RICE SOCIAL SCIENCES
Department of Linguistics

language data & nltk

LING 430 Computational Linguistics Fall 2026
lecture 3a
Sep 8 2026

Siyu Liang
siyu.liang@rice.edu

last time

- control flow: `if`, `for`, `while`
- dict

agenda

- Python on language data
- Zipf’s law
- nltk

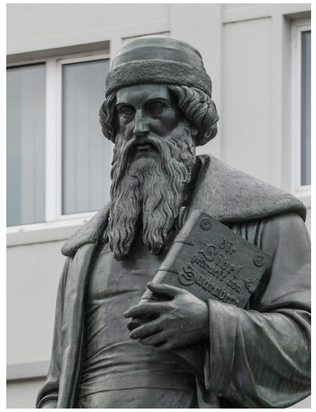
before we start

5 min

- go to this link → copy some texts elsewhere → paste it → submit
  - or type https://tinyurl.com/riceling
- requirements:
  - English only
  - natural language (no code, shopping list, etc.)
  - no private information
- we'll get back to it later

Python on language data

Project Gutenberg



Project Gutenberg

- an online collection of tens of thousands of free eBooks
- started as a one man's project and still volunteer-run
- named after **Johannes Gutenberg**, who invented mechanical printing press (and *'re'*-invented movable type)
- in the US, copy-rights generally expire 95 years after first publication date, so the dataset is mostly older books

our text today

the complete works of Shakespeare

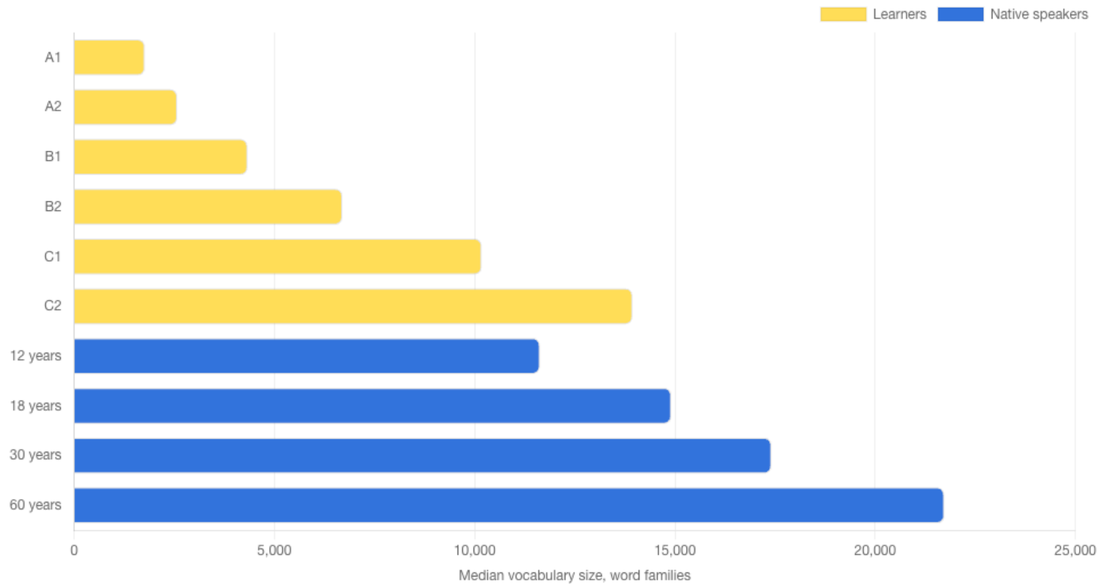
- run the first cell in our Python notebook
  - retrieving the entire works of Shakespeare text from the Gutenberg Project site
- when it finishes, you should see an output ‘5555356’ printed out below
- we could take a sneak peak after the run
  - `print(text[:200])`

counting words

- `.split()` splits one big string into words
  ```
  words = text.split()
  len(words)
  ```
  → 963460
- how many *different* words do you expect?

counting words

- `set()` gets all unique words
  `len(set(words))`
  → 71162
- that seems like a lot!
- for reference
  - a study by BRAVE, an English test website
- something must be going on!



