Words: Encodings & Tokenization

CS 490A, Fall 2021

Applications of Natural Language Processing
https://people.cs.umass.edu/~brenocon/cs490a_f21

Brendan O'Connor & Laure Thompson

College of Information & Computer Sciences
University of Massachusetts Amherst

Administrivia – Submissions

- Submit Exercise 0 if you haven't
- HW0 due Thursday (9/9) at **11:59pm EST**
- Future assignments will be due **before class**

Administrivia – Office Hours

- Akshay's OH: **Wednesday** 2-3:30pm , CS207
- Yuanguo's OH: **Thursday** 1-2:30pm, LGRT-T220
- Xiao's OH: **Friday** 10:15-11:45am, LGRT-T223

Administrivia – Schedule

- Schedule on website:

  people.cs.umass.edu/~brenocon/cs490a_f21/schedule.html

  - Readings and lectures will be posted here

- Complete **required** readings before class

  - “Reading” = required reading
  - “Optional” = supplemental

Input Collection

Word	Count
article	100
charming	32
dog	0
newspaper	29
poem	3
⋮	⋮

Working Representations

Frequency Analysis

Topic Modeling

Term-Covariate Ranking

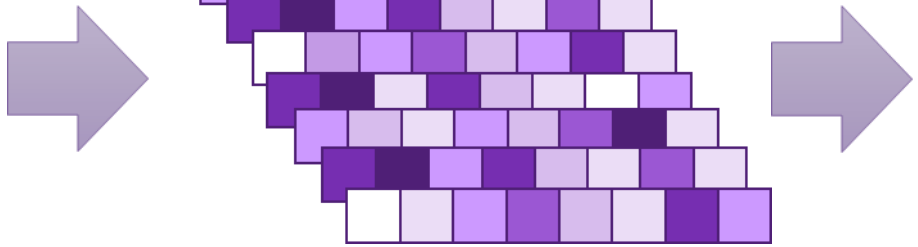
Supervised Learning

⋮

Downstream Analysis

Raw

Numerical



Frequency Analysis

Topic Modeling

Term-Covariate Ranking

Supervised Learning

⋮

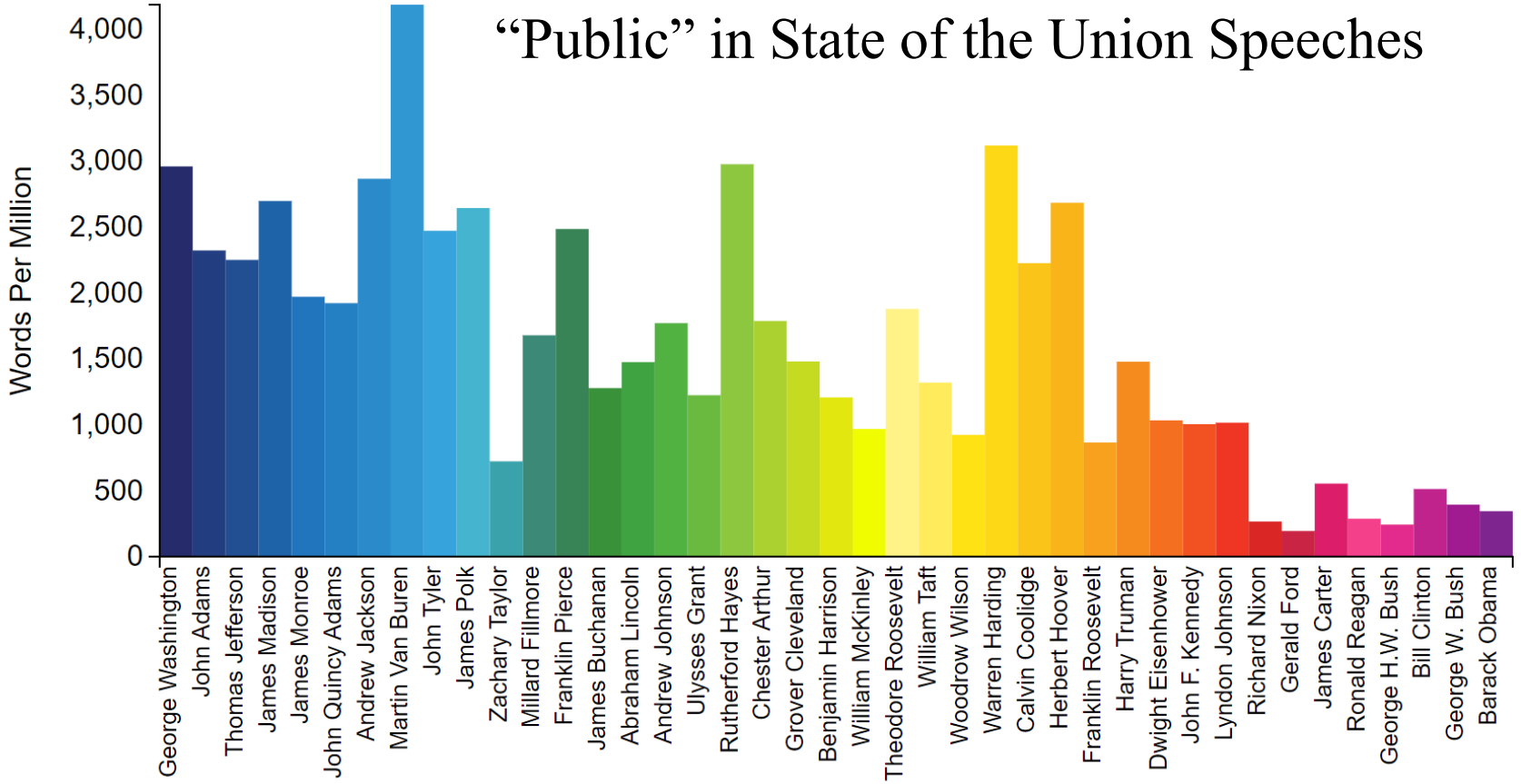
Input Collection

Vector Representations

Downstream Analysis

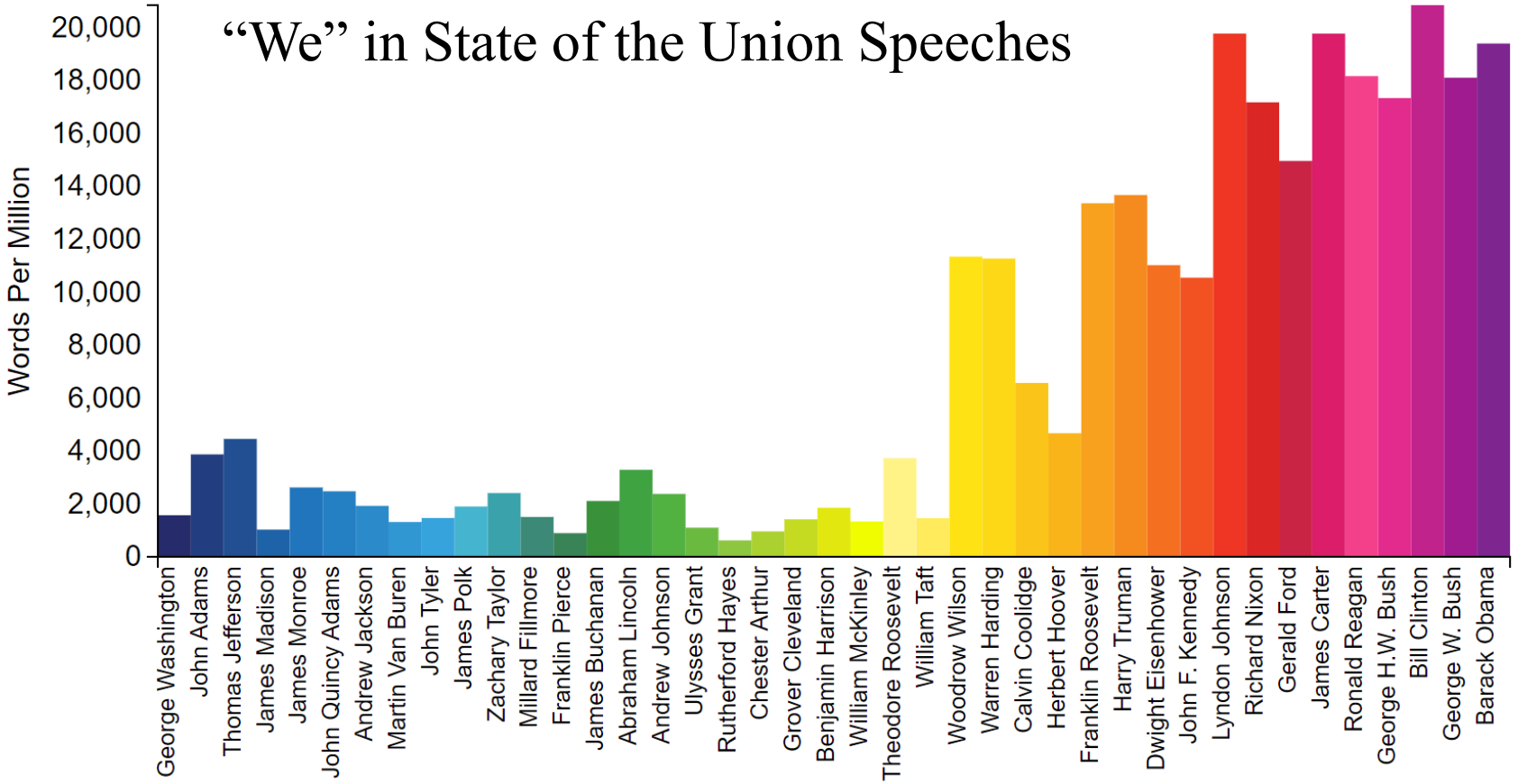
What can we do with word counts?

