IN4080 – 2020 FALL

NATURAL LANGUAGE PROCESSING

Jan Tore Lønning

Words, text processing

Lecture 2, 24 Aug

Today

Natural language:

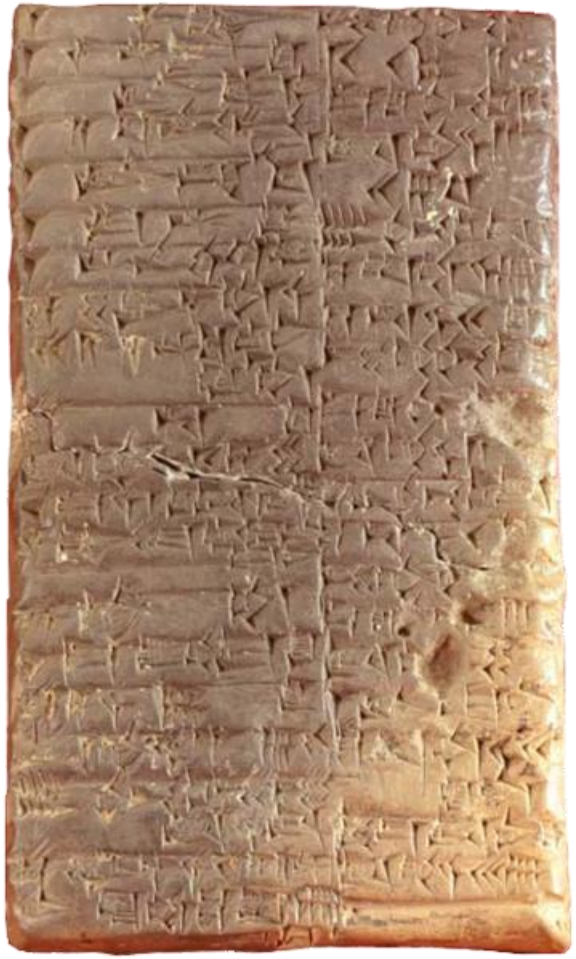
1. Words
2. Parts of speech
3. A little morphology

Processing – the first steps

4. Sentence splitting
5. Tokenization
6. Tagged text

(Natural) language

- Spoken vs written:
  - are not the same
- Writing is a fairly new invention
  - ~5000 years
  - Spoken 50-100,000 years
- Writing is (initially) a representation of spoken language



https://en.wikipedia.org/wiki/Language

Sentences and words

- A text can be broken up into a sequence of sentences.
  - A sentence is again a sequence of words.
    - The words may also have a structure.

- A language has a vocabulary, a finite set of words.
- We can produce and understand sentences we have not spoken/heard/read before if we know the words.

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. (wikipedia: Word)

Words: types and tokens

- One cat caught five mice and three cats caught one mouse
- How many words?

Words: types and tokens

- One cat caught five mice and three cats caught one mouse
- How many words?
  - 11 tokens, i.e., word occurrences
  - 9 types

Compare

- How many words did Shakespeare write ?
  - 884,647 (tokens)
- How many words did Shakespeare use?
  - 31,534 (types)

Words: types and tokens

- One cat caught five mice and three cats caught one mouse
- How many words?
  - 11 tokens, i.e., word occurrences
  - 9 types

```
In [79]: sent = "One cat caught five mice and three cats caught one mouse".split()
In [80]: len(sent)
Out[80]: 11
In [81]: len(set(sent))
Out[81]: 10
In [82]: len(set(w.lower() for w in sent))
Out[82]: 9
```

Lexeme and lemma

- One cat caught five mice and three cats caught one mouse
- How many words?
  - 11 tokens, i.e., word occurrences
  - 9 types
  - 7 lexemes

Lexeme	Lemma
one	
cat, cats	cat
caught	catch
five	
mouse, mice	mouse
three	
and	and

Lexeme and lemma

- A lexeme is an abstract unit of morphological analysis in linguistics, that roughly corresponds to a set of forms taken by a single word
- A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a lexeme
- (Beware that some use "lemma" where we use "lexeme".)

