Natural Language Processing (NLP) basics

NATURAL LANGUAGE PROCESSING WITH SPACY

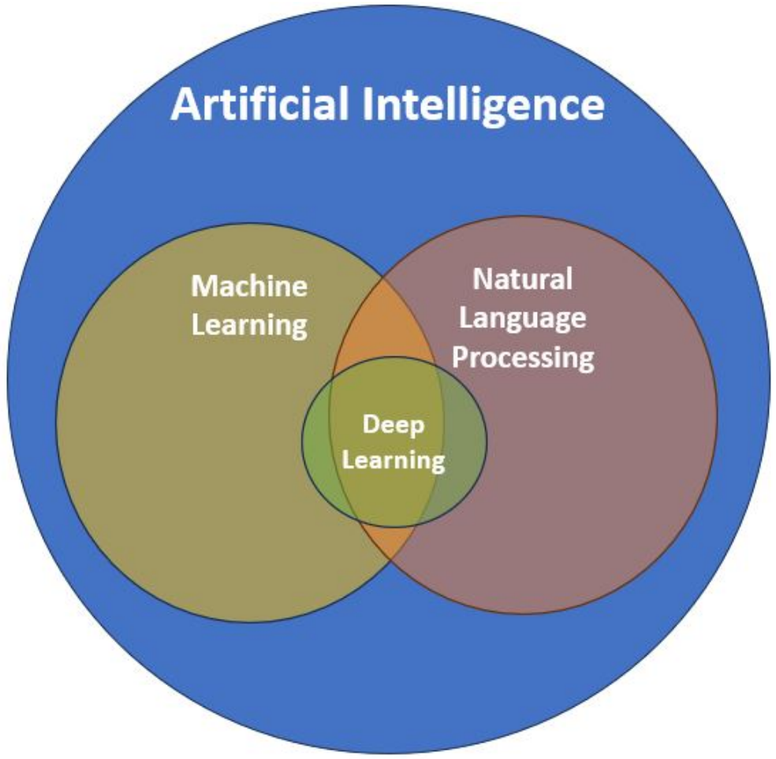


Azadeh Mobasher
Principal Data Scientist

datacamp

Natural Language Processing (NLP)

- A subfield of **Artificial Intelligence (AI)**
- Helps computers to understand **human language**
- Helps extract insights from unstructured data
- Incorporates **statistics, machine learning models** and **deep learning models**



NLP use cases

Sentiment analysis

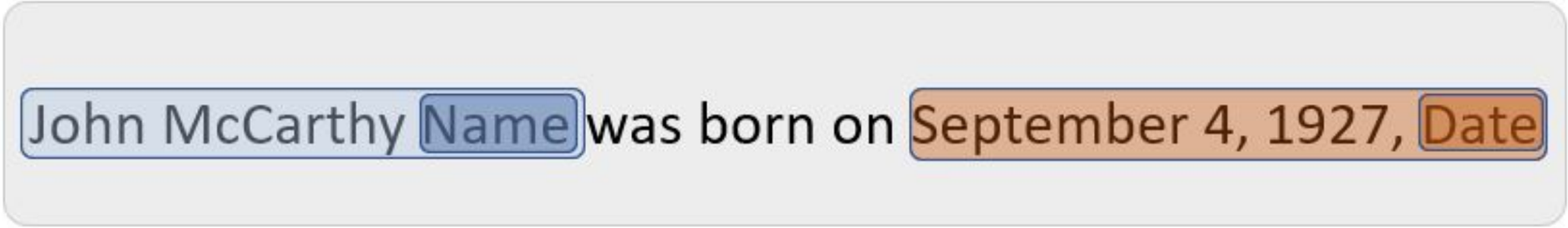
- Use of computers to determine the underlying subjective tone of a piece of writing



NLP use cases

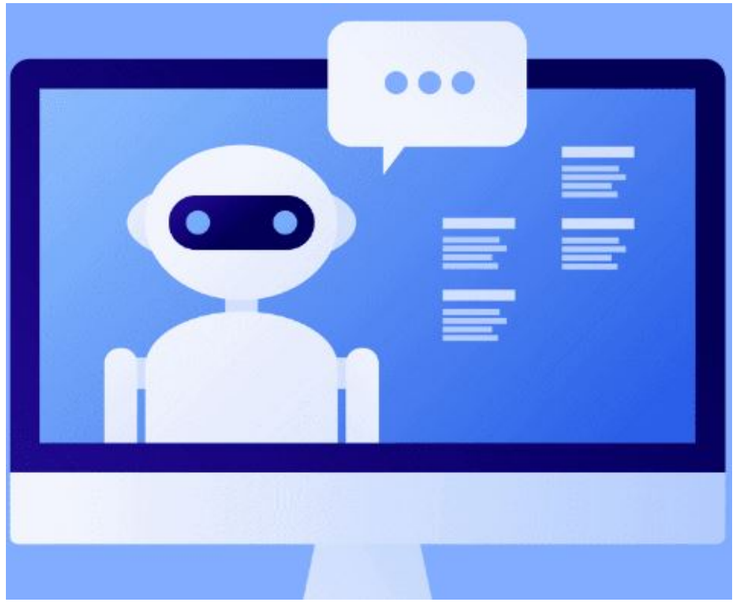
Named entity recognition (NER)

