LISBON
SCHOOL OF
ECONOMICS &
MANAGEMENT
UNIVERSIDADE DE LISBOA

Carlos J. Costa

NLP WITH NLTK AND OTHER LIBRARIES

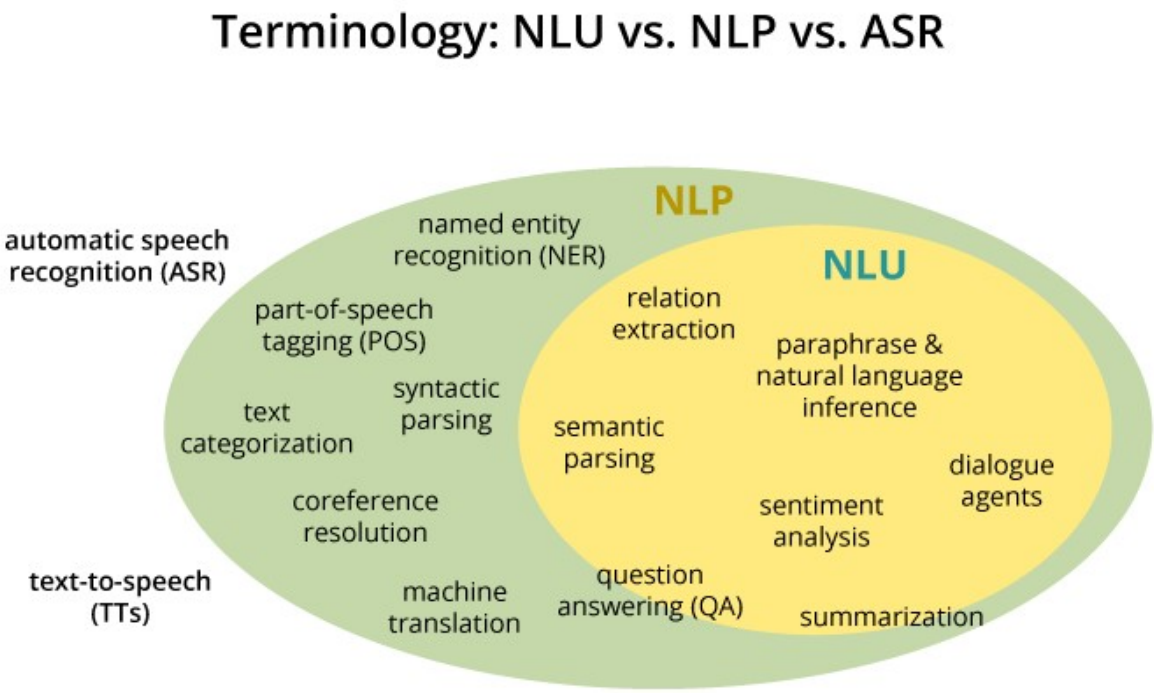
2021

Agenda

- NLP
- NLTK
- Stemming vs. Lemmatization
- Tokenization
- Sentiment Analysis
- Classification

NLP

- Natural language processing
- Subfield of artificial intelligence, linguistics, and computer science
- Create software to process and analyze large amounts of natural language data



Information Retrieval

Sentiment Analysis

Information Extraction

Natural Language Processing (NLP)

Machine Translation

Question Answering

NLTK

- Natural Language Toolkit
- A suite of text processing libraries for:
  - Classification
  - Tokenization
  - Stemming
  - Tagging
  - Parsing
  - Semantic reasoning
  - http://www.nltk.org/book/


Stemming

Stemming

- method of normalization of words in NLP
- words in a sentence are converted into a sequence to shorten its lookup
- words having the same meaning but have some variations according to the context or sentence are normalized.

