ML-BASED NLP

with spaCy & Python

CONTENT

- The NLP Pipeline
- Getting Started
- NLP Tasks
  - Tokenization
  - Part-of-Speech Tagging
  - Lemmatization
  - Named Entity Recognition
  - Dependency Parsing
  - Stop Words
  - Similarity
  - Sentiment
- Pretrained Language Models

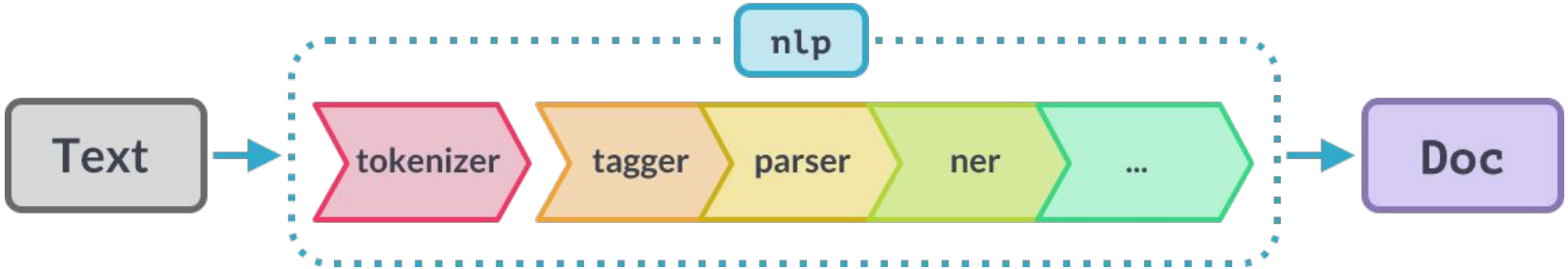
RECOMMENDED RESOURCES

- Free online course “spaCy 101”: https://spacy.io/usage/spacy-101
- spaCy’s model documentation website: https://spacy.io/models

THE NLP PIPELINE

THE NLP PIPELINE

FROM TEXT TO DOC



Source: https://spacy.io/usage/spacy-101

GETTING STARTED

Prerequisite: You need Python 3.x installed on your computer

Step 1: Install spaCy: `pip install spacy`

Step 2: Download a trained model: `python -m spacy download en_core_web_sm`

Step 3: Load spaCy and model from Python:

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("I love studying at the University of Applied Sciences in Osnabrück")
```

This is a specific trained model spaCy offers

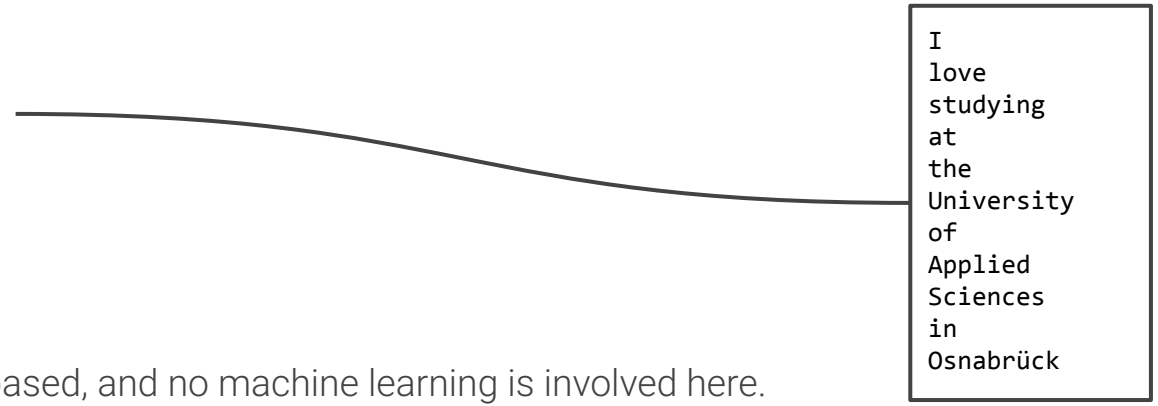
TOKENIZATION

TOKENIZATION

The first component in a NLP pipeline is the tokenizer. It splits the raw text into tokens:

```
doc = nlp("I love studying at the University of Applied Sciences in Osnabrück")

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



NOTE: The tokenizer is rule-based, and no machine learning is involved here.

