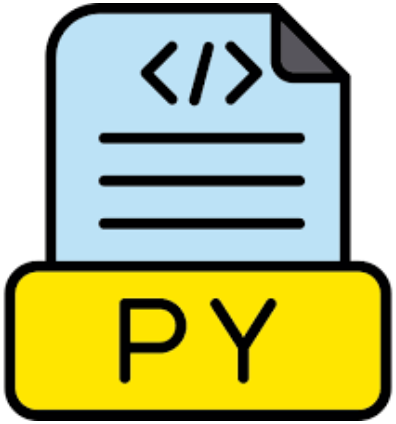
Introduction au Traitement Automatique des Langues

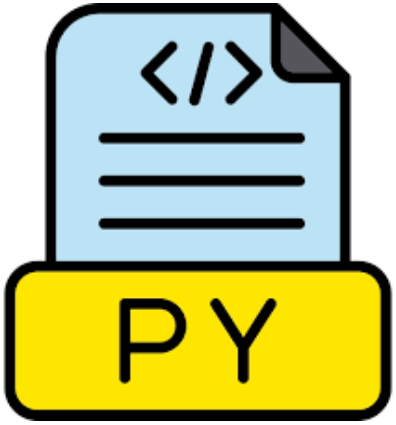
5 – Les niveaux de traitement – Le niveau Syntaxique

Niveaux de Traitement - Analyse Synatxique

- **Série TP 4 et 5 – Analyse Syntaxique avec NLTK et spaCy**



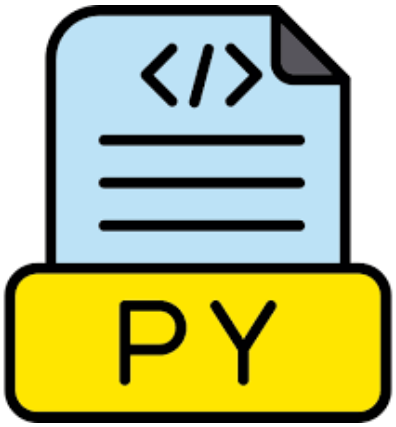
TP 4 – Parse Trees



TP 5 - spaCy

Niveaux de Traitement - Analyse Synatxique

- **Série TP 4 et 5 – Analyse Syntaxique avec NLTK et spaCy**

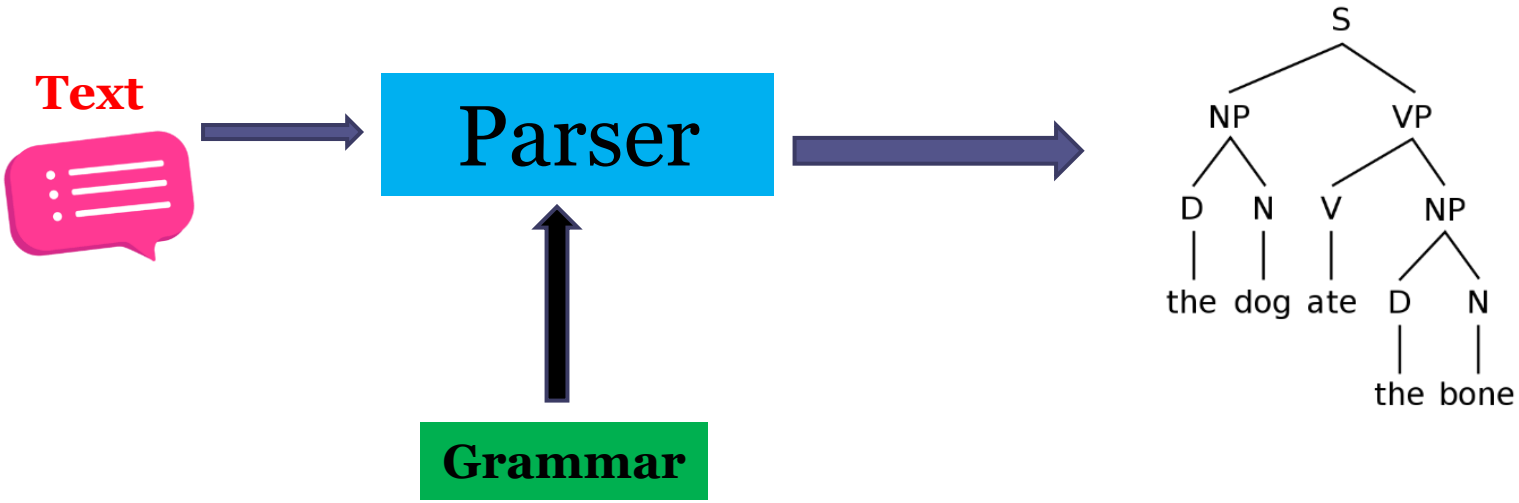


TP 4 – Parse Trees

TP 5 - spaCy

Série TP 4 – Arbres Syntaxiques avec NLTK

- **Analyse Syntaxique**



Série TP 4 – Arbres Syntaxiques avec NLTK

```
mygrammar_str = """
```

Grammar

```
"""
```

Série TP 4 – Arbres Syntaxiques avec NLTK

Grammar

```
mygrammar_str = """
  S -> NP VP
  VP -> V NP | V NP PP
  PP -> P NP
  NP -> Det N | Det N PP
  V -> 'saw' | 'ate' | 'walked' | 'shot'
  NP -> 'John' | 'Mary' | 'Bob'
  Det -> 'a' | 'an' | 'the' | 'my'
  N -> 'girl' | 'dog' | 'telescope' | 'bone'
  P -> 'in' | 'on' | 'by' | 'with'
  """
```

Série TP 4 – Arbres Syntaxiques avec NLTK

Grammar

```
mygrammar_str = """
  S -> NP VP
  VP -> V NP | V NP PP
  PP -> P NP
  NP -> Det N | Det N PP
  V -> 'saw' | 'ate' | 'walked' | 'shot'
  NP -> 'John' | 'Mary' | 'Bob'
  Det -> 'a' | 'an' | 'the' | 'my'
  N -> 'girl' | 'dog' | 'telescope' | 'bone'
  P -> 'in' | 'on' | 'by' | 'with'
  """
```

```
cfg = nltk.CFG.fromstring(mygrammar_str)
```

Série TP 4 – Arbres Syntaxiques avec NLTK

```
text = “the dog ate the bone”

tokens = nltk.word_tokenize(text)
```

Série TP 4 – Arbres Syntaxiques avec NLTK

https://www.nltk.org/api/nltk.parse.html

Parser

Série TP 4 – Arbres Syntaxiques avec NLTK

https://www.nltk.org/api/nltk.parse.html

Parser

```
chart_parser = ChartParser(cfg)
```

```
trees = chart_parser.parse(tokens)
```

