

Introduction to spaCy

ADVANCED NLP WITH SPACY



Ines Montani
spaCy core developer

The nlp object

```
# Import the English language class
from spacy.lang.en import English

# Create the nlp object
nlp = English()
```

- contains the processing pipeline
- includes language-specific rules for tokenization etc.

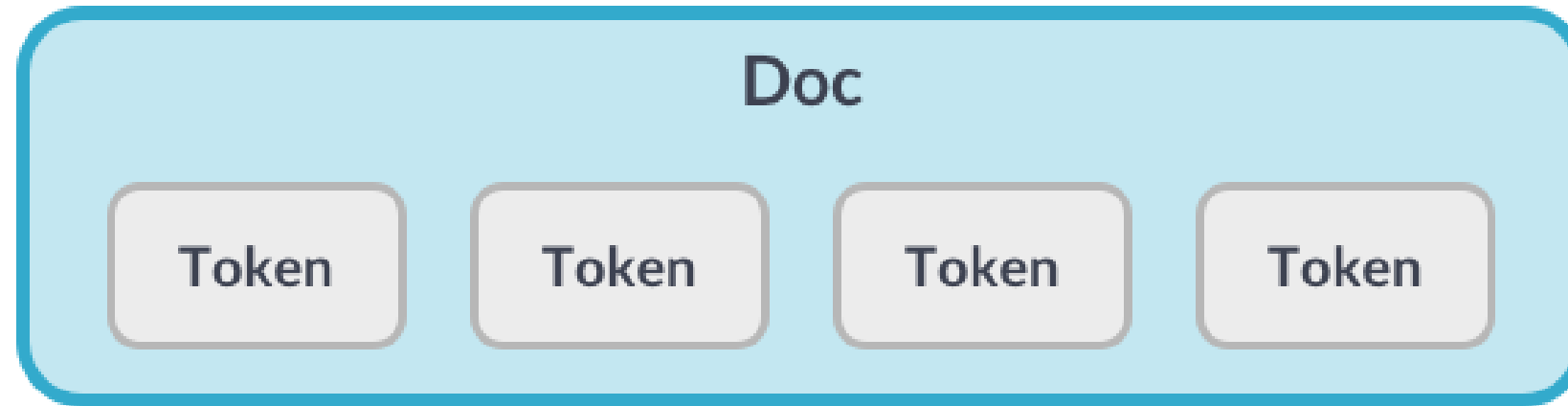
The Doc object

```
# Created by processing a string of text with the nlp object
doc = nlp("Hello world!")

# Iterate over tokens in a Doc
for token in doc:
    print(token.text)
```

```
Hello
world
!
```

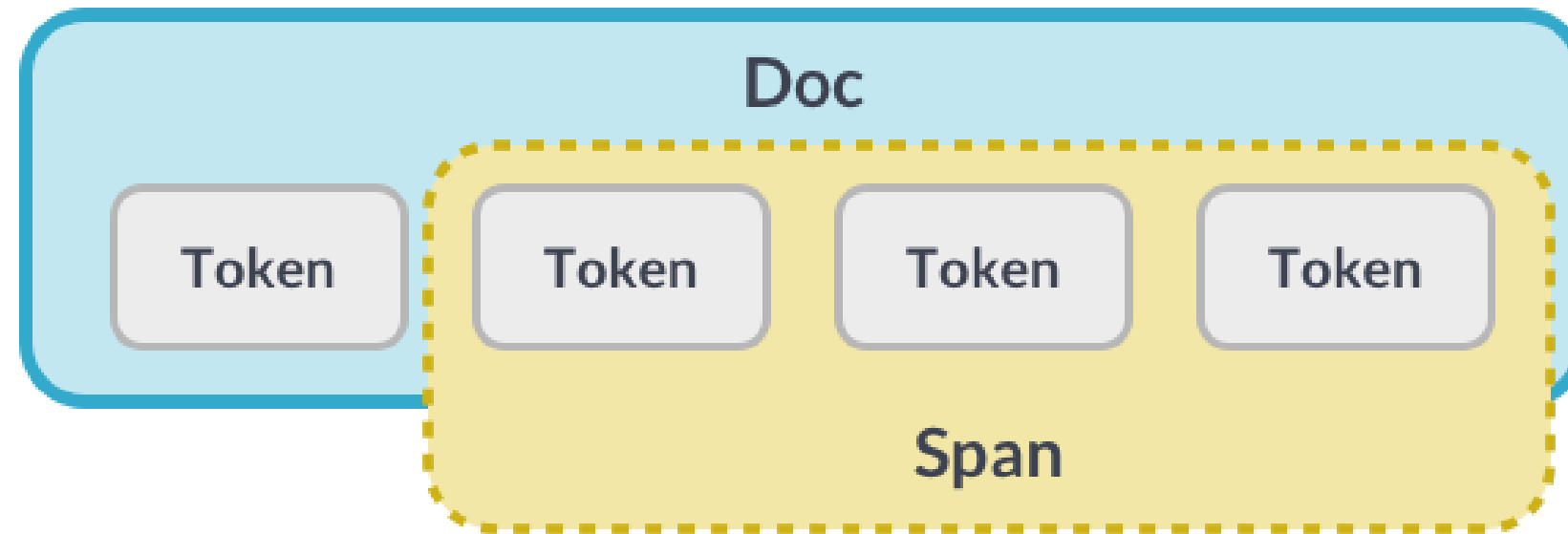
The Token object



```
doc = nlp("Hello world!")  
  
# Index into the Doc to get a single Token  
token = doc[1]  
  
# Get the token text via the .text attribute  
print(token.text)
```

```
world
```