PART-OF-SPEECH TAGGING

PART-OF-SPEECH TAGGING

The tagger determines the type of token and its role in the sentence:

```
doc = nlp("I love studying at the University of Applied Sciences in Osnabrück")

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

```
I PRON PRP
love VERB VBP
studying VERB VBG
at ADP IN
the DET DT
University PROPN NNP
of ADP IN
Applied PROPN NNP
Sciences PROPN NNPS
in ADP IN
Osnabrück PROPN NNP
```

NOTE: `spacy.explain("NNPS")` will print an explanation → “noun, proper plural”

LEMMATIZER

LEMMATIZER

EXAMPLE 1/2

The lemmatizer determines the base or canonical form (lemma) of a word:

```
doc = nlp("I love studying at the University of Applied Sciences in Osnabrück")

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

```
I I
love love
studying study
at at
the the
University University
of of
Applied Applied
Sciences Sciences
in in
Osnabrück Osnabrück
```

LEMMATIZER

EXAMPLE 2/2

Given a different context, the word “applied” is converted to its lemma form:

```
doc = nlp("I love data analytics when applied in practice")

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

```
I I
love love
data data
analytics analytic
when when
applied apply
in in
practice practice
```

NAMED ENTITY RECOGNITION (NER)

NAMED ENTITY RECOGNITION

The NER-component recognizes named entities such as famous people, places, or organisation:

```
doc = nlp("I love studying at the University of Applied Sciences in Osnabrück")

for entity in doc.ents:
     print(entity.text, entity.label_)
```

DEPENDENCY PARSING

DEPENDENCY PARSING

The dependency parser determines the syntactic relationship between tokens:

```
doc = nlp("I love studying at the University of Applied Sciences in Osnabrück")

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

```
I nsubj love True
love ROOT love False
studying xcomp love False
at prep studying False
the det University False
University pobj at False
of prep University False
Applied compound Sciences False
Sciences pobj of False
in prep University False
Osnabrück pobj in False
```

STOP WORDS

STOP WORDS

The NLP-pipeline determines stop words along with other useful characteristics:

```
doc = nlp("I love studying at the University of Applied Sciences in Osnabrück")

for token in doc:
    print(token.text, token.is_stop, token.is_alpha, token.shape)
```

```
I True True X
love False True xxxx
studying False True xxxx
at True True xx
the True True xxx
University False True Xxxxx
of True True xx
Applied False True Xxxxx
Sciences False True Xxxxx
in True True xx
Osnabrück False True Xxxxx
```

SIMILARITY

SIMILARITY

Models that ship with word vectors (`md`, `lg`) can calculate similarity scores between tokens and documents:

```
nlp = spacy.load("en_core_web_md")
doc1 = nlp("I love this course")
doc2 = nlp("I like this lecture")

print(doc1, "<->", doc2, doc1.similarity(doc2))
```

```
I love this course <-> I like this lecture 0.9248678087461984
```

NOTE: There is no objective definition of “similar” and the meaning depends on the use case.

SENTIMENT

COMING SOON

PRETRAINED LANGUAGE MODELS

PRETRAINED LANGUAGE MODELS

OVERVIEW

- spaCy offers variety of models for many languages.
- Models differ in **size** and **prediction speed** and **accuracy**:
  - Larger model → slower but higher accuracy
  - Smaller model → faster but less accuracy
- Englisch:
  - `en_core_web_sm` (12 MB) → smallest English model based trained on web-data
  - `en_core_web_md` (40 MB) → medium English model, contains word vectors
  - `en_core_web_lg` (560 MB) → large English model, highest accuracy
  - `en_core_web_trf` (438 MB) → Transformer-based model, GPU recommended