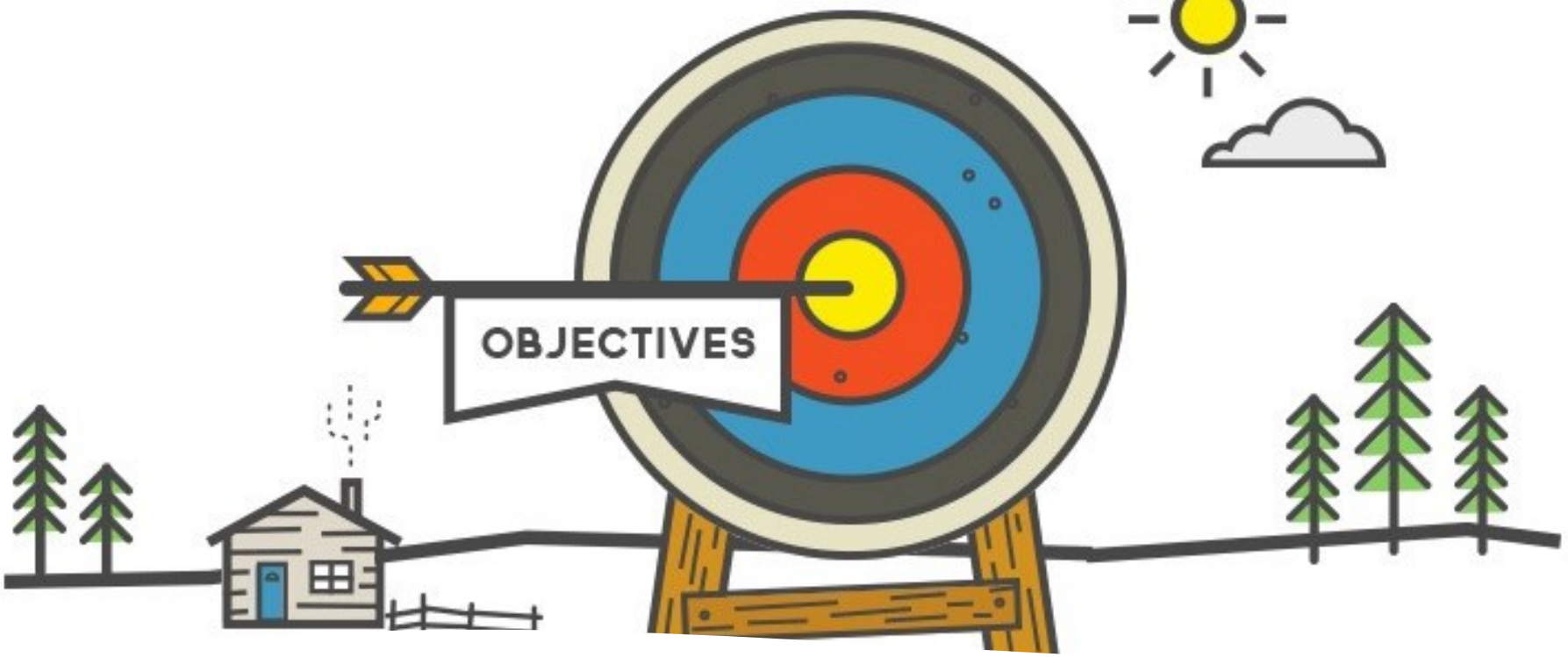
Lisbon School of Economics & Management
Universidade de Lisboa

U LISBOA
UNIVERSIDADE DE LISBOA

NATURAL LANGUAGE PROCESSING (NLP)