counting words

- `list.count(`*`value`*`)`
  - returns how many times a *value* appears in the list

    `words.count("love")`
    → 1364

    `words.count("love,")`
    → 425

    `words.count("love.")`
    → 211

    `words.count("Love")`
    → 80

counting words

first fix: capitalization

- lowercase everything before splitting
  ```
  lowered = text.lower().split()
  len(set(lowered))
  ```
  → 63100
- about 8k of unique “words” were only CapiTaliZation

counting words

second fix: punctuation

- a provided cell strips the punctuation around each word
- we'll learn about it next week, but just run it for now
  `len(set(cleaned_words))`
  → 31138
  `cleaned_words.count("love")`
  → 2291
- much bigger than the first count of 1,364

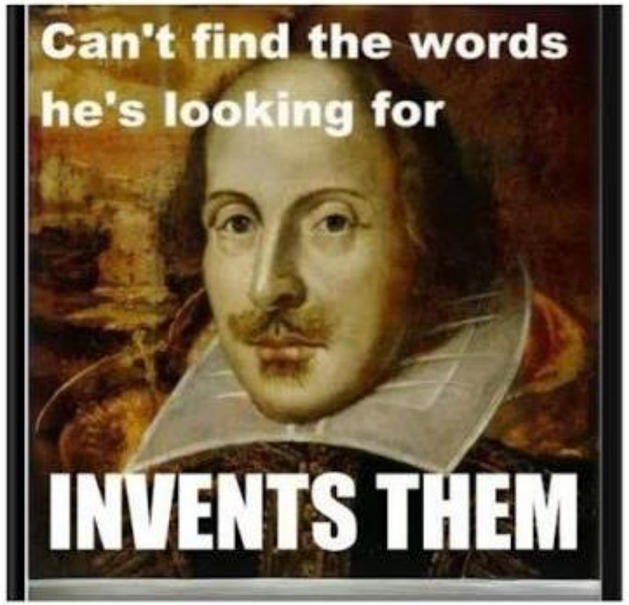
counting words

other fixes

- digits: sixty-seven vs 67
- hyphens: *ripe-red*, *urchin-snouted*, *sweet-smelling*
- apostrophes
  - *’d*: arm’d, kill’d, belov’d
  - *’s*: he’s, ’tis, the king’s
  - *’ll*: I’ll, we’ll, she’ll
- to be discussed later in the class

counting words

- every count depends on a definition of **word**
  - both a **linguistic** decision and an **engineering** decision
- published estimates of Shakespeare's vocabulary range from about 17,000 to about 30,000
- but now you know why people can disagree



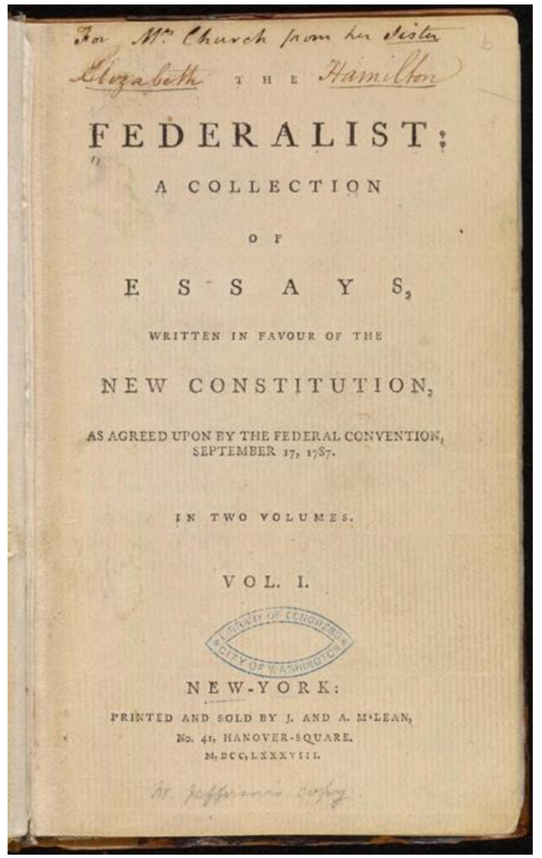
counting words

ranked by frequency

#	word	count	#	word	count
1	the	30,359	11	is	9,728
2	and	28,421	12	not	9,045
3	i	22,134	13	with	8,506
4	to	20,953	14	me	8,252
5	of	18,734	15	it	8,187
6	a	16,300	16	for	8,166
7	you	14,361	17	his	7,600
8	my	13,186	18	be	7,356
9	in	12,390	19	this	7,109
10	that	11,790	20	your	7,057