Return type:

iter(**Tree**)

Série TP 4 – Arbres Syntaxiques avec NLTK

https://www.nltk.org/api/nltk.parse.html

Parser

```
chart_parser = ChartParser(cfg)
trees = chart_parser.parse(tokens)

rd_parser = RecursiveDescentParser(cfg)
trees = rd_parser.parse(tokens)
```

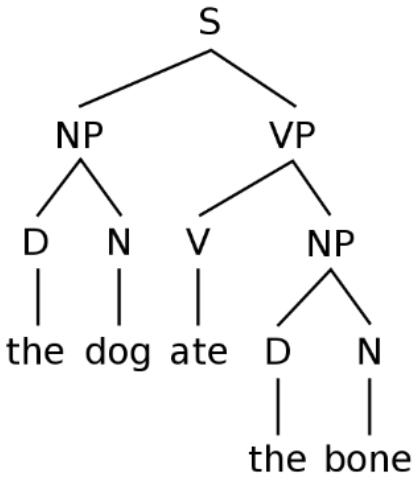
Série TP 4 – Arbres Syntaxiques avec NLTK

```
from nltk.parse.chart import BottomUpChartParser

cfg = nltk.CFG.fromstring("""
  S -> NP VP
  VP -> V NP
  NP -> 'I'  | 'John'  | 'apples'
  V -> 'eat'
""")

cyk_parser = BottomUpChartParser(cfg)
trees = cyk_parser.parse(tokens)
```

CYK Parser + CNF Grammar

Série TP 4 – Arbres Syntaxiques avec NLTK

```
for tree in trees:
    print(tree)
    tree.draw()
    tree.pretty_print()
```



(S (NP (Det the) (N dog)) (VP (V ate) (NP (Det the) (N bone))))

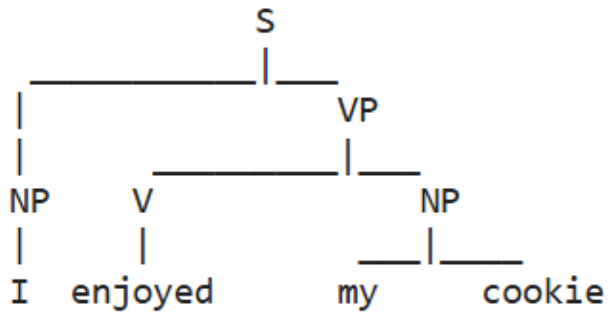
Série TP 4 – Arbres Syntaxiques avec NLTK

Tree objects from string

```
# Tree declaration in NLTK : Tree objects
mytree = nltk.Tree.fromstring('(S (NP I) (VP (V enjoyed) (NP my cookie)))')
print(mytree)
```

```
(S (NP I) (VP (V enjoyed) (NP my cookie)))
```

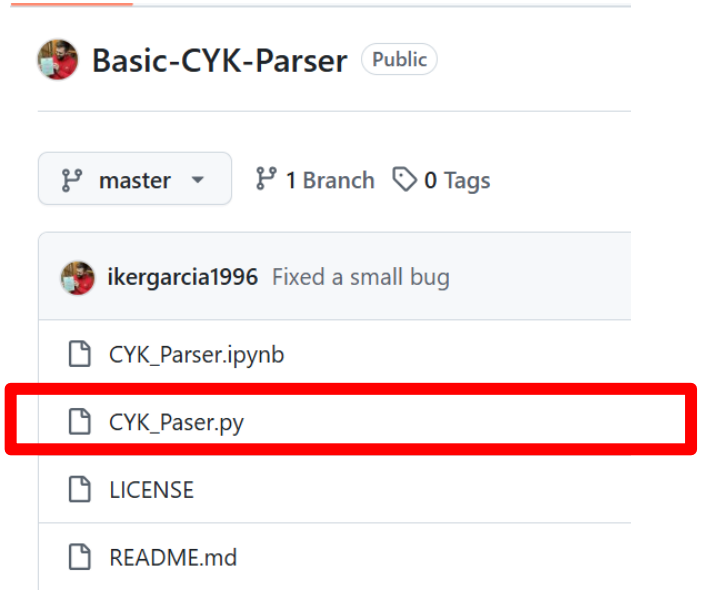
```
mytree.pretty_print()
```

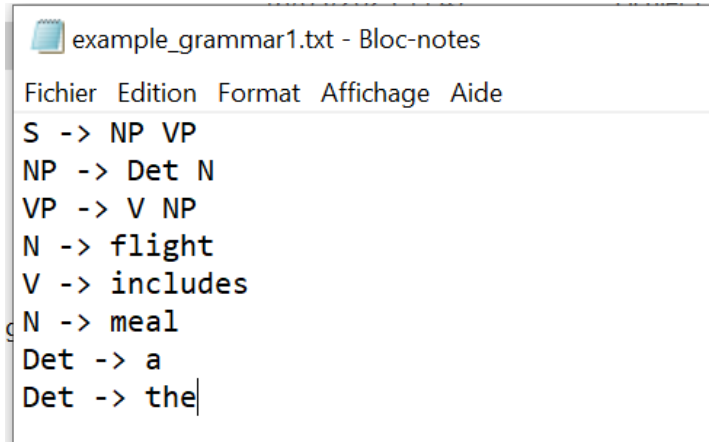


Série TP 4 – Arbres Syntaxiques avec NLTK

Basic CYK Parser in Python

- Source : https://github.com/ikergarcia1996/Basic-CYK-Parser
- Author : Iker García Ferrero
- File : CYK_Parser.py





Série TP 4 – Arbres Syntaxiques avec NLTK

```
from CYK_Paser import Grammar

grmr = Grammar('example_grammar1.txt')

text = "the flight includes a meal"

grmr.parse(text)
```

```
Applied Rule: NP[2,1] --> Det[1,1] N[1,2]
Applied Rule: NP[2,4] --> Det[1,4] N[1,5]
Applied Rule: VP[3,3] --> V[1,3] NP[2,4]
Applied Rule: S[5,1] --> NP[2,1] VP[3,3]
----------------------------------------
The sentence IS accepted in the language
Number of possible trees: 1
----------------------------------------
```

