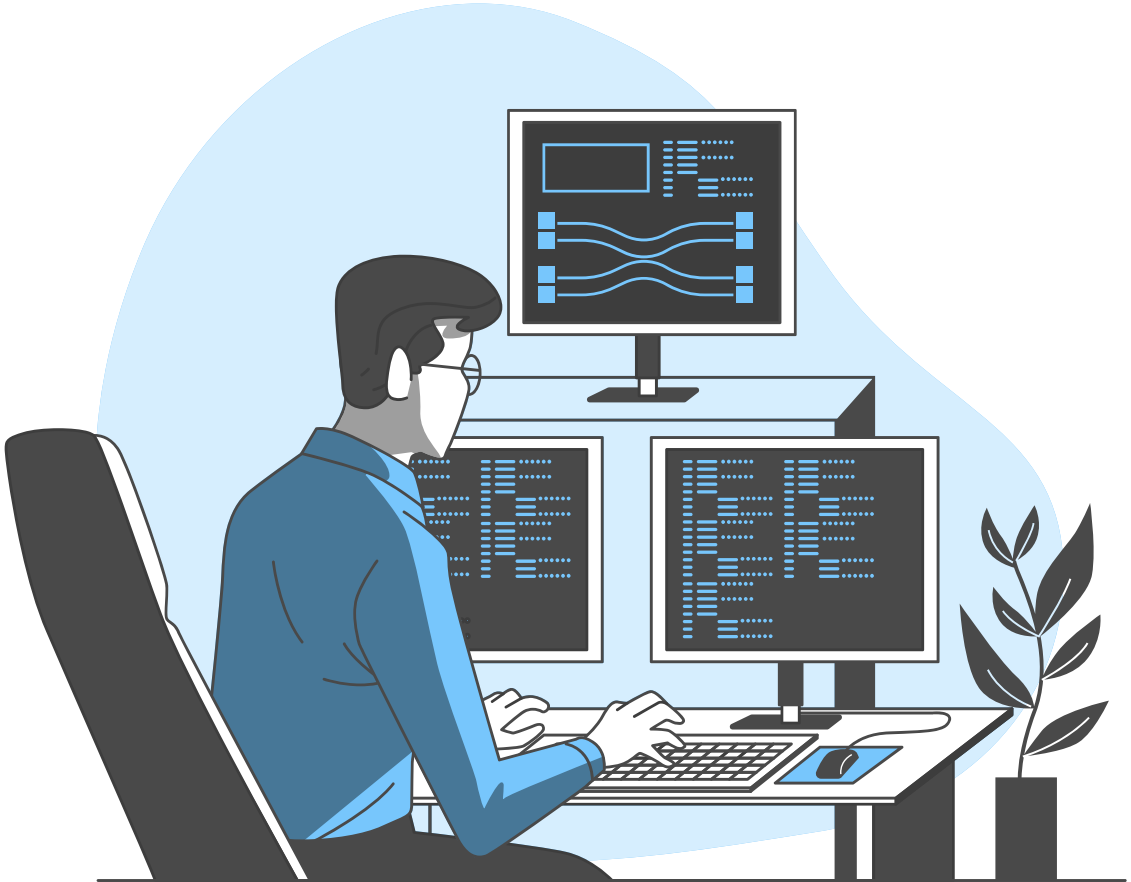


spaCy

2023/03/02

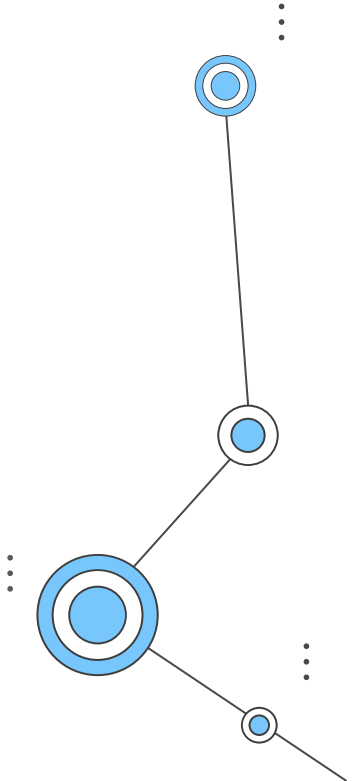
spaCy Intro

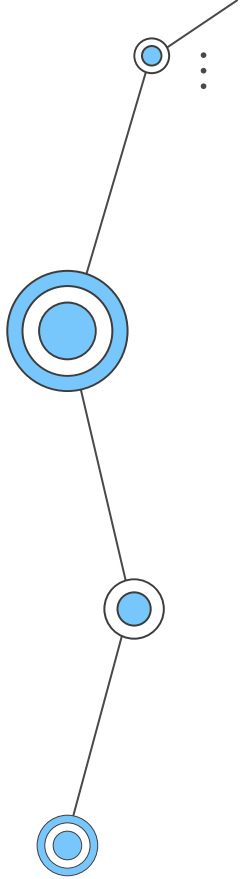
Features & Pipeline

spaCy & Models

Models

Conclusion





01 Quick Introduction

Why do we use spaCy?

Models & Languages · spaCy Usage Documentation

Feature comparison

Here's a quick comparison of the functionalities offered by spaCy, NLTK and CoreNLP.

	SPACY	NLTK	CORENLP
Programming language	Python	Python	Java / Python
Neural network models	✓	✗	✓
Integrated word vectors	✓	✗	✗
Multi-language support	✓	✓	✓
Tokenization	✓	✓	✓
Part-of-speech tagging	✓	✓	✓
Sentence segmentation	✓	✓	✓
Dependency parsing	✓	✗	✓
Entity recognition	✓	✓	✓
Entity linking	✓	✓	✗
Coreference resolution	✗	✗	✓

- To install spaCy

```
$ pip install spacy
```

- To install spaCy in one specific version of python

```
$ pip3.5 install spacy
```

- To install spaCy with conda

```
$ conda install -c conda-forge spacy
```

- To install spaCy on macOS/OS X

```
$ xcode-select -install
```

- To download language model

```
$ python -m spacy download en
$ python -m spacy download de
$ python -m spacy download fr
```

- Or you can choose what model you want to download

```
$ python -m spacy download en_core_web_md
```

- Now you can import spaCy, load model, and parse text

```
$ pip install spacy
$ python -m spacy download en
 import spacy
 nlp = spacy.load('en_core_web_md')
 doc = nlp('I have a ginger cat.')
```

```
$ pip install spacy
$ python -m spacy download en_core_web_md
 import spacy
