# Natual Language Processing

Alessia Mondolo

August 14, 2019

Academy AI

# Syntax Analysis

# Recap

**Syntax Analysis:**

- Words tokenization
- Sentences segmentation
- Part-of-Speech (PoS) tagging
- Lemmatizing
- Stemming

**Semantics Analysis:**

- Lexical semantics: Synonyms sets (Synset)
- Named Entity Recognition (NER)
- Word Sense Disambiguation (WSD)

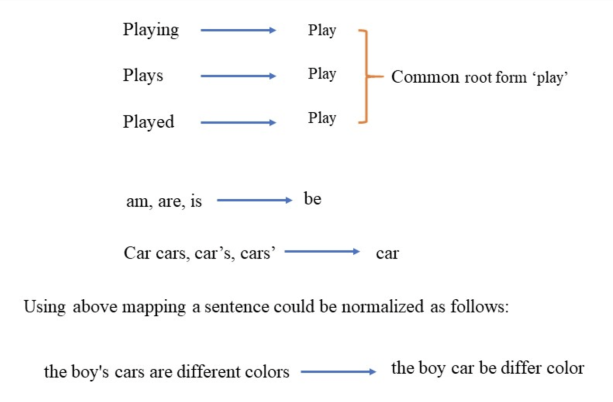
# Text normalization

Techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing:

- Words tokenization
- Sentence segmentation
- Stemming
- Lemmatization

# Inflection

**Inflection:** the modification of a word to express different grammatical categories such as tense, case, voice, aspect, person, number, gender, and mood. An inflection expresses one or more grammatical categories with a prefix, suffix or infix, or another internal modification such as a vowel change. Inflected words will have a common root form:



Playing → Play

Plays → Play

Played → Play

Common root form ‘play’

am, are, is → be

Car cars, car’s, cars’ → car

Using above mapping a sentence could be normalized as follows:

the boy's cars are different colors → the boy car be differ color

# Stemming and lemmatization

Stemming and Lemmatization helps us to achieve the **root forms** (sometimes called synonyms in search context) of inflected (derived) words. **Stemming is different to Lemmatization in the approach it uses to produce root forms of words and the word produced.**

Stemming and Lemmatization are widely used in **tagging systems, indexing, SEOs, Web search results, and information retrieval**. For example, searching for fish on Google will also result in fishes, fishing as fish is the stem of both words.

# Stemming

Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem **even if the stem itself is not a valid word in the Language**.

Stem (root) is the part of the word to which you add inflectional (changing/deriving) affixes such as (-ed,-ize, -s,-de,mis). So stemming a word or sentence may result in words that are not actual words. Stems are created by **removing the suffixes or prefixes used with a word**.

# Stemming with NLTK

**English stemmers:** `PorterStemmer`, `LancasterStemmer`

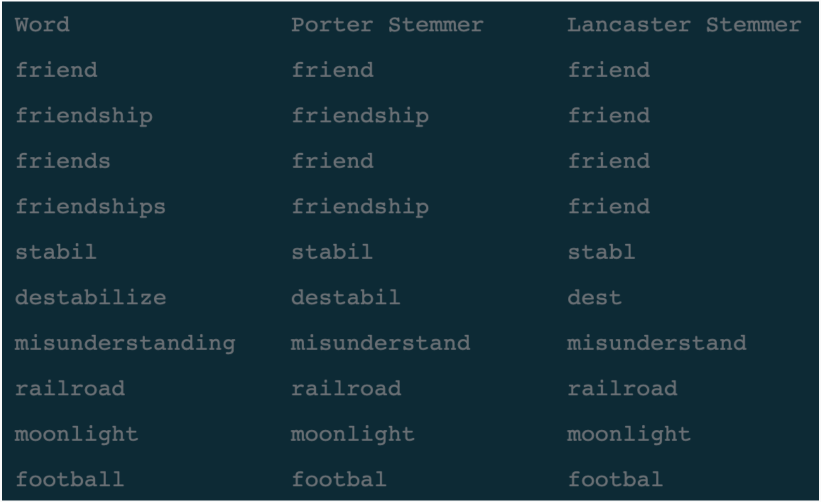
```
word_list =["friend", "friendship", "friends", "friendships","
                                    stabil","destabilize","
                                    misunderstanding","
                                    railroad","moonlight","
                                    football"]
print("{0:20}{1:20}{2:20}".format("Word","Porter Stemmer","
                                    lancaster Stemmer"))
for word in word_list:
    print("{0:20}{1:20}{2:20}".format(word.porter.stem(word),
                                        lancaster.stem(word)))
```

