Text Analysis and Retrieval

# 2. Basics of Natural Language Processing

Prof. Jan Šnajder

University of Zagreb
Faculty of Electrical Engineering and Computing (FER)

Academic Year 2022/2023



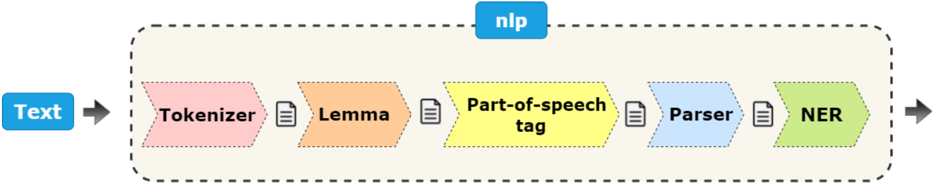
v3.0

# Motivation: NLP as preprocessing

- Most text analysis tasks benefit from natural language processing
- For basic IR, you don't need much: **tokenization** and a bit of **morphology processing** suffices
- For full-blown semantic text analysis, you need a lot: proper **morphology, syntax, and semantic processing**
- There are many tools available for these tasks (unfortunately, in most cases the best tools available work only for English)

# Outline

# Typical NLP pipeline



# Typical NLP pipeline

1. Language detection
2. Text cleanup (boilerplate removal / normalization / OCR-error correction, . . . )
3. Sentence segmentation
4. Tokenization
5. Morphological processing: stemming
6. POS tagging
7. Morphological processing: lemmatization
8. Syntactic processing: parsing

$\vdots$

Higher-level tasks (semantics, information extraction, . . . )

# Basic NLP pipeline

1. Language detection
2. Text cleanup (boilerplate removal / normalization / OCR-error correction, ...)
3. **Sentence segmentation**
4. **Tokenization**
5. **Morphological processing: stemming**
6. **POS tagging**
7. **Morphological processing: lemmatization**
8. Syntactic processing: parsing

⋮

Higher-level tasks (semantics, information extraction, ...)

# Basics of morphology

## Morphology

Branch of linguistics concerned with the internal structure of words. Words are made up of morphemes (= smallest linguistic pieces with a grammatical function).

1. **Inflectional morphology**: creating word-forms that express grammatical features
   - fish → fishes, Haus → Häuser, skup → najskupljoj
2. **Derivational morphology**: creating new words from existing ones
   - fish → fishery, Haus → Gehäuse, voće → voćnjak
3. **Compounding**: combine two or more existing words
   - sky + scraper, Kraft + fahr + zeug, vatro + gasac

# Quick test

Inflection, derivation, or compounding?

- EN: show $\rightarrow$ showed
- EN: big $\rightarrow$ bigger
- HR: novac $\rightarrow$ novčanik
- HR: kupiti $\rightarrow$ otkupljivanje
- EN: house $\rightarrow$ housing
- EN: run $\rightarrow$ runs
- DE: kaufen $\rightarrow$ verkaufen
- DE: kaufen $\rightarrow$ verkauft
- EN: house $\rightarrow$ housewife
- EN: tour $\rightarrow$ detoured

# Morphological normalization

- Transform each word to its **normalized form** (whatever it is)
- Two approaches:
  - **Stemming** – quick and dirty
  - **Lemmatization** – linguistically proper way of normalization

# Stemming

- Reduction of word-forms to **stems**
  - adjustments → adjust
  - defensible → defens
  - revivals → reviv
- Typically by **suffix stripping** plus some extra steps and checks
- Pros: simple and efficient
- Cons:
  - prone to **overstemming** and **understemming** errors
  - difficult to design for morphologically complex languages
  - imprecise (don't differentiate between inflection and derivation)

# Lemmatization

- Transformation of a word-form into a linguistically valid base form, called the **lemma** (the dictionary form)
  - nouns $\rightarrow$ singular nominative form
  - verbs $\rightarrow$ infinitive form
  - adjectives $\rightarrow$ singular, nominative, masculine, indefinitive, positive form
- A much more difficult task than stemming, especially for morphologically complex languages, for which you basically need:
  - a **morphological dictionary** that maps word-forms to lemmas
  - a **machine learning model**, trained on a large number of word-lemma pairs

# Parts-of-speech

- **Part of speech** is the grammatical category of a word
- Some parts of speech are universal across languages:
  - **Verbs** assert something about the subject of the sentence and express actions, events, or states of being
  - **Nouns** are words that we used to name a person, an animal, a place, a thing, or an abstract idea
  - **Adjectives** modify nouns and pronouns by describing, identifying, or quantifying them.
  - **Pronouns** replace nouns or another pronouns and are essentially used to make sentences less cumbersome and less repetitive
  - **Adverbs** modify a verb, an adjective, another adverb, a phrase, or a clause. An adverb indicates manner, time, place, cause, ...
  - **Prepositions**, **Conjunctions**, **Interjections** ...

