why use text?
- NLP workflow
- Regular expressions
- “Ask me anything”

Text formats

Revisited: Basic units of data

- Bits
  - Smallest unit of storage; a 0 or 1
  - With *n* bits, can store 2^n patterns
- Bytes
  - 8 bits = 1 byte (why 1 byte can store 256 patterns)
  - ``eight bit encoding" - used to represent characters, such as represented as = 01000001

ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Encoding

- a ``character set'' is a list of character with associated numerical representations
- ASCII: the original character set, uses just 7 bits (2^7) – see http://ergoemacs.org/emacs/unicode_basics.html
- ASCII was later extended, e.g. ISO-8859, using 8 bits (2^8)
- but this became a jungle, with no standards: see http://en.wikipedia.org/wiki/Character_encoding

Solution: Unicode

- Unicode was developed to provide a unique number (a “code point”) to every known character – even some that are”unknown"
- problem: there are more far code points than fit into 8-bit encodings. Hence there are multiple ways to *encode* the Unicode code points
- *variable-byte* encodings use multiple bytes as needed. Advantage is efficiency, since most ASCII and simple extended character sets can use just one byte, and these were set in the Unicode standard to their ASCII and ISO-8859 equivalents
- two most common are **UTF-8** and **UTF-16**, using 8 and 16 bits respectively

Things to watch out for

- Input texts can be very different
- Many text production software (e.g. MS Office-based products) still tend to use proprietary formats, such as Windows-1252
- Windows tends to use UTF-16, while Mac and other Unix-based platforms use UTF-8
- Your eyes can be deceiving: a client may display gibberish but the encoding might still be as intended
- No easy method of detecting encodings (except in HTML meta-data)

Document formats

- Many different formats contain text
- How many can you think of?
- What problems are encountered?

Why use text?

Measuring unobserverables

- psychological states
- sentiment
- “topics”
- Ideology: “left-right” policy positions
- corruption
- cultural values
- power