Série TP 4 – Arbres Syntaxiques avec NLTK

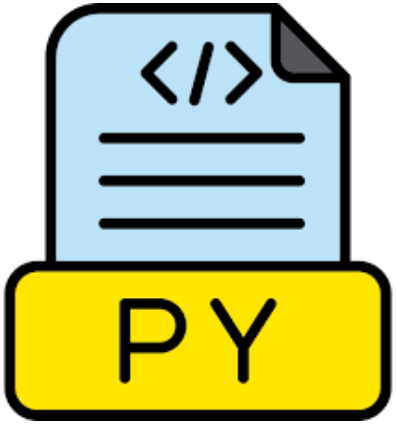
```
grmr = Grammar('example_grammar1.txt')

text = "the flight includes a meal"

grmr.print_parse_table()
```

```
------  ------  --------  -------  -----
['S']
[]      []
[]      []      ['VP']
['NP']  []      []        ['NP']
['Det'] ['N']   ['V']     ['Det']  ['N']
the     flight  includes  a        meal
------  ------  --------  -------  -----
```

Niveaux de Traitement - Analyse Synatxique

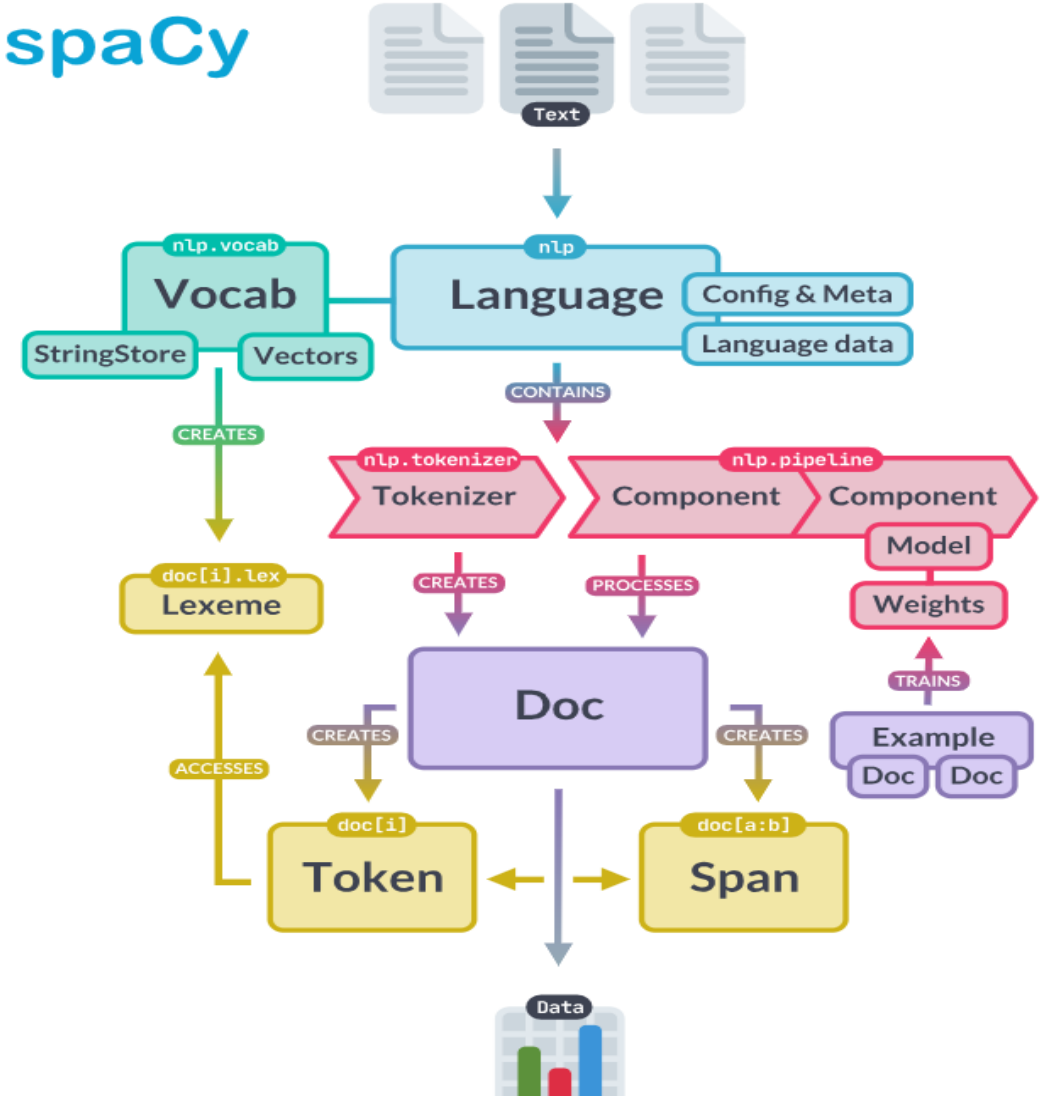
- **Série TP 4 et 5 – Analyse Syntaxique avec NLTK et spaCy**

TP 4 – Parse Trees



TP 5 - spaCy

Série TP 5 – Analyse Syntaxique avec spaCy



Série TP 5 – Analyse Syntaxique avec spaCy

- Load a pretrained spaCy statistical **language model**.

```
nlp = spacy.load('en_core_web_sm')
```

· **Size:** Package size indicator, sm (small), md (medium), lg (large) or trf (transformer).

The model uploaded above, *en_core_web_sm* is a small English pipeline trained on written web text (blogs, news, comments), that includes vocabulary, syntax and entities.

· **Type:** Capabilities (e.g., core for general-purpose pipeline with tagging, parsing, lemmatization and named entity recognition, or dep for only tagging, parsing and lemmatization). Core = Vocabulary, syntax, entities

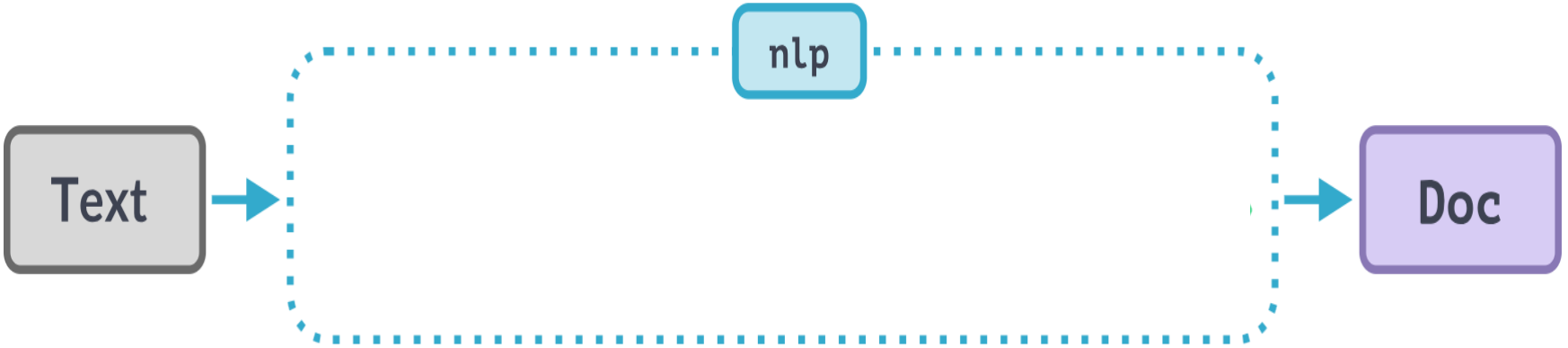
· **Genre:** Type of text the pipeline is trained on. Web = written text (blogs, news, comments)