Stemming

```
from nltk.stem import PorterStemmer
e_words= ["studies", "studying", "cries", "cry"]
ps =PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
    print(rootWord),
```

```
studi
studi
cri
cri
```


Lemmatization

Lemmatization

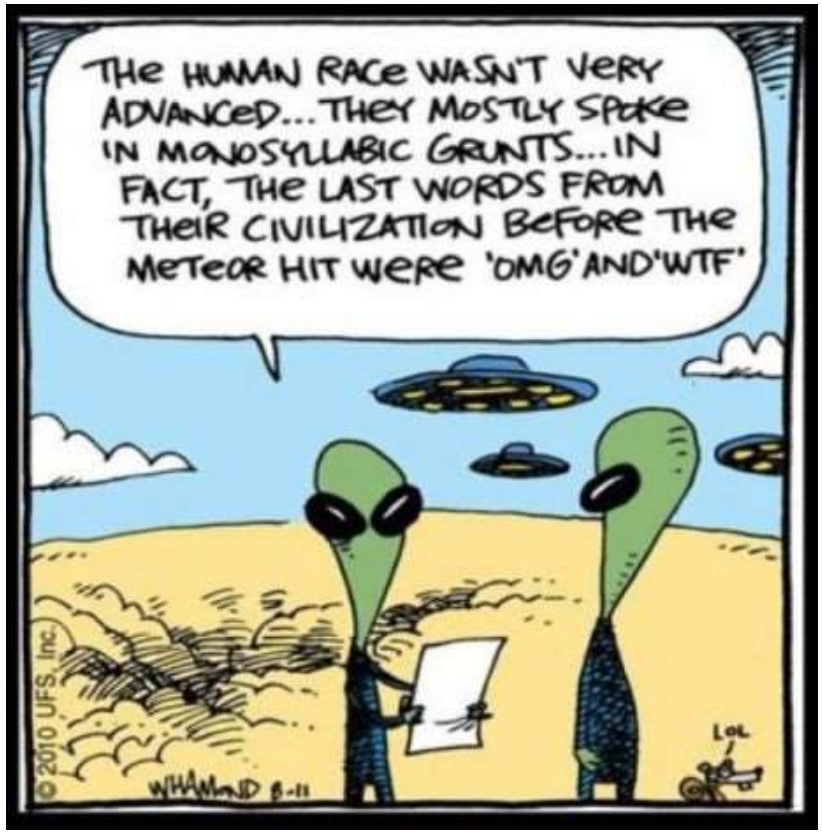
- algorithmic process of finding the lemma of a word depending on its meaning and context.
- refers to the morphological analysis of words
- aims to remove inflectional endings.
- returning the base or dictionary form of a word known as the lemma

Lemmatization

```
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
    print(wordnet_lemmatizer.lemmatize(w))
```

```
study
studying
cry
cry
```

Tokenization



Tokenization

- Splitting up a larger body of text into smaller lines

```
import nltk
sentence_data = "First, I will explain you how this work. Then, you will do it. "
nltk_tokens = nltk.sent_tokenize(sentence_data)
print (nltk_tokens)
```

```
['First, I will explain you how this work.', 'Then, you will do it.']
```