Example: budget debate

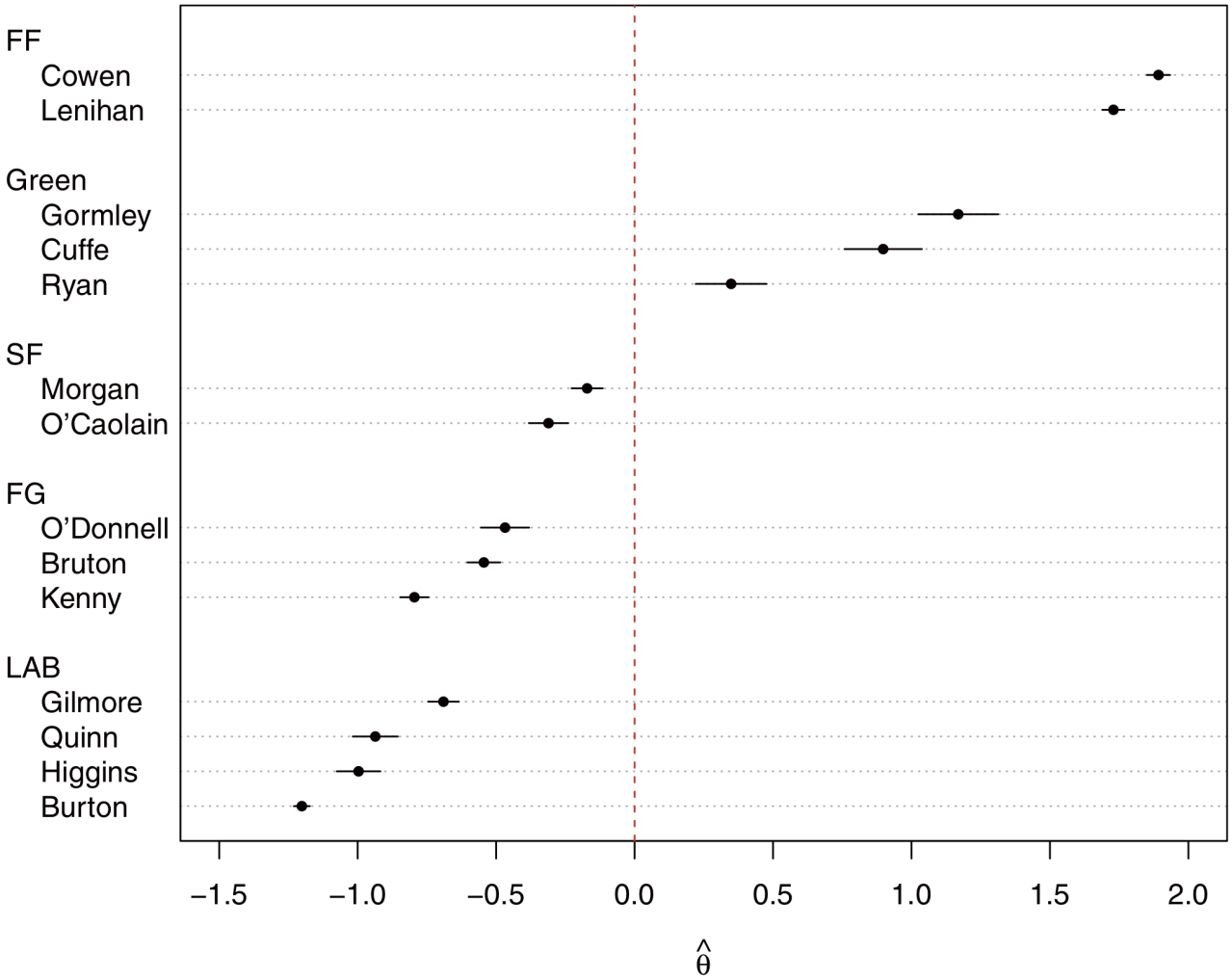


Fig. 2 Poisson scaling results.

Textual statistics

- Computed from lexical features, typically