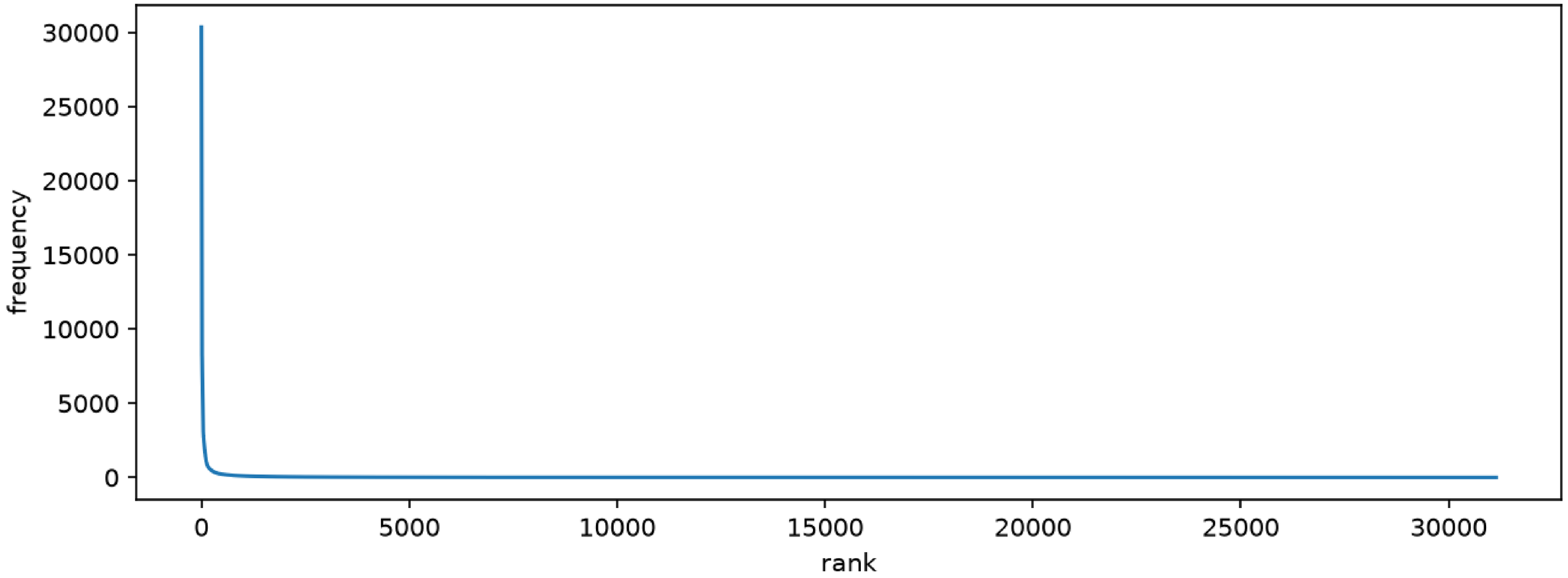
counting words

- **function words** say a lot about the author
- the pattern of common words usage can help to determine who wrote a text
- e.g. 85 essays in the *Federalist Papers* (1787–88) appeared under pseudonym “Publius”
  - twelve of them disputed between Hamilton and Madison
  - Mosteller & Wallace (1964) counted the function words
  - Hamilton used *upon* constantly; Madison almost never did
  - verdict: all twelve to Madison
- also: unmasking of “Robert Galbraith” as J. K. Rowling (2013)



