Natural Language Processing - 2nd Semester
(2024-2025)
1038141

1.5 - Words, Corpora, Text Normalization

SAPIENZA
UNIVERSITÀ DI ROMA

Prof. Stefano Faralli
faralli@di.uniroma1.it

Prof. Iacopo Masi
masi@di.uniroma1.it

**credits are reported in the last slide

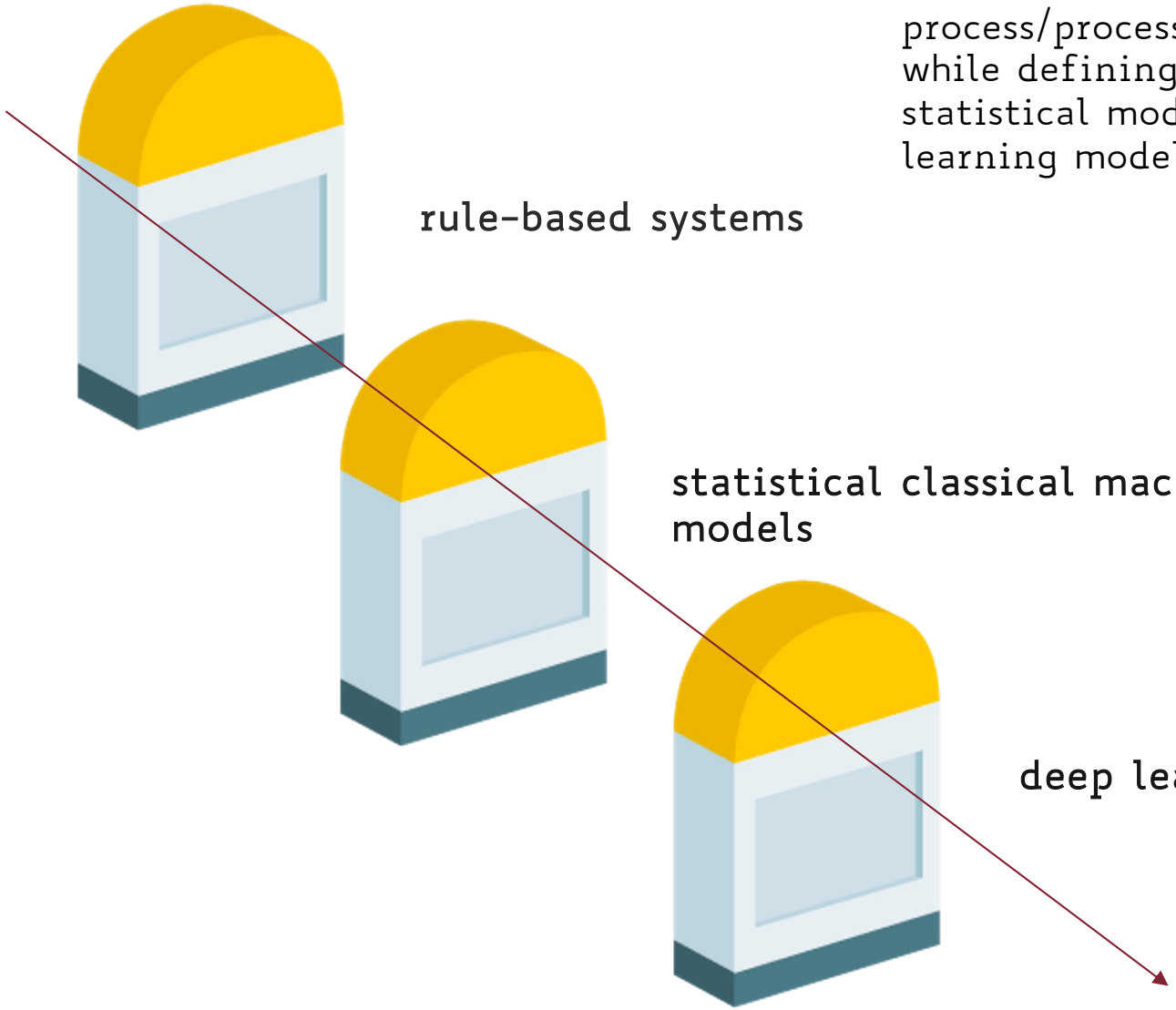
SAPIENZA
UNIVERSITÀ DI ROMA

1.5 - Words, Corpora and Text Normalization

- words
- corpora
- what counts as a word?
- lemmas and wordforms
- word tokens vs. word types
- text normalization: tokenization, lemmatization, segmentation
- Q&A

Milestones in NLP

today topics apply to the entire timeline and are related to how to pre-process/process text
while defining: a rule-based system, a statistical model or a machine/deep learning model.