- Locating and classifying named entities mentioned in unstructured text into pre-defined categories
- **Named entities** are real-world objects such as a person or location



NLP use cases

- Generate human-like responses to text input, such as **ChatGPT**



Introduction to spaCy

spaCy is a **free, open-source** library for NLP in **Python** which:

- Is designed to build systems for **information extraction**
- Provides **production-ready** code for NLP use cases
- Supports **64+** languages
- Is **robust** and **fast** and has **visualization libraries**



Install and import spaCy

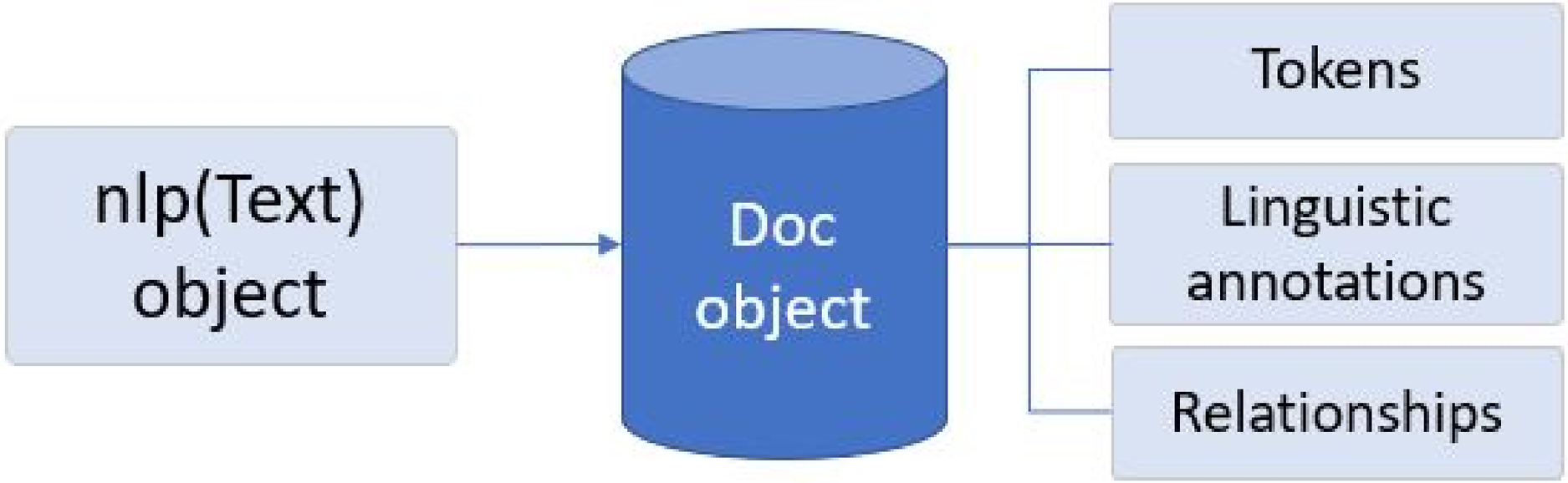
- As the first step, `spaCy` can be installed using the Python package manager pip
- `spaCy` trained models can be downloaded
- Multiple trained models are available for English language at spacy.io

```
$ python3 pip install spacy
```

```
python3 -m spacy download en_core_web_sm
import spacy
nlp = spacy.load("en_core_web_sm")
```

Read and process text with spaCy

- Loaded `spaCy` model `en_core_web_sm` = `nlp` object
- `nlp` object converts text into a `Doc` object (container) to store processed text



spaCy in action

- **Processing** a string using `spaCy`

```
import spacy
nlp = spacy.load("en_core_web_sm")
text = "A spaCy pipeline object is created."
doc = nlp(text)
```

- **Tokenization**
  - A `Token` is defined as the smallest meaningful part of the text.
  - **Tokenization:** The process of dividing a text into a list of meaningful tokens

```
print([token.text for token in doc])
```

```
['A', 'spaCy', 'pipeline', 'object', 'is', 'created', '.']
```

Let's practice!