**Non-English:** SnowballStemmers (Danish, Dutch, English, French, German, Hungarian, Italian, Norwegian, Porter, Portuguese, Romanian, Russian, Spanish, Swedish), ISRIStemmer (Arabic), RSLPStemmer (Portuguese)

```
spanishStemmer=SnowballStemmer("spanish")
spanishStemmer.stem("Corriendo")
```

Output: `'corr'`

# Porter vs. Lancaster



| Word | Porter Stemmer | Lancaster Stemmer |
|---|---|---|
| friend | friend | friend |
| friendship | friendship | friend |
| friends | friend | friend |
| friendships | friendship | friend |
| stabil | stabil | stabl |
| destabilize | destabil | dest |
| misunderstanding | misunderstand | misunderstand |
| railroad | railroad | railroad |
| moonlight | moonlight | moonlight |
| football | footbal | footbal |

## Porter vs. Lancaster

**Porter stemmer:**

- oldest one, originally developed in 1979, known for its simplicity and speed. It uses Suffix Stripping to produce stems
- does not follow linguistics, but rather a set of 5 rules for different cases that are applied in phases (step by step) to generate stems
- often generate stems that are not actual English words
- commonly useful in IR Environments for fast recall and fetching of search queries

**Lancaster stemmer:**

- developed in 1990 and uses a more aggressive approach than Porter Stemming Algorithm
- iterative algorithm with rules saved externally
- simple, but heavy stemming due to iterations and over-stemming may occur. Over-stemming causes the stems to be not linguistic, or they may have no meaning

# Lemmatization

Lemmatization, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called Lemma. A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a set of words.

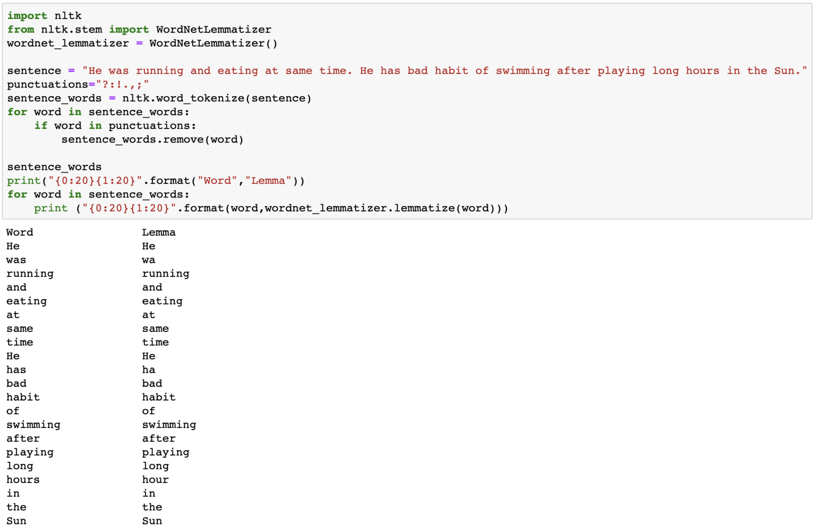
For example, runs, running, ran are all forms of the word run, therefore run is the lemma of all these words. Because lemmatization returns **an actual word** of the language, it is **used where it is necessary to get valid words**.

# Lemmatization with NLTK

NLTK provides `WordNet Lemmatizer` that uses the `WordNet` Database to lookup lemmas of words.