- examples are:
  - frequency analysis
  - readability
  - lexical diversity

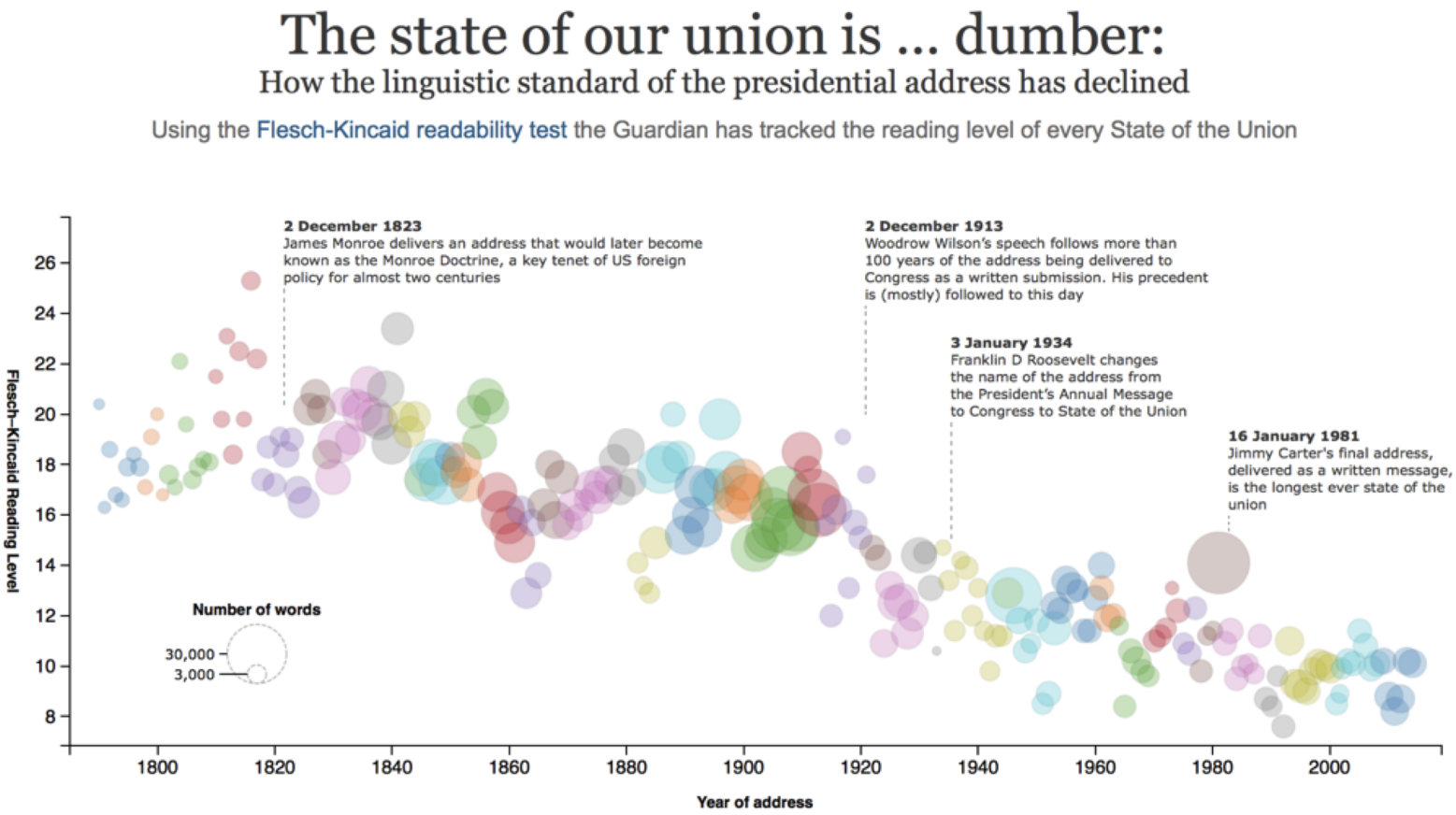
Example: "Readability" on US Presidential Speeches

