

# Text preprocessing



Prof Dr Marko Robnik-Šikonja  
Natural language processing, Edition 2026

# Lecture outline

- Text preprocessing and normalization

Read Chapter 2 in  
Daniel Jurafsky & James H. Martin. Speech and Language  
Processing, 3rd edition draft, 2026.

Some slides from this source



# Basic text preprocessing for the (classical) NLP pipeline

- document → paragraphs → sentences → words
- words and sentences ← POS tagging
- sentences ← syntactical and grammatical analysis
- still present in neural pipeline, but also splits word into subword tokens

# Text preprocessing

**Lool :-)**

- text normalization: transformation into a standard (canonic) form or any useful form, e.g., from non-standard language to standard
  - upper/lower casing
  - rediacritisation (e.g., for Slovene)
  - notation of acronyms
  - standard form of dates, time, and numbers
  - stress marks, quotation marks, punctuation,
  - non-informative words
  - spelling, e.g., US or GB
  - emoticons, emoji, hashtags, web links
  - editing and presentation markup, e.g., html tags
  - spelling correction
  - (subword) tokenization
  - lemmatization and stemming
- other forms of text preparation, e.g., extraction from PDFs, structured files like XML, web crawl, etc.

[illegible][illegible][illegible]

# Token, type, term

- A *token* is an instance of a sequence of characters in some text processing task that are grouped together as a useful semantic unit for processing.
- A *type* is the class of all tokens containing the same character sequence.
- A *term* is a (perhaps normalized) type that is included in the system's dictionary.
- *To sleep perchance to dream,*
- 5 tokens, 4 types (2 instances of *to*)
- if *to* is omitted from the index (as a stop word), then there will be only 3 terms: *sleep*, *perchance*, and *dream*
- Warning: neural processing brings some ambiguity what is a (subword) token, e.g., ambiguity -> ambig #u #ity

# Is the tokenization this simple?

```
## tokenizing a piece of text
doc = "I wrote this sentence"
for i, w in enumerate(doc.split(" ")):
    print("Token " + str(i) + ": " + w)
```

Token 0: I

Token 1: wrote

Token 2: this

Token 3: sentence

# How many words?

**$N$**  = number of tokens

**$V$**  = vocabulary = set of types,  **$|V|$**  is the size of vocabulary

Heaps Law = Herdan's Law:  $|V| = kN^\beta$  where often  $.67 < \beta < .75$

i.e., vocabulary size grows with  $>$  square root of the number of word tokens

|                                 | Tokens = $N$ | Types = $ V $ |
|---------------------------------|--------------|---------------|
| Switchboard phone conversations | 2.4 million  | 20 thousand   |
| Shakespeare                     | 884,000      | 31 thousand   |
| COCA, edition 2010              | 440 million  | 2 million     |
| Google N-grams                  | 1 trillion   | 13+ million   |

# Corpora

- Words don't appear out of nowhere.
- A text is produced by a specific writer(s), at a specific time, in a specific variety of a specific language, for a specific function.

# Corpora vary along dimension like

- **Language:** 7097 languages in the world
- **Variety**, like African American Language varieties.

- AAL Twitter posts might include forms like "*iont*" (*I don't*)

- **Code switching**, e.g., Spanish/English, Hindi/English:

S/E: Por primera vez veo a @username actually being hateful! It was beautiful:)

*[For the first time I get to see @username actually being hateful! it was beautiful:]*

H/E: dost tha or ra- hega ... dont worry ... but dherya rakhe

*["he was and will remain a friend ... don't worry ... but have faith"]*

- **Genre:** newswire, fiction, non-fiction, scientific articles, Wikipedia
- **Author demographics:** writer's age, gender, race, socioeconomic status, etc.

# Corpus datasheets

- **Motivation:** Why was the corpus collected, by whom, and who funded it?
- **Situation:** In what situation was the text written?
- **Collection process:** If it is a subsample how was it sampled? Was there consent? Pre-processing?
- **+Annotation process, language variety, speaker demographics**
- See, e.g., corpora on [Clarin.si](https://clarin.si)

# Text Normalization

- Many NLP task need text normalization:
  1. Segmenting/tokenizing words in running text
  2. Normalizing word formats
  3. Segmenting sentences in running text

# Simple Tokenization in UNIX

- (Inspired by Ken Church's UNIX for Poets.)
- Given a text file, output the word tokens and their frequencies
- Command tr (translate)

```
tr -sc 'A-Za-z' '\n' < shakes.txt
```