Analyze trends in *specific* language use



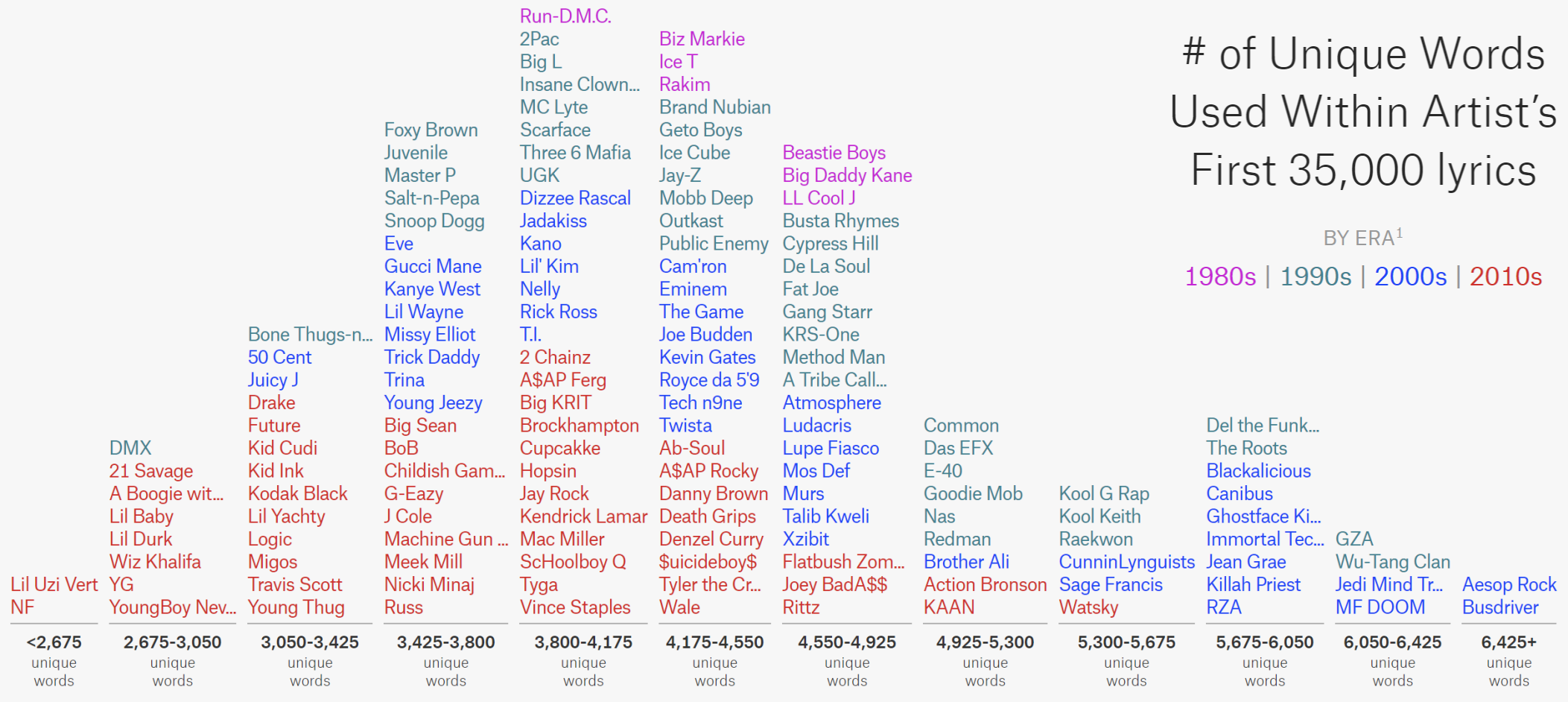
Schmidt & Fraas 2015

Analyze trends in *specific* language use



Schmidt & Fraas 2015

