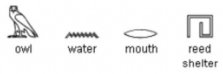
CAS CS 505
Lec 2

Basic Notions and Low-Level Text Normalization

1. Basic Notions: Characters, Words, Tokens, Documents, Corpora
    a. The human race has developed many different writing systems, based on several categories of graphemes (atmoic symbols). To vastly over-simplify, we have
        i. Alphabets: a set of < 100 symbols, each roughly corresponding to a speech sound:
            1. Example
                a. English: a b c … z
                b. Greek: alpha, beta, …
        ii. Syllabaries: a set of 100s of symbols, each roughly corresponding to spoken syllable
            1. Example

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

                a. Japanese: さか, *saka*, "hill"
        iii. Logographies: A set of 1000s of symbols, each roughly corresponding to a spoken word or concept
            1. Example

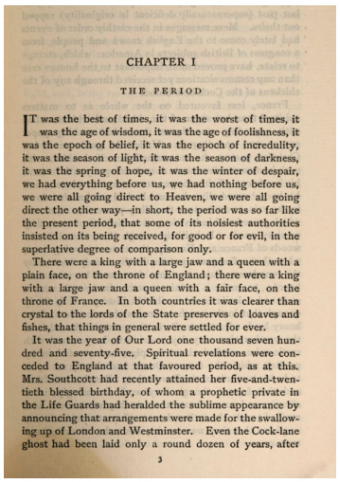
                Egyptian Hieroglyphs: 

                Chinese: 山 mountain. 刀 knife 下 down

                a.
    b. For NLP, we want to process text as a sequence of atomic symbols and these may be any of the preceding categories
    c. Thus: In NLP, textual data is presented in its most basic form as a sequence of atomic symbols from some finite collection (think Unicode!).
        i. Unicode is more modern version
    d. 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.
    e. 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.):

f.

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.

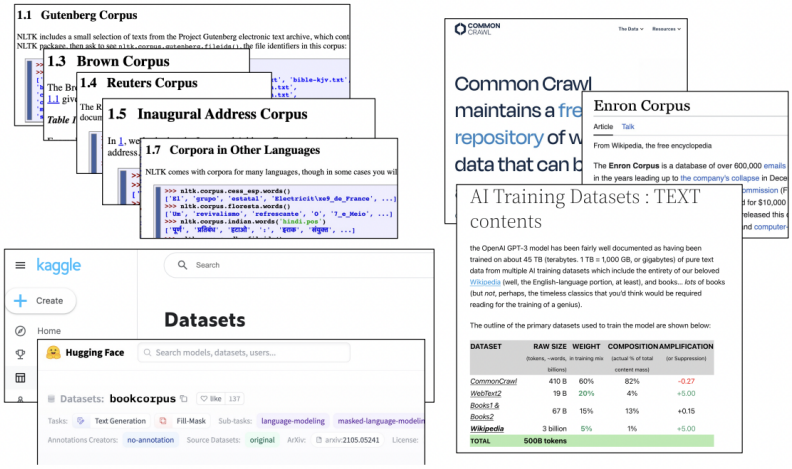
It was the year of Our Lord one thousand seven hundred and seventy-five. Spiritual revelations were conceded to England at that favoured period, as at this. Mrs. Southcott had recently attained her five-and-twentieth blessed birthday, of whom a prophetic private in the Life Guards had heralded the sublime 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

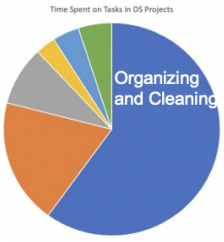
3

```
"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.  ..... "
```

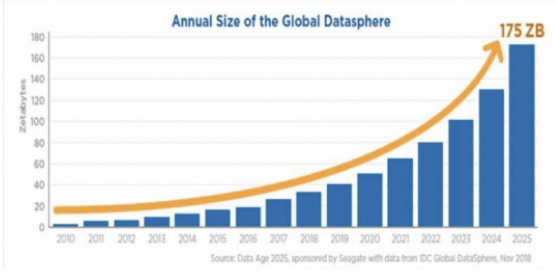
g. Although we will have occasion to use the string form when we study character-level machine learning models, almost all NLP uses data which has been grouped into larger units:
    i. Words: Sequence of characters, separated by white space or punctuation
    ii. Tokens: Words possibly preprocessed into some more useful form (the rest of this lecture)
    iii. Sentences: Sequences of words/tokens
    iv. Paragraphs: Sequences of sentences
    v. Chapters/Sections/Topics: Sequences of paragraphs
    vi. Document: Sequence of paragraphs
    vii. Corpus: Set of documents