# POS tagging

- Marking up a word in a text with its part of speech

**POS-tagged text**

A/DT Part-Of-Speech/NNP Tagger/NNP is/VBZ a/DT piece/NN of/IN software/NN that/WDT reads/VBZ text/NN in/IN some/DT language/NN and/CC assigns/VBZ parts/NNS of/IN speech/NN to/TO each/DT word/NN ,/, such/JJ as/IN noun/NN ,/, verb/NN ,/, adjective/NN ,/, etc./FW./.

- POS taggers assign tags from a finite predefined **tagset**
- State-of-the-art POS taggers are supervised machine learning models

# NLP libraries

- Stanford CoreNLP (http://nlp.stanford.edu/software/corenlp.shtml)
- NLTK (http://www.nltk.org/)
- **spaCy** (https://spacy.io/) ⇐ **recommended!**

# Learning outcomes 1

1. Describe the components of the basic NLP pipeline
2. Describe what POS tagging is and why we need it
3. Explain stemming and lemmatization, why we need it, and the difference between them
4. List the main NLP tools available

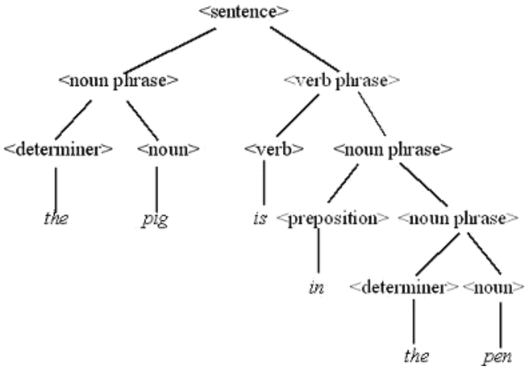
# Outline

# Grammars and parsers

- **Parsing** is the task of analyzing the grammatical structure of a sentence, which results in a syntax tree of the sentence
- Given a sequence of words, a parser forms units like subject, verb, object and determines the relations between them according to some grammar formalism
- Two types of parsers
  - **Constituency parsers/phrase structure tree (PST) parsers** – based on constituency/PS grammars
  - **Dependency parsers** – based on dependency grammars

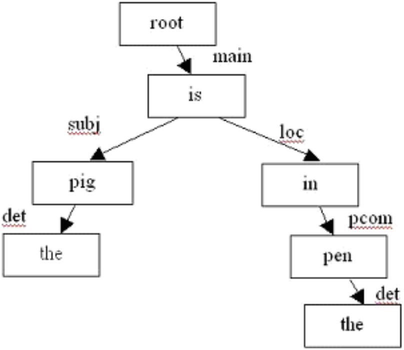
# Constituency parser

- Produces a tree that represents the syntactic structure of a sentence (i.e., a breakdown of the sentence)
- Words appear only as leaves of the tree



# Dependency parser

- Produces a tree of syntactic dependencies between pairs of words
- Each dependency relation has a governing word and a dependent word
- Verb is the syntactic center of the clause, all other words directly or indirectly dependent on the verb



# Universal dependencies (UD)

- Cross-linguistically consistent labels for multilingual parsing
  http://universaldependencies.org/#universal-dependencies-v2
- Universal POS Tags
  http://universaldependencies.org/u/pos/
- Universal Dependency Relations
  http://universaldependencies.org/u/dep/

# Universal dependencies (UD) parsers

- Stanford UD parser (English)
  https://nlp.stanford.edu/software/stanford-dependencies.shtml
  demo: http://nlp.stanford.edu:8080/corenlp/
- spaCy's dependency parser (English)
  https://spacy.io/api/dependencyparser
  very nice demo: https://explosion.ai/demos/displacy
- Google's SyntaxNet
  - Parsey McParseface (English)
    https://github.com/plowman/python-mcparseface
  - Parsey Universal (40 languages, including DE, HR, and SI)
    https://github.com/tensorflow/models/blob/master/research/syntaxnet/g3doc/universal.md

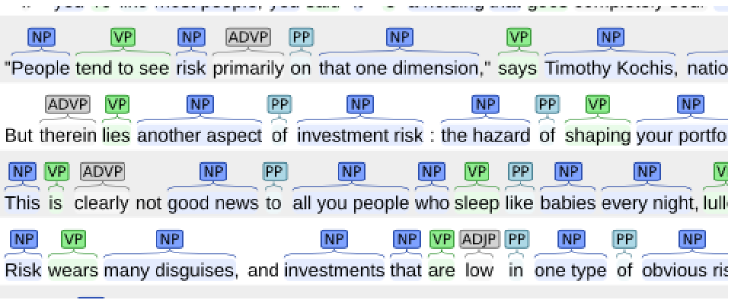
# Shallow parsing (aka "chunking")

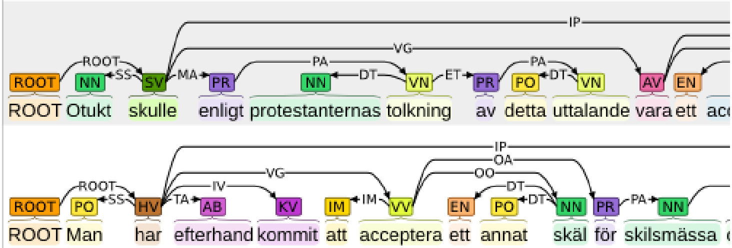
- Merely identifies the constituents (noun phrases, verb phrases, prepositional phrases, etc.), but does not specify their internal structure nor their role in the sentence

[ NP Jack and Jill ] [ VP went ] [ ADVP up ] [ NP the hill ] [ VP to fetch ] [ NP a pail ] [ PP of ] [ NP water ].