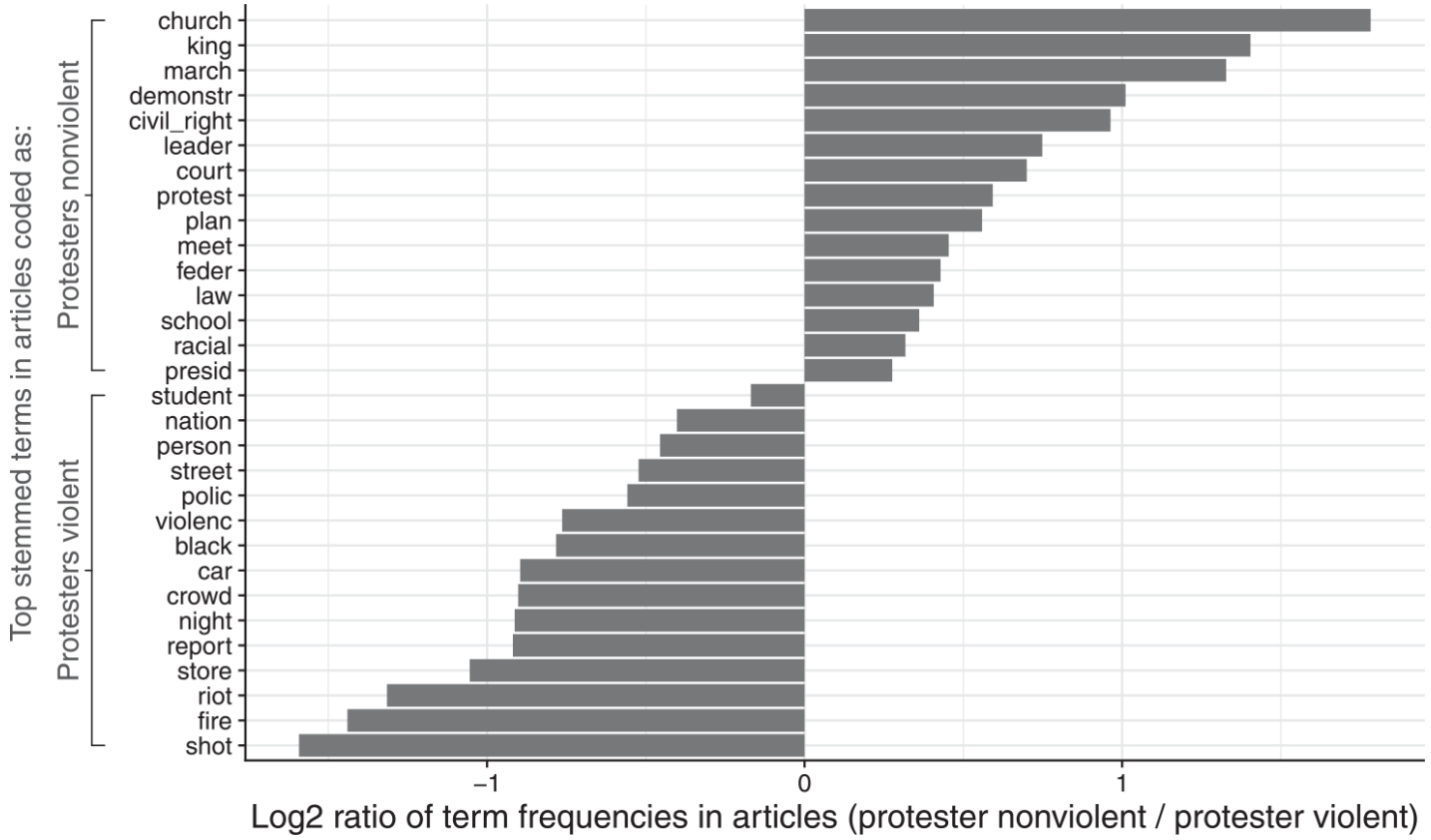
Compare *general* language use



Daniels 2019

Identify distinctive word use

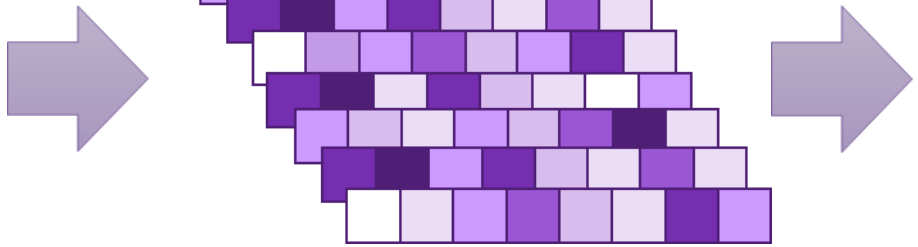
Corpus: News articles from the late 1960s