Words, Corpus

Definition: corpora

corpus (plural corpora), a computer-readable collection of text or speech.

Definition: utterance

an utterance is the spoken correlate of a sentence:

consider the following sentence:
"I do mainly business data processing"
and the following utterance:
"I do uh main- mainly business data processing"

[Jurafsky&Martin, 2022]

Words, Corpus

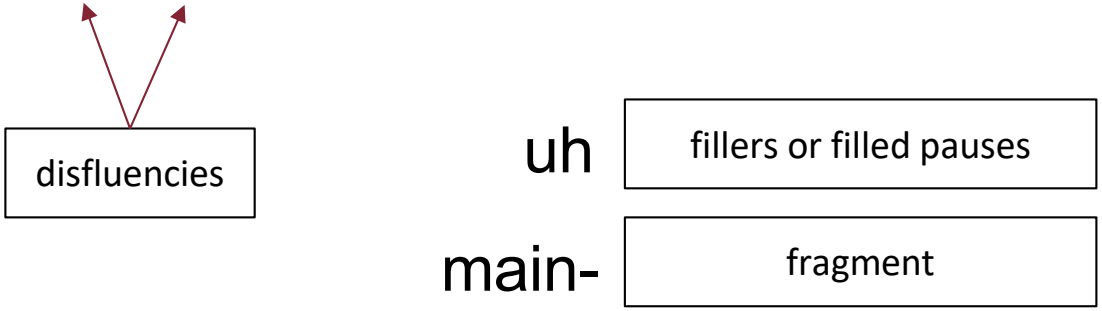
Definition: corpora

corpus (plural corpora), a computer-readable collection of text or speech.

Definition: utterance

an utterance is the spoken correlate of a sentence:

consider the following sentence:
"I do mainly business data processing"
and the following corresponding utterance:
"I do uh main- mainly business data processing"



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

[**Jurafsky&Martin, 2022**]

Words, Corpus

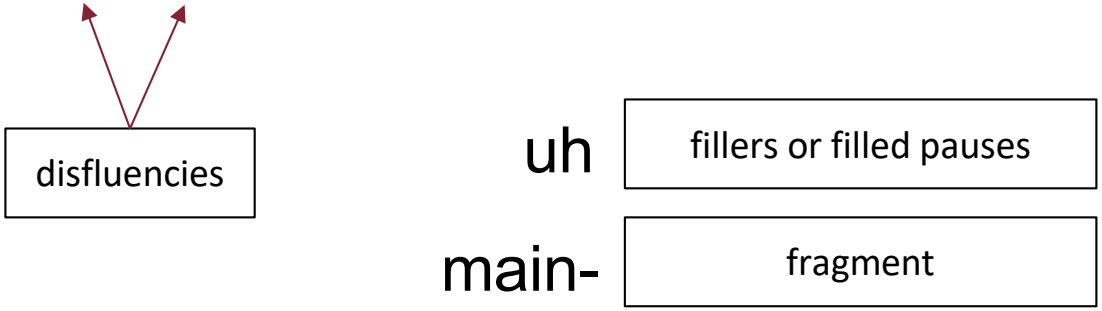
Definition: corpora

corpus (plural corpora), a computer-readable collection of text or speech.

Definition: utterance

an utterance is the spoken correlate of a sentence:

consider the following sentence:
"I do mainly business data processing"
and the following corresponding utterance:
"I do uh main- mainly business data processing"



What counts as a word?

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

What counts as a word?

the answer is:
it depends on the application

What counts as a word?

What counts as a word?

How many words are in the following sentence?

"He stepped out into the hall, was delighted to encounter a water brother."

What counts as a word?

How many words are in the following sentence?

"He stepped out into the hall, was delighted to encounter a water brother."

13 words if we don't count punctuation marks as words, 15 if we count punctuation. Whether we treat period ("."), comma (","), and so on as words **depends on the task.**

[Jurafsky&Martin, 2022]

What counts as a word?

Are capitalized tokens like “He” and uncapitalized tokens like “he” different?