Norwegian example

One lexeme

4 different forms of the same lexeme

One lemma

mann	N, sg, indef
mannen	N, sg, def
menn	N, pl, indef
mennene	N, pl, def

Today

Natural language:

1. Words
2. Parts of speech
3. A little morphology

Processing – the first steps

4. Sentence splitting
5. Tokenization
6. Tagged text

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
  - Noun names a thing, person, place,…
  - Verb: activity, event, state,…
- Morphological:
  - Similar inflection
  - Similar derivation patterns

N	V	N
Cats	chase	mice

N	cats, girl, boy, elephant, ..
V	ate, saw, chase, give

Some parts of speech

	Category	Subcategory	Example
N	Noun	Common noun	girl, boy, house, foot, information, …
		Proper noun	Mary, John, Paris, France, …
V	Verb		run, see, give, say, understand, …
A	Adjective		nice, bad, green, fantastic, …
P	Preposition		to, from, on, under, of, to, …
Pro	Pronoun		I, you, me, they, …
Adv	Adverb		not, often, nicely, ….
Det	Determiner		a, the, some, every, all, …

More parts of speech

- Agreement regarding the previous 7 categories (or at least the first 6)
- There are more categories, but the exact number and division may vary
  - E.g., some distinguish between conjunction and subjunction, some don't
- Additional categories for Norwegian (from Norsk referensegrammatikk):
  - Interjeksjon: *ja, æsj, hurra, ..*
  - Konjunksjon: *og, eller, ..* *(and, or, …)*
  - Subjunksjon: *at, hvis, fordi, …* *(that, if, because, …)*

Example: Universal POS tag set (NLTK)

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Subcategories

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

- Nouns:
  - Proper nouns (names): Kim, Johnson, Africa, UiO, …
  - Common nouns: *year, home, costs, time*
  - Nouns may vary with respect to gender (Norw., German, French)
    - Masc.: mann, Mann, homme
    - Fem.: kvinne, Frau, femme
    - Neut.: hus, Haus
- Pronouns:
  - Personal: I, you, she, he, …
  - Possessive: my, yours, his, hers, …
- Verbs:
  - Intransitive: sleep
  - Transitive: eat
  - Ditransitive: give
- etc.

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.

Today

Natural language:

1. Words
2. Parts of speech
3. A little morphology

Processing – the first steps

4. Sentence splitting
5. Tokenization
6. Tagged text

Morphology (the linguistic study of words)

Words are not simple atomic units – they have structure

1. Inflection
   - Different forms of the same lexeme
2. Word formation
   A. Derivation
      - *quick* → *quickly*
   B. Compounding
      - *Hjernehinnebetennelse*
      - *Scatterplot*
3. Clitics – not really words

1. Inflection: Nouns

Noun			
Singular		Plural	
Indef	Definite	Indef.	Definite
gutt	gutten	gutter	guttene
jente	jenta	jenter	jentene
barn	barnet	barn	barna

Each line is a lexeme

Lemma = indefinite singular

Distinguish	
Abstract feature	Realization
Indef.+pl	-er, -, …
Def., sg, neut	- et
Def., sg, fem	- a
Def., pl, neut	-a, -ene

