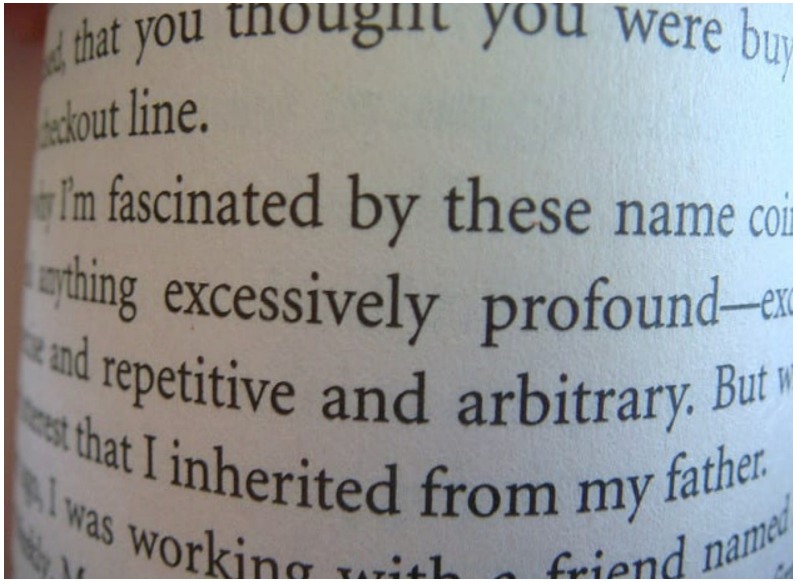
CS 505: Introduction to Natural Language Processing

Wayne Snyder
Boston University

Lecture 3: Basic Notions and Low-Level Text Normalization



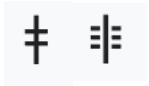
Basic Notions: Characters, Words, Tokens, Documents, Corpora

The human race has developed many different writing systems, based on several categories of graphemes (atomic symbols). To *vastly over-simplify*, we have

Alphabets: A set of < 100 symbols, each roughly corresponding to a speech sound:

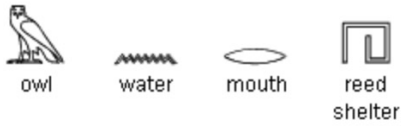
Example: English: a b c …. z Greek: $\alpha\ \beta\ \gamma\ \delta$ ... ω

Syllabaries: A set of 100s of symbols, each roughly corresponding to spoken syllable:

Linear B (early Greek):  pa-te = πατήρ = pater = ”father”

Japanese: さか, *saka*, "hill"

Logographies: A set of 1000s of symbols, each roughly corresponding to a spoken word or concept:

Egyptian Hieroglyphs: 

Chinese: 山 · mountain. 刀 knife 下 down

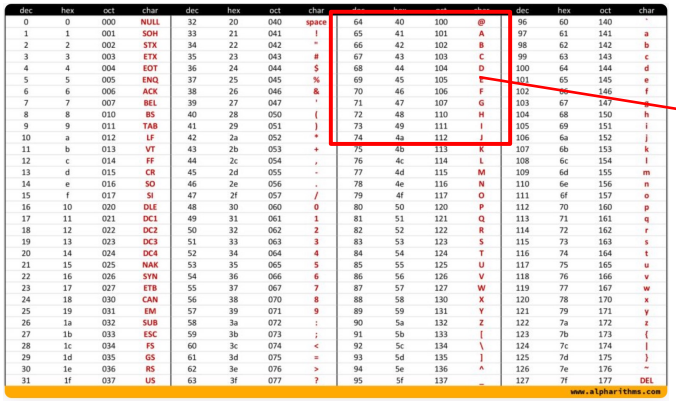
What about Emojis?

Basic Notions: Characters, Words, Tokens, Documents, Corpora

For NLP, we want to process text as a sequence of atomic symbols and these may be any of the preceeding categories.

Thus: In NLP, textual data is presented in its most basic form as a sequence of atomic symbols from some finite collection (think Unicode!).

In CS 505, our language is English, and this collection will be ASCII characters, and we will generally just call them characters. Thus, in its most basic form, a text is simply one long string.



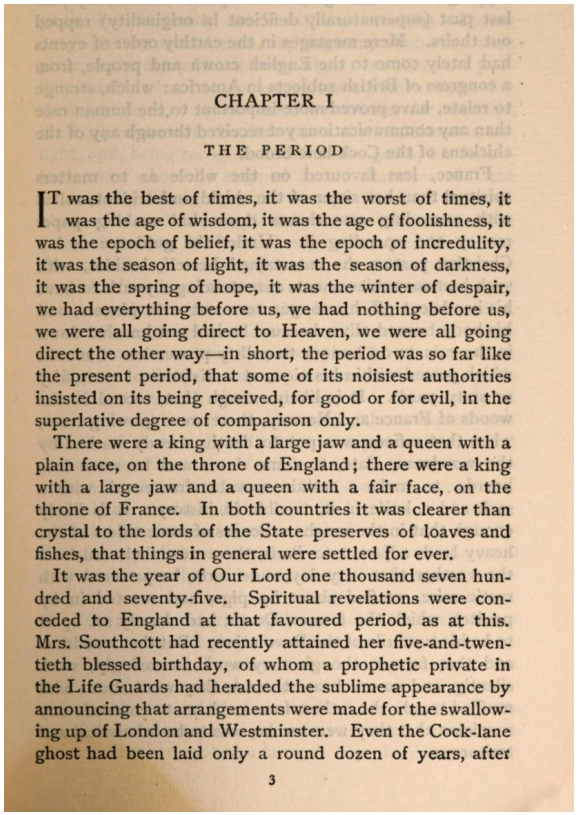
dec	hex	oct	char
64	40	100	@
65	41	101	A
66	42	102	B
67	43	103	C
68	44	104	D
69	45	105	E
70	46	106	F
71	47	107	G
72	48	110	H

Caveat: We may find it useful if we analyze social media texts to consider Emojis (which are given Unicode numbers!).