2. Corpora for NLP
    a. 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….
    b.



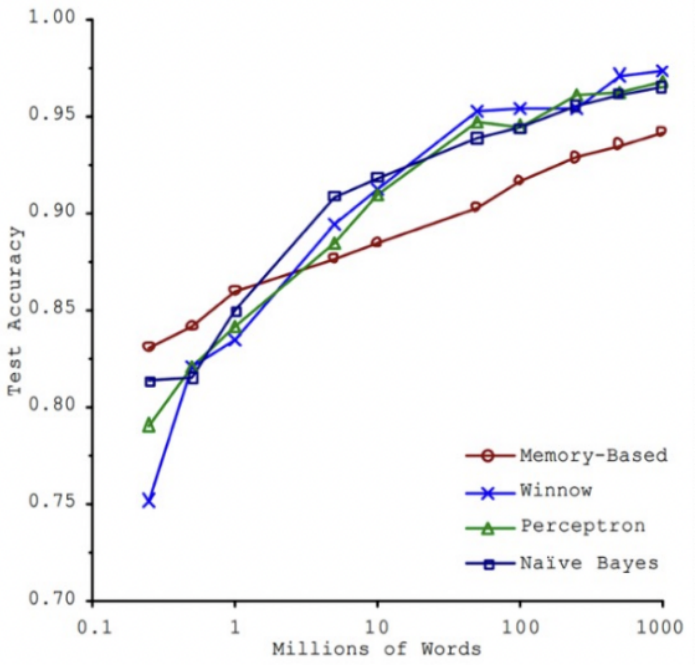
3. Textual Data Preparation
    a. Why do we need to learn low-level text processing?
        i. Because these corpora are for education, contests, creating general language models (e.g., chatGPT), etc. Most NLP projects involve taking some raw textual data and wrangling it into a corpus of your own.
    b. CrowdFlower, provider of a "data enrichment" platform for data scientists, conducted a survey of about 80 data scientists and found that data scientists spend
        i. 60% of the time in organizing and cleaning data
        ii. 19% of the time is spent in collecting datasets
        iii. 9% of the time is spent in mining the data to draw patterns
        iv. 3% of the time is spent in training the datasets
        v. 4% of the time is spent in refining the algorithms
        vi. 5% of the time is spent in other tasks
        vii. 

    c. ML algorithms are growing exponentially
        i. 

    d. Basic rule: the more parameters, the more data you need
    e. 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!
        i. 

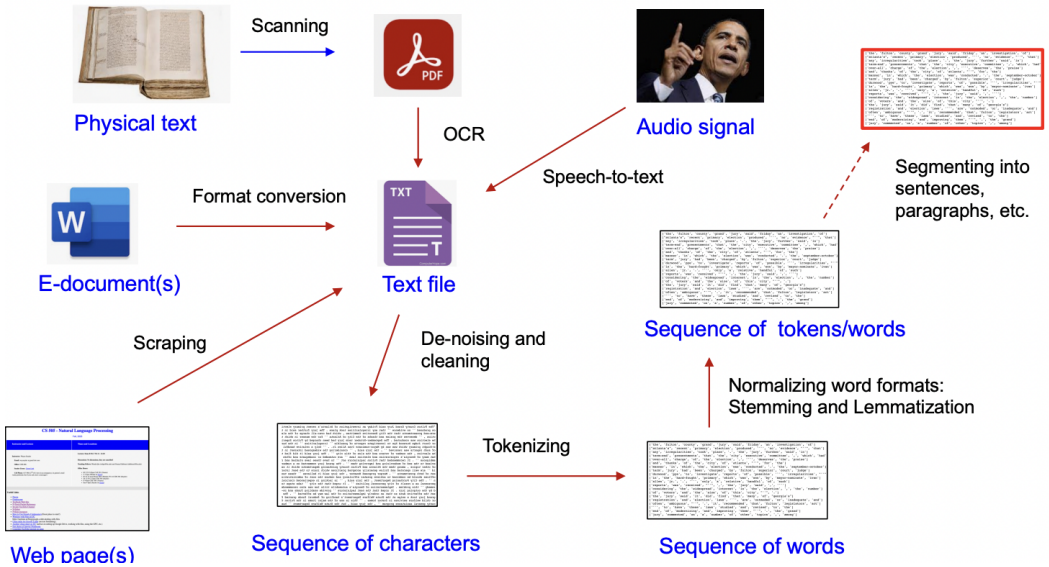
f. Plus, data is the most important resource – progress in NLP is overwhelmingly dependent on the amount of data, NOT refinements to the algorithms. (need an enormous amount of data → the more data you have, the better the solution)

i.



g. Most textual data preparation will follow some part of this flowchart

i.