Word Tokenization

- Two possible examples of

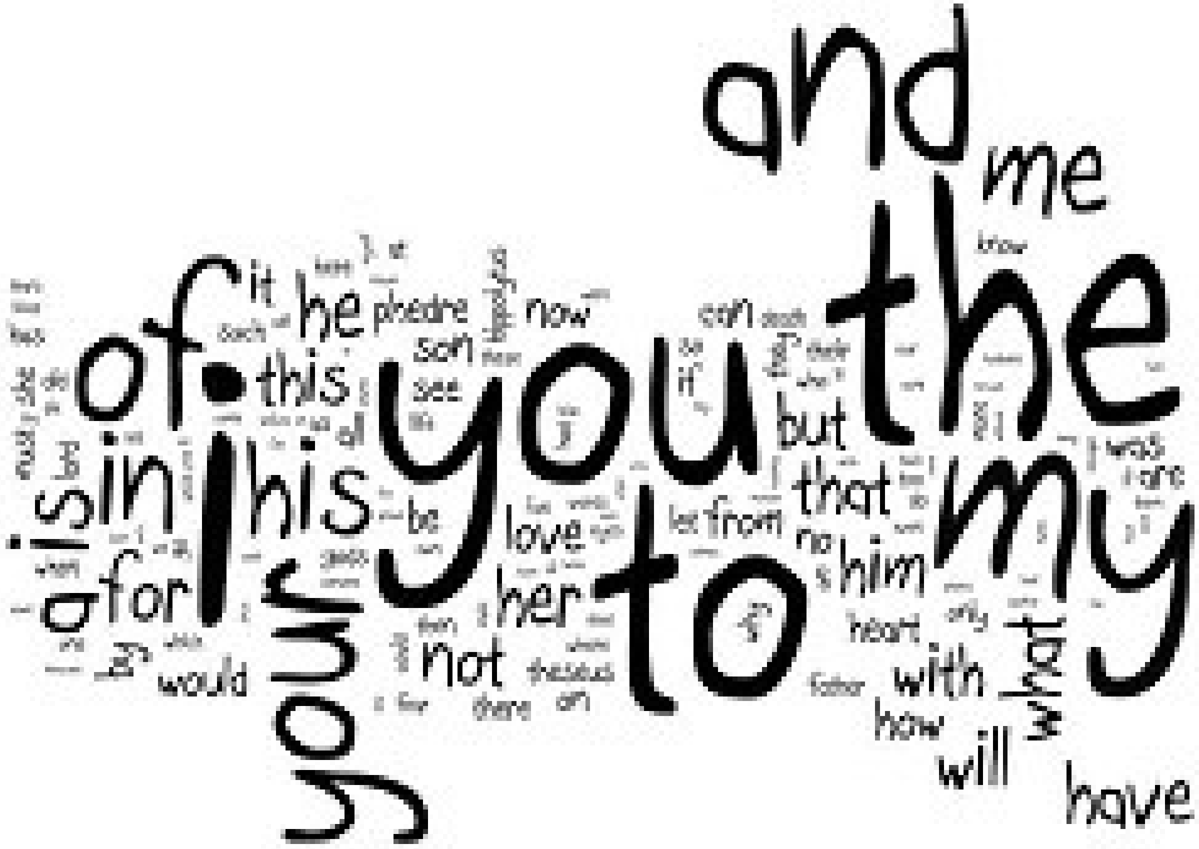
```
text = "First, I will explain you how this work. Then, you will do it."

import nltk
new_text = nltk.word_tokenize(text)
print (new_text)
```

```
['First', ',', 'I', 'will', 'explain', 'you', 'how', 'this', 'work',
'.', 'Then', ',', 'you', 'will', 'do', 'it', '.']
```

```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer(r'\w+')
new_text=tokenizer.tokenize(text)
print (new_text)
```

```
['First', 'I', 'will', 'explain', 'you', 'how', 'this', 'work', 'Then',
'you', 'will', 'do', 'it']
```