Note: Download the WordNet corpora from NLTK downloader before using the WordNet Lemmatizer: `nltk.download('wordnet')`

# Lemmatization with NLTK

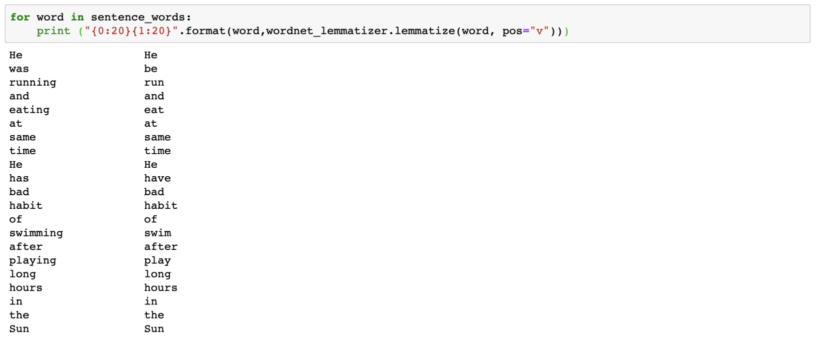


```
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

sentence = "He was running and eating at same time. He has bad habit of swimming after playing long hours in the Sun."
punctuations="?:!.,;"
sentence_words = nltk.word_tokenize(sentence)
for word in sentence_words:
    if word in punctuations:
        sentence_words.remove(word)

sentence_words
print("{0:20}{1:20}".format("Word","Lemma"))
for word in sentence_words:
    print ("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word)))
```

```
Word                Lemma
He                  He
was                 wa
running             running
and                 and
eating              eating
at                  at
same                same
time                time
He                  He
has                 ha
bad                 bad
habit               habit
of                  of
swimming            swimming
after               after
playing             playing
long                long
hours               hour
in                  in
the                 the
Sun                 Sun
```

# Lemmatization with NLTK

You need to provide the context in which you want to lemmatize that is the parts-of-speech (POS). This is done by giving the value for pos parameter in `wordnet_lemmatizer.lemmatize`.



```
for word in sentence_words:
    print ("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word, pos="v")))
```

```
He                  He
was                 be
running             run
and                 and
eating              eat
at                  at
same                same
time                time
He                  He
has                 have
bad                 bad
habit               habit
of                  of
swimming            swim
after               after
playing             play
long                long
hours               hours
in                  in
the                 the
Sun                 Sun
```

# Stemming or lemmatization?

Stemming and Lemmatization both generate the root form of the inflected words. The difference is that stem might not be an actual word whereas, lemma is an actual language word.

Stemming follows an algorithm with steps to perform on the words which makes it faster. Whereas, in lemmatization, you used WordNet corpus and a corpus for stop words as well to produce lemma which makes it slower than stemming. You also had to define a parts-of-speech to obtain the correct lemma.

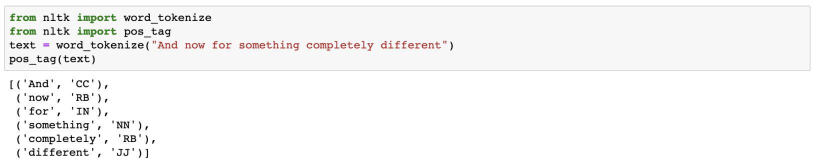
**If speed is focused then stemming should be used** since lemmatizers scan a corpus which consumed time and processing. It depends on the application you are working on that decides if stemmers should be used or lemmatizers. If you are building a language application in which **language is important you should use lemmatization** as it uses a corpus to match root forms.

# POS tagging

Parts of speech Tagging is responsible for reading the text in a language and assigning some specific tags (Parts of Speech) to each word. Example:
`Input: Everything to permit us.`
`Output: [(’Everything’, NN),(’to’, TO), (’permit’, VB), (’us’, PRP)]`

**Steps involved:**

- Tokenize text (`word_tokenize`)
- Apply pos_tag to above step: `nltk.pos_tag(tokenize_text)`



```
from nltk import word_tokenize
from nltk import pos_tag
text = word_tokenize("And now for something completely different")
pos_tag(text)
```

```
[('And', 'CC'),
 ('now', 'RB'),
 ('for', 'IN'),
 ('something', 'NN'),
 ('completely', 'RB'),
 ('different', 'JJ')]
```