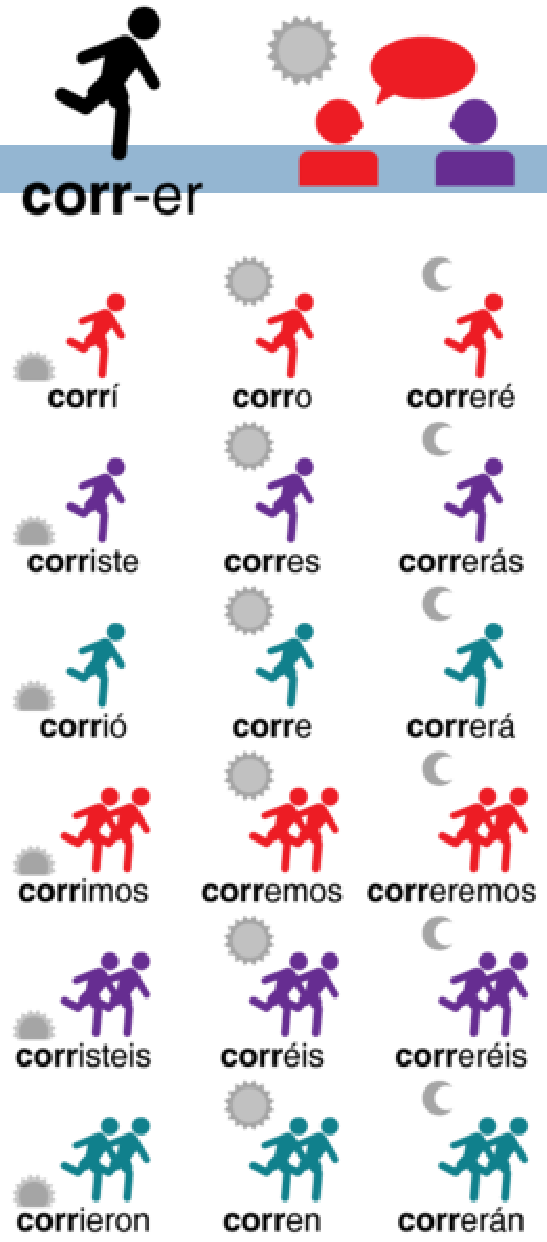
1b. Inflection: verbs

V, verb				
infinitiv	**presens**	**past**	**perfect**	**imperative**
kaste	kaster	kastet kasta	kastet kasta	kast
bygge	bygger	bygde bygget	bygd bygget	bygg
gå	går	gikk	gått	gå
English				
walk	walk/ walks	walked	walked	walk
run	run	ran	run	run

Example: Spanish (wikipedia)

Past – present – future

- Singular:
  - 1. pers
  - 2.pers
  - 3.pers
- Plural
  - 1. pers
  - 2.pers
  - 3.pers



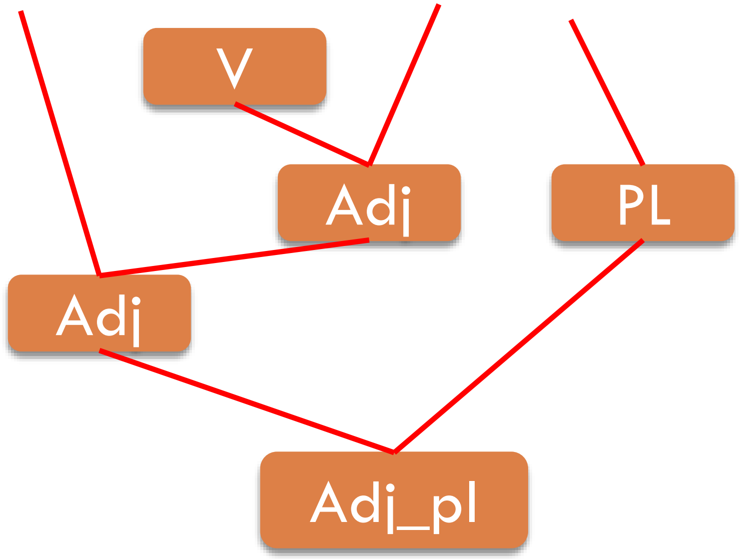
https://en.wikipedia.org/wiki/Grammatical_conjugation