- spaCy demo
  http://textanalysisonline.com/spacy-noun-chunks-extraction

# Parsing vs. chunking



# Learning outcomes 2

1. Describe what parsing is and why we need it
2. Differentiate between phrase-based and dependency-based parsing
3. Describe what chunking is and why we need it
4. List the main tools available for parsing/chunking

# Outline

1. Basic NLP pipeline

2. Syntactic parsing

3. Corpora & language modeling

# Corpora

- **Text corpus** (plural: **corpora**): large and structured set of texts, used for **corpus linguistic analyses** and for the development of **natural language models** (primarily machine learning models)
- Popular corpora (English):
  - Brown Corpus (1M words)
  - British National Corpus – BNC (100M words)
  - Wall Street Journal Corpus (30M words)
- Web as a Corpus (WaC): ukWaC, frWaC, deWaC, hrWaC
  - WaCky - The Web-As-Corpus Kool Yinitiative (`http://wacky.sslmit.unibo.it`)

# Language modeling

- Probabilistic models of text, used for two purposes:
  1. determine the probability of the next word in a sequence
  2. determine the probability of a word sequence
- We'd like to compute the probability

$$P(w_1, w_2, \ldots, w_{n-1}, w_n) = P(w_1^n)$$

- This can be rewritten using the chain rule

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\cdots P(w_n|w_1^{n-1}) = \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

- All we need now is to estimate these probabilities. . .

# Language modeling

- Naive solution: **maximum likelihood estimates (MLE)** from corpus

$$P(w_k|w_1^{k-1}) = \frac{C(w_1^k)}{C(w_1^{k-1})}$$

  where $C(\cdot)$ is the number of occurrences in the corpus
- This would fail because of **sparsity**
- Solution: **approximate** by considering only $N-1$ preceding words

$$P(w_k|w_1^{k-1}) \approx P(w_k|w_{k-N+1}^{k-1})$$

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_{k-N+1}^{k-1})$$

# Language modeling

- MLE:

$$P(w_k|w_{k-N+1}^{k-1}) = \frac{C(w_{k-N+1}^{k})}{C(w_{k-N+1}^{k-1})}$$

**Language model MLE**

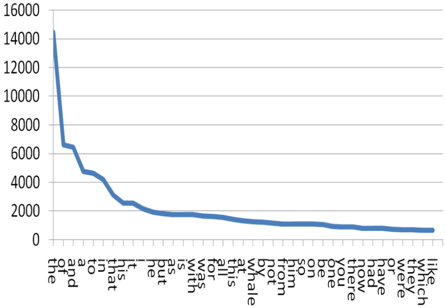
I saw a white fluffy. . .

- Bigram model ($N = 2$):
  $P(\text{rabbit}|\text{I saw a white fluffy}) \approx \frac{C(\text{fluffy rabbit})}{C(\text{fluffy})}$
- Trigram model ($N = 3$):
  $P(\text{rabbit}|\text{I saw a white fluffy}) \approx \frac{C(\text{white fluffy rabbit})}{C(\text{white fluffy})}$

- Increasing $N$ increases the accuracy, but also memory usage!

# A problem with MLE: Zipf's law

- Zipf's law (Zipf, 1949) states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table
- Example: sorted word counts in Herman Melville's "Moby Dick"




- **Happax legomena** account for $\sim 50\%$ of the words in corpus

# Neural language models (NLMs)


- Dan Jurafsky (2018). **Neural Networks and Neural Language Models**. (SLP draft chapter)
  https://web.stanford.edu/~jurafsky/slp3/7.pdf
- Yoshua Bengio et al. (2003). **A neural probabilistic language model**. Journal of machine learning research.
  http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

# Learning outcomes 3

1. Describe what a corpus is, why we need it, and name a few
2. Describe what a language model is and what it's used for
3. Write down the MLE probability for an $N$-gram language model
4. Differentiate between statistical and neural language models

# Study assignment

1. Study TAR slides "Basics of NLP":
   https://www.fer.unizg.hr/_download/repository/TAR-02-NLP.pdf
2. Read Jurafsky's chapter on dependency parsing (section 18.1):
   https://web.stanford.edu/~jurafsky/slp3/18.pdf
3. Read Jurafsky's chapter on n-gram LMs:
   https://www.fer.unizg.hr/_download/repository/TAR-2020-reading-01.pdf
4. Read Jurafsky's chapter on neural LMs (focus on section 7.5):
   https://web.stanford.edu/~jurafsky/slp3/7.pdf
5. Familiarize yourself with SOTA in LM:
   https://paperswithcode.com/task/language-modelling
   https://github.com/sebastianruder/NLP-progress/blob/master/english/language_modeling.md
6. Self-check against learning outcomes!