- A corpus of State of the Union addresses, given each January by the US President
- Uses a descriptive index of "reading ease", developed in educational science, known as the Flesch-Kincaid measure

$$206.835 - 1.015\left(\frac{\#\text{ of words}}{\#\text{ of sentences}}\right) - 84.6\left(\frac{\#\text{ of syllables}}{\#\text{ of words}}\right)$$

- Widely identified trend that this is greatly increasing over time: SOTU addresses are getting simpler

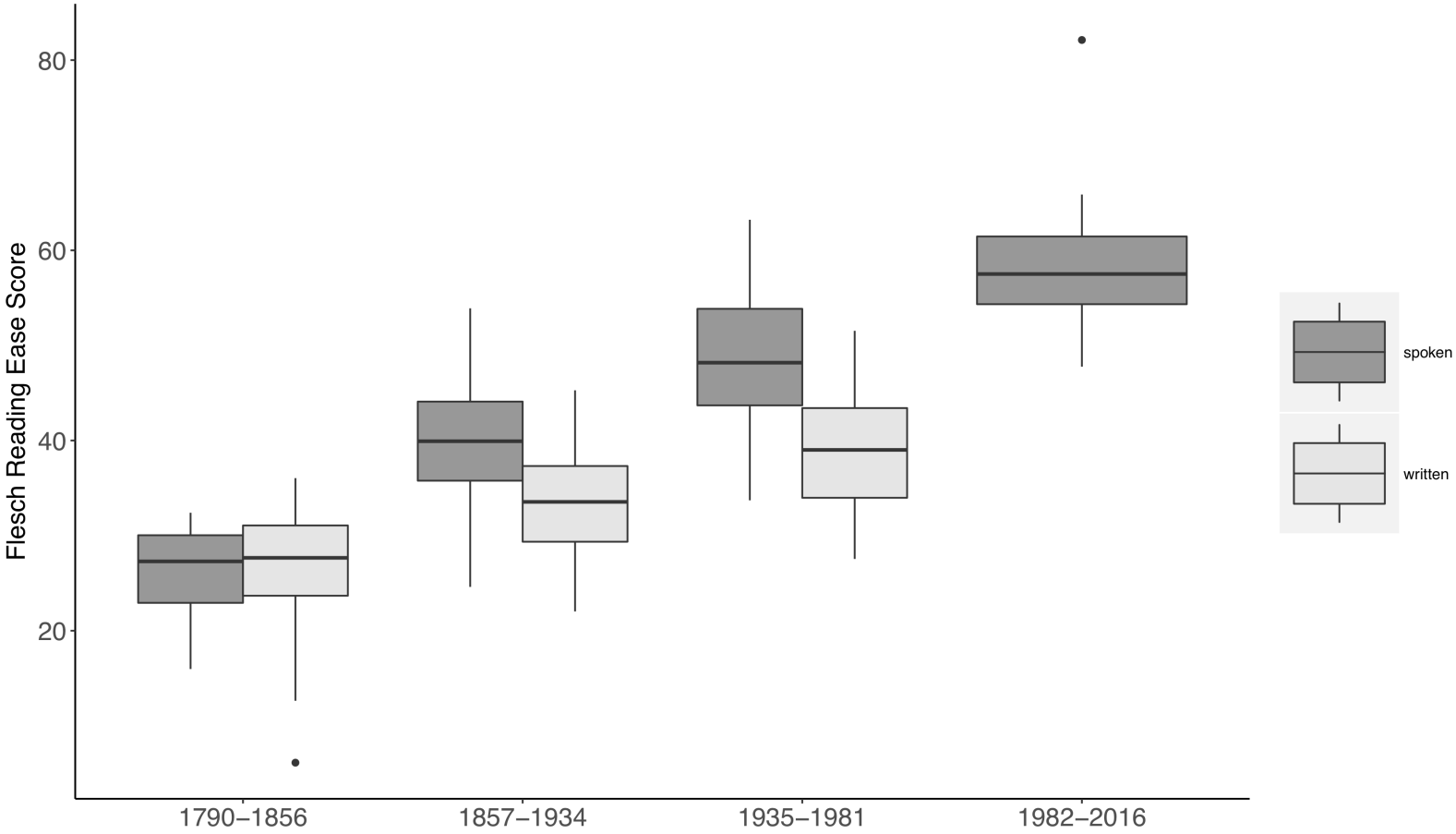
Visualized


The state of our union is ... dumber:
How the linguistic standard of the presidential address has declined
Using the Flesch-Kincaid readability test the Guardian has tracked the reading level of every State of the Union
2 December 1823
James Monroe delivers an address that would later become known as the Monroe Doctrine, a key tenet of US foreign policy for almost two centuries
2 December 1913
Woodrow Wilson's speech follows more than 100 years of the address being delivered to Congress as a written submission. His precedent is (mostly) followed to this day
3 January 1934
Franklin D Roosevelt changes the name of the address from the President's Annual Message to Congress to State of the Union
16 January 1981
Jimmy Carter's final address, delivered as a written message, is the longest ever state of the union
Number of words
30,000
3,000
Flesch-Kincaid Reading Level
26
24
22
20
18
16
14
12
10
8
1800
1820
1840
1860
1880
1900
1920
1940
1960
1980
2000
Year of address

But not a general trend

- Only appears in US presidential State-of-the-Union addresses
- Partly an effect of changing audiences and the mode of delivery: from written to spoken form
- Did not hold true for other forms of political communication, either in the US or comparatively