2. Word formation

- Morpheme: smallest meaning-bearing unit
- Root: angripe
- Prefix: u-
- Suffix: -lig, -e
- Other languages: infix, circumfix

uangipelige (unassailable)

u+angripe+lig+e



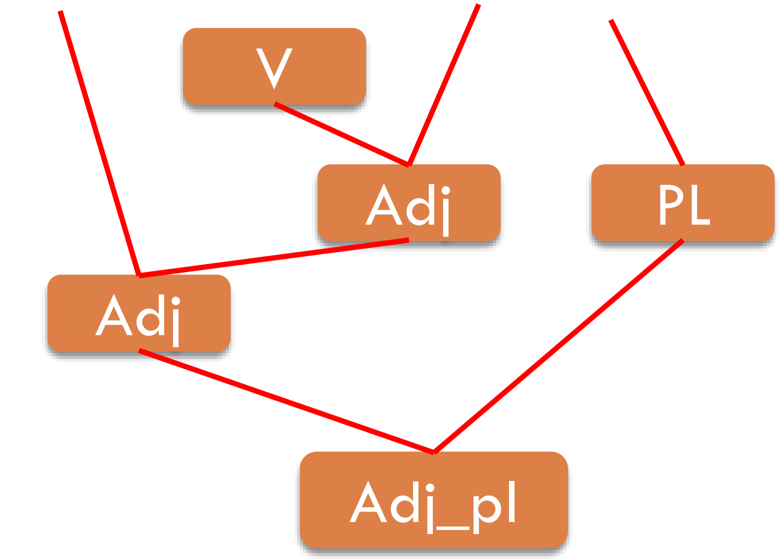
2 Word formation: derivation

- Combine a word stem with a grammatical morpheme
- Might result in a different POS

	Resulting word class			
Verb, infinite	Adjective	Noun	Noun	Noun
	-ende	-ing	-er	-
kaste	kastende	kasting	(en) kaster	(et) kast
throw	throwing	throwing	thrower	(a) throw

uangipelige (unassailable)

u+angripe+lig+e



Two derivations
followed by one inflection

2B. Word formation: Compounding

- A compound gets properties from the last part
  - *god*: Adj + *snakke*:V ➔ *godsnakke*: V
  - *fiske*: V + *konkurranse*: N ➔ *fiskekonkurranse*: N

4. Clitics

- Not full words
- Function morphologically as affixes, but syntactically as words
  - *Mary's car*
  - *I've done that*

- To alternative approaches to *Mary's car's* etc.:
  - One token: *Mary's* is a form of *Mary*
  - Two tokens, nouns + clitic, *Mary -s*

Changes in sounds and orthography

- Inflection and derivation is not always simple concatenation
- Sound changes/changes to orthography
  - *model*: V + -*ed*: past → *modelled* (or *modeled*)
  - *supply*: N + -*s*: pl → *supplies* (not *supplys*)
  - *calf*: N + -*s*: pl → *calves* (not *calfs*)
  - Etc.

Today

Natural language:

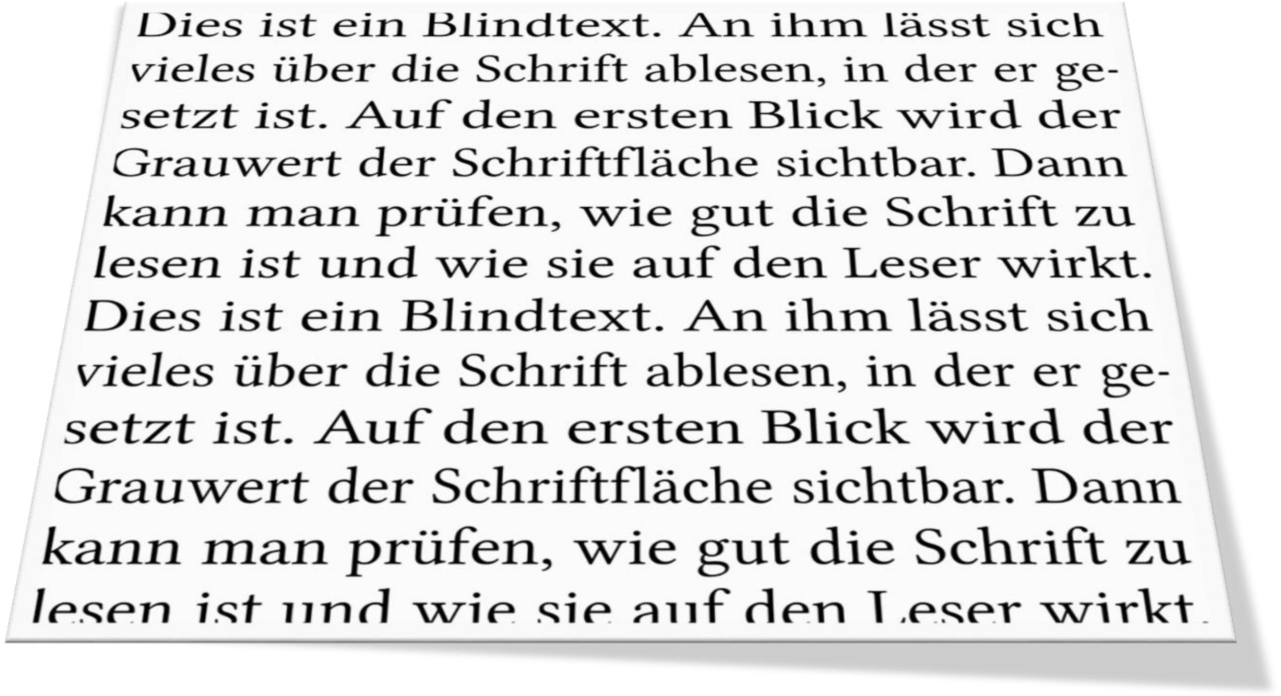
1. Words
2. Parts of speech
3. A little morphology