The Span object



```
doc = nlp("Hello world!")

# A slice from the Doc is a Span object
span = doc[1:4]

# Get the span text via the .text attribute
print(span.text)
```

```
world!
```

Lexical attributes

```
doc = nlp("It costs $5.")
print('Index:    ', [token.i for token in doc])
print('Text:     ', [token.text for token in doc])
print('is_alpha:', [token.is_alpha for token in doc])
print('is_punct:', [token.is_punct for token in doc])
print('like_num:', [token.like_num for token in doc])
```

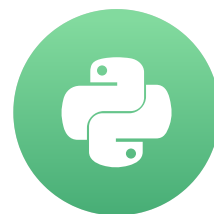
```
Index:    [0, 1, 2, 3, 4]
Text:     ['It', 'costs', '$', '5', '.']
is_alpha: [True, True, False, False, False]
is_punct: [False, False, False, False, True]
like_num: [False, False, False, True, False]
```

Let's practice!

ADVANCED NLP WITH SPACY

Statistical Models

ADVANCED NLP WITH SPACY



Ines Montani
spaCy core developer

What are statistical models?

- Enable spaCy to predict linguistic attributes *in context*
 - Part-of-speech tags
 - Syntactic dependencies
 - Named entities
- Trained on labeled example texts
- Can be updated with more examples to fine-tune predictions

Model Packages



```
import spacy
```

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

- Binary weights
- Vocabulary
- Meta information (language, pipeline)

Predicting Part-of-speech Tags

```
import spacy

# Load the small English model
nlp = spacy.load('en_core_web_sm')

# Process a text
doc = nlp("She ate the pizza")

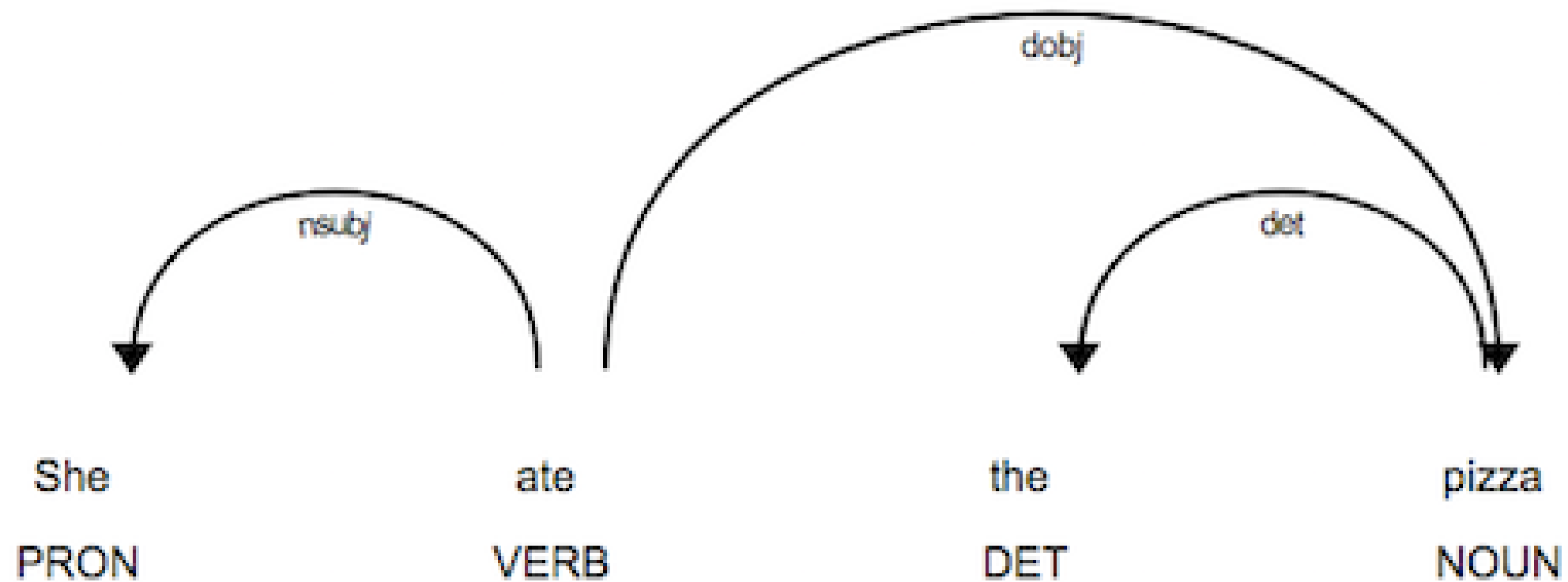
# Iterate over the tokens
for token in doc:
    # Print the text and the predicted part-of-speech tag
    print(token.text, token.pos_)
```

```
She PRON
ate VERB
the DET
pizza NOUN
```