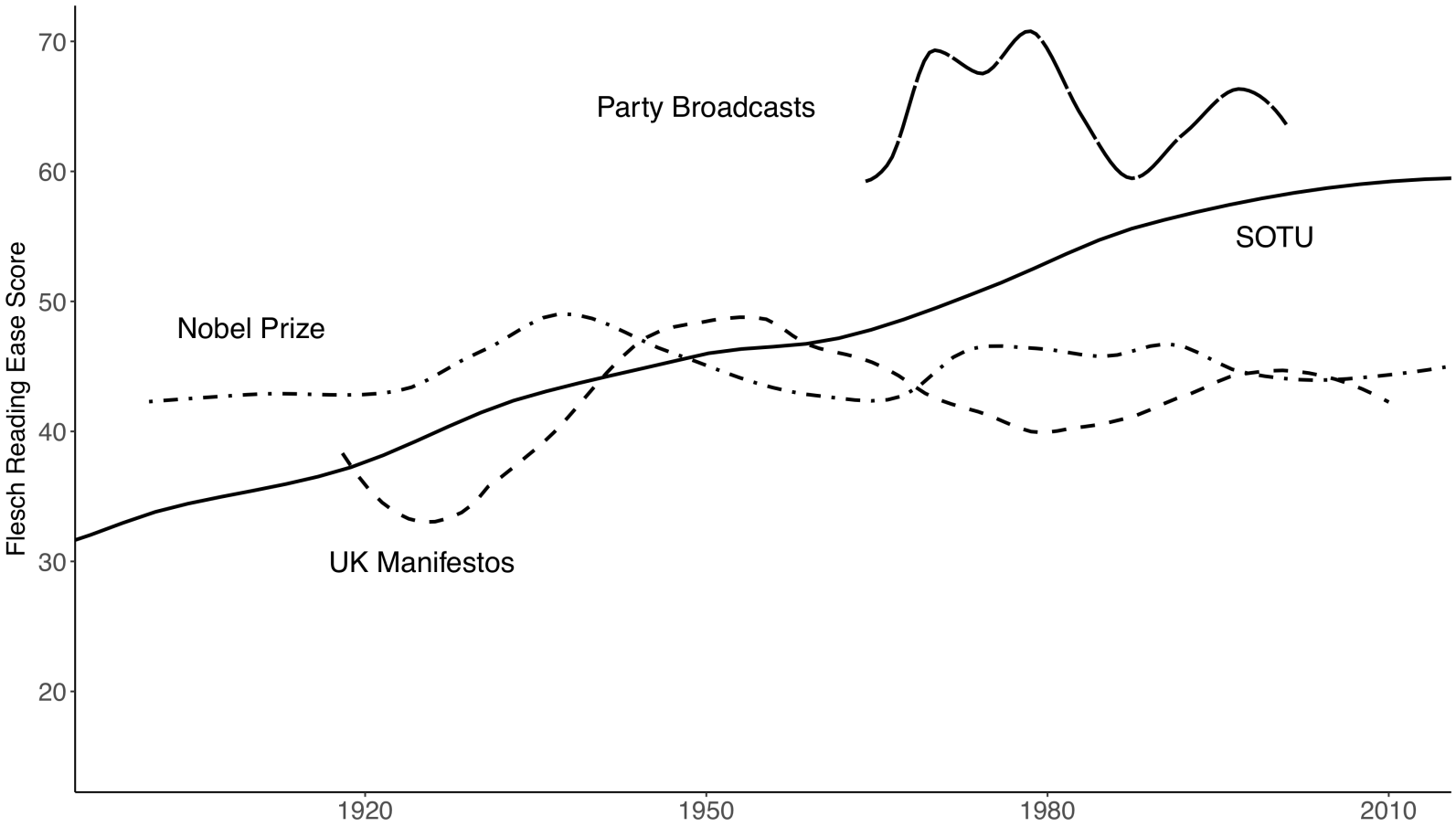
Differences in Reading Ease by Delivery Method of Speech



Not evident in other US texts

Not evident in other contexts

Prediction


Flesch Reading Ease Score
70
60
50
40
30
20
1920
1950
1980
2010
Party Broadcasts
SOTU
Nobel Prize
UK Manifestos

“Ground truth” (for supervised learning)

- Human annotation: Part of the research process
  - Authors
  - Students
  - Crowd workers
- Accompanies the data: Part of the data generation process
  - Star ratings for reviews - most movie sentiment databases (Pang, Lee, and Vaithyanathan 2002)
  - Votes for legislative bills - Thomas, Pang and Lee (2006)