https://spacy.io/usage/models

https://medium.com/eni-digitalks/text-preprocessing-nlp-fundamentals-with-spacy-54f32e520bc8

spaCy

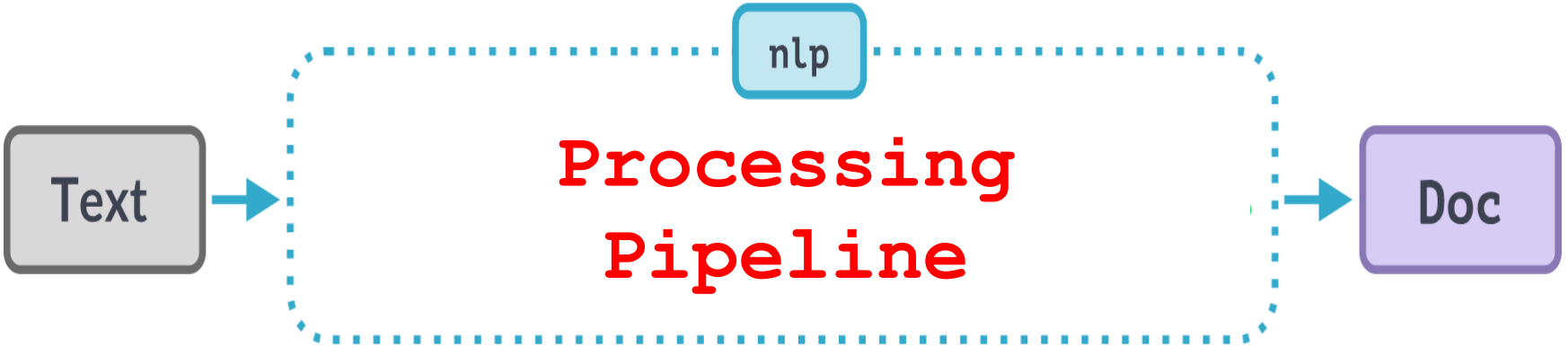
text = "I prefer the morning flight through Denver."




```
doc = nlp(text)
```

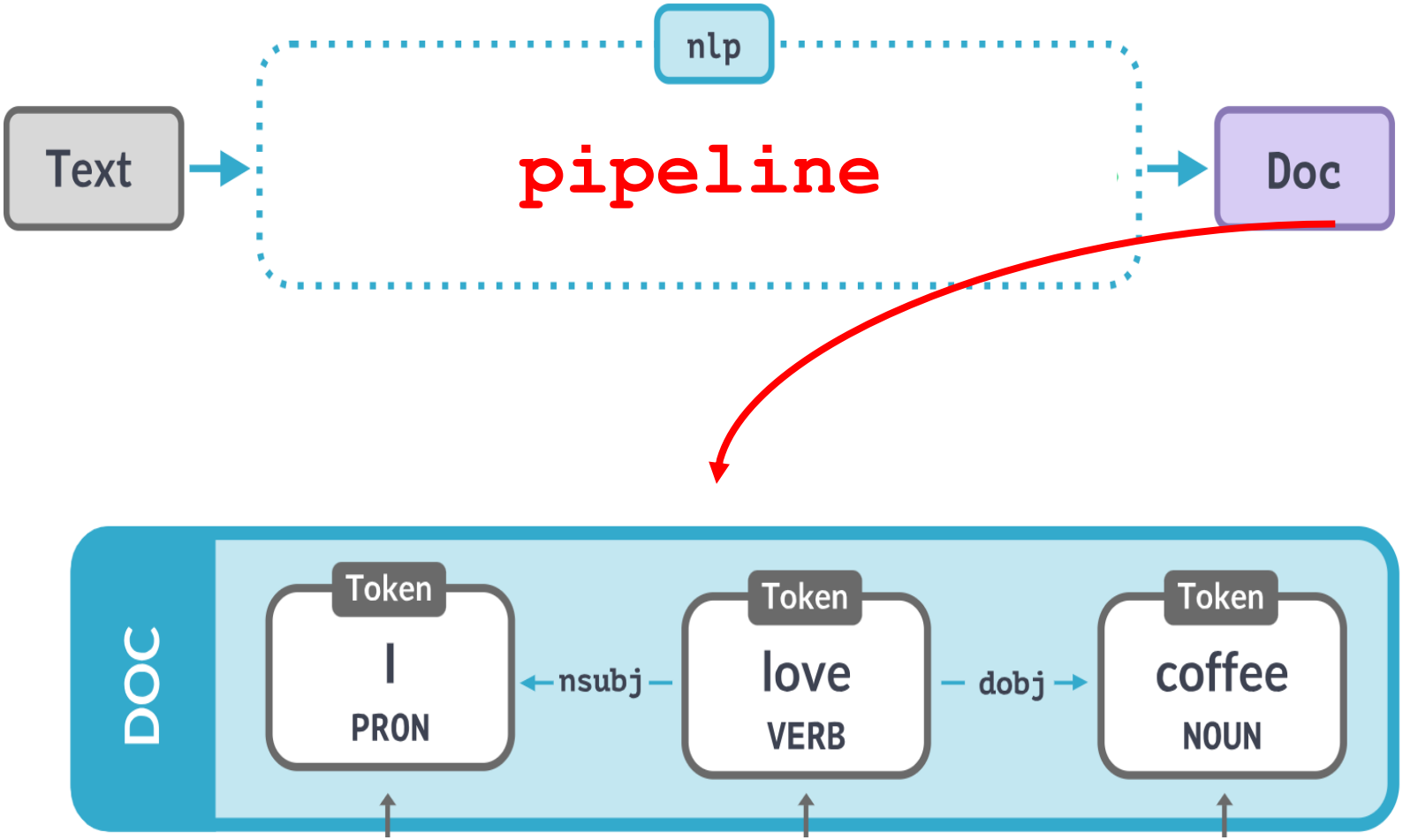
spaCy

text = "I prefer the morning flight through Denver."