Stopword

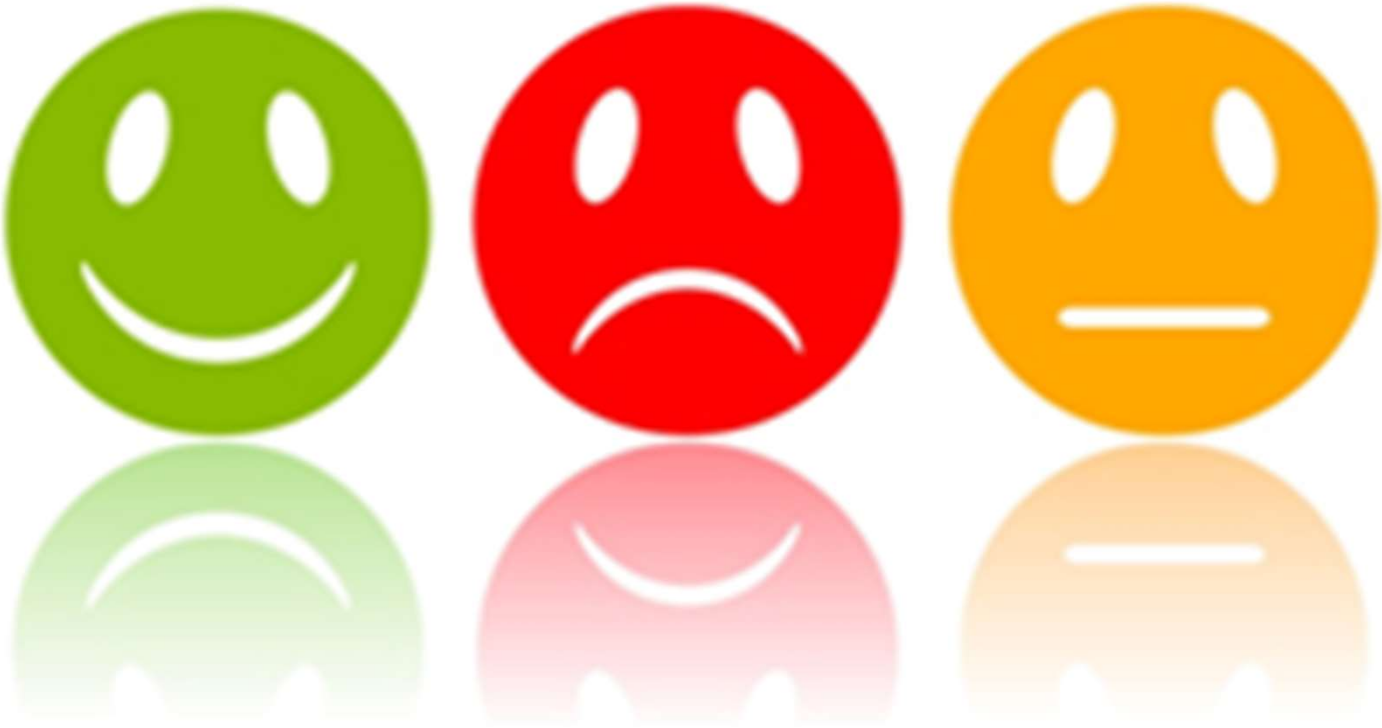


and
me
the
you
to
my
of
I
his
in
is
a
your
for
this
he
it
phedre
son
see
now
can
but
that
let
from
no
him
love
her
be
not
theseus
would
heart
with
what
how
will
have

Stopword

- Removing stop words is an essential step in NLP text processing
- filtering out high-frequency words that add little or no semantic value to a sentence
- for example to, at, for, is, etc.

Sentiment Analysis



Sentiment Analysis

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```
text="I am the winner!"
```

```
vader = SentimentIntensityAnalyzer()
```

```
vader.polarity_scores(text)
```

```
{'neg': 0.0, 'neu': 0.328, 'pos': 0.672, 'compound': 0.6239}
```

Classification

```
from nltk.corpus import names
import random
import nltk.classify as nltk
```

```
def gender_features(word):
    return {'last_letter': word[-1]}
```

```
gender_features('John')
```

```
{'last_letter': 'n'}
```

```
labeled_names = ([(name, 'male') for name in names.words('male.txt')] +
                 [(name, 'female') for name in names.words('female.txt')])
```

```
random.shuffle(labeled_names)
```

```
featuresets = [(gender_features(n), gender) for (n, gender) in labeled_names]
```

```
train_set, test_set = featuresets[500:], featuresets[:500]
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

```
print(nltk.accuracy(classifier, test_set))
```

```
0.758
```

Classification

- https://www.nltk.org/book/ch06.html
- https://pythonprogramming.net/text-classification-nltk-tutorial/
- https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

Semantics

```
import nltk
from nltk.sem import Valuation, Model

v = [('adam', 'b1'), ('betty', 'g1'), ('fido', 'd1'),
     ('girl', set(['g1', 'g2'])), ('boy', set(['b1', 'b2'])),
     ('dog', set(['d1'])),
     ('love', set([('b1', 'g1'), ('b2', 'g2'), ('g1', 'b1'), ('g2', 'b1')]))]
val = Valuation(v)
dom = val.domain
m = Model(dom, val)
```

```
g = nltk.sem.Assignment(dom)
```

```
m.evaluate('all x.(boy(x) -> - girl(x))', g)
```

```
True
```