[**Jurafsky&Martin, 2022**]

What counts as a word?

Are capitalized tokens like "He" and uncapitalized tokens like "he" different?

it depends on the task.

[Jurafsky&Martin, 2022]

Lemma vs. wordforms

In English "cat" and "cats" are inflected forms, sharing the same lemma "cat".

[Jurafsky&Martin, 2022]

Lemma vs. wordforms

In English "cat" and "cats" are inflected forms, sharing the same lemma "cat".

definition

A **lemma** is a set of lexical forms having the same **stem**, the same major **part-of-speech**, and the same **word sense**.

definition

A **wordform** is a full inflected or derived form of a word.

word tokens vs. word types

definition

Word Types is the set of the distinct words in a corpus. Let **V** be the vocabulary of corpus, **|V|** is the number of types.

definition

Word Tokens is the collection of running words. We indicate with **N** the number of tokens.

word tokens vs. word types

In the following sentence:

"They picnicked by the pool, then lay back on the grass and looked at the stars."

How many word tokens?

How many word types?

word tokens vs. word types

In the following sentence:

"They picnicked by the pool, then lay back on the grass and looked at the stars."

if we ignore punctuation, we can count 16 tokens and 14 types.

text normalization

Before almost any natural language processing of a text, the text has to be normalized. At least three tasks are commonly applied as part of any normalization process:

1. **Tokenizing** (segmenting) **words**
2. **Normalizing word formats**
3. **Segmenting sentences**

tokenizing

do you remember the following definition?

Word Tokens is the collection of running words. We indicate with N the number of tokens.

definition

Tokenization is the process of emitting the word tokens of a given corpus.

tokenization: issues

- **Finland's capital** => **Finland**? **Finlands**? **Finland's**?
- **Hewlett-Packard** => **Hewlett** and **Packard** as two tokens?
  - State of the art: break up hyphenated sequence.
  - **co-education**
  - **lowercase**, **lower-case**, **lower case** ?
  - It can be effective to get the user to **put in** possible hyphens
- **San Francisco**: one token or two?
  - How do you decide it is one token?
- **what're -> what are**
- **we're -> we are**
- [French] ***L'ensemble*** → one token or two?
- [German] ***Lebensversicherungsgesellschaftsangestellter***

  'life insurance company employee'

multiword expressions dictionary, named entity recognition task

expansion of clitic contractions

compounds

[Jurafsky&Martin, 2022]

The Penn Treebank tokenization, standard

This standard separates out clitics (doesn't becomes does plus n't), keeps hyphenated words together, and separates out all punctuation.

example:

Input: "The San Francisco-based restaurant," they said, "doesn't charge $10".

Output: "␣The␣San␣Francisco-based␣restaurant␣,␣"␣they␣said␣,␣"␣does␣n't␣charge␣$␣10␣"␣.|