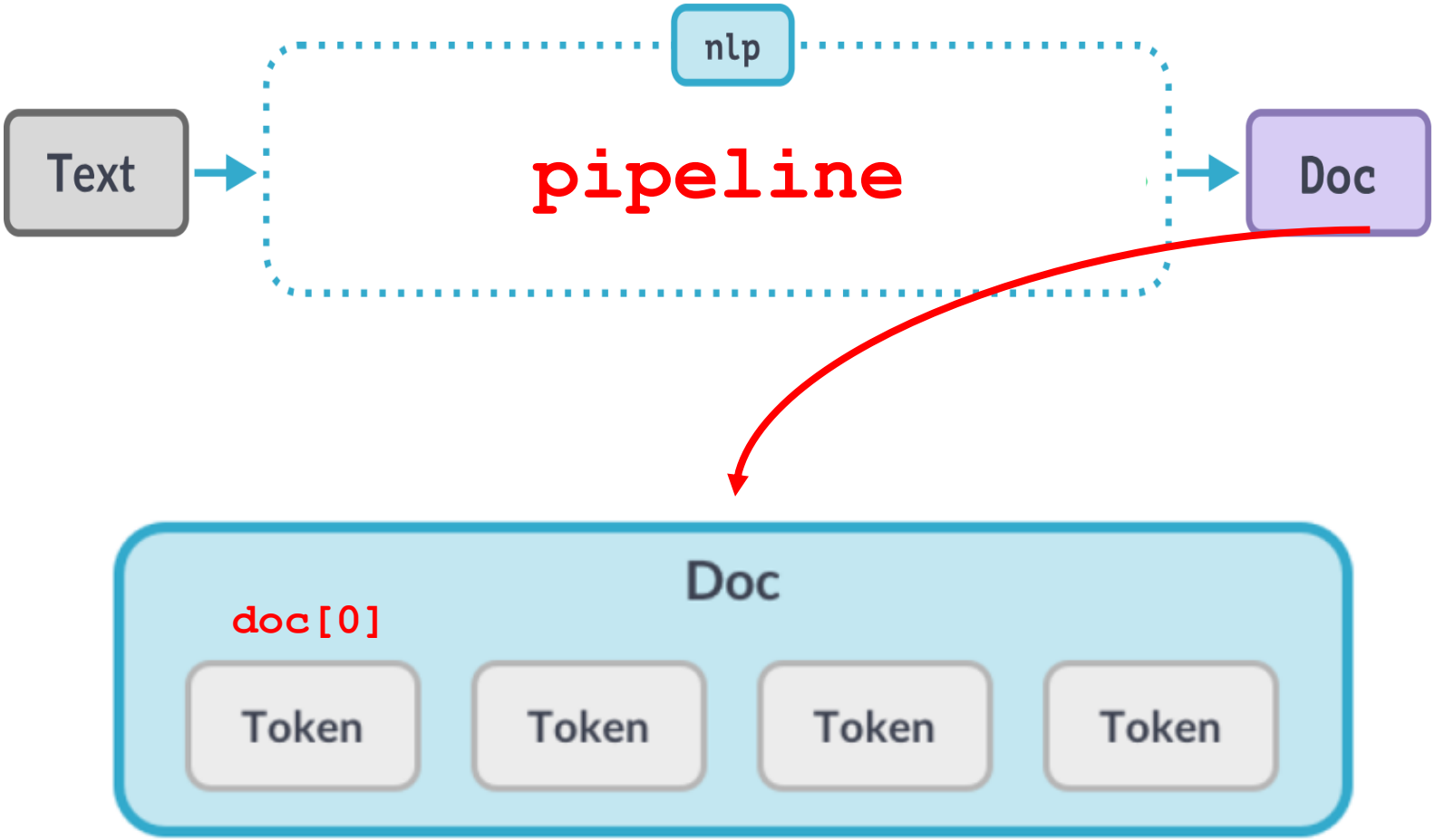



```
doc = nlp(text)
```

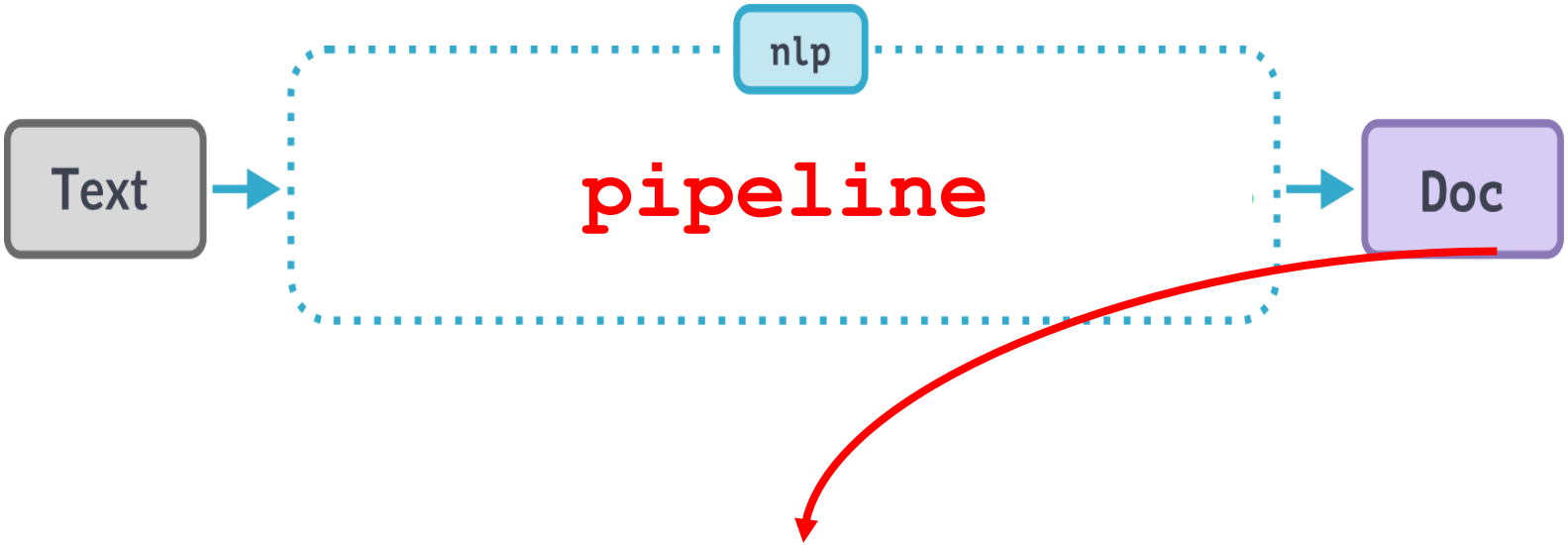
SpaCy

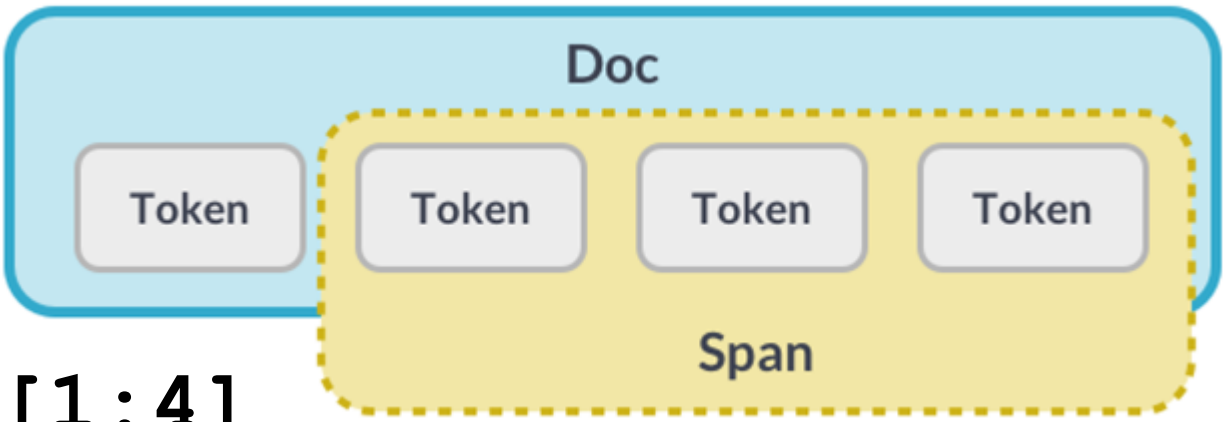

nlp
Text
pipeline
Doc
DOC
Token
I
PRON
nsubj
Token
love
VERB
dobj
Token
coffee
NOUN

SpaCy


nlp
Text
pipeline
Doc
Doc
doc[0]
Token
Token
Token
Token
I prefer the morning flight through Denver

