# Sentences and Words

* A **text** can be broken up into a sequence of sentences
  + A sentence is again a sequence of words
* A **language** (formal or natural) consists of
  + A vocabulary, a finite set of words
  + Syntactic rules for how the words can be combined into sentences
* In natural languages, the words also have an inner structure
* In linguistics, a word of a spoken language can be defined as **the smallest sequence of phonemes** that can be uttered in isolation with objective or practical meaning

# Types and Tokens

One cat caught five mice and three cats caught one mouse

* 11 **tokens**, i.e. word occurrences
* 9 **types**, i.e. 9 different words
* To get tokens, we can use .split() method
* To get types, we have to use set() function first

# Lexeme and Lemma

* **Lexeme**: An abstract unit of morphological analysis in linguistics, that roughly corresponds to a set of forms taken by a single word (several forms of one word can be found in one text, e.g. mouse, mice)
* **Lemma**: Canonical form, dictionary form, or citation form of a lexeme

# Part of Speech / Word Class / Lexical Category

Category of words with similar grammatical properties:

* **Syntactic**: occur in similar places, can replace each other
* **Semantic**: similar type of meaning (Jack-he-the man)
* **Morphological**: similar inflection, similar derivation patterns

# Universal POS Tag Set (NLTK)

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag** | **Meaning** | **Subcategory** | **English Examples** |
| ADJ | Adjective |  | New, good, high |
| ADP | Adposition (preposition) |  | On, of, at, with, by, into, under |
| ADV | Adverb |  | Really, already, still |
| CONJ | Conjunction |  | And, or, but, if, while |
| DET | Determiner |  | The, a, some, most |
| NOUN | Noun | Proper | Mary, John, Paris, France |
|  |  | Common | Girl, oy, house, foot |
| NUM | Numeral |  | Twenty-four, fourth, 1991 |
| PRT | Particle |  | At, on, out, over, per, that |
| PRON | Pronoun |  | He, their, her, its, my, I, us |
| VERB | Verb |  | Is, say, told, given, playing |
| . | Punctuation marks |  | ,.;:!?” |
| X | Other |  | Ersatz, esprit, dunno, gr8, univeristy |

# Subcategories

The POSs can have subcategories which differ in distribution, semantics, morphology

* Nouns
  + Proper nouns (names): Kim, Johnson, Africa, UiO
  + Common nouns: year, home, costs, time
  + Nouns may vary with respect to gender (Masculine, feminine, neuter)
* Pronouns
  + Personal: I, you, he, she
  + Possessive: my, your, her, his
* Verbs
  + Intransitive (do not take an object): to sleep
  + Transitive (take an object): to eat
  + Ditransitive (take direct and indirect object): to give

# Open and Closed Classes

* An **open class** accepts the addition of new words (N, V, Adj, Adv, Int)
* A **closed class** rarely accepts new words (Det, Pro, Prep, Conj., Subj.)

# Morphology

* = linguistic study of words
* **Morpheme**: smallest meaning-bearing unit
* Words are not simple atomic units but they have structure
* Inflection
  + Different forms of the same lexeme
* Word formation
  + Derivation: quick 🡪 quickly
  + Compounding: hjernehinnebetennelse, scatterplot
* Clitics:
  + not really words (he’s)
  + function morphologically as affixes, but syntactically as words
  + Two alternative approaches  
    One token: *Mary’s* is a form of Mary  
    Two tokens, noun + clitic, *Mary -s*

Changes in sounds and orthography

* Inflection and derivation is not always simple concatenation
* Sound changes are changes in orthography can occur (to model 🡪 modelled)

# Text Processing: First Steps

* A text in raw form is a sequence of characters
* First steps in processing texts:
  + Split text into sentences
  + Split sentences into words
* Beware, often we have to do some cleaning first, e.g. remove markup or consider character encoding

# Sentence Segmentation

* Sentences are natural units for many tasks: translation, various types of “understanding”, parsing, tagging
* What is a sentence?
  + Where should we split? There is mainly consensus, but there are some corner cases:

Is “:” a sentence boundary?  
Embedded sentences or direct speech  
Incomplete utterances, particularly in speech, SMS

* Example: Is colon a sentence-splitter?
  + Examples are split in ntlk.brown.sents() but not in nltk.sent\_tokenize()
* How?
  + Hand-written rules
  + Various types of machine learning

Supervised vs. unsupervised  
Alternative machine learners

* The problem
  + Split a text into sentences 🡪 split at .!?
  + But what about abbreviations?
  + Main steps in Kiss and Strunkt: Punkt (2006)
    - Unsupervised recognition of abbreviations: a language-independent model, train the model on text for the specific languages
    - Deciding split or not: recognize the abbreviations in the text, split after sentence boundary which is not part of abbreviations, new round of decisions whether to split or not after abbreviations

# Tokenization

* After sentence splitting one gets a string of characters, e.g. “For example, this isn’t a well-formed example.”
* We want to split into a list of words
* Results could be:

| For | example |,| this | is | n’t | a | well-formed | example |.|

Penn TreeBank-style (PTB)

| For example, | this | isn’t | a | well- | formed | example. |

English Resource Grammar-style (ERG)

|for | example | this | is | not | a | well-formed | example |

* Punctuation:
  + (1) Separate tokens, (2) part of words, (3) remove
* Isn’t, doesn’t etc.: (1) split, (2) keep, (3) normalize
* Multiword expressions: (2) one token (1),(3) one token per word
* Hyphens: when to split? How?
* Case folding (lowercasing) or not?
* In addition, there are special constructions like decimal numbers, urls, etc.

# How to tokenize

* The cheapest way in Python:
  + words = s.split()
* If we prefer “example” to “example.” We could proceed
  + clean\_words = [w.strip(“.,:;=!”) for w in words]
* To keep “.” As a separate token, you must be more refined
* In NLTK for English, we can use the word\_tokenize
  + words = nltk.word\_tokenize(s)

# nltk.word\_tokenize()

* Penn-treebank tokens (nearly)
* English – no language specific options
* Uses regular expressions
* Splits on whitespace, also for numbers

# Using NLTK

In [36]: raw='This item consists of several sentences. It should be illustrative'   
In [37]: sents = nltk.sent\_tokenize(raw)   
In [38]: for i in sents: print(i)   
This item consists of several sentences.   
It should be illustrative

In [39]: tokenized = [**nltk.word\_tokenize(s)** for s in sents]   
In [40]: tokenized   
Out[40]: [['This', 'item', 'consists', 'of', 'several', 'sentences', '.'],   
['It', 'should', 'be', 'illustrative']]

# Text Normalization

* Should we lower-case or not?
  + Depends on the application!
  + [[w.lower() for w in set] for sent in sentences]
  + For some applications, e.g. search, it is useful to unify the various forms of a lexeme
    - Mice-mouse, caught-catch
    - **Lemmatization**: uses a lexicon and tagging to find the corresponding lemma
    - **Stemming**: uses rules to remove suffixes and identify the root

# Ambiguity

* Is what makes NLP hard and fun
* POS:
  + Noun or verb: eats shoots and leaves
  + Verb or preposition: like
* Word sense: bank, file
* Structural: She saw a man with binoculars
* Sounds

# Tagged Corpora

* In a **tagged corpus**, the word occurrences are disambiguated with respect to parts of speech (and possibly subcat and form)
* Good data for training various machine learning tasks (tags make useful features)
* Explore the frequency and positions of tags – When does a determiner occur in front of a verb?
* Possible to explore the occurrences of the word with the tag

# Tagged Text and Tagging

* In **tagged text**, each token is assigned a part of speech-tag (POS)
* A **tagger** is a program which automatically ascribes tags to words in text
* From the context we are most often able to determine the tag

# Various POS tag sets

* A tagged text is tagged according to a fixed small set of tags
* There are various tag sets
* Brown Tagset: 87 tags, Penn-treebank tags: 35 + 9 punctuation tags, Universal POS Tagset: 12 tags