Wasow 2020

Raw

Numerical



Frequency Analysis

Topic Modeling

Term-Covariate Ranking

Supervised Learning

⋮

Input Collection

Vector Representations

Downstream Analysis

Raw

Input Collection

Numerical

Vector Representations

Frequency Analysis

Topic Modeling

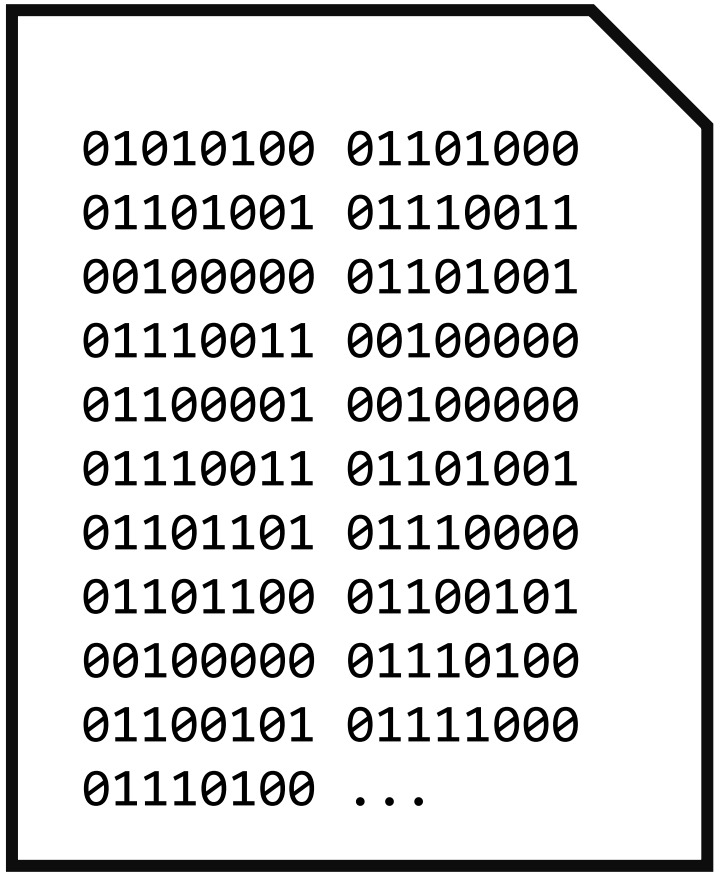
Term-Covariate Ranking

Supervised Learning

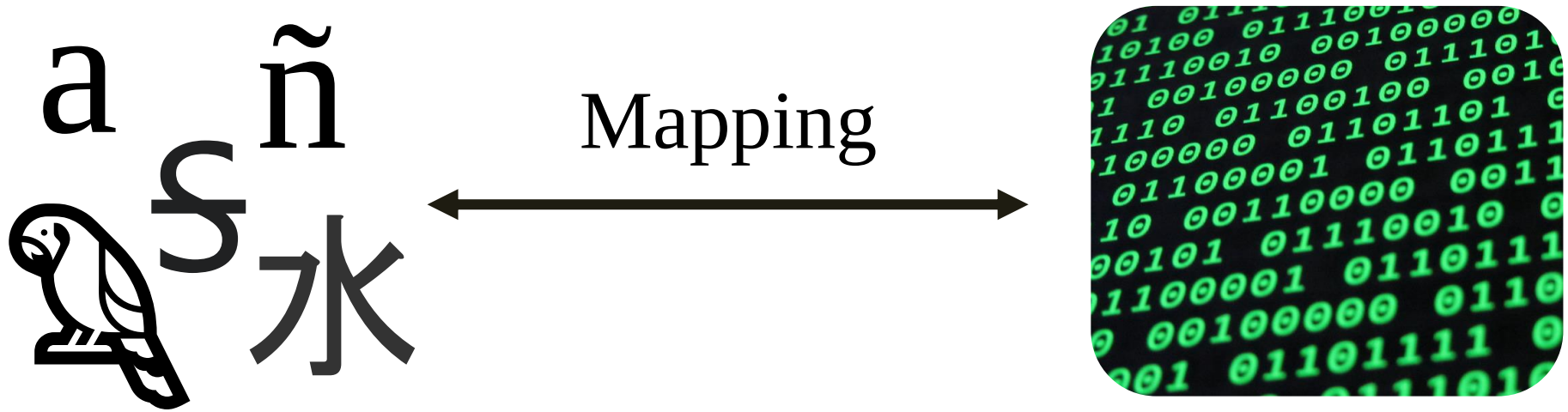
⋮

Downstream Analysis

This is a simple sentence.



From Bits to Characters


a
ñ
Mapping

Bacon's cipher

Strategy: Assign each letter in the English alphabet a unique value

A → 0	X → 23
B → 1	Y → 24
C → 2	Z → 25

Q: How many bits do we need to uniquely encode these numbers?

Bacon's cipher

Strategy: Assign each letter in the English alphabet a unique value

A → `00000`	X → `01111`
B → `00001`	Y → `11000`
C → `00010`	Z → `11001`