 nlp = spacy.load('en_core_web_md')
 doc = nlp('I have a ginger cat.')
```

spaCy linguistic features

Tokenization
Lemmatization

Cf. stemming

POS tagging

Dependency Parsing

Sentence structure displayed via dependencies among the tokens.

NER

Rule-Based Matching

E.g. Hashtag and emoji

Vectorization

slido

Which one of the following is not a token?

ⓘ Start presenting to display the poll results on this slide.

slido

Not "lemma"?

ⓘ Start presenting to display the poll results on this slide.

spaCy linguistic features

Tokenization
Lemmatization

Cf. stemming

POS tagging

Dependency Parsing

Sentence structure displayed via dependencies among the tokens.

NER

Rule-Based Matching

E.g. Hashtag and emoji

Vectorization

Chinese NLP pipeline in spaCy

01 Tokenization

02 POS tagging

03 Dependency parsing

04 NER

```
# load language model
nlp = spacy.load('zh_core_web_lg')
```

```
# parse text
doc1 = nlp("我最喜歡自然語言處理了")
doc2 = nlp("我最喜歡自然語言處理了。所以秒選謝舒凱老師的課！")
```

```
# doc 2

print(doc2.text)
print(doc2[1])
sen = list(doc2.sents)
print(sen)
print(doc2.ents)
```

Chinese NLP pipeline in spaCy

01 Tokenization

02 POS tagging

03 Dependency parsing

04 NER

```
doc = nlp("我最喜歡自然語言處理了")

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

```
我 PRON PN nsubj True
最 ADV AD advmod True
喜歡 VERB VV ROOT False
自然 NOUN NN compound:nn False
語言 NOUN NN nsubj False
處理 VERB VV ccomp False
了 PART AS aux:asp True
```

Chinese NLP pipeline in spaCy

01 Tokenization

02 POS tagging

03 Dependency parsing

04 NER