Basic Notions: Characters, Words, Tokens, Documents, Corpora

This string form of a text is the minimal representation of the information content of the text (excluding formatting, diagrams, different fonts, illustrations, etc.) and may include some minimal formatting (white space, \n, \t, etc.):

A Tale of Two Cities, Charles Dickens



CHAPTER I

THE PERIOD

IT was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of light, it was the season of darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way—in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

There were a king with a large jaw and a queen with a plain face, on the throne of England; there were a king with a large jaw and a queen with a fair face, on the throne of France. In both countries it was clearer than crystal to the lords of the State preserves of loaves and fishes, that things in general were settled for ever.

It was the year of Our Lord one thousand seven hundred and seventy-five. Spiritual revelations were conceded to England at that favoured period, as at this. Mrs. Southcott had recently attained her five-and-twentieth blessed birthday, of whom a prophetic private in the Life Guards had heralded the sublime appearance by announcing that arrangements were made for the swallowing up of London and Westminster. Even the Cock-lane ghost had been laid only a round dozen of years, after

3

```
"CHAPTER I\nThe Period\nIt was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way—in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.\nThere were a king with a large jaw and a queen with a plain face, on the throne of England; there were a king with a large jaw and a queen with a fair face, on the throne of France. In both countries it was clearer than crystal to the lords of the State preserves of loaves and fishes, that things in general were settled for ever.  ..... "
```

Basic Notions: Characters, Words, Tokens, Documents, Corpora

Although we will have occasion to use the string form when we study character-level machine learning models, almost all NLP uses data which has been grouped into larger units:

Words: Sequence of characters, separated by white space or punctuation;

Tokens: Words possibly preprocessed into some more useful form (the rest of this lecture);

Sentences: Sequences of words/tokens

Paragraphs: Sequences of sentences

Chapters/Sections/Topics: Sequences of paragraphs

Optional

Document: Sequence of paragraphs

Corpus: Set of documents

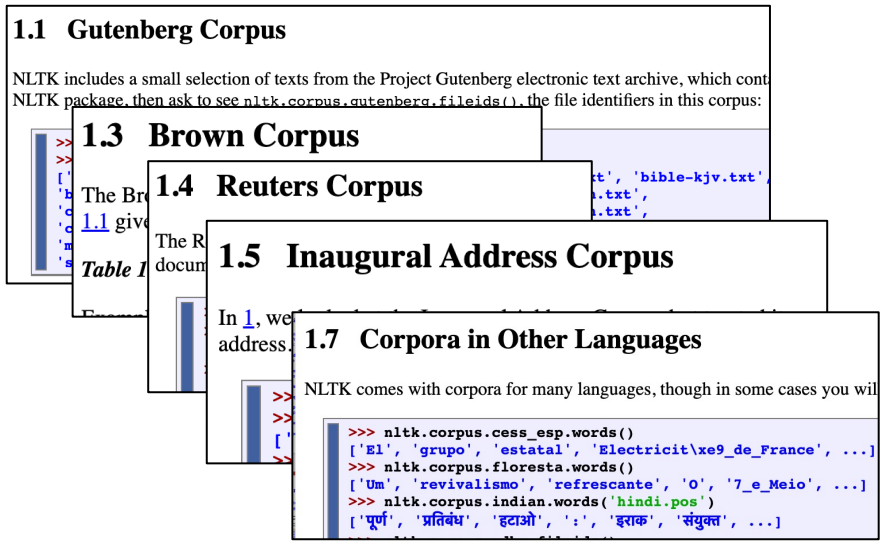
Brown Corpus:

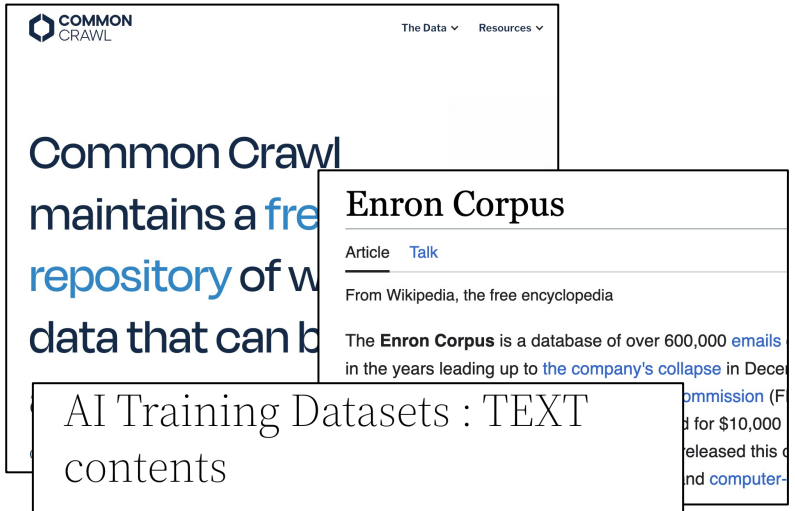
```
String form (a list of characters):

"The Fulton County Grand Jury said Friday an investigation of Atlanta

Words (a list of strings):

['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

Sentences (a list of lists of strings):

[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an',
'primary', 'election', 'produced', '``', 'no', 'evidence', "''", 'tha
'.'], ['The', 'jury', 'further', 'said', 'in', 'term-end', 'presentme
ttee', ',', 'which', 'had', 'over-all', 'charge', 'of', 'the', 'elect
nd', 'thanks', 'of', 'the', 'City', 'of', 'Atlanta', "''", 'for', 'th
'was', 'conducted', '.'], ...]

Paragraphs (a list of lists of lists of strings):

[[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an'
'primary', 'election', 'produced', '``', 'no', 'evidence', "''", 'tha
'.']], [['The', 'jury', 'further', 'said', 'in', 'term-end', 'present
mittee', ',', 'which', 'had', 'over-all', 'charge', 'of', 'the', 'ele
'and', 'thanks', 'of', 'the', 'City', 'of', 'Atlanta', "''", 'for', '
n', 'was', 'conducted', '.']], ...]
```