Carlos J. Costa

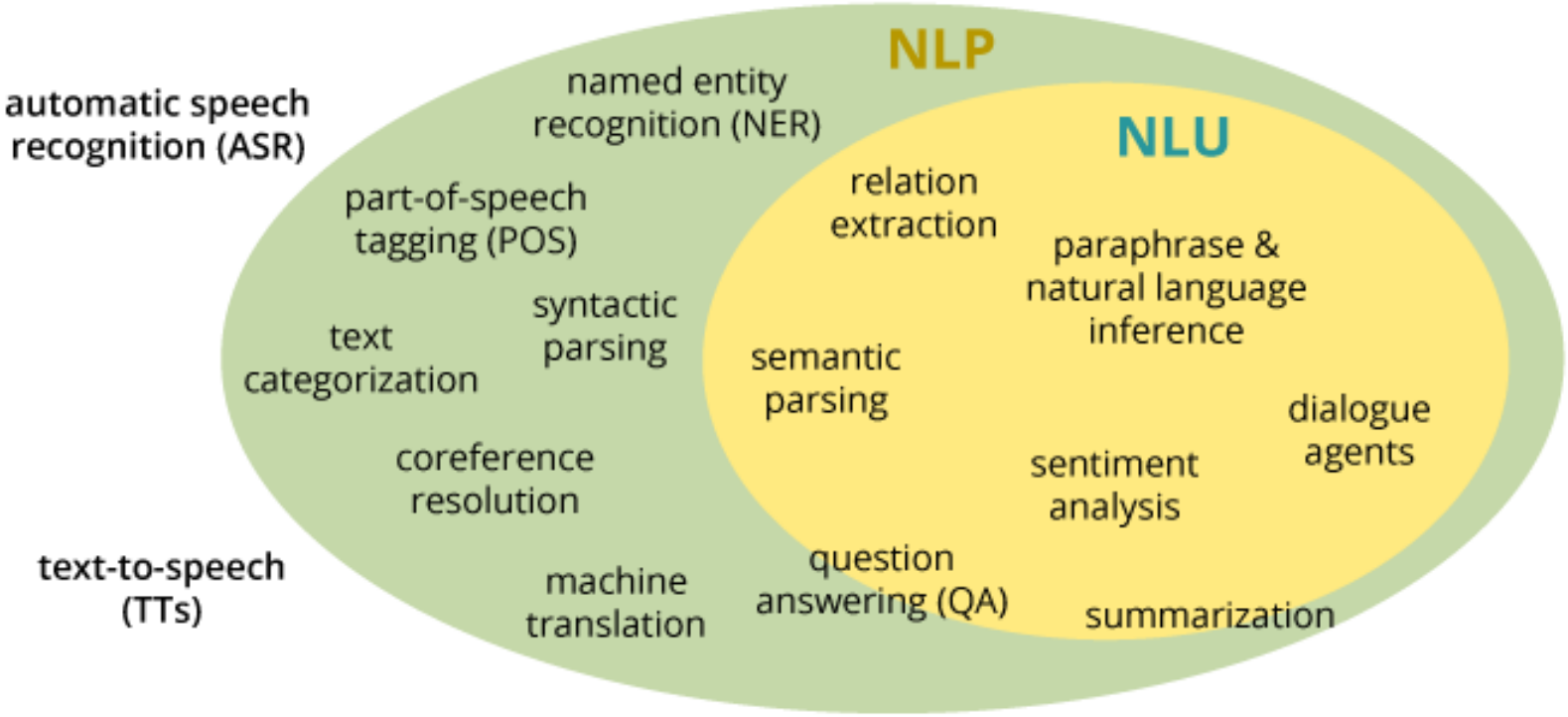


Learning Goals

- Understand the context of NLP
- Explain main concepts
- Use main libraries

Table of Contents

- Definition
- Main Concepts
- Libraries



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

NLP

- Tokenization
- Stemming
- Lemmatization
- Part-of-Speech (POS) Tagging
- Named Entity Recognition (NER)
- Sentiment Analysis
- Language Modeling
- Machine Translation
- Text Summarization
- Information Extraction
- Question Answering
- Text Classification
- Topic Modeling
- Dependency Parsing
- Discourse Analysis

Tokenization

Breaking down text into smaller units such as words, phrases, or sentences.

```
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.']
```

```
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']
```

Stemming

Removing suffixes or prefixes from words to obtain their root form.

```
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

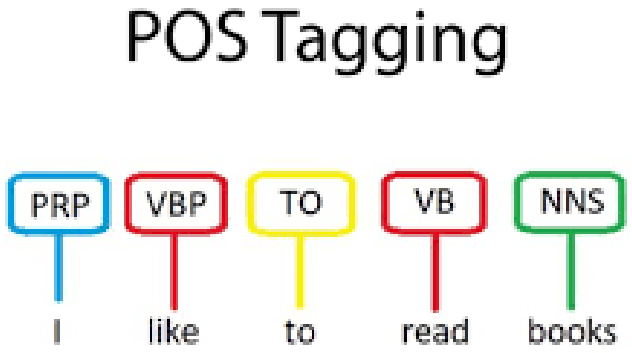
Reducing words to their base or dictionary form while still ensuring they are valid words.