Regular expressions-based implementation

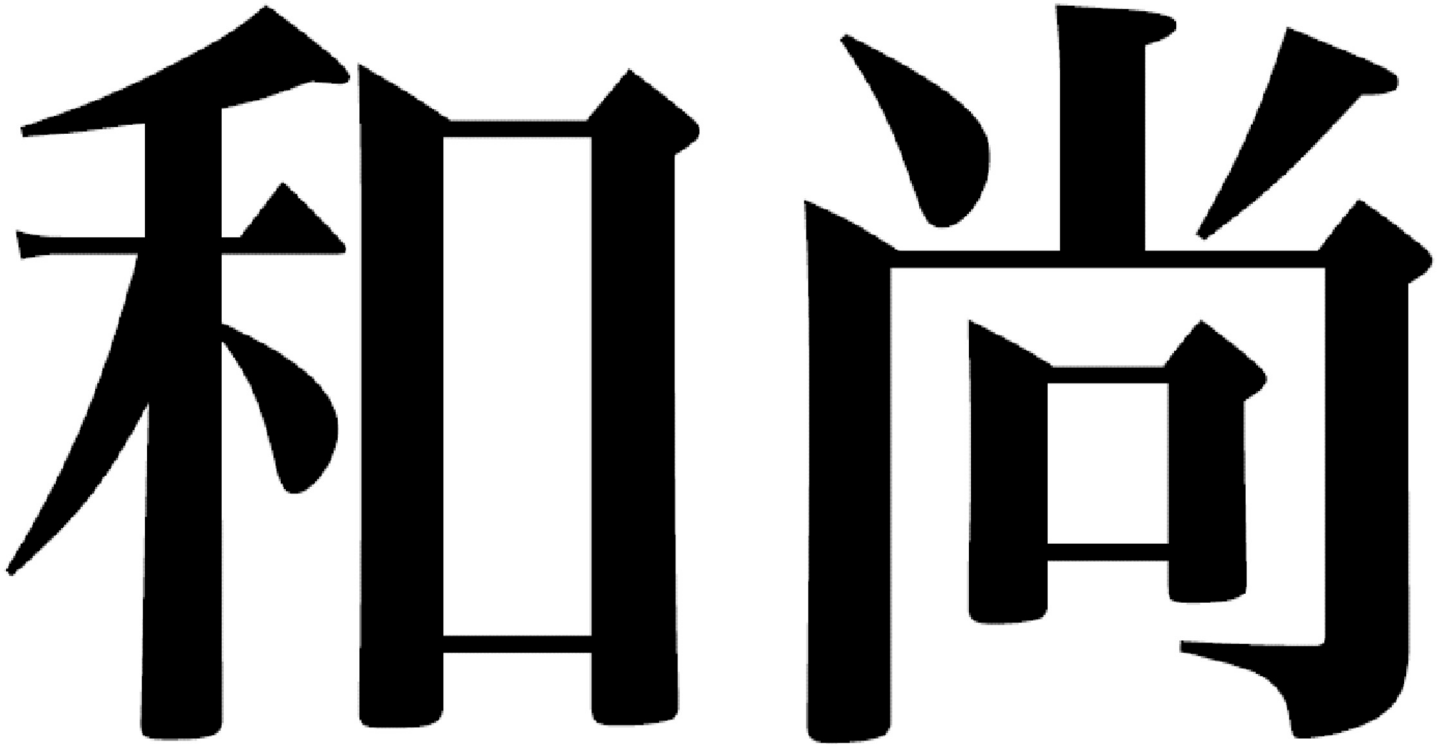
Fast deterministic algorithms.

```
>>> 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', '...']
```

https://www.nltk.org/

[Jurafsky&Martin, 2022]
[Bird et. al 2009]

Additional tokenization issues

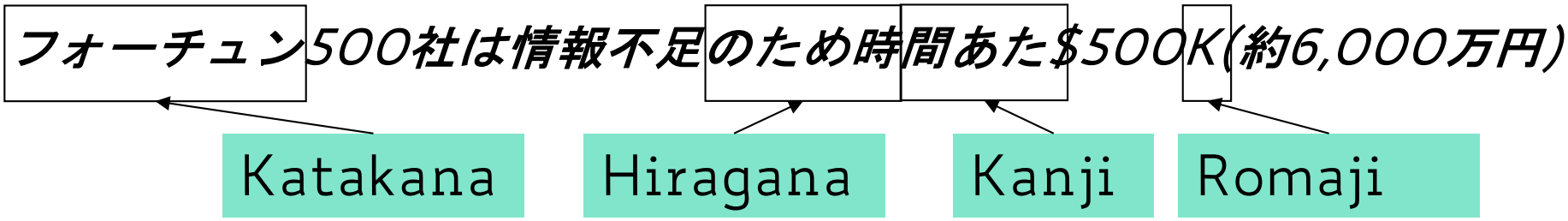


- The two characters can be treated as one word meaning ‘monk’ or as a sequence of two words meaning ‘and’ and ‘still’.

[Jurafsky&Martin, 2022]

Additional tokenization issues

- Chinese and Japanese have no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - Not always guaranteed a unique tokenization
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats



[Jurafsky&Martin, 2022]

Byte-pair encoding for tokenization

Used in the GPT pipeline

Token Learner

Token Parser

[Jurafsky&Martin, 2022]
[Sennrich et al. 2016]

Byte-pair encoding for tokenization: token learner

function BYTE-PAIR ENCODING(strings C, number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters
for $i = 1$ **to** k **do** # merge tokens til k times
  $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C
  $t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating
  $V \leftarrow V + t_{NEW}$ # update the vocabulary
  Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus
return V

1. begins with a vocabulary that is just the set of all individual characters.
2. examines the training corpus, chooses the two symbols that are most frequently adjacent (say 'A', 'B'), adds a new merged symbol 'AB' to the vocabulary, and replaces every adjacent 'A' 'B' in the corpus with the new 'AB'.
3. It continues to count and merge, creating new longer and longer character strings, until k merges have been done creating k novel tokens; k is thus is a parameter of the algorithm.
4. The resulting vocabulary consists of the original set of characters plus k new symbols.