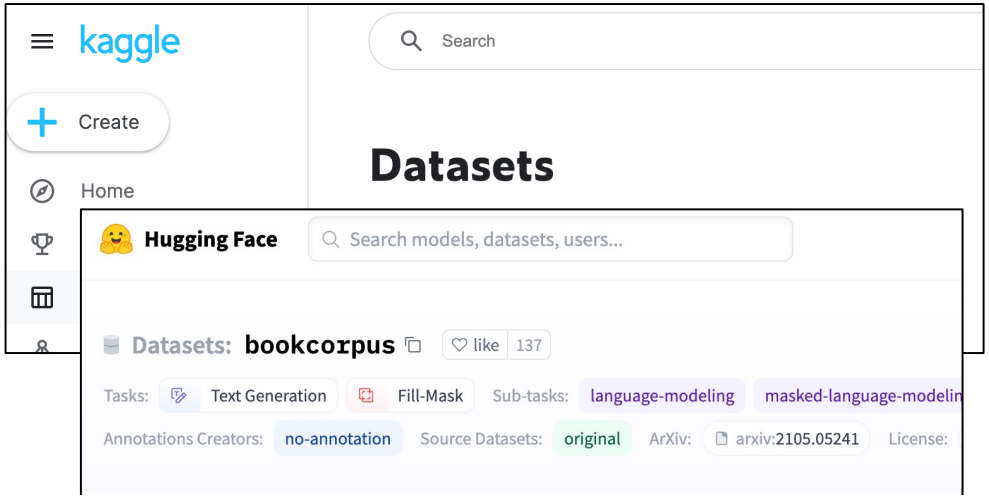
ID	File	Genre	Description
A16	ca16	news	Chicago Tribune: *Society Reportage*
B02	cb02	editorial	Christian Science Monitor: *Editorials*
C17	cc17	reviews	Time Magazine: *Reviews*
D12	cd12	religion	Underwood: *Probing the Ethics of Realtors*
E36	ce36	hobbies	Norling: *Renting a Car in Europe*
F25	cf25	lore	Boroff: *Jewish Teenage Culture*
G22	cg22	belles_lettres	Reiner: *Coping with Runaway Technology*
H15	ch15	government	US Office of Civil and Defence Mobilization: *The Family Fallout Shelter*
J17	cj19	learned	Mosteller: *Probability with Statistical Applications*
K04	ck04	fiction	W.E.B. Du Bois: *Worlds of Color*
L13	cl13	mystery	Hitchens: *Footsteps in the Night*
M01	cm01	science_fiction	Heinlein: *Stranger in a Strange Land*
N14	cn15	adventure	Field: *Rattlesnake Ridge*
P12	cp12	romance	Callaghan: *A Passion in Rome*
R06	cr06	humor	Thurber: *The Future, If Any, of Comedy*

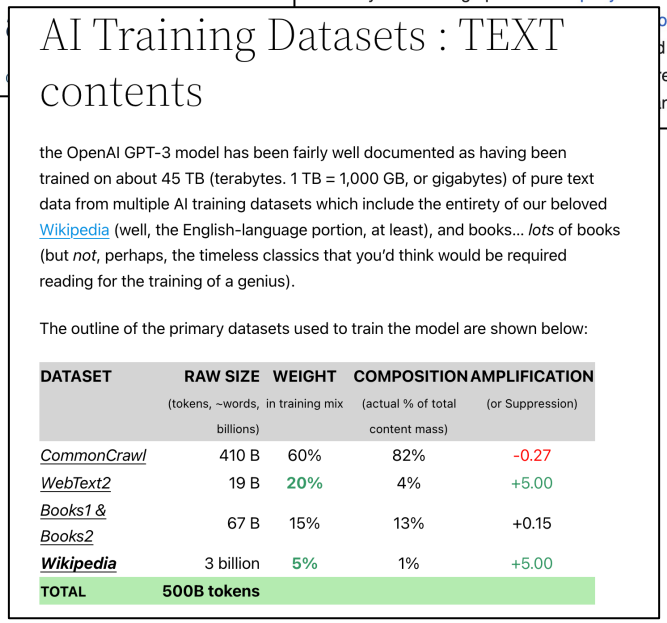
Corpora for Natural Language Processing

There are many publicly-available corpora for NLP, often categorized (and preprocessed) for specific tasks, in various languages, etc. Dr Google will help you find these….









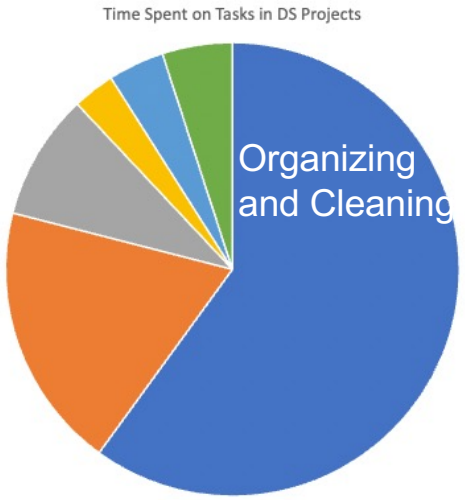
Textual Data Preparation

Why do we need to learn low-level text processing?

Because these corpora are for education, contests, creating general language models (e.g., chatGPT), etc. Most NLP projects involve taking some raw textual data and wrangling it into a corpus of your own.

