

# Morphology

## (Two-level morphology and Stemming)

Ing. Roberto Tedesco, PhD

[roberto.tedesco@polimi.it](mailto:roberto.tedesco@polimi.it)



NLP – AA 20-21

# Preprocessing

- The first task to do in the “NLP toolchain”
- Extracting text from textual documents
  - txt, HTML, e-mail, ...
  - Discard tags and other format-specific commands
- Extracting text from binary documents
  - Word, PDF, ...
  - Much more complex (PDF...)
- Spelling correction?
- Text encoding:
  - ASCII, ISO 8859-1 (Latin 1), UTF-8, ...
  - Languages with diacritical signs (like Italian “Questa è così”, but also English “Naïve”)

# Morphology

- Analysis and description of the structure of words
- **Morpheme**: the smallest linguistic unit that has semantic meaning
  - E.g.: unbelievably → un-believe-able-ly
- Morphemes are divided into:
  - **Root**: the base form (*believe*)
  - Affixes: **prefix** (*un-*), **infix** (*-able-*), or **suffix** (*-ly*)

# Lexicon

- Morphemes compose *lexemes*
- **Lexeme**: unit of lexical meaning that exists regardless of the number of inflectional endings it may have or the number of words it may contain
  - E.g.: BELIEVE, NEW YORK, RUN
- **Lemma**: the canonical form of a lexeme
  - E.g.: TO RUN
- **Lexicon**: a set of lexemes
  - Lexicons for NLP usually contain affixes and other info
- A **word** is, in general, a inflected form of a lexeme
  - E.g.: *unbelievably*, *runs*

# Composing morphemes

- **Morphologic rules:**

- Restrict the ordering of morphemes (morphotactics)
- E.g.: PLURAL NOUN = SINGULAR NOUN + PL

- **Orthographic rules:**

- aka “spelling rules” or “two-level rules”
- E.g.: fox + s → foxes; un-believeable-ly → unbelievably

- **Lexicons in NLP:**

- Define base forms
  - E.g.: fox: NOUN, SINGULAR, ...
- Address irregular forms
  - E.g.: wrote → root: write; mice → root: mouse
- Define affix morphemes
  - E.g.: PL → s

# Lexicon syntax

\lf **`mouse**  
\lx N  
\alt Suffix  
\fea **irreg**  
\gl1

\lf **`mice**  
\lx N  
\alt Clitic  
\fea **pl irreg**  
\gl1 `mouse

\lf **`fox**  
\lx **N**  
\alt Suffix  
\gl1

Part-of-speech:  
Noun

Has irregular plural

Is the irregular  
plural form  
of "mouse"

\lf **+s**  
\lx INFL  
\alt End  
\fea v/v s  
\gl1 **+3SG**  
\gl2 +3SG

+s: 3<sup>rd</sup> singular  
person

\lf **+s**  
\lx IC\_SUFF  
\alt Clitic  
\fea n-aj/n reg  
\gl1 **+PL**  
\gl2 +PL

+s: plural

## 7



# Recognition: with morphologic rules

- **Foxes** → **fox+PL**, noun
  - **fox+PL** disambiguated, as **fox** is a noun, and nouns have +PL but not +3SG
- **Talk**
  - Can be noun or verb
  - No disambiguation!
  - Context is needed
- POS taggers are needed
  - These tools leverage the context



# FreeLing

- Multilanguage (Spanish, English, Italian)
- Based on a statistical model
  - It is actually a POS tagger (more on that later...)
  - Can analyze the morphologic structure of the word (affixes, ...), for some languages (try “*prendimelo*”)
  - Can return the base-form and POS
- Modern approach: we have lots of space to store all inflexed forms...

## Write your sentences

that

## Analysis options

- ☒ Multiword detection
- ☒ Number recognition
- ☒ Date/Time recognition
- ☒ Quantities, ratios, and percentages
- ☒ Named Entity detection
- ☐ Named Entity classification
- ☒ No sense annotation
- ☐ WN sense annotation: All senses
- ☐ WN sense annotation: Most frequent sense

Select language

English

Select output

Morphological Analysis

Submit

## Analysis Results

### Sentence #1

that

*that*

IN

0.573717

*that*

WDT

0.239164

*that*

DT

0.184577

*that*

RB

0.00252238

*that*

WP

1.92548e-05

# Lexicon-free methods: stemming

- Simple algorithms
  - No lexicon needed
  - Often used for Information Retrieval
- Why is it useful for Information Retrieval?
  - Ideally: assign unique ID (a *stem*) to all inflected forms of a given base form
    - E.g.: (dog, dogs, doggy) → ID1, (cat, cats, catlike) → ID2, ...
  - Apply to the document collection:
    - Doc1: "... dog..." → doc1: "... ID1 ..."
    - Doc2: "... doggy..." → doc2: "... ID1 ..."
  - And to the query:
    - "dogs" → "ID1"; the system finds [doc1, doc2]

# Lexicon-free methods: Porter

- Porter Stemming Algorithm ('80)
- A set of *rewriting rules*
  - E.g.: +ATIONAL → +ATE (es: rel~~ational~~ → relat~~e~~),  
+ING → ε (es: motor~~ing~~ → motor), ...
- Simple, but error prone
  - Collisions (Two different words, the same stem)  
Word: Policy → stem: police  
Word: Police → stem: police  
Searching for “policy” I can also find docs containing “police”

# References

- FreeLing
  - <http://garraf.epsevg.upc.es/freeling>
- Porter
  - Stemmer for the English language
  - <http://www.tartarus.org/~martin/PorterStemmer/>
  - Available in C, Java e much more languages
- Other stemming algorithms
  - Lancaster Stemming Algorithm:  
<http://www.comp.lancs.ac.uk/computing/research/stemming/index.htm>
  - Snowball (programming language for stemmers)  
<http://snowball.tartarus.org/>