Ascii – 7 bits

USASCII code chart

$b_7 b_6 b_5$ →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits ↓ b_4 b_3 b_2 b_1	Column → / Row ↓ 0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

A → 65

B → 66

Z → 90

a → 97

b → 98

z → 123

Ascii – 7 bits

USASCII code chart

$b_7 b_6 b_5$ → / Bits ↓	000	001	010	011	100	101	110	111
$b_4 b_3 b_2 b_1$ ↓ / Row ↓ Column →	0	1	2	3	4	5	6	7
0000 0	NUL	DLE	SP	0	@	P	`	p
0001 1	SOH	DC1	!	1	A	Q	a	q
0010 2	STX	DC2	"	2	B	R	b	r
0011 3	ETX	DC3	#	3	C	S	c	s
0100 4	EOT	DC4	$	4	D	T	d	t
0101 5	ENQ	NAK	%	5	E	U	e	u
0110 6	ACK	SYN	&	6	F	V	f	v
0111 7	BEL	ETB	'	7	G	W	g	w
1000 8	BS	CAN	(	8	H	X	h	x
1001 9	HT	EM	)	9	I	Y	i	y
1010 10	LF	SUB	*	:	J	Z	j	z
1011 11	VT	ESC	+	;	K	[	k	{
1100 12	FF	FS	,	<	L	\	l	|
1101 13	CR	GS	-	=	M	]	m	}
1110 14	SO	RS	.	>	N	^	n	~
1111 15	SI	US	/	?	O	_	o	DEL