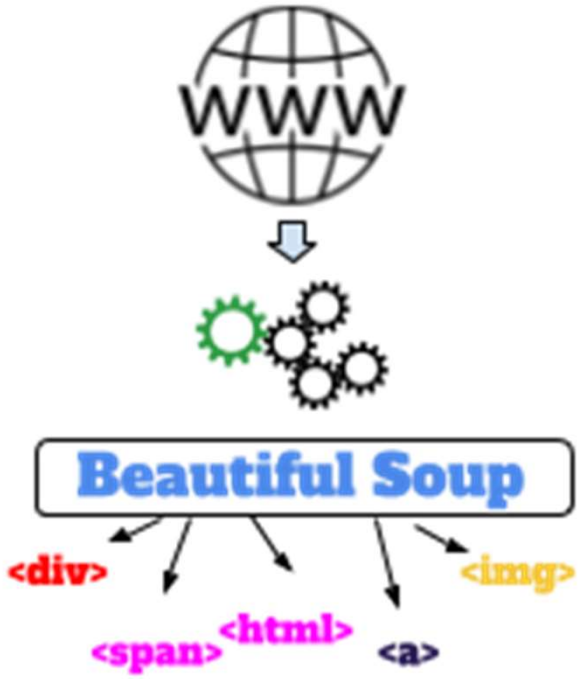
- http://www.nltk.org/howto/semantics.html

Other tools…


Web Scraping

Web Scraping

- Beautiful Soup is a HTML parser.
- This Python library is designed for screen-scraping projects.
- Three features make it powerful:
  - navigating, searching, and modifying a parse tree
  - converts incoming documents to Unicode and outgoing documents to UTF-8
  - sits on top of popular Python parsers like lxml and html5lib.
- https://www.crummy.com/software/BeautifulSoup/

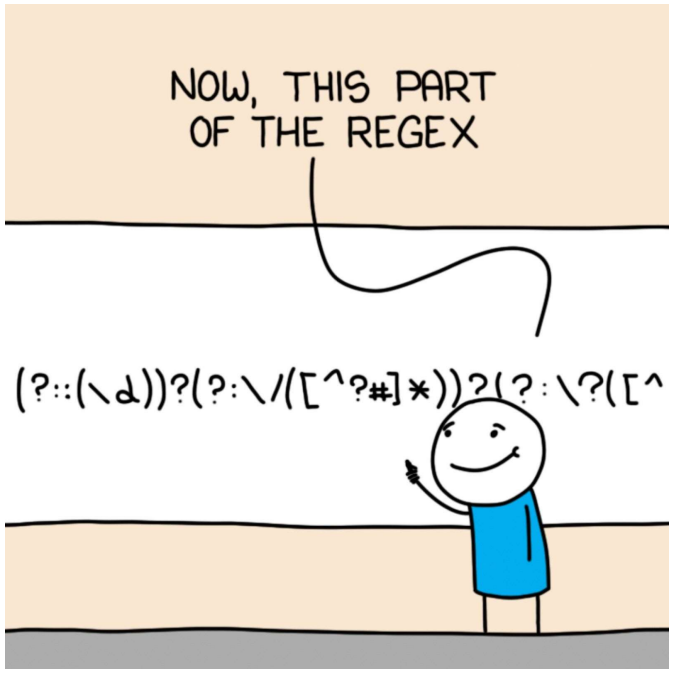


Scraping

```
import urllib.request
response =  urllib.request.urlopen('https://en.wikipedia.org/wiki/Digital_transformation')
html = response.read()
```

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html,'html5lib')
text = soup.get_text(strip = True)
```

Regular Expression



- Sequence of characters that specifies a search pattern.
- https://docs.python.org/3/howto/regex.html

Other libraries

References

- https://www.guru99.com/nltk-tutorial.html