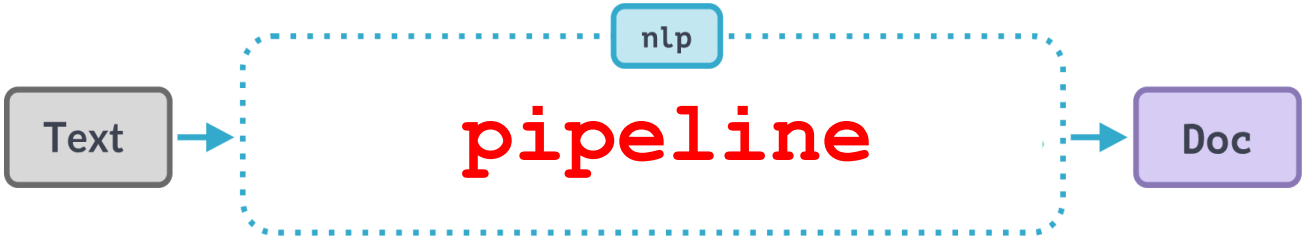
SpaCy

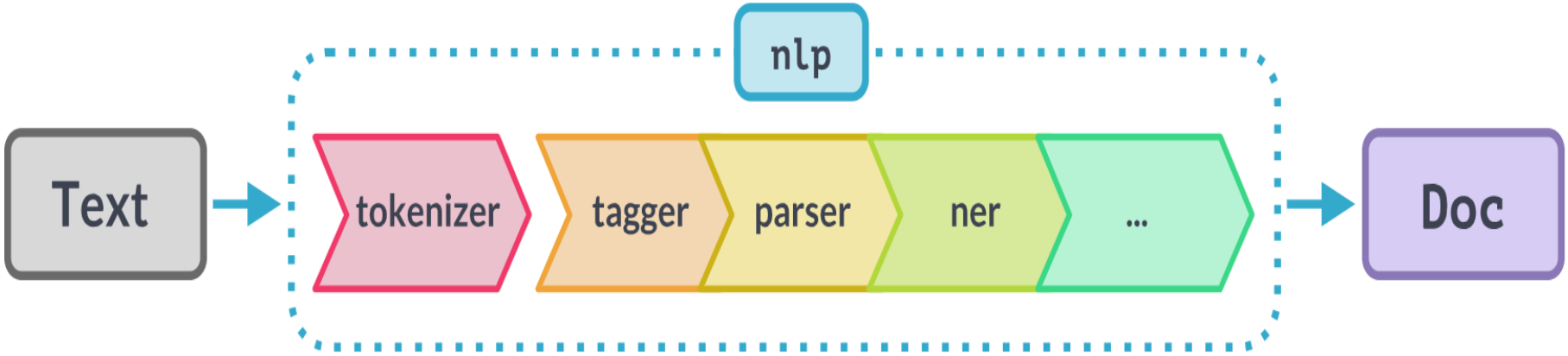





```
doc[1:4]
```

SpaCy

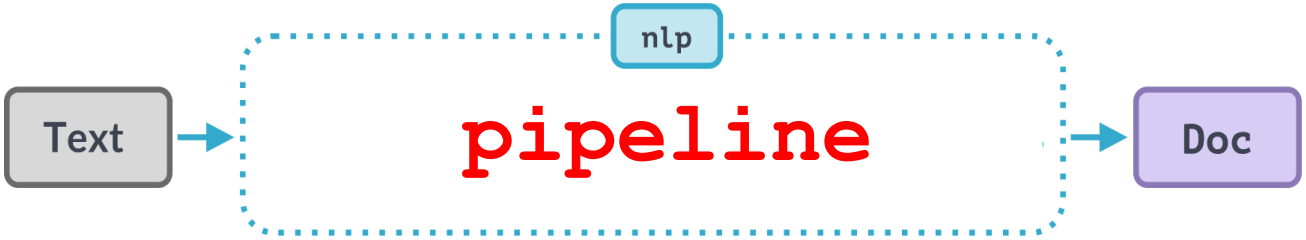




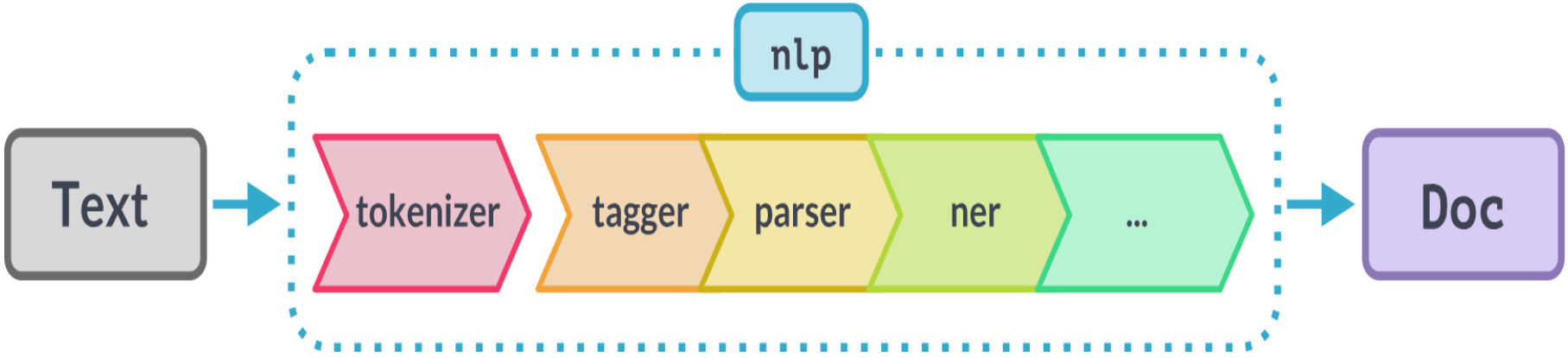
`nlp.pipe_names`

`nlp.component_names`

SpaCy



['tok2vec', 'tagger', 'parser', 'ner', 'attribute_ruler', 'lemmatizer']

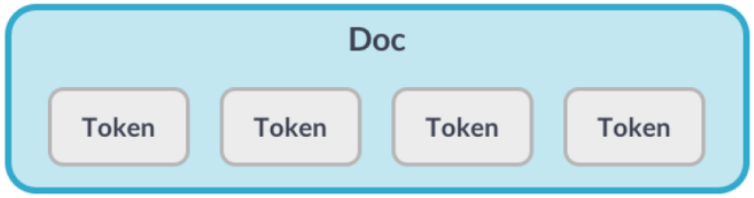


`nlp.pipe_names` `nlp.component_names`

SpaCy

```
for token in doc:
```

SpaCy - Tokenization




```
for token in doc:
    print(token.text, token.orth_)
```

SpaCy - Lemmatization


```
for token in doc:
    print(token.text, token.orth_)
    print(token.lemma_)
```

SpaCy -- POS-Tagging


```
for token in doc:
    print(token.text, token.orth_)
    print(token.lemma_)
    print(token.pos_, token.tag_)
```

Part-of-Speech Tagging: `token.pos_` et `token.tag_`

Tag types:

-Coarse-grained (**Universal Part-of-Speech Tagset**) : Noun, verb, adjective, etc. https://universaldependencies.org/u/pos/

-Fine-grained (**Penn Treebank tagset**): https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html • noun-proper-singular, noun-proper-plural, nouncommon-mass, .. • verb-past, verb-present-3rd, verb-base, ... • adjective-simple, adjective-comparative, ...