# Lexical semantics: synsets

# WordNet

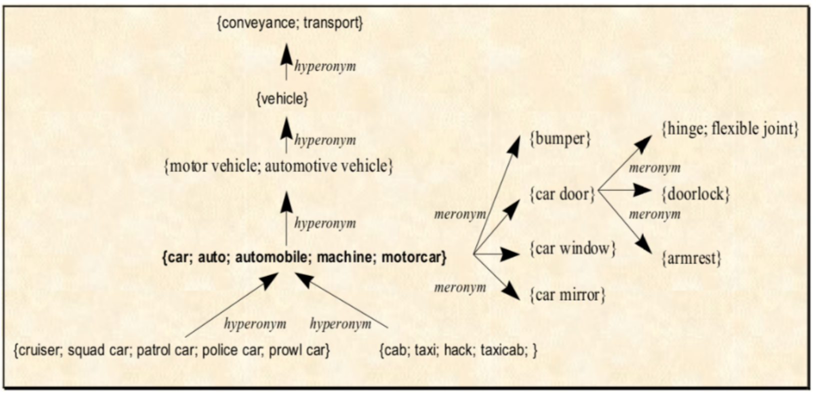
- Free large lexical database of English
- Contains only nouns, verbs, adjectives and adverbs
- Words are grouped into synonyms sets (synsets)
- each synset has an associated gloss and some examples
- synsets are interlinked by means of lexical relations
- Source: `http://wordnetweb.princeton.edu/perl/webwn`

# Lexical relations

- **Synonym:** same meaning. Ex: age - historic period
- Antonym: opposite meaning. Ex: dark - light
- Homophome: same sound. Ex: son - sun
- Homograph: same written form. Ex: lead (noun - verb)
- Polysemy: different related meaning. Ex: newspaper (paper - firm)
- Homonymy: different unrelated meaning. Ex: position (place - status)
- **Hypernym:** parent. Ex: cat - feline
- **Hyponym:** child. Ex: feline - cat
- Holonymy: group, whole. Ex: class - student
- Meronymy: member, part. Ex: student - class
- Metonymy: substitution of entity. Ex: We ordered many delicious dishes at the restaurant.

# Lexical relations

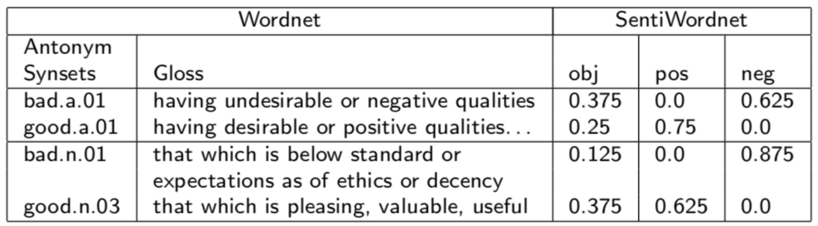
Example of lexical realtion net:



# SentiWordNet

Extension of wordnet that adds for each synset 3 measures:

- positive_score
- negative_score
- objective_score = 1 - positive_score - negative_score



| Wordnet | | SentiWordnet | | |
|---|---|---|---|---|
| Antonym Synsets | Gloss | obj | pos | neg |
| bad.a.01 | having undesirable or negative qualities | 0.375 | 0.0 | 0.625 |
| good.a.01 | having desirable or positive qualities. . . | 0.25 | 0.75 | 0.0 |
| bad.n.01 | that which is below standard or expectations as of ethics or decency | 0.125 | 0.0 | 0.875 |
| good.n.03 | that which is pleasing, valuable, useful | 0.375 | 0.625 | 0.0 |