Examples

Get out the vote: Determining support or opposition from Congressional floor-debate transcripts

Matt Thomas, Bo Pang, and Lillian Lee
Department of Computer Science, Cornell University
Ithaca, NY 14853-7501
mattthomas84@gmail.com, pabo@cs.cornell.edu, llee@cs.cornell.edu

Abstract

We investigate whether one can determine from the transcripts of U.S. Congressional floor debates whether the speeches represent support of or opposition to proposed legislation. To address this problem, we

online accessibility of politically oriented texts in particular, however, is a phenomenon that some have gone so far as to say will have a potentially society-changing effect.

In the United States, for example, governmental bodies are providing and soliciting political documents via the Internet, with lofty goals in

Examples

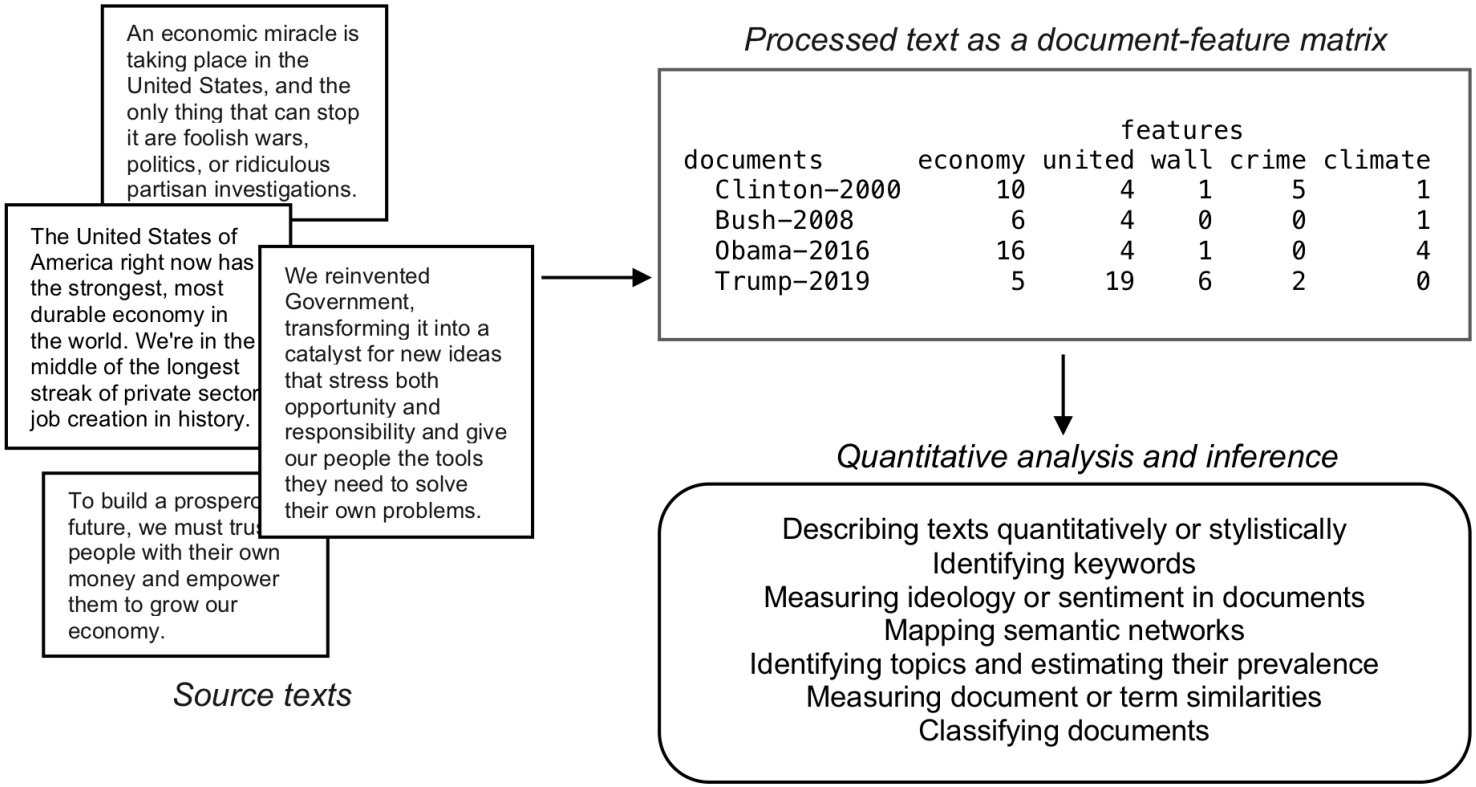
we seek to determine from the transcripts of U.S. Congressional floor debates whether each "speech" (continuous single-speaker segment of text) represents support for or opposition to a proposed piece of legislation. Note that from an experimental point of view, this is a very convenient problem to work with because we can automatically determine ground truth (and thus avoid the need for manual annotation) simply by consulting publicly available voting records.