[Jurafsky&Martin, 2022]
[Sennrich et al. 2016]

Byte-pair encoding for tokenization: token learner

corpus

```
5   l o w _
2   l o w e s t _
6   n e w e r _
3   w i d e r _
2   n e w _
```

vocabulary

```
_, d, e, i, l, n, o, r, s, t, w
```


- splits words by punctuation and counts the word token occurrences.
- input corpus of 18 word tokens with counts for each word (the word *low* appears 5 times, the word *newer* 6 times, and so on)
- the starting vocabulary consists of 11 letters.

[Jurafsky&Martin, 2022]
[Sennrich et al. 2016]

Byte-pair encoding for tokenization: token learner

corpus

5 l o w _
2 l o w e s t _
6 n e w e r _
3 w i d e r _
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w

most frequent pair of symbols is **e r** ; (9 occurrences)

- **e r** are merged; **we save the rule (e r -> er)**
- the **e r** occurrences are replaced in the corpus with **er;**
- the **er** symbol is then added to the vocabulary;

corpus

5 l o w _
2 l o w e s t _
6 n e w er _
3 w i d er _
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er

[Jurafsky&Martin, 2022]
[Sennrich et al. 2016]

Byte-pair encoding for tokenization: token learner

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er
2 l o w e s t _	
6 n e w er _	
3 w i d er _	
2 n e w _	

- the algorithm now pick **er _** (9 occurrences); we save the rule (er _ -> er_)
- the **er _** occurrences are merged and replaced in the corpus with **er_**;
- the **er_** symbol is then added to the vocabulary;

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2 l o w e s t _	
6 n e w er_	
3 w i d er_	
2 n e w _	

[Jurafsky&Martin, 2022]
[Sennrich et al. 2016]

Byte-pair encoding for tokenization: token learner

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2 l o w e s t _	
6 n e w er_	
3 w i d er_	
2 n e w _	

- the algorithm now pick **n e** (8 occurrences); we save the rule (n e -> ne)
- the **n e** occurrences are merged and replaced in the corpus with **ne**;
- the **ne** symbol is then added to the vocabulary;

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne
2 l o w e s t _	
6 ne w er_	
3 w i d er_	
2 ne w _	

[Jurafsky&Martin, 2022]
[Sennrich et al. 2016]

Byte-pair encoding for tokenization: token parser

The token parser applies the same replacements learned on training data to the test corpus.

- `e r -> er`
- `er _ -> er_`
- `n e -> ne`
- `ne w-> new`
- `l o -> lo`
- `lo w -> low`
- `new er_ -> newer_`
- `low _ -> low_`

Eventually, only rare tokens will be represented as parts.

[Jurafsky&Martin, 2022]
[Sennrich et al. 2016]

Word Normalization

definition

Word normalization is the task of putting words/tokens in a standard format, choosing a single normal form for words with multiple forms like *USA* and *US* or *uh-huh* and *uhhuh*.

definition

Case folding is a form of normalization. Mapping everything to **lower case** means that *Woodchuck* and *woodchuck* are represented identically, which is very helpful for generalization in many tasks, such as information retrieval or speech recognition. For sentiment analysis and other text classification tasks, information extraction, and machine translation, by contrast, case can be quite helpful and case folding is generally not done.

[Jurafsky&Martin, 2022]
[Sennrich et al. 2016]