counting words

the distribution

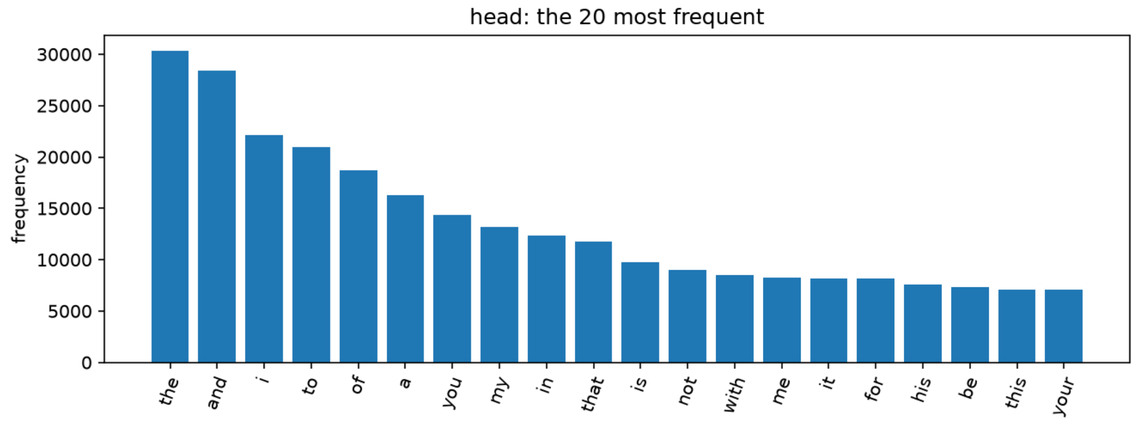


Zipf’s law

Zipf's law

the head

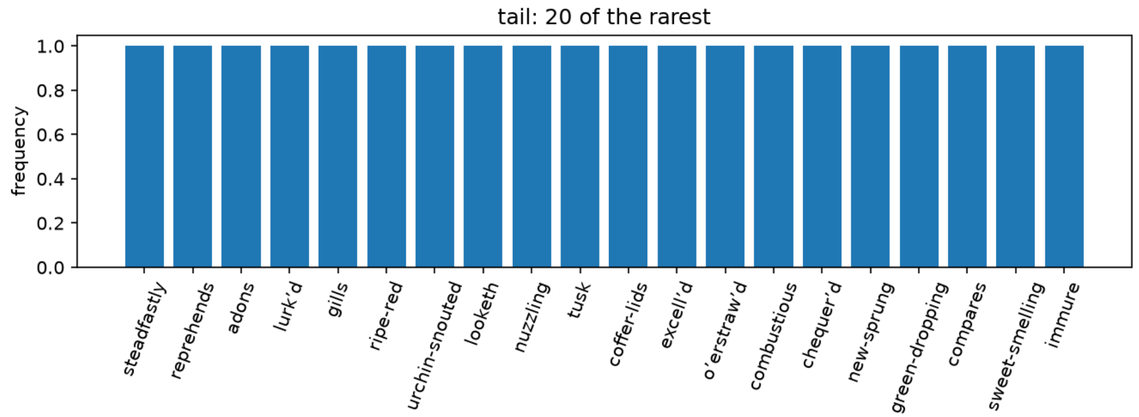
- a few words are very frequent
  - *the*, *and*, *I*, *to*, *of*, *a*, …
- for Shakespeare
  - *the* makes up 3.07% of everything he wrote
  - top 100 words make up 52.3%



Zipf’s law

the looooooong tail

- most words are not so frequent
  - proper names, rare items, etc.
- for Shakespeare
  - 8,790 (35.9%) words appear exactly once
  - *honorificabilitudinitatibus* (the longest word, *Love's Labour's Lost)*
  - *anthropophaginian* (in *Merry Wives)*