```
下課 NOUN noun NT temporal noun
我 PRON pronoun PN pronoun
要 VERB verb VV other verb
知道 VERB verb VC 是 (copula)
寶雅 PROPN proper noun NR proper noun
屈臣氏 PROPN proper noun NR proper noun
他們 PRON pronoun PN pronoun
```

token.pos_:
coarse-grained pos

token.tag_:
fine-grained pos

Chinese NLP pipeline in spaCy

01 Tokenization

02 POS tagging

03 **Dependency parsing**

04 NER

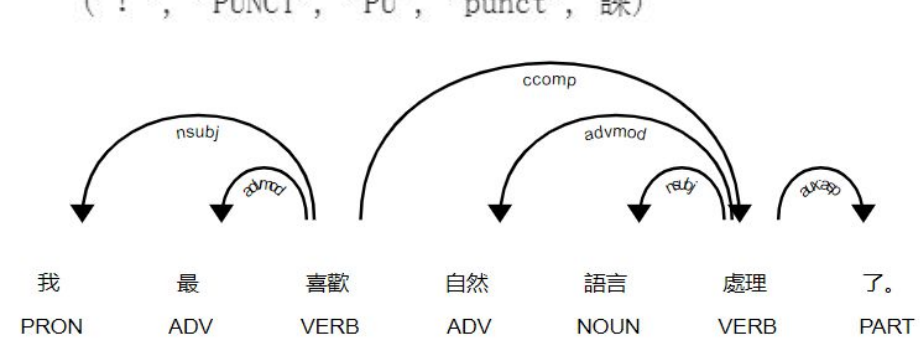
```
('我', 'PRON', 'PN', 'nsubj', 喜歡)
('最', 'ADV', 'AD', 'advmod', 喜歡)
('喜歡', 'VERB', 'VV', 'ROOT', 喜歡)
('自然', 'ADV', 'AD', 'advmod', 處理)
('語言', 'NOUN', 'NN', 'nsubj', 處理)
('處理', 'VERB', 'VV', 'ccomp', 喜歡)
('了', 'PART', 'SP', 'aux:asp', 處理)
('。', 'PUNCT', 'PU', 'punct', 喜歡)
('所以', 'ADV', 'AD', 'advmod', 課)
('秒選', 'PROPN', 'NR', 'name', 謝)
('謝', 'PROPN', 'NR', 'dep', 課)
('舒凱', 'PROPN', 'NR', 'compound:nn', 老師)
('老師', 'NOUN', 'NN', 'nmod:assmod', 課)
('的', 'PART', 'DEG', 'case', 老師)
('課', 'NOUN', 'NN', 'ROOT', 課)
('！', 'PUNCT', 'PU', 'punct', 課)
```

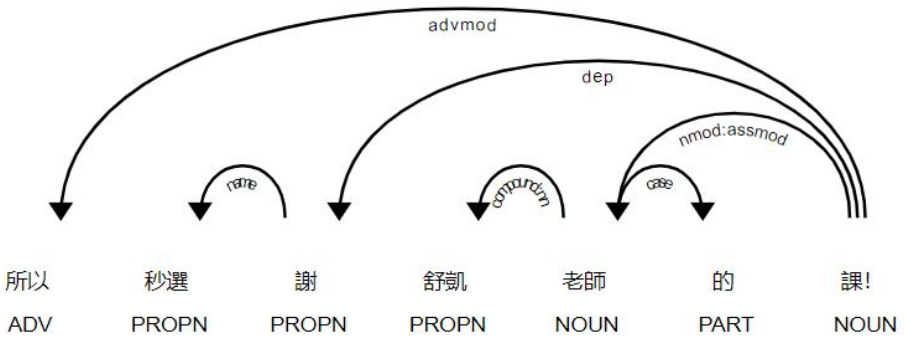
Chinese NLP pipeline in spaCy

03 Dependency parsing

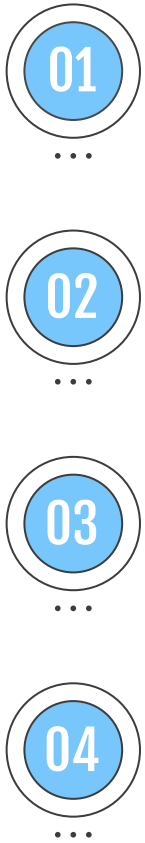
```
('我', 'PRON', 'PN', 'nsubj', 喜歡)
('最', 'ADV', 'AD', 'advmod', 喜歡)
('喜歡', 'VERB', 'VV', 'ROOT', 喜歡)
('自然', 'ADV', 'AD', 'advmod', 處理)
('語言', 'NOUN', 'NN', 'nsubj', 處理)
('處理', 'VERB', 'VV', 'ccomp', 喜歡)
('了', 'PART', 'SP', 'aux:asp', 處理)
('。', 'PUNCT', 'PU', 'punct', 喜歡)
('所以', 'ADV', 'AD', 'advmod', 課)
('秒選', 'PROPN', 'NR', 'name', 謝)
('謝', 'PROPN', 'NR', 'dep', 課)
('舒凱', 'PROPN', 'NR', 'compound:nn', 老師)
('老師', 'NOUN', 'NN', 'nmod:assmod', 課)
('的', 'PART', 'DEG', 'case', 老師)
('課', 'NOUN', 'NN', 'ROOT', 課)
('！', 'PUNCT', 'PU', 'punct', 課)
```

token 對這個 head 有這樣的依存關係





Chinese NLP pipeline in spaCy



Tokenization

POS tagging

Dependency parsing

NER