Word Normalization

definition

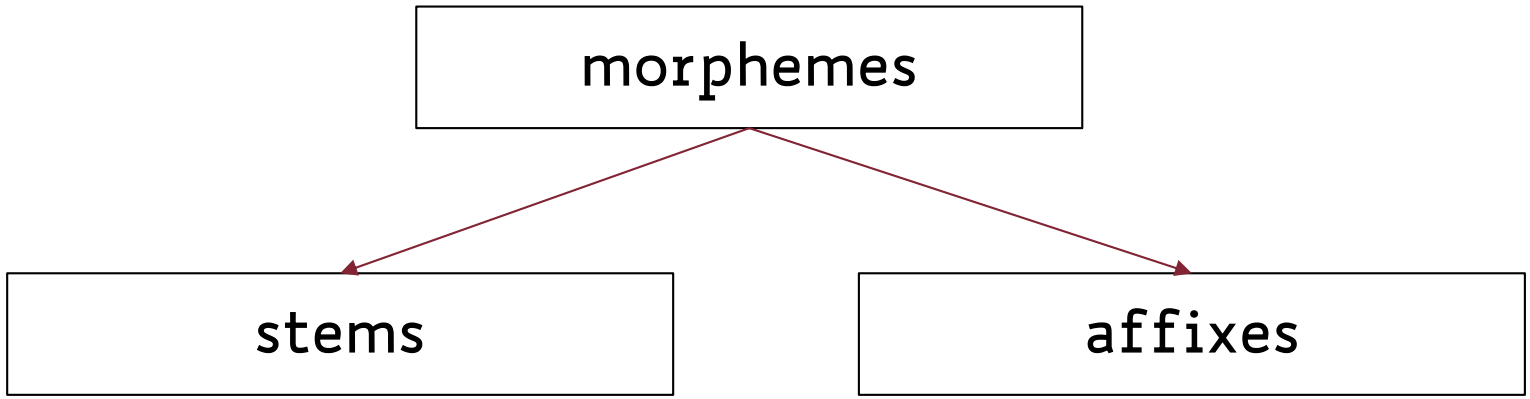
Lemmatization is the task of determining that two words have the same root,
despite their surface differences. The words *am*, *are*, and *is* have the shared lemma *be*, the words *dinner* and *dinners* both have the lemma *dinner*.

The lemmatized form of a sentence like:
"He is reading detective stories."
would thus be:
"He be read detective story."

[Jurafsky&Martin, 2022]

Word Normalization

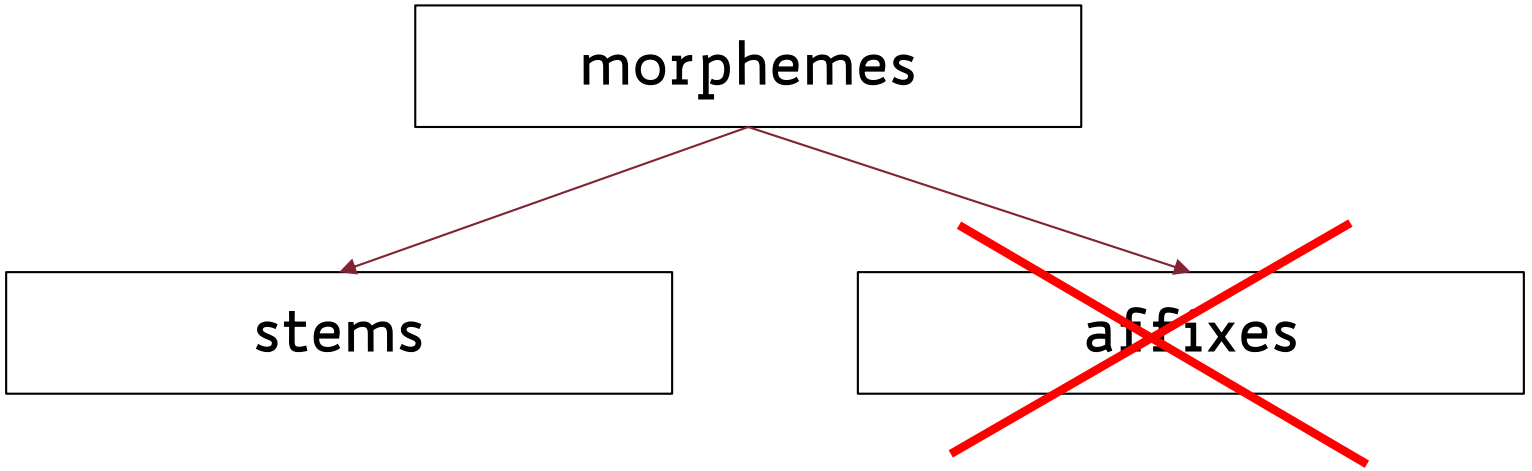
Lemmatization is done by means of **morphological parsing.**
In morphology, words are seen as built up from smaller **meaning-bearing units** called **morphemes.**



the word *fox* is built up from one morpheme i.e., *fox*
but the word *cats* is built up from the morpheme *cat* and the morpheme *-s*
A morphological parser splits words into morphemes.