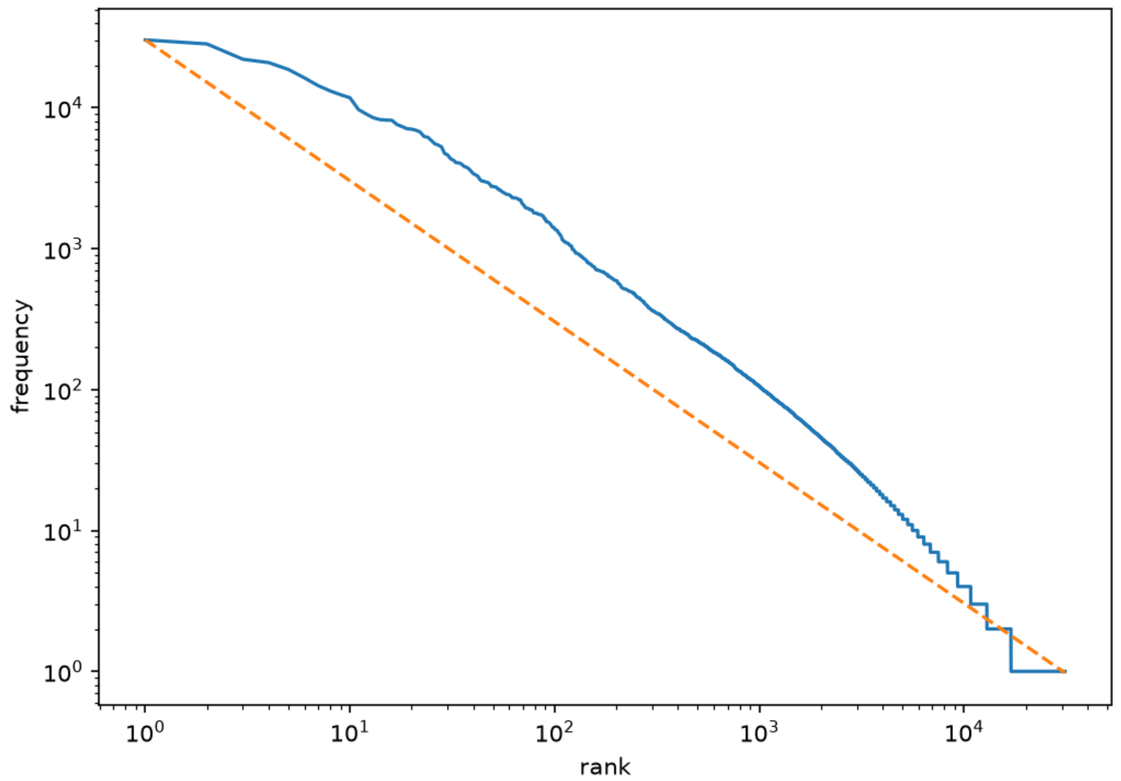
Zipf's law

- **George Kingsley Zipf** (1902–1950), a linguist at Harvard
- a word's frequency is inversely proportional to its rank
  - the 2nd most frequent word occurs about half as often as the 1st, the 3rd about a third, …