NATURAL LANGUAGE PROCESSING WITH SPACY

datacamp

spaCy basics

NATURAL LANGUAGE PROCESSING WITH SPACY



Azadeh Mobasher
Principal Data Scientist

datacamp

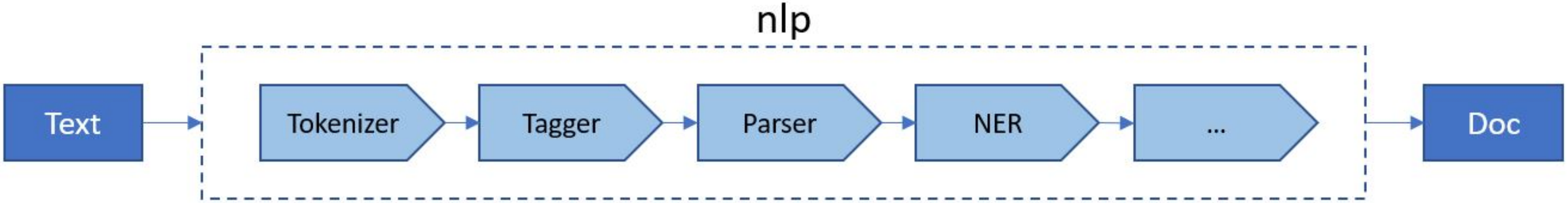
spaCy NLP pipeline

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("Here's my spaCy pipeline.")
```

- Import `spaCy`
- Use `spacy.load()` to return `nlp`, a `Language` class
  - The `Language` object is the text processing pipeline
- Apply `nlp()` on any text to get a `Doc` container

spaCy NLP pipeline

`spaCy` applies some processing steps using its `Language` class:



Container objects in spaCy

- There are multiple data structures to represent text data in `spaCy` :

Name	Description
`Doc`	A container for accessing linguistic annotations of text
`Span`	A slice from a `Doc` object
`Token`	An individual token, i.e. a word, punctuation, whitespace, etc.

Pipeline components

- The `spaCy` language processing pipeline always depends on the loaded model and its capabilities.

Component	Name	Description
Tokenizer	Tokenizer	Segment text into tokens and create `Doc` object
Tagger	Tagger	Assign part-of-speech tags
Lemmatizer	Lemmatizer	Reduce the words to their root forms
EntityRecognizer	NER	Detect and label named entities

Pipeline components

- Each component has unique features to process text
  - **Language**
  - **DependencyParser**
  - **Sentencizer**

Tokenization

- Always the first operation
- All the other operations require tokens
- Tokens can be words, numbers and punctuation

```
import spacy
nlp = spacy.load("en_core_web_sm")

doc = nlp("Tokenization splits a sentence into its tokens.")
print([token.text for token in doc])
```

```
['Tokenization', 'splits', 'a', 'sentence', 'into', 'its', 'tokens', '.']
```

Sentence segmentation

- More complex than tokenization
- Is a part of `DependencyParser` component

```
import spacy
nlp = spacy.load("en_core_web_sm")

text = "We are learning NLP. This course introduces spaCy."
doc = nlp(text)
for sent in doc.sents:
    print(sent.text)
```

```
We are learning NLP.
This course introduces spaCy.
```

Lemmatization

- A **lemma** is a the base form of a token
- The lemma of **eats** and **ate** is **eat**
- Improves accuracy of language models

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("We are seeing her after one year.")
print([(token.text, token.lemma_) for token in doc])
```

```
[('We', 'we'), ('are', 'be'), ('seeing', 'see'), ('her', 'she'),
('after', 'after'), ('one', 'one'), ('year', 'year'), ('.', '.')]
```

Let's practice!

NATURAL LANGUAGE PROCESSING WITH SPACY

Linguistic features in spaCy

NATURAL LANGUAGE PROCESSING WITH SPACY



Azadeh Mobasher
Principal Data Scientist

datacamp

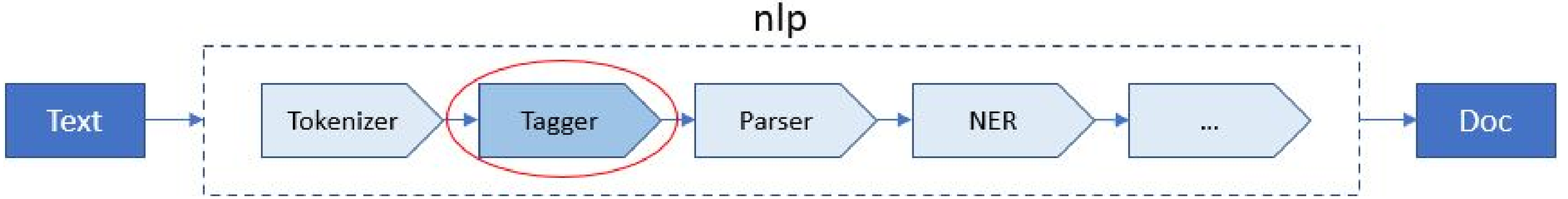
POS tagging

- **Categorizing** words grammatically, based on function and context within a sentence