Predicting Syntactic Dependencies

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

```
She PRON nsubj ate  
ate VERB ROOT ate  
the DET det pizza  
pizza NOUN dobj ate
```



Label	Description	Example
nsubj	nominal subject	She
dobj	direct object	pizza
det	determiner (article)	the

Predicting Named Entities

Apple ORG is looking at buying U.K. GPE startup for \$1 billion MONEY

```
# Process a text
doc = nlp(u"Apple is looking at buying U.K. startup for $1 billion")

# Iterate over the predicted entities
for ent in doc.ents:
    # Print the entity text and its label
    print(ent.text, ent.label_)
```

```
Apple ORG
U.K. GPE
$1 billion MONEY
```

Tip: the explain method

Get quick definitions of the most common tags and labels.

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

```
Countries, cities, states'
```

```
spacy.explain('NNP')
```

```
'noun, proper singular'
```

```
spacy.explain('dobj')
```

```
'direct object'
```

Let's practice!

ADVANCED NLP WITH SPACY

Rule-based Matching

ADVANCED NLP WITH SPACY



Ines Montani
spaCy core developer

Why not just regular expressions?

- Match on `Doc` objects, not just strings
- Match on tokens and token attributes
- Use the model's predictions
- Example: "duck" (verb) vs. "duck" (noun)

Match patterns

- Lists of dictionaries, one per token
- Match exact token texts

```
[ { 'ORTH' : 'iPhone' }, { 'ORTH' : 'X' } ]
```

- Match lexical attributes

```
[ { 'LOWER' : 'iphone' }, { 'LOWER' : 'x' } ]
```

- Match any token attributes

```
[ { 'LEMMA' : 'buy' }, { 'POS' : 'NOUN' } ]
```

Using the Matcher (1)

```
import spacy

# Import the Matcher
from spacy.matcher import Matcher

# Load a model and create the nlp object
nlp = spacy.load('en_core_web_sm')

# Initialize the matcher with the shared vocab
matcher = Matcher(nlp.vocab)

# Add the pattern to the matcher
pattern = [{'ORTH': 'iPhone'}, {'ORTH': 'X'}]
matcher.add('IPHONE_PATTERN', None, pattern)

# Process some text
doc = nlp("New iPhone X release date leaked")

# Call the matcher on the doc
matches = matcher(doc)
```

Using the Matcher (2)

```
# Call the matcher on the doc
doc = nlp("New iPhone X release date leaked")
matches = matcher(doc)

# Iterate over the matches
for match_id, start, end in matches:
    # Get the matched span
    matched_span = doc[start:end]
    print(matched_span.text)
```

iPhone X

- `match_id` : hash value of the pattern name
- `start` : start index of matched span
- `end` : end index of matched span

Matching lexical attributes

```
pattern = [  
    {'IS_DIGIT': True},  
    {'LOWER': 'fifa'},  
    {'LOWER': 'world'},  
    {'LOWER': 'cup'},  
    {'IS_PUNCT': True}  
]
```

```
doc = nlp("2018 FIFA World Cup: France won!")
```

2018 FIFA World Cup:

Matching other token attributes

```
pattern = [  
    {'LEMMA': 'love', 'POS': 'VERB'},  
    {'POS': 'NOUN'}  
]
```

```
doc = nlp("I loved dogs but now I love cats more.")
```

loved dogs

love cats

Using operators and quantifiers (1)

```
pattern = [  
    {'LEMMA': 'buy'},  
    {'POS': 'DET', 'OP': '?'}, # optional: match 0 or 1 times  
    {'POS': 'NOUN'}  
]
```

```
doc = nlp("I bought a smartphone. Now I'm buying apps.")
```

```
bought a smartphone  
buying apps
```


Using operators and quantifiers (2)

	Description
<code>{ 'OP' : '!' }</code>	Negation: match 0 times
<code>{ 'OP' : '?' }</code>	Optional: match 0 or 1 times
<code>{ 'OP' : '+' }</code>	Match 1 or more times
<code>{ 'OP' : '*' }</code>	Match 0 or more times

Let's practice!

ADVANCED NLP WITH SPACY