Not always obvious

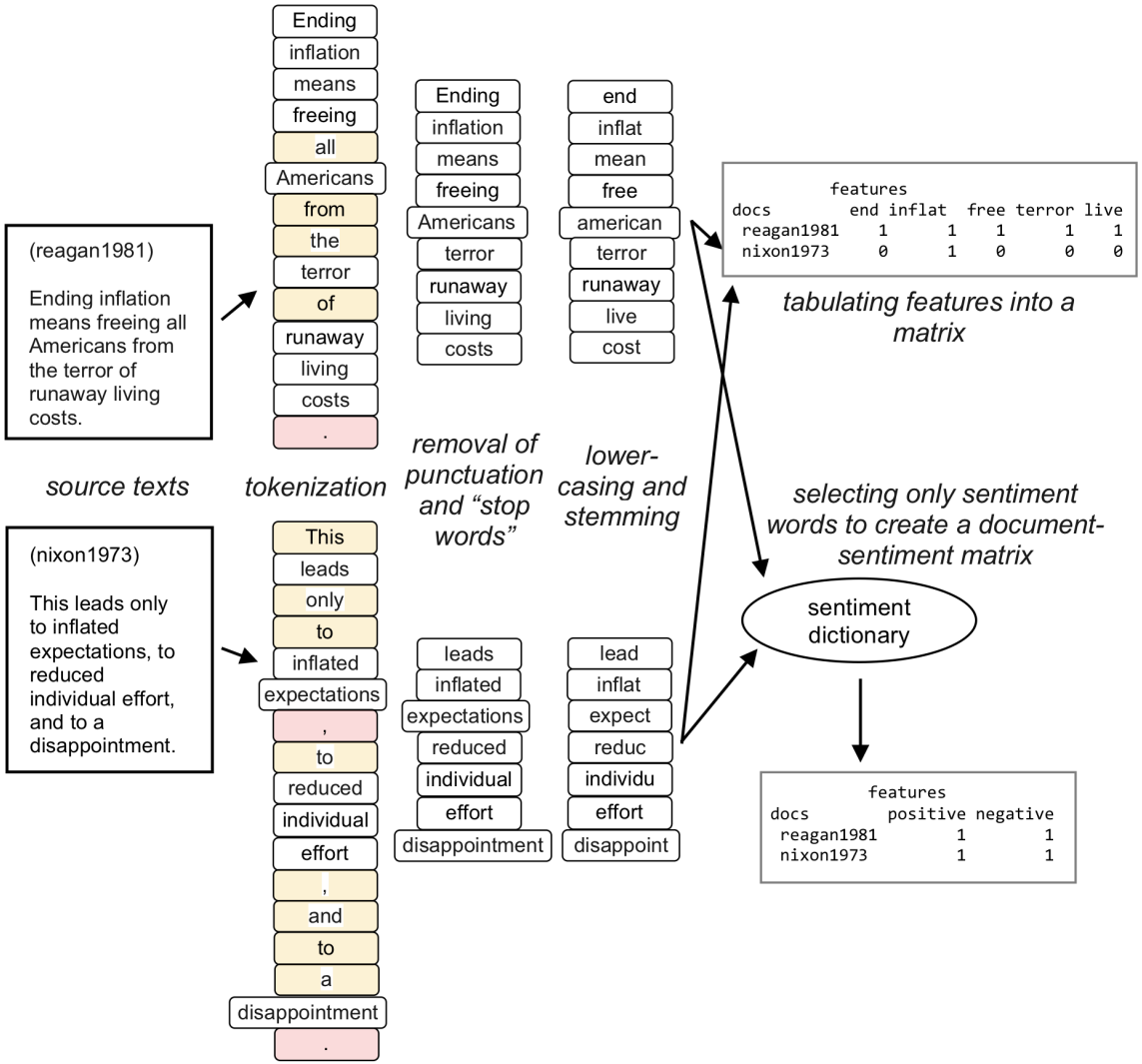
- text is not like cats v. dogs
- sometimes not even cats v. dogs is clear