Zipf's law

- usually drawn on a log–log scale

$$word\ frequency \propto \frac{1}{word\ rank}$$



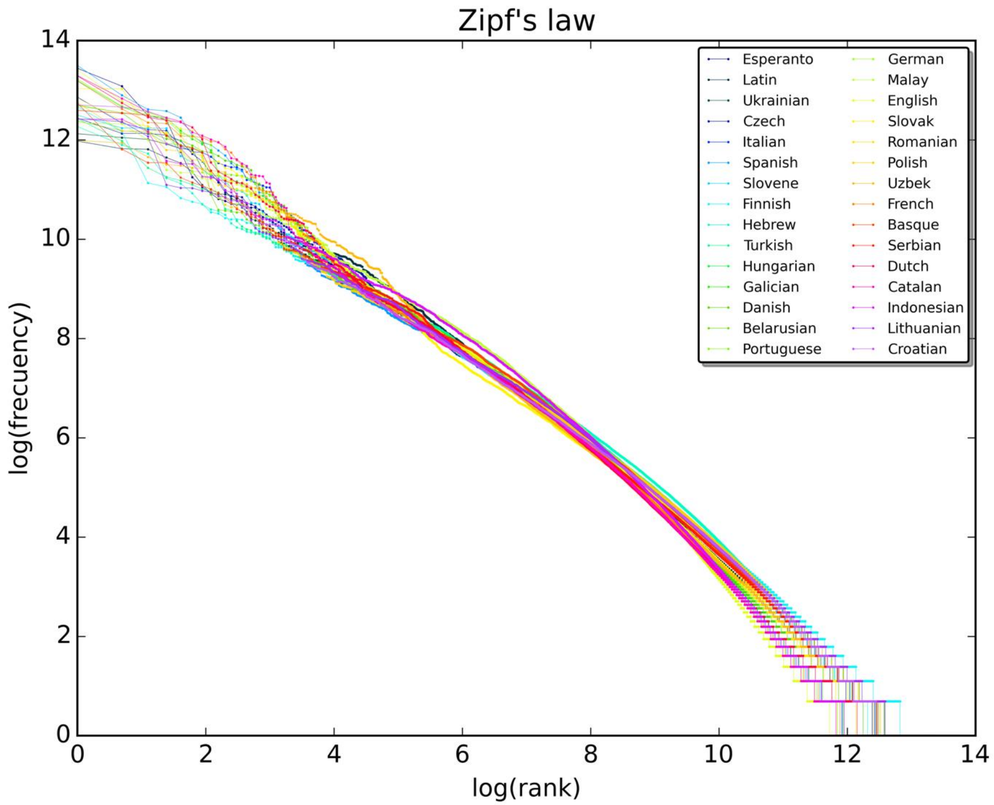
Zipf's law

in Shakespeare

rank	word	count
1	the	30,359
10	that	11,790
100	first	1,383
1,000	hadst	105
10,000	ponderous	4

Zipf's law

- it holds for every language anyone has checked
- rank vs frequency for Wikipedia data in thirty languages
- question: what does it say about natural language?



the class corpus

nltk

doing linguistics with Python

- you can do a great deal of string processing yourself
  - write your own code for linguistic tasks
- but there are many **libraries** that do things for you
  - **nltk**: the Natural Language Toolkit
  - spaCy: spacy.io
  - Stanza: stanfordnlp.github.io/stanza
  - …

doing linguistics with Python

using libraries

- pros
  - other people will know what you used
  - less work for you, and easier to maintain
- cons
  - might not do exactly what you expect
  - bugs are harder to trace
  - version incompatibilities, security vulnerabilities

nltk

the Natural Language Toolkit

- a collection of teaching and demonstration tools by Steven Bird and colleagues
- used to search and analyze texts, annotate them, classify them, generate them
- not built for large-scale applications (spaCy or Stanza), but widely used where speed is not crucial

Steven Bird

nltk

import

- import the whole library, then the book's nine texts
  ```
  import nltk
  from nltk.book import *
  ```
  → text1: Moby Dick by Herman Melville 1851
  → text2: Sense and Sensibility by Jane Austen 1811
  → text3: The Book of Genesis
  → …
- you get nine variables for whole texts

nltk

- text1 is an nltk.text.Text object
  `text1.name`
  → 'Moby Dick by Herman Melville 1851'
  `len(text1)`
  → 260819
  `text1.count("monstrous")`
  → 10
- an *attribute* is a property the object has: `.name`
- a *method* is something it can do: `.count()`
- both are reached with the dot, like `.split()` and `.lower()`

nltk

concordance

- every occurrence of a word, centered, with its context
  `text1.concordance("monstrous")`
  → Displaying 11 of 11 matches:
  …

nltk

methods & arguments

- arguments can be optional, and can be named
  `text1.concordance("monstrous", lines=3)`
  `text1.concordance("monstrous", 40, 3)`
  → Displaying 3 of 11 matches:
  → was of a most monstrous size . ... Thi
  → Touching that monstrous bulk of the wh
  → enish array of monstrous clubs and spea
- `.concordance()` takes up to three arguments: the word, the display width (default 79), the number of lines (default 25)
- omit an argument and the default is used
- `help(text1.concordance)` shows the relevant documentation

nltk

.similar()