A → 65 = 100 0001

B → 66 = 100 0010

Z → 90 = 101 1010

a → 97 = 110 0001

b → 98 = 110 0010

z → 123 = 111 1010

Aside – End of Line Characters

USASCII code chart

$b_7 b_6 b_5$ → / Bits ↓	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 Row ↓ Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

LF = Line Feed = \n

CR = Carriage Return = \r

<u>End of Line Defaults</u>

- Unix: \n
- Windows: \r\n

Latin-1 (ISO-8859-1): Ascii + an extra bit

8_ 128																
9_ 144																
A_ 160	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	SHY 00AD	® 00AE	¯ 00AF
B_ 176	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C_ 192	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D_ 208	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E_ 224	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F_ 240	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

ISO-8859-7: A different ASCII extension

8_ 128																
9_ 144																
A_ 160	NBSP 00A0	‘ 2018	’ 2019	£ 00A3	€ 20AC	₯ 20AF	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ͺ 037A	« 00AB	¬ 00AC	SHY 00AD		— 2015
B_ 176	° 00B0	± 00B1	² 00B2	³ 00B3	΄ 0384	΅ 0385	Ά 0386	· 00B7	Έ 0388	Ή 0389	Ί 038A	» 00BB	Ό 038C	½ 00BD	Ύ 038E	Ώ 038F
C_ 192	ΐ 0390	Α 0391	Β 0392	Γ 0393	Δ 0394	Ε 0395	Ζ 0396	Η 0397	Θ 0398	Ι 0399	Κ 039A	Λ 039B	Μ 039C	Ν 039D	Ξ 039E	Ο 039F
D_ 208	Π 03A0	Ρ 03A1		Σ 03A3	Τ 03A4	Υ 03A5	Φ 03A6	Χ 03A7	Ψ 03A8	Ω 03A9	Ϊ 03AA	Ϋ 03AB	ά 03AC	έ 03AD	ή 03AE	ί 03AF
E_ 224	ΰ 03B0	α 03B1	β 03B2	γ 03B3	δ 03B4	ε 03B5	ζ 03B6	η 03B7	θ 03B8	ι 03B9	κ 03BA	λ 03BB	μ 03BC	ν 03BD	ξ 03BE	ο 03BF
F_ 240	π 03C0	ρ 03C1	ς 03C2	σ 03C3	τ 03C4	υ 03C5	φ 03C6	χ 03C7	ψ 03C8	ω 03C9	ϊ 03CA	ϋ 03CB	ό 03CC	ύ 03CD	ώ 03CE	

Unicode: A single mapping to rule them all

Standard that maps characters to *codepoints*

Unicode 14.0 is releasing next week!

- Overall: 144,697 characters, 159 scripts
- New: 838 characters, 5 scripts, 37 emojis

Unicode Encodings: UTF-8

- Variable length encoding: Represents each *codepoint* as **1-4 bytes**
- **Compatible** with ASCII
- Python 3’s default text encoding

Unicode Encodings: UTF-16

- Variable length encoding: Represents each *codepoint* as **2 or 4** bytes
- **Not** compatible with ASCII

Q: When might we prefer UTF-16 over UTF-8?