Workflow

Text analysis workflow



Tokenization



Dictionary approaches

- Map each word or phrase to a “dictionary” of words, associated with a known “sentiment” or psychological state
- Treats matches within each dictionary as equivalent
- Dictionary choice is important, but the computer does all of the counting – so perfectly reliable
- Examples: Linguistic Inquiry and Word Count, or the General Inquirer

Dictionary example (from LIWC 2015)

```
Dictionary object with 1 key entry.
- [posemo]:
- like, like*, :), (:, accept, accepta*, accepted, accepting, accepts, active, …
interests, invigor*, joke*, joking, jolly, joy*, keen*, kidding,
kind, kindly, kindn*, kiss*, laidback, laugh*, legit, libert*,
likeab*, liked, likes, liking, livel*, lmao*, lmfao*, lol, love, loved, lovelier, ...
```

Problems

- “polysemy” – multiple meanings. kind has three!
- From State of the Union corpus: 318 matches
  - kind/NOUN – 95%
  - kind (of)/ADVERB – 1%
  - kind/ADJECTIVE – 4%
- These are known as false positives
- Other problem: false negatives (what we miss)
  - Missed: kindliness
  - Also missed: altruistic and magnanimous

Regular expressions

Regular expressions

- a concise and flexible tool for describing patterns in strings
- the “glob” format (as in the dictionary examples) is a simplified version
- Very good coverage in *R for Data Science*, chapter on “strings”
- Also: Jeffrey Fried, *Mastering Regular Expressions*

Matching

- simplest: match exact “strings”

```
library("stringr")
x <- c("apple", "banana", "pear")
str_extract(x, "an")
```

```
## [1] NA   "an" NA
```

Matching - case sensitive

- case is an option

```
bananas <- c("banana", "Banana", "BANANA")
str_detect(bananas, "banana")
```

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

```
str_detect(bananas, regex("banana", ignore_case = TRUE))
```

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

Wildcards

- `.` matches any character
- `*` matches no or more of the preceding character
- `+` matches one or more of the preceding character
- `()` allow grouping
- `[]` defines character classes
- `\p{}` defines categories

For more details

- R package **stringr**
  - including an excellent tutorial "Regular expressions"
- R package **stringi**
- base `grep()` is deprecated but possible to use

And don't forget the humble "glob"

- beginning and end of match are built into the pattern
- `*` means no or any characters
- `?` means any single character
- built into **quanteda**: `?quanteda::valuetype`

Back to the dictionary example

```
Dictionary object with 1 key entry.
- [posemo]:
- like, like*, :), (:, accept, accepta*, accepted, accepting, accepts, active, …
interests, invigor*, joke*, joking, jolly, joy*, keen*, kidding,
kind, kindly, kindn*, kiss*, laidback, laugh*, legit, libert*,
likeab*, liked, likes, liking, livel*, lmao*, lmfao*, lol, love, loved, lovelier, ...
```

How do you do more with text?

Take **MY429: Quantitative Text Analysis** (Lent Term)

Week 8 Lab

Textual data hackathon!