- words that appear in the same contexts as monstrous
  `text1.similar("monstrous")  # Melville`
  → true contemptible christian abundant few part mean careful …
  `text2.similar("monstrous")  # Austen`
  → very so exceedingly heartily a as good great extremely …
- for Austen it patterns with *very* and *exceedingly*
  - an intensifier, as in “a monstrous pretty girl”

nltk

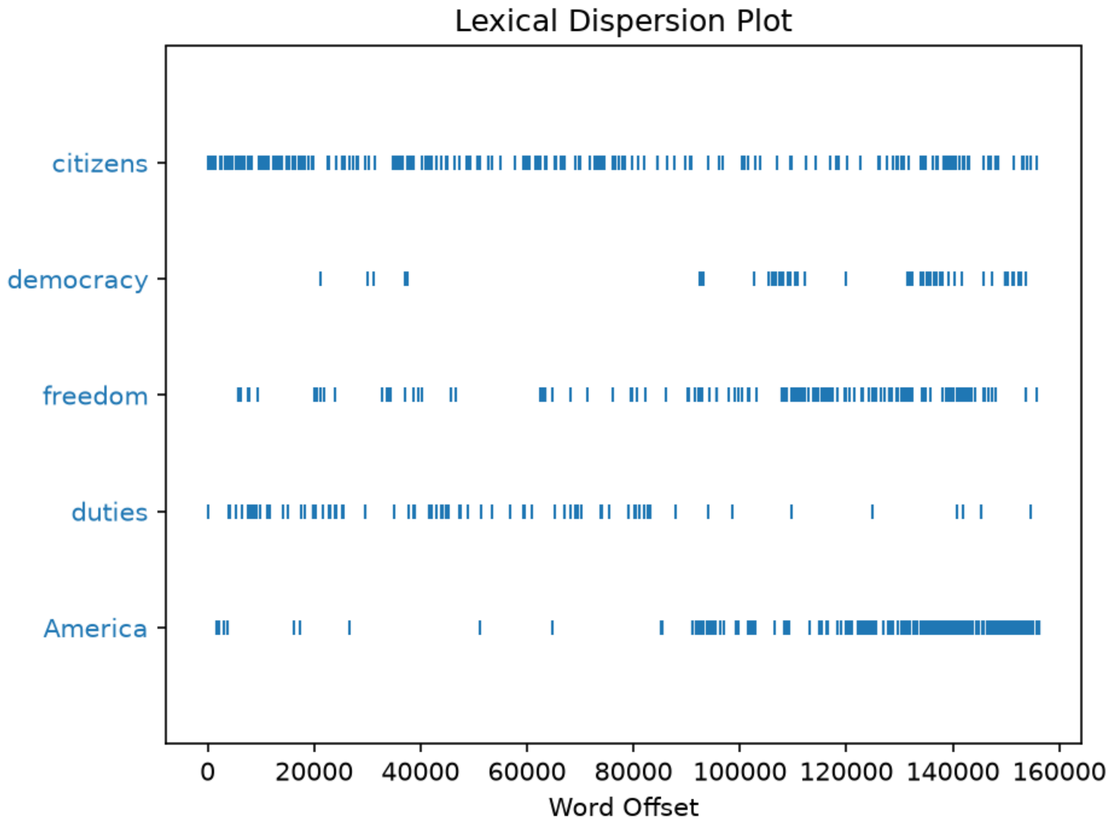
.similar()

- the notebook wraps the Shakespeare text as an nltk Text called shakes
  `shakes.similar("love")`
  → that this will you death have it life all heart me for and see blood …
- *love* keeps company with *death, life, heart,* and *blood*

nltk

.dispersion_plot()

- where in a text a word occurs `text4.dispersion_plot(["citizens", "democracy", "freedom", "duties", "America"])`
- `text4` is every inaugural address since 1789, in order, so the x-axis is time



activity

- play around with the texts in nltk
  - do similar things for different texts
  - look up new functions
  - share what you find

next time

- Thursday
  - functions, files & text normalization
  - *Think Python*, Ch. 3: Functions
  - *Think Python*, Ch. 13, first half: Files
- **HW 1 due Thursday, Sep 10**
- HW 2 to be released Thursday
  - due dates have been updated for future homeworks