CrowdFlower, provider of a "data enrichment" platform for data scientists, conducted a survey of about 80 data scientists and found that data scientists spend –

- 60% of the time in organizing and cleaning data.
- 19% of the time is spent in collecting datasets.
- 9% of the time is spent in mining the data to draw patterns.
- 3% of the time is spent in training the datasets.
- 4% of the time is spent in refining the algorithms.
- 5% of the time is spent in other tasks.



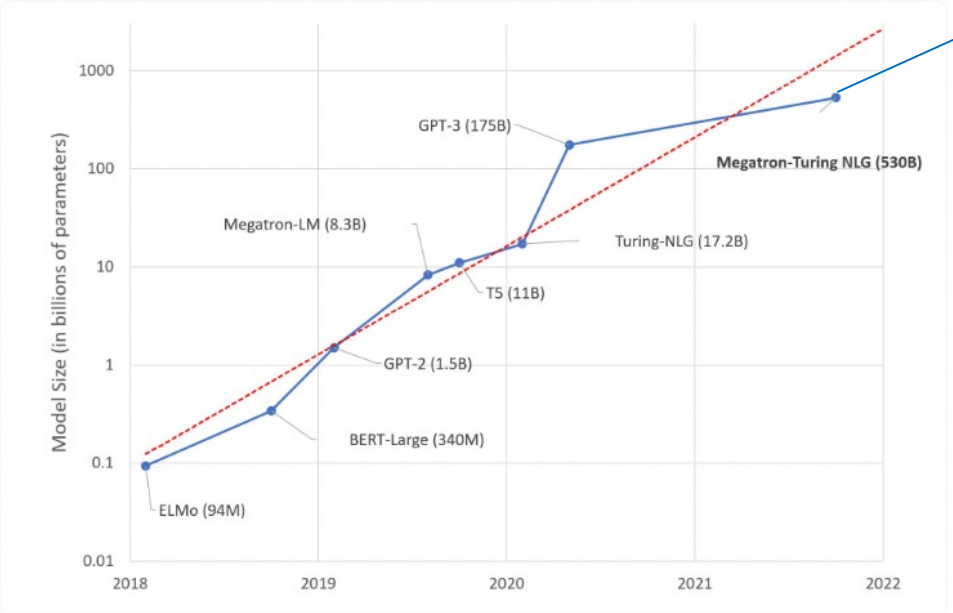
At the same time.....

Textual Data Preparation

ML Algorithms are growing exponentially!

GPT-4: 1.76 trillion parameters!

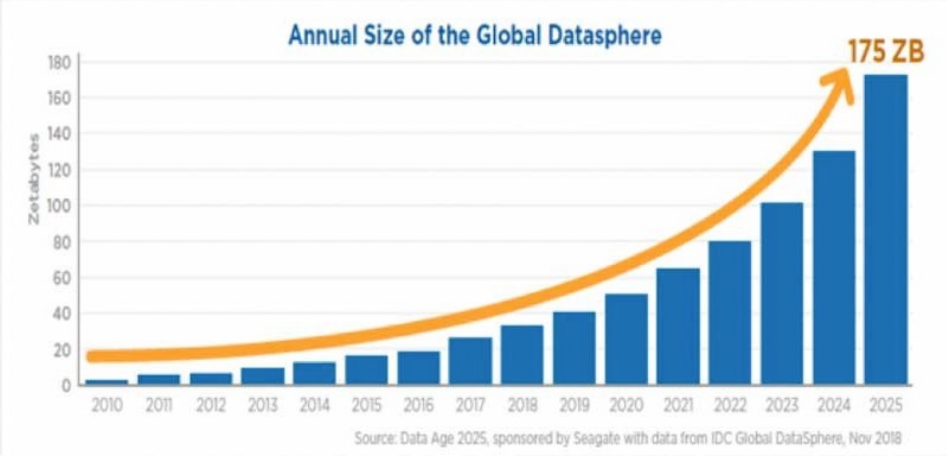
Log Scale

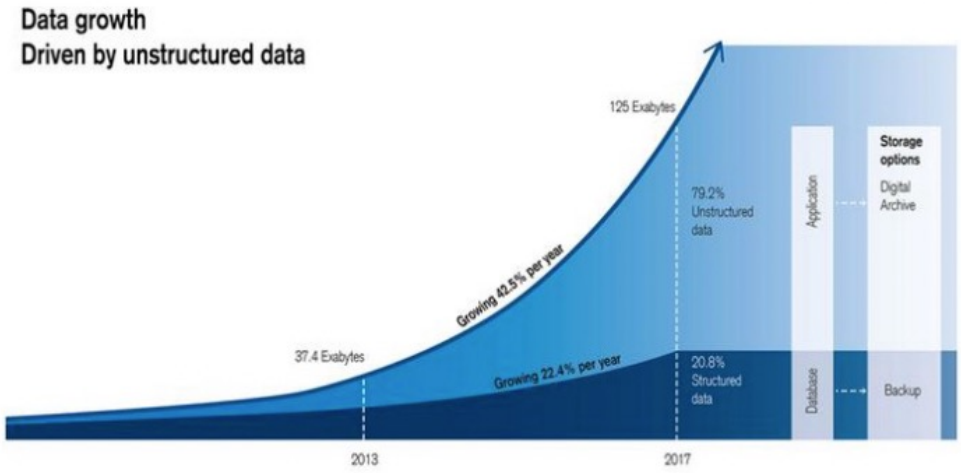


Basic rule: The more parameters, the more data you need... AND...

Textual Data Preparation

The amount of available data
is growing exponentially,
and it is mostly unstructured.
It is simply impossible to process
this tsunami of data by (human)
hand!