```
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
```

Part-of-Speech (POS) Tagging

Assigning grammatical categories (such as noun, verb, adjective, etc.) to words in a sentence.




```python
#  Part-of-Speech (POS) Tagging

import nltk

text = "I am waiting for the end of the class."
tokens = nltk.word_tokenize(text)
pos_tags = nltk.pos_tag(tokens)
print(pos_tags)

```

```
[('I', 'PRP'), ('am', 'VBP'), ('waiting', 'VBG'), ('for', 'IN'), ('the', 'D
T'), ('end', 'NN'), ('of', 'IN'), ('the', 'DT'), ('class', 'NN'), ('.', '.')]
```

Named Entity Recognition (NER)

Identifying and classifying named entities (such as person names, locations, organizations, etc.) in text.

```python
# Named Entity Recognition (NER):

import nltk

text = "The campus of ISEG is in Lisbon, Portugal. Carlos works there."
tokens = nltk.word_tokenize(text)
tags = nltk.pos_tag(tokens)
entities = nltk.chunk.ne_chunk(tags)
print(entities)
```

```
(S
  The/DT
  campus/NN
  of/IN
  (ORGANIZATION ISEG/NNP)
  is/VBZ
  in/IN
  (GPE Lisbon/NNP)
  ,/,
  (GPE Portugal/NNP)
  ./.
  (PERSON Carlos/NNP)
  works/VBZ
  there/RB
  ./.)
```

Sentiment Analysis

Determining the sentiment or opinion expressed in text, typically categorized as positive, negative, or neutral.

```
#  Sentiment Analysis:

from nltk.sentiment.vader import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()
text = "I love this movie! It's amazing."
sentiment_scores = analyzer.polarity_scores(text)
print(sentiment_scores)
```

```
{'neg': 0.0, 'neu': 0.266, 'pos': 0.734, 'compound': 0.8516}
```

Language Modeling

Building statistical models to predict the next word in a sequence of words, often used in machine translation, autocomplete, and speech recognition.

```
:
```

```python
# Language Modeling:

import nltk
# nltk.download('gutenberg')
#nltk.corpus.gutenberg.fileids()

text = nltk.corpus.gutenberg.raw('bible-kjv.txt')
#text = nltk.corpus.gutenberg.raw('shakespeare-hamlet.txt')
words = nltk.word_tokenize(text)
bigrams = nltk.ngrams(words, 2)
model = nltk.ConditionalFreqDist(bigrams)
print(model["face"].most_common(5))  # Predict next word after "to"

```

```
[('of', 109), (',', 79), ('to', 43), ('.', 31), ('from', 20)]
```

Machine Translation

Translating text from one language to another automatically using computational methods.

```python

from deep_translator import GoogleTranslator

def translate_text(text, dest_lang='es'):
    try:
        translated_text = GoogleTranslator(source='auto', target=dest_lang).translate(text)
        return translated_text
    except Exception as e:
        return f"An error occurred: {str(e)}"

# Example usage
translated_text = translate_text("Das funktioniert tatsächlich gut.", dest_lang='pt')
print(translated_text)
```

```
Isso realmente funciona bem.
```

Text Summarization

Generating concise summaries of longer text documents while preserving the most important information.

```python
from transformers import pipeline

# Load the summarization pipeline with a specific model
summarizer = pipeline("summarization", model="t5-base", tokenizer="t5-base")

# Example text
text = "We are currently exploring the applications of Artificial Intelligence in management and economics. Our focus exte