SpaCy -- POS-Tagging

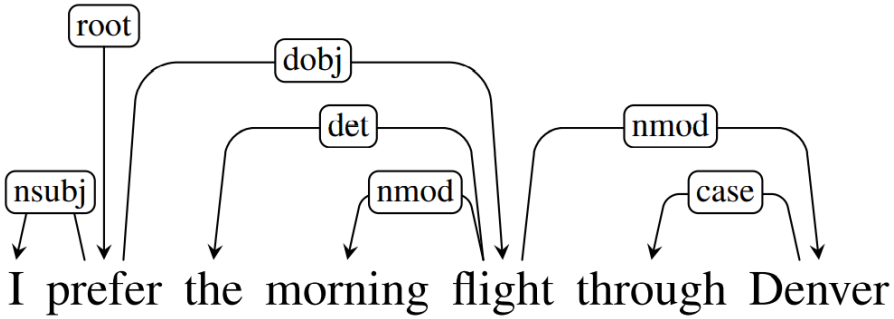

```
for token in doc:
    print(token.text, token.orth_)
    print(token.lemma_)
    print(token.pos_, token.tag_)
```

```
# Label explanations
spacy.explain("VBP")
```

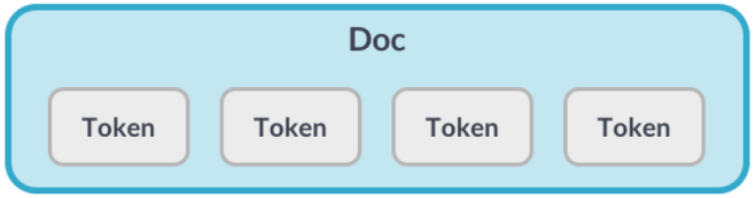
```
'verb, non-3rd person singular present'
```

SpaCy - Dependency Parsing


```
for token in doc:
    print(token.text, token.orth_)
    print(token.lemma_)
    print(token.pos_, token.tag_)
    print(token.dep_, token.head)
```

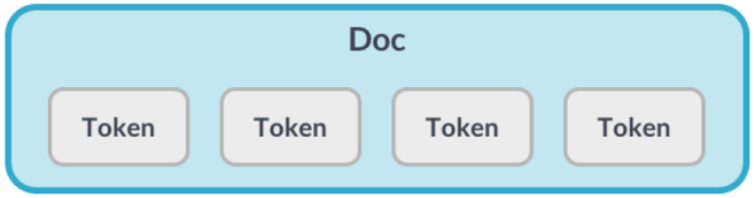


SpaCy - StopWords



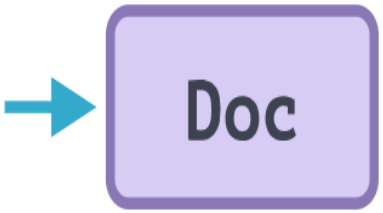

```
for token in doc:
    print(token.text, token.orth_)
    print(token.lemma_)
    print(token.pos_, token.tag_)
    print(token.dep_, token.head)
    print(token.is_stop)
```

SpaCy



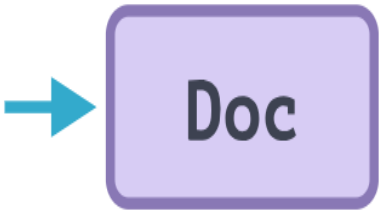

```
for token in doc:
    print(token.text, token.orth_)
    print(token.lemma_)
    print(token.pos_, token.tag_)
    print(token.dep_, token.head)
    print(token.is_stop)
    print(token.lang_)
```