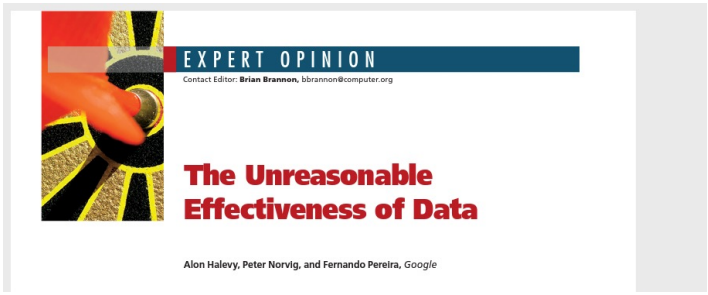
125 Exabytes of enterprise data was stored in 2017; 80% was unstructured data. (Source: Credit Suisse)

Examples of Data Volumes

	Unit	Value	Example
10^3	Kilobytes (KB)	1,000 bytes	a paragraph of a text document
10^6	Megabytes (MB)	1,000 Kilobytes	a small novel
10^9	Gigabytes (GB)	1,000 Megabytes	Beethoven's 5th Symphony
10^{12}	Terabytes (TB)	1,000 Gigabytes	all the X-rays in a large hospital
10^{15}	Petabytes (PB)	1,000 Terabytes	half the contents of all US academic research libraries
10^{18}	Exabytes (EB)	1,000 Petabytes	about one fifth of the words people have ever spoken
10^{21}	Zettabytes (ZB)	1,000 Exabytes	as much information as there are grains of sand on all the world's beaches
10^{24}	Yottabytes (YB)	1,000 Zettabytes	as much information as there are atoms in 7,000 human bodies

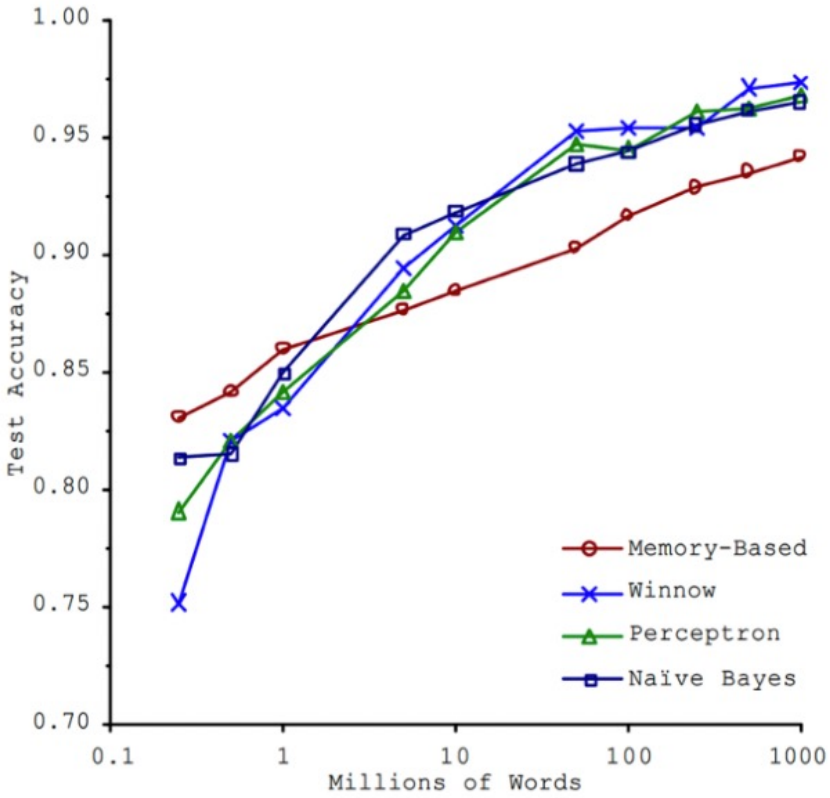
Textual Data Preparation

Plus, data is the most important resource – progress in NLP is overwhelmingly dependent on the amount of data, NOT refinements to the algorithms.



"...consider an experiment done by Microsoft in 2001. Researchers ran a side-by-side test to evaluate the merits of 4 of different approaches to ML translation. They trained each model from scratch with the same input data, running a series of trials with varying data set sizes from 100k to 1 billion words."

"... simple models and a lot of data trump more elaborate models based on less data....."



Textual Data Preparation

Most textual data preparation will follow some part of this flowchart:

Physical text

Scanning

PDF

OCR

Audio signal

Speech-to-text

Segmenting into sentences, paragraphs, etc.

E-document(s)

Format conversion

TXT

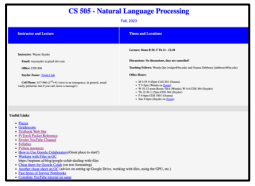
Text file

Sequence of tokens/words

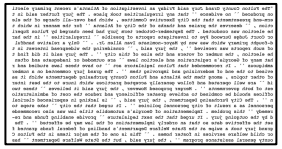
Scraping

De-noising and cleaning

Normalizing word formats: Stemming and Lemmatization



Web page(s)