# Generate summary
summary = summarizer(text, max_length=50, min_length=10, do_sample=False)[0]['summary_text']

# Print the summary
print(summary)
```

```
our focus extends beyond studying the use of AI to include the management of artificial intelligence . we have been significa
nt in organizing annual Artificial Intelligence and Management workshops since 2019 .
```

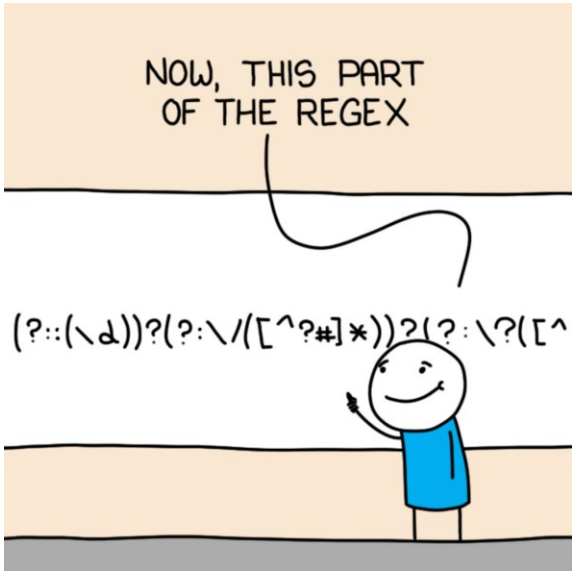
```python
# Information Extraction:

import re

text = "João Duque is the dean of ISEG, University of Lisbon. Luís Carriço is the dean of FCUL, University of Lisbon."
matches = re.findall(r'([A-Za-zÀ-ÖØ-öø-ÿ]+)\s+([A-Za-zÀ-ÖØ-öø-ÿ]+)\s+is\s+the\s+dean\s+of\s+([^,.]+)', text)
print(matches)

```

```
[('João', 'Duque', 'ISEG'), ('Luís', 'Carriço', 'FCUL')]
```

Information Extraction

Automatically extracting structured information from unstructured text, such as extracting entities, relations, and events.

Regular Expression



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

Question Answering

Building systems that can understand and answer questions posed in natural language.

```python
#Question Answering:

from transformers import pipeline

qa_pipeline = pipeline("question-answering", model="bert-large-uncased-whole-word-masking-finetuned-squad",
                       tokenizer="bert-large-uncased-whole-word-masking-finetuned-squad")
context = """Natural Language Processing (NLP) is a subfield of artificial intelligence that focuses on the
           interaction between computers and humans through natural language. """
question = "What is NLP?"
answer = qa_pipeline(question=question, context=context)
print(answer['answer'])

```

```
Natural Language Processing
```

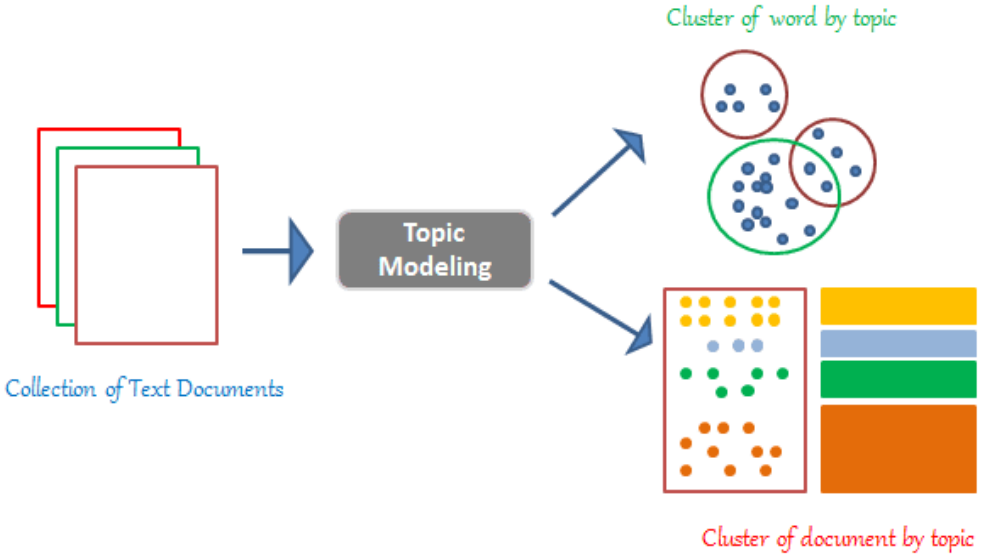
Text Classification

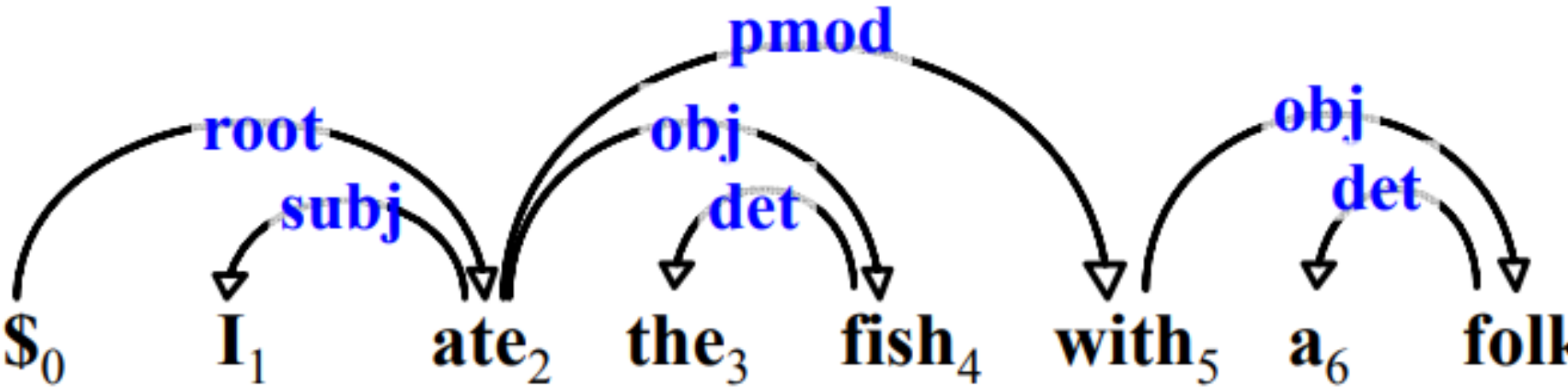
Categorizing text documents into predefined categories or classes, such as spam detection, sentiment classification, topic classification, etc.