Processing – the first steps

4. Sentence splitting
5. Tokenization
6. Tagged text

Text processing: first steps

- A text in raw form is a sequence of characters
- Our first steps in processing it:
  1. Split the text into sentences
  2. Split the sentences into words
- Beware: often we have to do some cleaning first,
  - E.g. remove markup (html, xml,..)
  - Consider character encoding


Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt. Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt.

Sentence segmentation

- Why?
  - Sentences are natural units for many tasks:
    translation, various types of "understanding", parsing, tagging, etc.
- What is a sentence?
  - i.e., where should we (as humans split)?
  - There is mainly consensus, but there are some corner cases:
    - Is ':' a sentence boundary?
    - Embedded sentences, direct speech.
    - Incomplete utterances, particularly in speech, SMS, etc.

Question: Is colon a sentence-splitter?

- When is colon used: https://en.oxforddictionaries.com/punctuation/colon
- These examples are split in nltk.brown.sents()
- But nltk.sent_tokenize() will not split them
- Beware of these types of quirks for downstream tasks!

There are a number of ways this could happen, the churchmen pointed out, and here is an example:
Last month in Ghana an American missionary discovered when he came to pay his hotel bill that the usual rate had been doubled.

When he protested , the hotel owner said :
``Why do you worry?´´

Sentence segmentation

- How?
  - Hand-written rules
  - Various types of machine learning
    - Supervised or unsupervised
    - Alternative machine learners
- One example, Kiss and Strunk: Punkt (2006):
  - Uses unsupervised machine learning
  - Implemented as *nltk.sentence_tokenize()*.
  - Trained for various languages, including Norwegian.

The problem

- Split a text into sentences.
- ``How difficult could that be?”:
  - ``Split at: . ! ?” (and possibly ":")
- What about e.g. abbreviations?
  - ``Okay, not after abbreviations”
- What about abbreviations at the end of a sentence?
- This is the main problem according to K&S.

Punkt, main steps

- Unsupervised recognition of abbreviations:
  - A language-independent model
  - Train the model on text for the specific language
- Deciding split or not:
  - Recognize the abbreviations in the text
  - Split after sentence boundary (. ? !) which is not part of abbrevs.
  - New round with decisions whether to split or not after abbrevs.