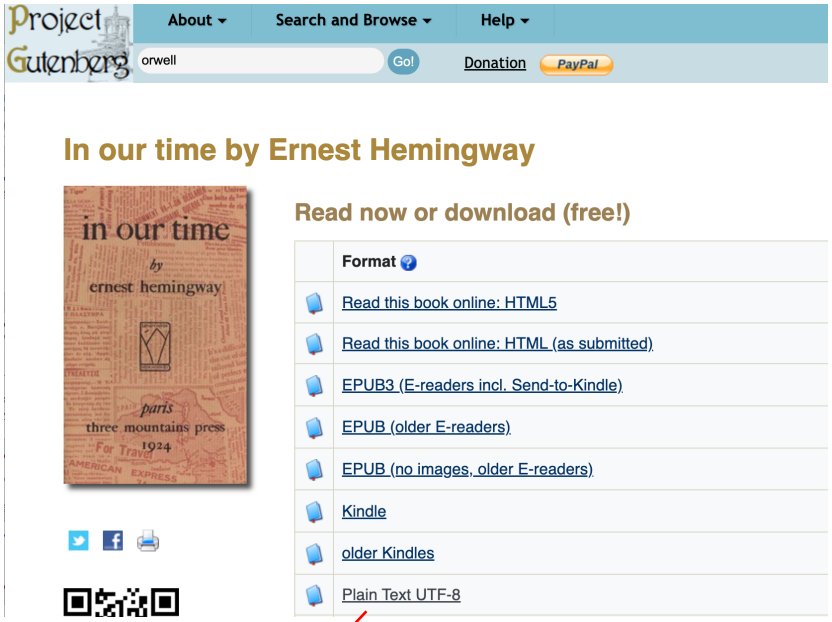
Sequence of characters

Tokenizing

Sequence of words

Tokenizing: Separating a string into words

Example




```
in our time



chapter 1


Everybody was drunk. The whole battery was drunk going along the road
in the dark. We were going to the Champagne. The lieutenant kept
riding his horse out into the fields and saying to him, “I’m drunk, I
tell you, mon vieux. Oh, I am so soused.” We went along the road all
night in the dark and the adjutant kept riding up alongside my
kitchen and saying, “You must put it out. It is dangerous. It will be
observed.” We were fifty kilometers from the front but the adjutant
worried about the fire in my kitchen. It was funny going along that
road. That was when I was a kitchen corporal.
```

Tokenizing: Separating a string into words

First try: separate on white space

```
in our time



chapter 1


Everybody was drunk. The whole battery was drunk going along the road
in the dark. We were going to the Champagne. The lieutenant kept
riding his horse out into the fields and saying to him, “I’m drunk, I
tell you, mon vieux. Oh, I am so soused.” We went along the road all
night in the dark and the adjutant kept riding up alongside my
kitchen and saying, “You must put it out. It is dangerous. It will be
observed.” We were fifty kilometers from the front but the adjutant
worried about the fire in my kitchen. It was funny going along that
road. That was when I was a kitchen corporal.
```

```
1  import re
2
3  separator_1 = '\s+'          # one or more white space characters
4
5  print(re.split(separator_1,text),'\n')
6
```

```
['in', 'our', 'time', 'chapter', '1', 'Everybody', 'was', 'drunk.', 'The', 'whole', 'battery', 'was', 'drunk', 'goin
g', 'along', 'the', 'road', 'in', 'the', 'dark.', 'We', 'were', 'going', 'to', 'the', 'Champagne.', 'The', 'lieutenan
t', 'kept', 'riding', 'his', 'horse', 'out', 'into', 'the', 'fields', 'and', 'saying', 'to', 'him,', '“I’m', 'drun
k,', 'I', 'tell', 'you,', 'mon', 'vieux.', 'Oh,', 'I', 'am', 'so', 'soused.”', 'We', 'went', 'along', 'the', 'road',
'all', 'night', 'in', 'the', 'dark', 'and', 'the', 'adjutant', 'kept', 'riding', 'up', 'alongside', 'my', 'kitchen',
'and', 'saying,', '“You', 'must', 'put', 'it', 'out.', 'It', 'is', 'dangerous.', 'It', 'will', 'be', 'observed.”', 'W
e', 'were', 'fifty', 'kilometers', 'from', 'the', 'front', 'but', 'the', 'adjutant', 'worried', 'about', 'the', 'fir
e', 'in', 'my', 'kitchen.', 'It', 'was', 'funny', 'going', 'along', 'that', 'road.', 'That', 'was', 'when', 'I', 'wa
s', 'a', 'kitchen', 'corporal.']
```

Tokenizing: Separating a string into words

First try: separate on white space

```
in our time



chapter 1


Everybody was drunk. The whole battery was drunk going along the road
in the dark. We were going to the Champagne. The lieutenant kept
riding his horse out into the fields and saying to him, “I’m drunk, I
tell you, mon vieux. Oh, I am so soused.” We went along the road all
night in the dark and the adjutant kept riding up alongside my
kitchen and saying, “You must put it out. It is dangerous. It will be
observed.” We were fifty kilometers from the front but the adjutant
worried about the fire in my kitchen. It was funny going along that
road. That was when I was a kitchen corporal.
```

```
1  import re
2
3  separator_1 = '\s+'          # one or more white space characters
4
5  print(re.split(separator_1,text),'\n')
6
```

```
['in', 'our', 'time', 'chapter', '1', 'Everybody', 'was', 'drunk.', 'The', 'whole', 'battery', 'was', 'drunk', 'goin
g', 'along', 'the', 'road', 'in', 'the', 'dark.', 'We', 'were', 'going', 'to', 'the', 'Champagne.', 'The', 'lieutenan
t', 'kept', 'riding', 'his', 'horse', 'out', 'into', 'the', 'fields', 'and', 'saying', 'to', 'him,', '“I’m', 'drun
k,', 'I', 'tell', 'you,', 'mon', 'vieux.', 'Oh,', 'I', 'am', 'so', 'soused.”', 'We', 'went', 'along', 'the', 'road',
'all', 'night', 'in', 'the', 'dark', 'and', 'the', 'adjutant', 'kept', 'riding', 'up', 'alongside', 'my', 'kitchen',
'and', 'saying,', '“You', 'must', 'put', 'it', 'out.', 'It', 'is', 'dangerous.', 'It', 'will', 'be', 'observed.”', 'W
e', 'were', 'fifty', 'kilometers', 'from', 'the', 'front', 'but', 'the', 'adjutant', 'worried', 'about', 'the', 'fir
e', 'in', 'my', 'kitchen.', 'It', 'was', 'funny', 'going', 'along', 'that', 'road.', 'That', 'was', 'when', 'I', 'wa
s', 'a', 'kitchen', 'corporal.']
```