```python
# Text Classification:

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X = ["I love this movie!", "This movie is terrible."]
y = [1, 0]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = make_pipeline(CountVectorizer(), MultinomialNB())
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
```

Topic Modeling

Identifying topics or themes present in a collection of documents, often using techniques like Latent Dirichlet Allocation (LDA).





Dependency Parsing

Analyzing the grammatical structure of sentences to determine the relationships between words.

Discourse Analysis

Studying the organization and structure of connected texts beyond the sentence level, including coherence, cohesion, and rhetorical relations.

```python
# Discourse Analysis

from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize

# Example text with multiple sentences
text = """
Natural Language Processing (NLP) is a subfield of artificial intelligence.
It focuses on the interaction between computers and humans through natural language.
NLP techniques are used in various applications such as machine translation, sentiment
analysis, and text summarization.
"""

# Tokenize the text into sentences
sentences = sent_tokenize(text)

# Tokenize each sentence into words and remove stopwords
stop_words = set(stopwords.words('english'))
tokenized_sentences = [word_tokenize(sentence) for sentence in sentences]
filtered_sentences = [[word for word in tokens if word.lower() not in stop_words] for tokens in tokenized_sentences]

# Compute lexical chains based on word similarity
def compute_lexical_chains(sentences):
    chains = []
    for i, sentence in enumerate(sentences):
        chain = []
        for word in sentence:
            for j, prev_sentence in enumerate(sentences[:i]):
                if word in prev_sentence:
                    chain.append((word, j))
                    break
        chains.append(chain)
    return chains

# Compute lexical chains for the example
lexical_chains = compute_lexical_chains(filtered_sentences)

# Print the lexical chains for each sentence
for i, chain in enumerate(lexical_chains):
    print(f"Sentence {i+1} Lexical Chain: {chain}")

```

```
Sentence 1 Lexical Chain: []
Sentence 2 Lexical Chain: [('.', 0)]
Sentence 3 Lexical Chain: [('NLP', 0), ('.', 0)]
```

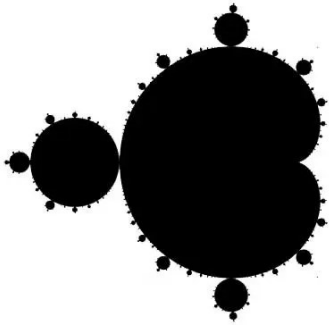
Web Scraping

- 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/

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.

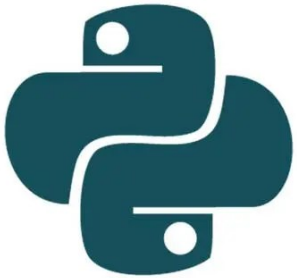
Libraries



TextBlob

spaCy

GENSIM
topic modelling for humans



NLTK

NLTK

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

Conclusions

- NLP as subset of AI and CS
- Main concepts and Applications
- Main libraries