Today

Natural language:

1. Words
2. Parts of speech
3. A little morphology

Processing – the first steps

4. Sentence splitting
5. Tokenization
6. Tagged text

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 it into (a list of) words
- What should the result be?
  1. |For|example|,|this|is|n’t|a|well-formed| example|.|
  2. |For example,|this|isn’t|a|well- |formed| example.|
  3. |for|example|this|is|not|a|well-formed|example|
     - (1) is Penn TreeBank-style (PTB)
     - (2) is English Resource Grammar-style (ERG)

Tokenization - alternatives

1. |For|example|,|this|is|n’t|a|well-formed| example|.|
2. |For example,|this|isn’t|a|well- |formed| example.|
3. |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)
  - How does this tokenize the ``for example”-sentence?

nltk.word_tokenize()

- Penn-treebank tokens (nearly)
- English - no language specific options
- Uses regular expressions
- Splits on white space, also for numbers
  - 500 000
  - Phone: 987 65 432
  - (Works for English:
    - 500,000
    - 987-65-432)

Example

1. s="It listed his wife's age as 74 and place of birth as Opelika , Ala."
2. ['It', 'listed', 'his', "wife's", 'age', 'as', '74', 'and', 'place', 'of', 'birth', 'as', 'Opelika', ',', 'Ala.', '.']
3. ['It', 'listed', 'his', 'wife', "'s", 'age', 'as', '74', 'and', 'place', 'of', 'birth', 'as', 'Opelika', ',', 'Ala', '.']

- (1) is a sentence from the Brown corpus
- It comes in a tokenized form as (2)
  - nltk.corpus.brown.sents()[36]
- But the result becomes (3) if we use
  - nltk.word_tokenize(s)

  on (1).
- Moral: Be conscious about the tools you use

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']]
```

Can use 'Norwegian' as parameter

Not optimal for Norwegian

Other tools

- There are several freely available tool kits for tokenization, etc.
- For example, spacy
- Beware, they may deliver slightly different results.

Text normalization

- Should we lower-case or not?
  - Depends on the application
  - [[w.lower() for w in sent] 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

Today

Natural language:

1. Words
2. Parts of speech
3. A little morphology

Processing – the first steps

4. Sentence splitting
5. Tokenization
6. Tagged text

Ambiguity…

- …is what makes natural language processing…
  - …hard/fun
- POS:
  - noun or verb: *eats shoots and leaves* (joke)
  - 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:
  - The 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, e.g.
  - How often is ``likes" used as a noun compared to 20 years ago?

Tagged text and tagging