SpaCy




```
doc.text

list(doc.sents)

list(doc.noun_chunks)
```

SpaCy - Chunking




```
doc.text
```

```
list(doc.sents)
```

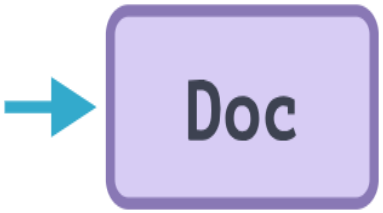
```
list(doc.noun_chunks)
```

```
list(doc.noun_chunks)[1].text
```

```
list(doc.noun_chunks)[1].label_
```

```
list(doc.noun_chunks)[1].root
```

SpaCy - Named Entity Recognition




```
doc.text
```

```
list(doc.sents)
```

```
list(doc.noun_chunks)
```

```
doc.ents
```

I prefer the morning flight through Denver

```
doc.ents[0].text
```

```
(Denver,)
```

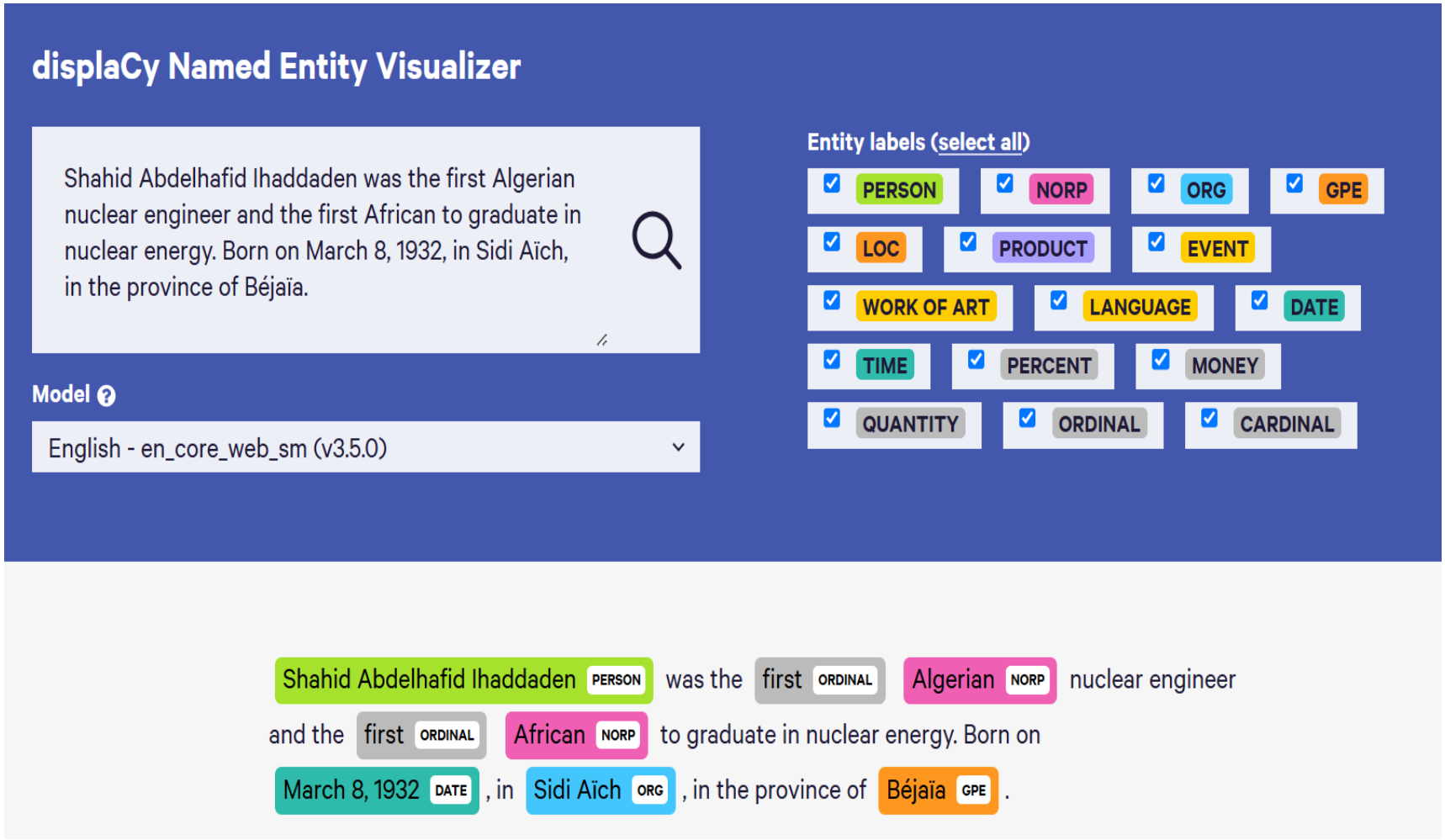
```
doc.ents[0].label_
```

```
Denver
```

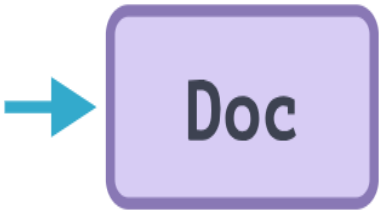
```
'GPE'
```

SpaCy - Named Entity Recognition

`doc.ents`



SpaCy




```
doc.text
```

```
list(doc.sents)
```

```
list(doc.noun_chunks)
```

```
doc.ents
```

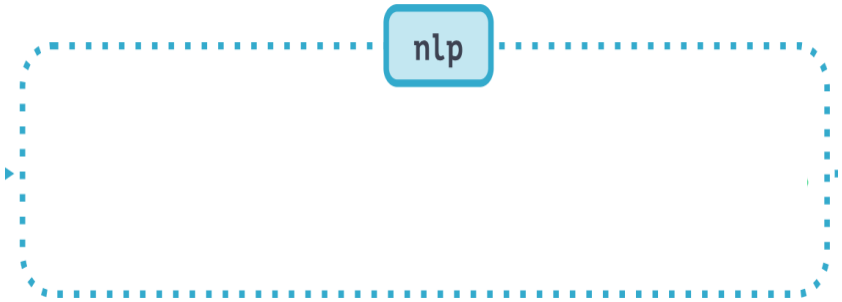
```
doc.ents[0].text
```

```
doc.ents[0].label_
```

```
spacy.explain('GPE')
```

Countries, cities, states

SpaCy




```
nlp.pipeline

nlp.pipe_names

nlp.component_names

list(nlp.Defaults.stop_words)

list(nlp.Defaults.tokenizer_exceptions)

list(nlp.Defaults.prefixes)

list(nlp.Defaults.suffixes)
```