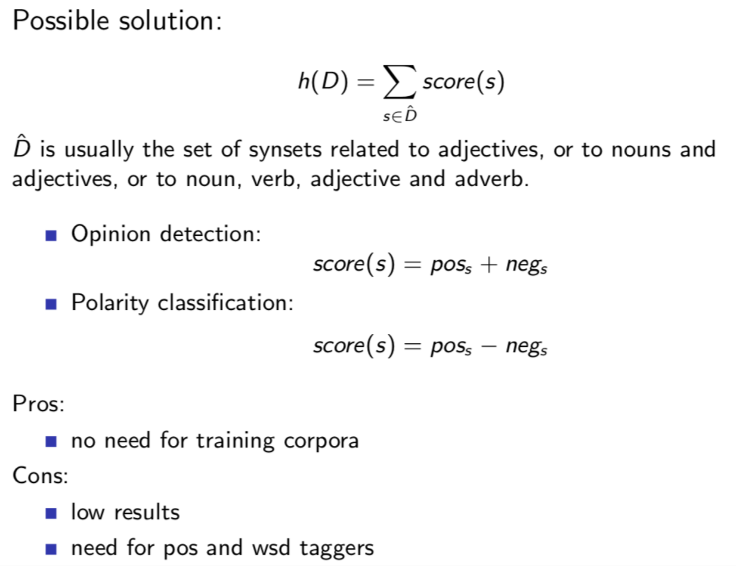
# Sentiment Analysis

# Sentiment Analysis

Different subtasks:

- Opinion detection: given a piece of text (document or sentence), is it an objective text or a subjective one?
- Polarity classification: given a subjective piece of text, is it a positive opinion or a negative one?
- Opinion extraction: given a subjective piece of text, recognise the focuses of the opinion (templates ¡entity, aspect, polarity¿).

# Unsupervised Sentiment Analysis



Possible solution:

$$h(D) = \sum_{s \in \hat{D}} score(s)$$

$\hat{D}$ is usually the set of synsets related to adjectives, or to nouns and adjectives, or to noun, verb, adjective and adverb.

- Opinion detection:

$$score(s) = pos_s + neg_s$$

- Polarity classification:

$$score(s) = pos_s - neg_s$$

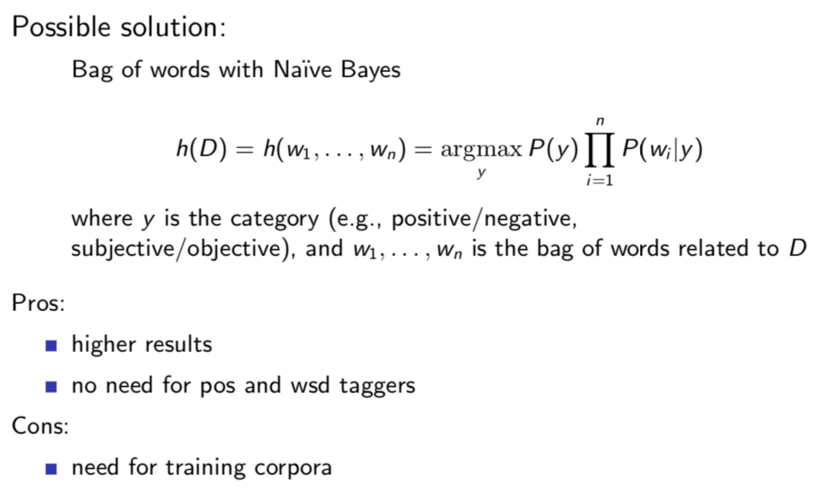
Pros:

- no need for training corpora

Cons:

- low results
- need for pos and wsd taggers

# Supervised Sentiment Analysis



Possible solution:

Bag of words with Naïve Bayes

$$h(D) = h(w_1, \ldots, w_n) = \operatorname*{argmax}_y P(y) \prod_{i=1}^{n} P(w_i|y)$$

where $y$ is the category (e.g., positive/negative, subjective/objective), and $w_1, \ldots, w_n$ is the bag of words related to $D$

Pros:

- higher results
- no need for pos and wsd taggers

Cons:

- need for training corpora

# Hybrid approach for Sentiment Analysis

Possible solution:

- Combine two supervised methods with SentiWordnet method
- I.e., consensuate the output of the three methods, using voting, for instance:
  - if at least 2 of the methods answer y then output y else output the answer of the method with better accuracy in the training corpus
  - The combination improves the results of the isolated methods

# Today’s homework

# Exercise

1. Extend the solution of exercise 4 from yesterday's session to the use of lemmas and other preprocess issues.
2. Notebook