Tokenizing: Separating a string into words

Second try: separate on white space and punctuation:

```
in our time



chapter 1


Everybody was drunk. The whole battery was drunk going along the road
in the dark. We were going to the Champagne. The lieutenant kept
riding his horse out into the fields and saying to him, “I’m drunk, I
tell you, mon vieux. Oh, I am so soused.” We went along the road all
night in the dark and the adjutant kept riding up alongside my
kitchen and saying, “You must put it out. It is dangerous. It will be
observed.” We were fifty kilometers from the front but the adjutant
worried about the fire in my kitchen. It was funny going along that
road. That was when I was a kitchen corporal.
```

```
1 separator_2 = '[\s,;:.!?]+'     # one or more white space or punctuation characters
2
3 print(re.split(separator_2,text),'\n')
```

```
['in', 'our', 'time', 'chapter', '1', 'Everybody', 'was', 'drunk', 'The', 'whole', 'battery', 'was', 'drunk', 'goin
g', 'along', 'the', 'road', 'in', 'the', 'dark', 'We', 'were', 'going', 'to', 'the', 'Champagne', 'The', 'lieutenan
t', 'kept', 'riding', 'his', 'horse', 'out', 'into', 'the', 'fields', 'and', 'saying', 'to', 'him', '“I’m', 'drunk',
'I', 'tell', 'you', 'mon', 'vieux', 'Oh', 'I', 'am', 'so', 'soused', '”', 'We', 'went', 'along', 'the', 'road', 'al
l', 'night', 'in', 'the', 'dark', 'and', 'the', 'adjutant', 'kept', 'riding', 'up', 'alongside', 'my', 'kitchen', 'an
d', 'saying', '“You', 'must', 'put', 'it', 'out', 'It', 'is', 'dangerous', 'It', 'will', 'be', 'observed', '”', 'We',
'were', 'fifty', 'kilometers', 'from', 'the', 'front', 'but', 'the', 'adjutant', 'worried', 'about', 'the', 'fire',
'in', 'my', 'kitchen', 'It', 'was', 'funny', 'going', 'along', 'that', 'road', 'That', 'was', 'when', 'I', 'was',
'a', 'kitchen', 'corporal', '']
```

Tokenizing: Separating a string into words

Third try: separate on anything other than a word character

```
in our time



chapter 1


Everybody was drunk. The whole battery was drunk going along the road
in the dark. We were going to the Champagne. The lieutenant kept
riding his horse out into the fields and saying to him, “I’m drunk, I
tell you, mon vieux. Oh, I am so soused.” We went along the road all
night in the dark and the adjutant kept riding up alongside my
kitchen and saying, “You must put it out. It is dangerous. It will be
observed.” We were fifty kilometers from the front but the adjutant
worried about the fire in my kitchen. It was funny going along that
road. That was when I was a kitchen corporal.
```

```
1 separator_3 = '\W+'          # one or more non-word characters
2
3 print(re.split(separator_3,text),'\n')
```

```
['in', 'our', 'time', 'chapter', '1', 'Everybody', 'was', 'drunk', 'The', 'whole', 'battery', 'was', 'drunk', 'goin
g', 'along', 'the', 'road', 'in', 'the', 'dark', 'We', 'were', 'going', 'to', 'the', 'Champagne', 'The', 'lieutenan
t', 'kept', 'riding', 'his', 'horse', 'out', 'into', 'the', 'fields', 'and', 'saying', 'to', 'him', 'I', 'm', 'drun
k', 'I', 'tell', 'you', 'mon', 'vieux', 'Oh', 'I', 'am', 'so', 'soused', 'We', 'went', 'along', 'the', 'road', 'all',
'night', 'in', 'the', 'dark', 'and', 'the', 'adjutant', 'kept', 'riding', 'up', 'alongside', 'my', 'kitchen', 'and',
'saying', 'You', 'must', 'put', 'it', 'out', 'It', 'is', 'dangerous', 'It', 'will', 'be', 'observed', 'We', 'were',
'fifty', 'kilometers', 'from', 'the', 'front', 'but', 'the', 'adjutant', 'worried', 'about', 'the', 'fire', 'in', 'm
y', 'kitchen', 'It', 'was', 'funny', 'going', 'along', 'that', 'road', 'That', 'was', 'when', 'I', 'was', 'a', 'kitch
en', 'corporal', '']
```

Tokenizing: Separating a string into words

But it gets worse!

- Can't just blindly remove punctuation or non-word characters:
  - m.p.h., Ph.D., AT&T, cap'n
  - prices ($45.55)
  - dates (01/02/06)
  - URLs (http://www.stanford.edu)
  - hashtags (#nlproc)
  - email addresses (someone@cs.colorado.edu)
- Clitic: a word that doesn't stand on its own
  - "are" in we're, French "je" in j'ai, "le" in l'honneur
- When should multiword expressions (MWE) be words?
  - New York, rock 'n' roll

Tokenizing: Separating a string into words

Even worse, many languages do not use consistently use spaces or punctuation to separate words, and the tokenization is quite complicated!

