Spoken Human Robot Interaction

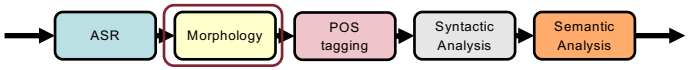
Morphology

SAPIENZA
UNIVERSITÀ DI ROMA

Andrea Vanzo, **Daniele Nardi**
lastname@dis.uniroma1.it
Department of Computer, Control, and Management Engineering
"Sapienza" University of Rome, Italy

Artificial Intelligence
AY 2018/19

Our Command Interpretation pipeline



- An essential pre-processing step before performing any NLP task
  - Verify the morphological correctness of words
  - Extract features useful for subsequent steps
- Handling of text as string of chars
  - Cleaning off undesired chars
  - Tokenize text
  - Tokenize words

Morphological ambiguities

- "The price is $24.30. No room for further discounts."
  - Multiple periods…
- "We're the best"
  - Does "we're" compose a single token? Where is the verb?
- "book books"
  - Am I reserving some books? Or someone called "Book" is reserving? Or I am just enumerating the singular and plural of "book"? What is the role of *s?*

Neologisms

- Slang
  - *tlkin* instead of *talking*
  - *touchdooown* → touchdown
- Social networks-inspired
  - #workhard
  - @Pontifex
  - *BTW* → *by the way*
  - *WTF* → … (you know what it means)
  - …

Text normalization

Main tasks:

- Tokenization
  - Input: Running text + models/rules
  - Output: Tokens composing the text
- Lemmatization
  - Stemming
    - Input: word + models/rules
    - Output: root of the word
  - Morphological Analysis
    - Input: words + models/rules
    - Output: root of the word + morphological information

Tokenization

Segmentation of running text into words (or *tokens*)

- Clitic
  - A *clitic* is a part of a word that can't stand on its own, and can only occur when it is attached to another word
    - Example: *we're* and *we are*
- Punctuation
  - Example: *m.p.h.* stands for *miles per hour*
- Prices and dates
  - $45.55 and 01/02/06 are atomic tokens

Some languages do not use spaces between words (e.g. Chinese, Japanese. Thai).

Penn "Treebank" tokenization standard

The most used tokenization standard

- **Based on RegEx**
  - `/[A-Z][a-z]*/`
- Treats all the above linguistic phenomena…
  - Clitics
  - Punctuation
  - Dates and Prices

…and more

ELIZA (Weizembaum, 1966)

A simple pattern matching on the tokens

```
User:   You are like my father in some ways.
ELIZA:  WHAT RESEMBLANCE DO YOU SEE
User:   You are not very aggressive but I think you don't want me to
        notice that.
ELIZA:  WHAT MAKES YOU THINK I AM NOT VERY AGGRESSIVE
User:   You don't argue with me.
ELIZA:  WHY DO YOU THINK I DON'T ARGUE WITH YOU
User:   You are afraid of me.
ELIZA:  DOES IT PLEASE YOU TO BELIEVE I AM AFRAID OF YOU
```

Weizenbaum (1966)

Ancestor of Modern Chatbots: More on regular expressions and the design of chatbots, next class

Lemmatization

Determining whether two words have the same root

- *am*, *are*, and *is* have the shared lemma *be*
- *dinner* and *dinners* have the lemma *dinner*

Representing a word by its lemma is important:

- In some tasks
  - web search (catch the meaning of the query)
  - generalize the lexicon and reduce uncertainty
- In morphologically complex languages
  - Arabic, Turkish, …

Lemmatization

Two approaches:

- Stemming
  - Naïve approach based on brute chopping
  - Not needed in E
- Morphology parsing
  - Complete and extensive morphology analysis

Stemming: Porter Stemmer

- Return the ”stem” of the words in a text
  - Input: *talk**ing***
  - Output: *talk*
- Porter stemmer 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
  - SSES → SS (e.g., grasses → grass)
  - ATIONAL → E (e.g., computational → compute)

Morphology

Study of the way words are built up from smaller units called **morphemes**

- Two broad classes of morphemes
  - Stems: define the main meaning
  - Affixes: add additional meaning
    - Prefixes
    - Suffixes
- Example: *cats*
  - The morpheme *cat* is the stem
  - The morpheme *-s* is the affix (suffix in this case)

Morphology: taxonomy

- Concatenative
- Agglutinative

- Inflectional Morphology
  - Nouns
  - Verbs
  - Irregularities
- Derivational Morphology
  - Nominalization
  - Adjectivization

Morphology: concatenative vs agglutinative

- Each (natural) language has its own morphology
  - Concatenative: affix* + stem + affix*
    - Italian
      - gatt-**o** (gender)
      - **s**-conosciuto (meaning)
    - English
      - buy-**er**
      - **un**-able (meaning)
  - Agglutinative: (stem* + affix*)*
    - Turkish
      - Muvaffakiyetsizleştiricileştiriveremeyebileceklerimizdenmişsiniz
      - "You happen to have been from among those whom we will not be able to easily/quickly make a maker of unsuccessful ones"

Morphology

- Inflectional Morphology
  - Combination of a stem and an affix resulting in a word of the **same category**
    - cat → cat-s
    - cut → cut-t-ing

- Derivational Morphology
  - Combination of a stem and an affix resulting in a word of a **different category**
    - transport → transport-able
    - tokenize → tokeniz-ation

Inflectional Morphology

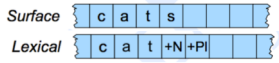
- Nouns
  - 2 inflexions
    - Plural: cat → cat**s**
    - Possessive: childrens → childrens**'**
- Verbs
  - 4 inflexions
    - stem: walk
    - **s** form: walk → walk**s**
    - past form: walk → walk**ed**
    - **ing** form: walk → walk**ing**
- Irregularities
  - mouse → mice
  - child → children
  - be → am, are, were, was, been

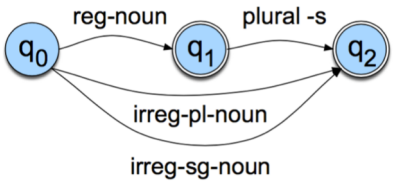
Derivational Morphology

- Nominalization (verb, adjective → noun)
  - 4 inflexions
    - **-ation**: computerize → computerization
    - **-ee**: employ → employee
    - **-er**: kill → killer
    - **-ness**: happy → happiness
- Adjectivization (verb, noun → adjective)
  - 3 inflexions
    - **-al**: computation → computational
    - **-able**: contain → containable
    - **-less**: shame → shameless

Morphology Analysis: FST

- Finite State Transducers: FSA + 2 memory *tapes*
  - A tape for reading the input
  - A tape for writing the output





Available tools

- Stanford Core NLP (Java, but APIs for other programming languages)
  - Collection of several annotators/parsers
    - Morphology Analysis
    - POS tagging
    - Syntactic Parsing
    - …
  - Probably the most famous tool for NLP
- Natural Language ToolKit - NLTK (Python)
  - Same functionalities of Stanford Parser

References

- Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. https://web.stanford.edu/~jurafsky/slp3/

Chapter 2 (but for reg-exp 2.1, refer also to AIML)