[Jurafsky&Martin, 2022]

Porter Stemmer



A brutal way of performing lemmatization consist of removing all the affix morphemes from the tokens. Lemmatization is here performed as the art of **stemming**, hence as the process of emitting words' stem.

[Jurafsky&Martin, 2022]
[Porter 1980]

Porter Stemmer

The Porter stemmer applied to the following paragraph:

"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."

produces the following stemmed output:

"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"

https://textanalysisonline.com/nltk-porter-stemmer

[Jurafsky&Martin, 2022]
[Porter 1980]

Porter Stemmer: cascade

The algorithm is based on series of rewrite rules run in series, as a cascade, in which the output of each pass is fed as input to the next pass; here is a sampling of the rules:

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

More details in the original paper **[Porter 1980]**;

Implementations, full list of cascade rules and more can be found:

https://tartarus.org/martin/

Sentence Segmentation

definition

Sentence Segmentation is the task of identifying sentences' boundaries from text.

let's try together the NLT sentence segmentation implementation:

https://textanalysisonline.com/nltk-sentence-segmentation

[Jurafsky&Martin, 2022]

Sentence Segmentation

The most useful cues for segmenting a text into sentences are **punctuation**, like **periods**, **question marks**, and **exclamation points.**

Question marks and exclamation points are relatively unambiguous markers of sentence boundaries.

Periods, on the other hand, are more ambiguous.

The period character "." is ambiguous between a sentence boundary marker and a marker of abbreviations like *Mr.* or *Inc.* The previous sentence that you just read showed an even more complex case of this ambiguity, in which the final period of Inc. marked both an abbreviation and the sentence boundary marker. For this reason, sentence tokenization and word tokenization may be addressed jointly.

[Jurafsky&Martin, 2022]

Sentence Segmentation

In general, **sentence tokenization** methods work by first deciding (based on rules or machine learning) whether a period is part of the word or is a sentence-boundary marker.
An abbreviation dictionary can help determine whether the period is part of a commonly used abbreviation; the dictionaries can be hand-built or machinelearned, as can the final sentence splitter.

In the **Stanford CoreNLP toolkit** [Manning et al., 2014], for example, sentence splitting is rule-based, a deterministic consequence of tokenization; a sentence ends when a sentence-ending punctuation (., !, or ?) is not already grouped with other characters into a token (such as for an abbreviation or number), optionally followed by additional final quotes or brackets.
https://stanfordnlp.github.io/CoreNLP/

[Jurafsky&Martin, 2022]
[Manning et al.,

Q&A

Resources and References

[Jurafsky&Martin, 2022] Jurafsky and Martin. Speech and Language Processing, Prentice Hall, third edition https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf

[Bird et. al 2009] Bird, S., Klein, E., and Loper, E. (2009). Natural Language Processing with Python. O'Reilly. https://www.oreilly.com/library/view/natural-language-processing/9780596803346/

[Sennrich et al. 2016] Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. ACL. https://aclanthology.org/P16-1162/

[Porter 1980] Porter, M. F. (1980). An algorithm for suffix stripping. Program 14(3), 130–137. http://www.cs.toronto.edu/~frank/csc2501/Readings/R2_Porter/Porter-1980.pdf

[Manning et al., 2014] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. ACL. https://aclanthology.org/P14-5010.pdf

Notebook:

https://github.com/iacopomasi/NLP/blob/main/notebooks/Part_1_5_Words_Corpora_Text_Normalization.ipynb

**Credits

The slides of this part of the course are the result of a personal reworking of the slides and of the course material from different sources:

1. The NLP course of Prof. Roberto Navigli, Sapienza University of Rome
2. The NLP course of Prof. Simone Paolo Ponzetto, University of Mannheim, Germany
3. The NLP course of Prof. Chris Biemann, University of Hamburg, Germany
4. The NLP course of Prof. Dan Jurafsky, Stanford University, USA

Highly readable font Biancoenero© by biancoenero edizioni srl, designed by Umberto Mischi, with the consultancy of Alessandra Finzi, Daniele Zanoni and Luciano Perondi (Patent no. RM2011OOOO128).

Available free of charge for all institutions and individuals who use it for non-commercial purposes.