Change all non-alpha to newlines

```
    | sort
```

Sort in alphabetical order

```
    | uniq -c
```

Merge and count each type

```
1945 A
    72 AARON
    19 ABBESS
     5 ABBOT
... ..
    25 Aaron
     6 Abate
     1 Abates
     5 Abbess
     6 Abbey
     3 Abbot
.... ..
```

# Issues in Tokenization

- Can't just blindly remove punctuation:
  - [m.p.h.](#), [Ph.D.](#), [AT&T](#), [cap'n](#).
  - prices ([\\$45.55](#))
  - dates ([01/02/06](#));
  - URLs; (<http://www.stanford.edu>),
  - hashtags ([#nlp](#)),
  - email addresses ([someone@cs.colorado.edu](mailto:someone@cs.colorado.edu)).
- Clitics: a part of a word that can't stand on its own
  - [we're](#) → we are
  - French [j'ai](#), [l'honneur](#)
  - Slovene: a [b' šlo](#)
- Can "Multiword Expressions (MWE) be words?
  - [New York](#), [rock 'n' roll](#)

# Issues in Tokenization

- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??

# Tokenization in NLTK

Bird et al. (2009)

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

# Tokenization: language issues

- French
  - ***L'ensemble*** → one token or two?
    - *L* ? *L'* ? *Le* ?
    - Want *l'ensemble* to match with *un ensemble*
- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - ‘life insurance company employee’
  - German information retrieval needs **compound splitter**

# Word Tokenization in Chinese

- Also called **Word Segmentation**
- Chinese words are composed of characters called **hanzi**
- Each one represents a meaning unit called a morpheme.
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.
- But deciding what counts as a word is complex and not agreed upon
- Standard baseline segmentation algorithm:
  - Maximum Matching (also called Greedy)
- So in Chinese it's common not to do word segmentation at all
- But in Thai and Japanese, it's required
- The standard algorithms are neural sequence models trained by supervised machine learning.

# Words in preprocessing

- Lexical analysis (tokenizer, word segmented), not just spaces
- 1,999.00€    1.999,00€!
- Ravne na Koroškem
- Port-au-prince
- Rules, finite automata, statistical models, dictionaries (of proper names), lexicons, ML models

# Term normalization

- Why we need to “normalize” terms
  - Information Retrieval (IR): indexed text & query terms must have the same form.
    - We want to match **U.S.A.** and **USA**
    - uhhuh or uh-huh
    - Fed or fed
    - am, is be, are
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
  - Enter: **window**     Search: **window, windows**
  - Enter: **windows**     Search: **Windows, windows, window**
  - Enter: **Windows**     Search: **Windows**
- Potentially more powerful, but less efficient

# Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., *General Motors*
    - *Fed* vs. *fed*
    - *SAIL* vs. *sail*
- For many uses case is helpful
  - sentiment analysis, machine translation (MT), information extraction
  - *US* versus *us* is important

# Lemmatization

- Reduce inflections or variant forms to base form
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation
  - Slovene **hočem** ('I want'), **hočeš** ('you want') have the same lemma as **hoteti** 'want'

# Morphology

- Morphemes:
  - Small meaningful units that make up words
  - **Stems**: The core meaning-bearing units
  - **Affixes**: Bits and pieces that adhere to stems
    - Often with grammatical functions
- Morphological Parsers:
  - Parse *cats* into two morphemes *cat* and *s*
  - Parse Spanish *amaren* ('if in the future they would love') into morpheme *amar* 'to love', and the morphological features *3PL* and *future subjunctive*.

# Dealing with complex morphology is sometimes necessary

- Some languages requires complex morpheme segmentation
  - Turkish
  - **Uygarlastiramadiklarimizdanmissinizcasina**
  - `(behaving) as if you are among those whom we could not civilize’
  - **Uygar** `civilized’ + **las** `become’
    - + **tir** `cause’ + **ama** `not able’
    - + **dik** `past’ + **lar** `plural’
    - + **imiz** `p1pl’ + **dan** `abl’
    - + **mis** `past’ + **siniz** `2pl’ + **casina** `as if’

# Lemmatization

- Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item.
- Lemmatization difficulty is language dependent i.e., depends on morphology
- *English*
  - *walk, walked, walking, walks, ne pa walker*
  - *go, goes, going, gone, went*
- *Slovene*
  - *priiti, pridem, prideš, pride, prideva, prideta, pridejo, pridemo, pridete, pridejo, ne pa prihod, prihodnost, prihajanje, prišlec*
  - *vlak, vlaka, vlaku, vlakom, vlakov, vlakoma, vlakih, vlaki, vlake*
  - *jaz, mene, meni, mano*
  - *Gori na gori gori!*
  - *Gori, na gori gori!*
- Use rules, dictionaries, lexicons, machine learning models
- Ambiguity resolution may be difficult