POS	Description	Example
VERB	Verb	run, eat, ate, take
NOUN	Noun	man, airplane, tree, flower
ADJ	Adjective	big, old, incompatible, conflicting
ADV	Adverb	very, down, there, tomorrow
CONJ	Conjunction	and, or, but

POS tagging with spaCy

- POS tagging confirms the meaning of a word
- Some words such as **watch** can be both noun and verb
- `spaCy` captures POS tags in the `pos_` feature of the nlp pipeline
- `spacy.explain()` explains a given POS tag



POS tagging with spaCy

```
verb_sent = "I watch TV."

print([(token.text, token.pos_,
spacy.explain(token.pos_))
for token in nlp(verb_sent)])
```

```
[('I', 'PRON', 'pronoun'),
('watch', 'VERB', 'verb'),
('TV', 'NOUN', 'noun'),
('.', 'PUNCT', 'punctuation')]
```

```
noun_sent = "I left without my watch."

print([(token.text, token.pos_,
spacy.explain(token.pos_))
for token in nlp(noun_sent)])
```

```
[('I', 'PRON', 'pronoun'),
('left', 'VERB', 'verb'),
('without', 'ADP', 'adposition'),
('my', 'PRON', 'pronoun'),
('watch', 'NOUN', 'noun'),
('.', 'PUNCT', 'punctuation')]
```

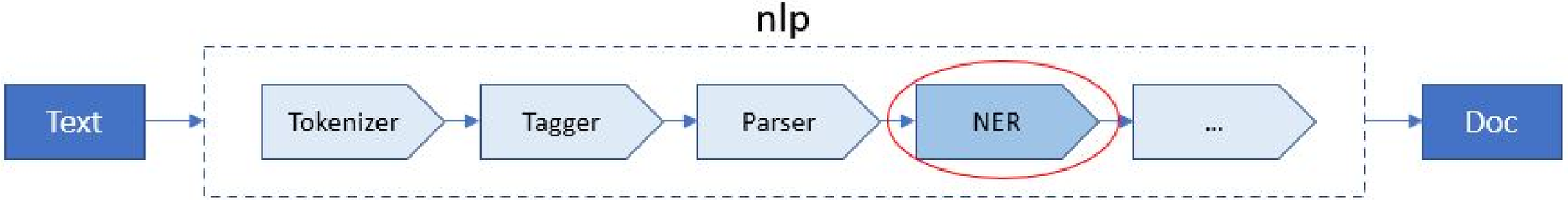
Named entity recognition

- A **named entity** is a word or phrase that refers to a specific entity with a name
- **Named-entity recognition** (NER) classifies named entities into pre-defined categories

Entity type	Description
PERSON	Named person or family
ORG	Companies, institutions, etc.
GPE	Geo-political entity, countries, cities, etc.
LOC	Non-GPE locations, mountain ranges, etc.
DATE	Absolute or relative dates or periods
TIME	Time smaller than a day

NER and spaCy

- `spaCy` models extract named entities using the `NER` pipeline component
- Named entities are available via the `doc.ents` property
- `spaCy` will also tag each entity with its entity label (`.label_`)



NER and spaCy

```
import spacy
nlp = spacy.load("en_core_web_sm")
text = "Albert Einstein was genius."
doc = nlp(text)
print([(ent.text, ent.start_char,
ent.end_char, ent.label_) for ent in doc.ents])
```

```
>>> [('Albert Einstein', 0, 15, 'PERSON')]
```

NER and spaCy

- We can also access entity types of each token in a `Doc` container

```
import spacy
nlp = spacy.load("en_core_web_sm")
text = "Albert Einstein was genius."
doc = nlp(text)
print([(token.text, token.ent_type_) for token in doc])
```

```
>>> [('Albert', 'PERSON'), ('Einstein', 'PERSON'),
('was', ''), ('genius', ''), ('.', '')]
```

displaCy

- `spaCy` is equipped with a modern visualizer: `displaCy`
- The `displaCy` entity visualizer highlights named entities and their labels

```
import spacy
from spacy import displacy

text = "Albert Einstein was genius."
nlp = spacy.load("en_core_web_sm")
doc = nlp(text)
displacy.serve(doc, style="ent")
```



Let's practice!

NATURAL LANGUAGE PROCESSING WITH SPACY