4. Tokenizing: Separating a string into words
    a. First try: separate on white space

```
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.
```

```
import re

separator_1 = '\s+'          # one or more white space characters

print(re.split(separator_1,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."', '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.']
```

        i. Punctuations were included → period, comma, quotations, and etc.

    b. Second try: separate on white space and punctuation

```
separator_2 = '[\s,;:.!?]+'     # one or more white space or punctuation characters

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', '']
```

        i. Still included the quotation marks (better since there are no periods/commas)

    c. Third try: separate on anything other than a word character

```
separator_3 = '\W+'          # one or more non-word characters

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', '']
```

        i. Pretty clean except there are some errors → I'm becomes I & m, and etc.

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

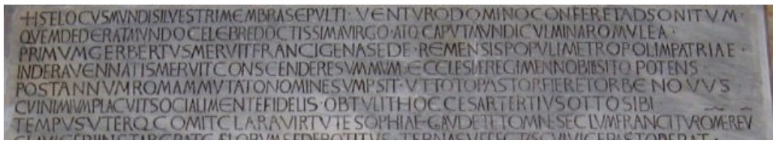
莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达

Sharapova now lives in US southeastern Florida

Classical Sanskrit

पश्यैतां पाण्डुपुत्राणामाचार्य महतीं चमूम् ।

Latin Inscriptions:



h. In the case of English and many western languages, there are a variety of useful approaches
    i. Rule-based methods, perhaps with list of exceptions
    ii. Sequence-to-Sequence methods
        1. Hidden Markov Models
        2. Neural Networks
i. Example: Tokenization in the 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', '...']
```

j. Example: SpaCy uses a multi-phase approach based on rules and exceptions
   i. First, the tokenizer split the text on whitespace similar to the split() function
   ii. Then, the tokenizer checks whether the substring matches the tokenizer exception rules
   iii. 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
   iv. 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
   v.

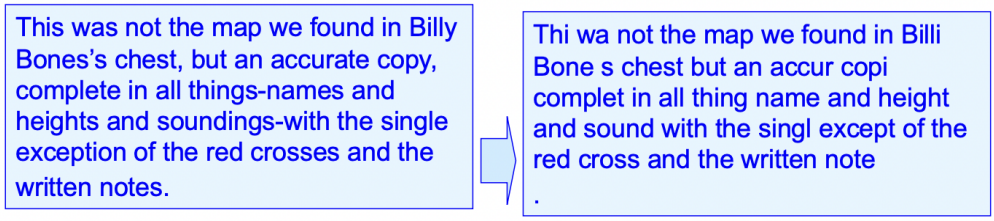
```
import spacy

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\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

5. Word Normalization; Stemming and Lemmatization
   a. Normalization is putting words into a standard format
   b. Simple: Make text case-insensitive by converting all to lower case
   c. More complex:
      i. Misspellings
         1. “ In tge beginning…. “
      ii. Abbreviations:
         1. PHD PhD Ph.D Ph.D.
         2. etc. &c
         3. US U.S. U.S.A.
      iii. Hyphenation:
         1. lowercase lower-case
      iv. Miscellaneous:
         1. uhhuh uh-huh

d. Stemming: remove suffixes
    i. Not always good
    ii. Useful when we do classification
    iii. Not useful with translation
e. likes, liked, likely, liking, likable Stem: like
f. Naïve method: Chop off last part of work (based on list of cases)!
    i. Pretty well: chop off last 20% of the word (first part of the word was enough to distinguish)



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

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

    i.
    ii. Does not always work but is good enough
h. 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*

    i.
    ii. Rules that amount to bunch of cases (better on classification)

6. Sentence Segmentation
    a. Segmenting into sentences is based on punctuation
    b. !, ? Are mostly unambiguous but period "." is very ambiguous
        i. Usually "." is a sentence boundary. Especially followed by capital letter.
        ii. But Abbreviations like Inc. or Dr. Snyder && Numbers like .02% or 4.3
    c. 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!
    d. Sentence segmentation can then often be done by rules based on this tokenization.