```
[('They', 'PRP'), ('saw', 'VBD'), ('a', 'DT'), ('saw', 'NN'), ('.', '.')]
[('They', 'PRP'), ('like', 'VBP'), ('to', 'TO'), ('saw', 'VB'), ('.', '.')]
[('They', 'PRP'), ('saw', 'VBD'), ('a', 'DT'), ('log', 'NN')]
```

- In tagged text each token is assigned a "part of speech" (POS) tag
- A tagger is a program which automatically ascribes tags to words in text
  - We will return to how they work
- From the context we are (most often) able to determine the tag.
  - But some sentences are genuinely ambiguous and hence so are the tags.

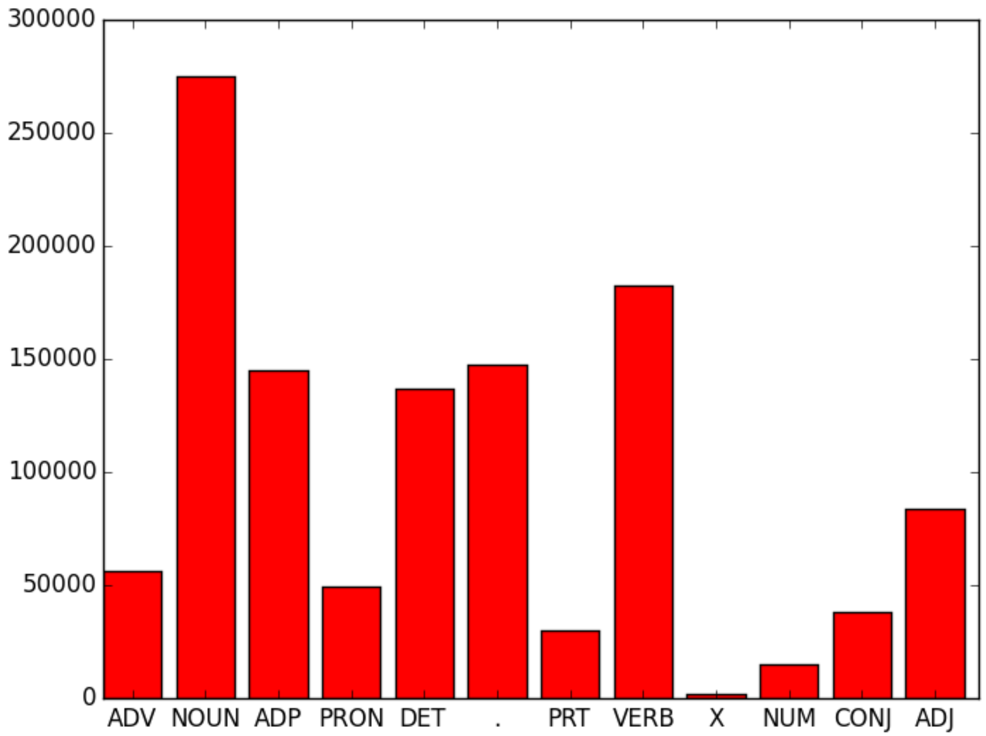
Various POS tag sets

- A tagged text is tagged according to a fixed small set of tags.
- There are various such tag sets.
- Brown tagset:
  - Original: 87 tags
  - Versions with extended tags <original>-<more>
    - Comes with the Brown corpus in NLTK
- Penn treebank tags: 35+9 punctuation tags
- Universal POS Tagset, 12 tags, (see NLTK book, web)

Universal POS tag set (NLTK)

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Distribution of universal POS in Brown



Cat	Freq
ADV	56 239
NOUN	275 244
ADP	144 766
NUM	14 874
DET	137 019
.	147 565
PRT	29 829
VERB	182 750
X	1 700
CONJ	38 151
PRON	49 334
ADJ	83 721

Brown vs. Penn: Nouns

Penn treebank

NN	Noun, sing. or mass	*llama*
NNS	Noun, plural	*llamas*
NNP	Proper noun, singular	*IBM*
NNPS	Proper noun, plural	*Carolinas*

Brown, original

NN	(common) singular or mass noun	time, world, work, school, family, door
NN$	possessive singular common noun	father's, year's, city's, earth's
NNS	plural common noun	years, people, things, children, problems
NNS$	possessive plural noun	children's, artist's parent's years'
NP	singular proper noun	Kennedy, England, Rachel, Congress
NP$	possessive singular proper noun	Plato's Faulkner's Viola's
NPS	plural proper noun	Americans Democrats Belgians Chinese Sox
NPS$	possessive plural proper noun	Yankees', Gershwins' Earthmen's
NR	adverbial noun	home, west, tomorrow, Friday, North,
NR$	possessive adverbial noun	today's, yesterday's, Sunday's, South's
NRS	plural adverbial noun	Sundays Fridays

Brown vs. Penn: Verb

VB	Verb, base form	*eat*
VBD	Verb, past tense	*ate*
VBG	Verb, gerund	*eating*
VBN	Verb, past participle	*eaten*
VBP	Verb, non-3sg pres	*eat*
VBZ	Verb, 3sg pres	*eats*

Penn treebank

VB	verb, base form	make, understand, try, determine, drop
VBD	verb, past tense	said, went, looked, brought, reached kept
VBG	verb, present participle, gerund	getting, writing, increasing
VBN	verb, past participle	made, given, found, called, required
VBZ	verb, 3rd singular present	says, follows, requires, transcends

Brown

Today

Natural language:

1. Words
2. Parts of speech
3. A little morphology

Processing – the first steps

4. Sentence splitting
5. Tokenization
6. Tagged text

```
sentences =
nltk.sent_tokenize(raw)

tokenized = [nltk.word_tokenize(s)
               for s in sents]

[[w.lower() for w in sent]
           for sent in tokenized]
```