UTF-8 vs. UTF-16

UTF-8 > UTF-16

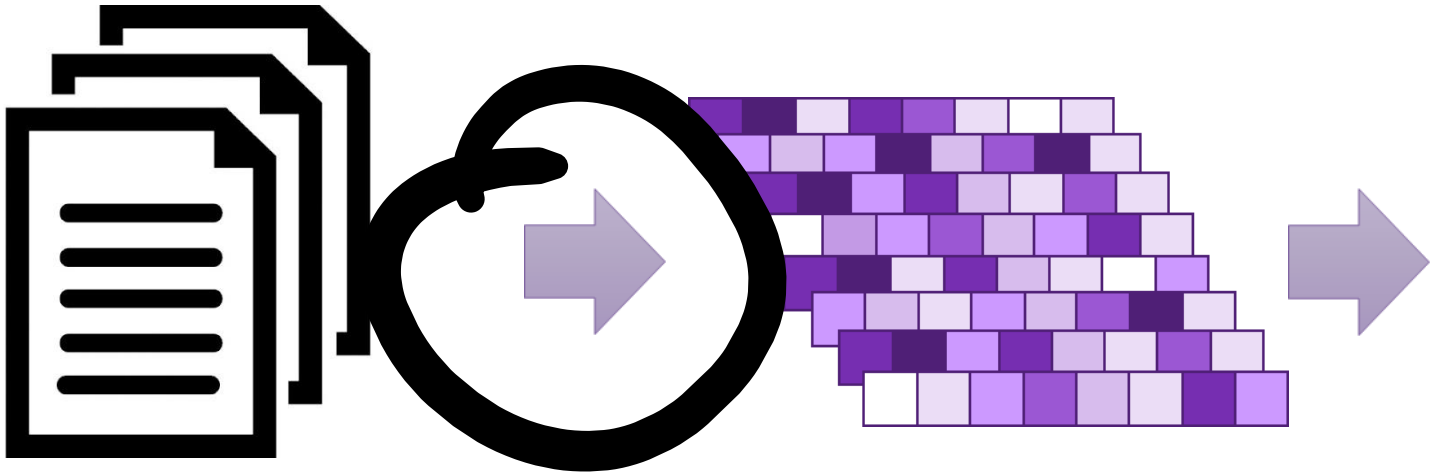
- Need ASCII compatibility
- More efficient for languages that can be represented by 1-2 bytes in UTF-8 (e.g. English, Spanish, Greek)

UTF-16 > UTF-8

- Majority of characters would be represented by 3 bytes in UTF-8, these are represented by 2 bytes in UFT-16 (e.g. Chinese, Japanese, Korean, Vietnamese)

Raw

Numerical



Frequency Analysis

Topic Modeling

Term-Covariate Ranking

Supervised Learning

⋮

Input Collection

Vector Representations

Downstream Analysis

Regular Expressions

Regular Expressions

- A formal language for specifying specific text strings
- How can we search for any of these?

woodchuck
woodchucks
Woodchuck
Woodchucks

Regular Expressions: Character Classes

- Disjunction of listed characters

Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

- Ranges

Pattern	Matches	Example
`[A-Z]`	An upper case letter	**D**renched Blossoms
`[a-z]`	A lower case letter	**m**y beans were impatient
`[0-9]`	A single digit	Chapter **1**: Down the Rabbit Hole

Regular Expressions: Negation

- Negation of class: [^Ss] (^ only means negation in first position)

Pattern	Matches	Example
`[^A-Z]`	Not an upper case letter	`Oyfn pripetchik`
`[^Ss]`	Neither `S` nor `s`	`I have no exquisite reason”`
`[^e^]`	Neither `e` nor `^`	`Look here`
`a^b`	The pattern `a^b`	`Look up a^b now`

Regular Expressions: More disjunction

- Woodchuck is another name for groundhog

Pattern	Matches
`groundhog|woodchuck`	
`yours|mine`	yours mine
`a|b|c`	= `[abc]`
`[gG]roundhog|[wW]oodchuck`	

Regular Expressions: ? * + .

Pattern	Matches	Examples
`colou?r`	Optional previous char	**color** **colour**
`oo*h!`	0 or more of previous char	**oh!** **ooh!** **oooh!** **ooooh!**
`o+h!`	1 or more of previous char	**oh!** **ooh!** **oooh!** **ooooh!**
`baa+`		**baa** **baaa** **baaaa** **baaaaa**
`beg.n`		**begin** **begun** **began** **beg1n**

Demo / Activity