Meni je vzel z mize (zapestnico).
- Quick solutions and heuristics, in English just remove suffixes: *-ing, -ation, -ed, ...*
- essential approach for morphologically rich languages (Slavic, Arabic, Turkish, Spanish, etc)

# Stemming

- stem: the root or main part of a word, to which inflections or formative elements are added
- in English
- simple solution: remove affixes

***for example compressed and compression are both accepted as equivalent to compress.***



for exampl compress and  
compress ar both accept  
as equival to compress

- Stemmer operates on a single word *without* knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on part of speech (meeting: a lemma is to meet or a meeting). Speed!
- Potter algorithm
- rare nowadays

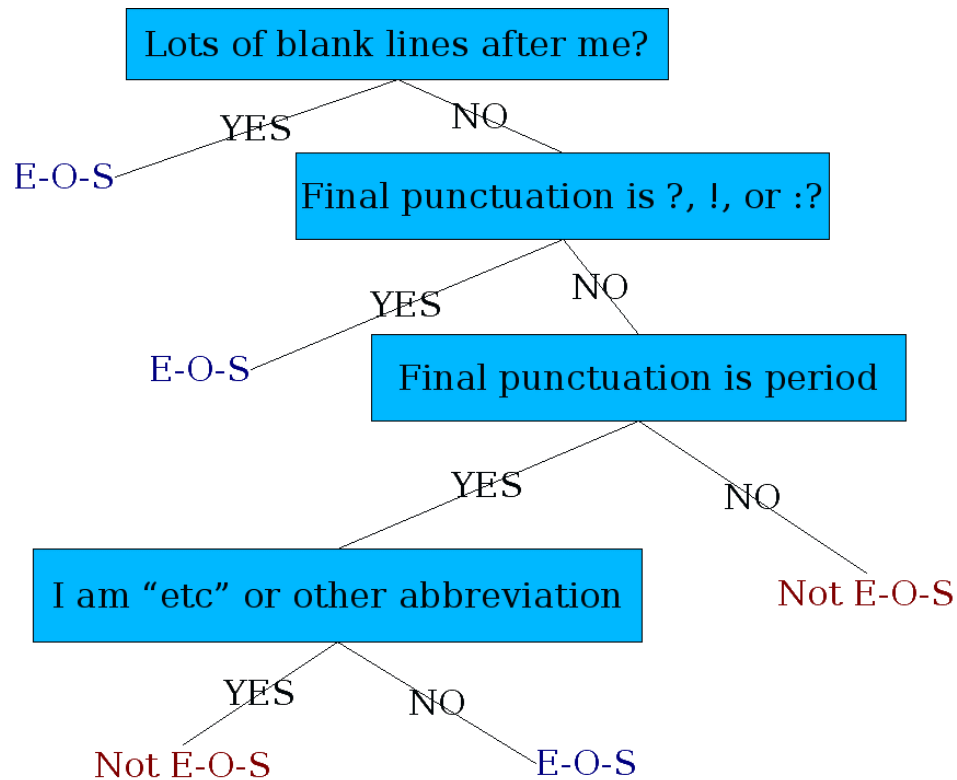
# Sentences

- sentence delimiters – punctuation marks and capitalization are insufficient
- E.g., remains of 1. Timbuktu from 5c BC, were discovered by dr. Barth.
- Regular expressions, rules, manually segmented corpora

# Sentence segmentation

- !, ? are relatively unambiguous
- Period “.” is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary ML classifier
  - Looks at a “.”
  - Decides EndOfSentence/NotEndOfSentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

# Determining if a word is end-of-sentence: a Decision Tree



## More sophisticated features

- Case of word with “.”: Upper, Lower, Cap, Number
- Case of word after “.”: Upper, Lower, Cap, Number
- Numeric features
  - Length of word with “.”
  - Probability(word with “.” occurs at end-of-s)
  - Probability(word after “.” occurs at beginning-of-s)

# Tools

- every NLP library has a tokenizer, sentence delimiter, lemmatizer, e.g., NLTK, spaCy, Gensim
- for Slovene: CLASSLA-Stanza
- <https://www.cjvt.si/viri/>
- <https://github.com/clarinsi>
- for nonstandard Slovene (twits, forum messages)
  - **Nikola Ljubešić, Tomaž Erjavec, Darja Fišer:** Orodja za procesiranje nestandardne slovenščine. V Fišer, D. (ur). 2018. Viri, orodja in metode za analizo spletne slovenščine. Ljubljana: Znanstveni založbi Filozofske fakultete Univerze v Ljubljani.