Chinese

莎拉波娃现在居住在美国东南部的佛罗里达。

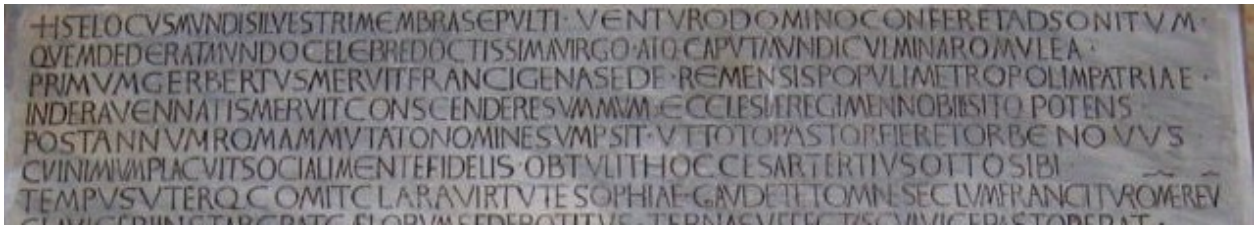
莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达

Sharapova now lives in US southeastern Florida

Classical Sanskrit

पश्यैतां पाण्डुपुत्राणामाचार्य महतीं चमूम् ।

Latin Inscriptions:



Tokenizing: Separating a string into words

In the case of English and many western languages, there are a variety of useful approaches:

- Rule-based methods, perhaps with list of exceptions
- Sequence-to-Sequence methods:
  - Hidden Markov Models;
  - Neural Networks

Tokenizing: Separating a string into words

Example: Tokenization in the Natural Language Toolkit (NLTK) uses rules based on regular expressions.

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)    # set flag to allow verbose regexps
...     ([A-Z]\.)+        # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*        # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?  # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.            # ellipsis
...   | [][.,;"'?():-_`]  # these are separate tokens; includes ], [
... '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

Tokenizing: Separating a string into words

Example: SpaCy uses a multi-phase approach based on rules and exceptions:

- First, the tokenizer split the text on whitespace similar to the split() function.
- Then the tokenizer checks whether the substring matches the tokenizer exception rules. For example, "don't" does not contain whitespace, but should be split into two tokens, "do" and "n't", while "U.K." should always remain one token.
- Next, it checks for a prefix, suffix, or infix in a substring, these include commas, periods, hyphens, or quotes. If it matches, the substring is split into two tokens.

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp(text)
tst = [ token.text for token in doc]

for k in range(0,len(tst),10):
  print(tst[k:k+10])
```

```
['in', 'our', 'time', '\n\n\n\n\n', 'chapter', '1', '\n\n\n', 'Everybody', 'was', 'drunk']
['.', 'The', 'whole', 'battery', 'was', 'drunk', 'going', 'along', 'the', 'road']
['in', 'the', 'dark', '.', 'We', 'were', 'going', 'to', 'the', 'Champagne']
['.', 'The', 'lieutenant', 'kept', 'riding', 'his', 'horse', 'out', 'into', 'the']
['fields', 'and', 'saying', 'to', 'him', ',', '"', 'I', "'m", 'drunk']
[',', 'I', 'tell', 'you', ',', 'mon', 'vieux', '.', 'Oh', ',']
['I', 'am', 'so', 'soused', '.', '"', 'We', 'went', 'along', 'the']
['road', 'all', 'night', 'in', 'the', 'dark', 'and', 'the', 'adjutant', 'kept']
['riding', 'up', 'alongside', 'my', 'kitchen', 'and', 'saying', ',', '"', 'You']
['must', 'put', 'it', 'out', '.', 'It', 'is', 'dangerous', '.', 'It']
['will', 'be', 'observed', '.', '"', 'We', 'were', 'fifty', 'kilometers', 'from']
['the', 'front', 'but', 'the', 'adjutant', 'worried', 'about', 'the', 'fire', 'in']
['my', 'kitchen', '.', 'It', 'was', 'funny', 'going', 'along', 'that', 'road']
['.', 'That', 'was', 'when', 'I', 'was', 'a', 'kitchen', 'corporal', '.']
```

+ Code + Text

Word Normalization; Stemming and Lemmatization

Normalization is putting words into a standard format

Simple: Make text case-insensitive by converting all to lower case

More complex:

- Misspellings: " In tge beginning…. "
- Abbreviations:

  PHD PhD Ph.D Ph.D.

  etc. &c

  US U.S. U.S.A.
- Hyphenation:

  lowercase lower-case
- Miscellaneous:

  uhhuh uh-huh

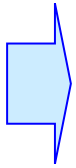
Word Normalization; Stemming and Lemmatization

Stemming: Remove suffixes

likes, liked, likely, liking, likable Stem: like

Naïve method: Chop off last part of work (based on list of cases)!

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.



Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note
.

Word Normalization; Stemming and Lemmatization

Stemming: Remove suffixes

likes, liked, likely, liking, likable Stem: like

Better: Multi-phase rule-based systems such as the Porter Stemmer (available in NLTK):

$$\begin{aligned} \text{ATIONAL} &\rightarrow \text{ATE} \quad \text{(e.g., relational} \rightarrow \text{relate)} \\ \text{ING} &\rightarrow \epsilon \quad \text{if stem contains vowel (e.g., motoring} \rightarrow \text{motor)} \\ \text{SSES} &\rightarrow \text{SS} \quad \text{(e.g., grasses} \rightarrow \text{grass)} \end{aligned}$$

Word Normalization; Stemming and Lemmatization

Lemmatization: Represent all words by their lemma, the shared root word (dictionary headword):

- *am, are, is* → *be*
- *car, cars, car's, cars'* → *car*
- Spanish quiero ('I want'), quieres ('you want')

→ querer 'want'

- *He is reading detective stories*

→ *He be read detective story*

Sentence Segmentation

Segmenting into sentences is based on punctuation.

!, ? Are mostly unambiguous but period "." is very ambiguous

- Usually "." is a sentence boundary. Especially followed by capital letter. But:
- Abbreviations like Inc. or Dr. Snyder
- Numbers like .02% or 4.3

Common algorithm: Tokenize first: use rules or ML to classify a period as either (a) part of the word or (b) a sentence-boundary. Capitalization can help too!

Sentence segmentation can then often be done by rules based on this tokenization.