```
for  ent  in  doc3.ents:
    print(((
        ent.text,
        ent.start_char,
        ent.end_char,
        ent.label_
    )))
```

```
('spacy', 0, 5, 'PERSON')
('屈臣氏', 10, 13, 'ORG')
('寶雅', 14, 16, 'ORG')
('家樂福', 18, 21, 'ORG')
```

Chinese NLP pipeline in spaCy

01 Tokenization

02 POS tagging

03 Dependency parsing

04 NER

```
('賓雅', 13, 15, 'ORG', 'Companies, agencies, institutions, etc.')
('屈臣氏比較', 21, 26, 'ORG', 'Companies, agencies, institutions, etc.')
('他們', 29, 31, 'PERSON', 'People, including fictional')
```

02

spaCy and Models

Vectorization

01 Gensim word2vec/ fasttext pretrained model (Chinese wiki)

02 spaCy 3.0 - transformers (bert-base-chinese as example)

```
tokens = nlp("防疫就要宅在家")

for token in tokens:
    print(token.text, token.has_vector, token.vector_norm, token.is_oov, token.vector)
```

```
防疫 True 37.577972 False [ 1.3487   -0.69083   1.8575    0.03564   2.132     2.8709
  0.60342  -4.8321    1.1779    1.6164   -0.54163   1.8831   -1.4367
```

calculate similarity: doc1.similarity(doc2)

Vectorization

01

Gensim word2vec/ fasttext pretrained model (Chinese wiki)

02

spaCy 3.0 - transformers (bert-base-chinese as example)

```
[ ]  import spacy
     nlp = spacy.load("zh_core_web_trf")
```

```
[ ]  doc = nlp("防疫就要宅在家")
     for token in doc:
         print(token.text, token.vector, token.dep_, token.pos_)
```

```
防疫 [] nsubj VERB
就要 [] dep VERB
宅 [] ROOT VERB
在家 [] advmod:rcomp VERB
```

```
doc2 = nlp("沐浴球全台康是美，寶雅，tomad's以及各大通路都買得到")
for ent in doc2.ents:
    print([ent.text, ent.start_char, ent.end_char, ent.label_])
```

```
['沐', 0, 1, 'PRODUCT']
['浴球', 1, 3, 'PRODUCT']
['台康', 4, 6, 'GPE']
['是', 6, 7, 'PRODUCT']
['美', 7, 8, 'ORG']
['寶雅', 9, 11, 'ORG']
['to', 12, 14, 'ORG']
["mad's", 14, 19, 'ORG']
```

Conclusion

:)

Clear & nice pipelines for different languages

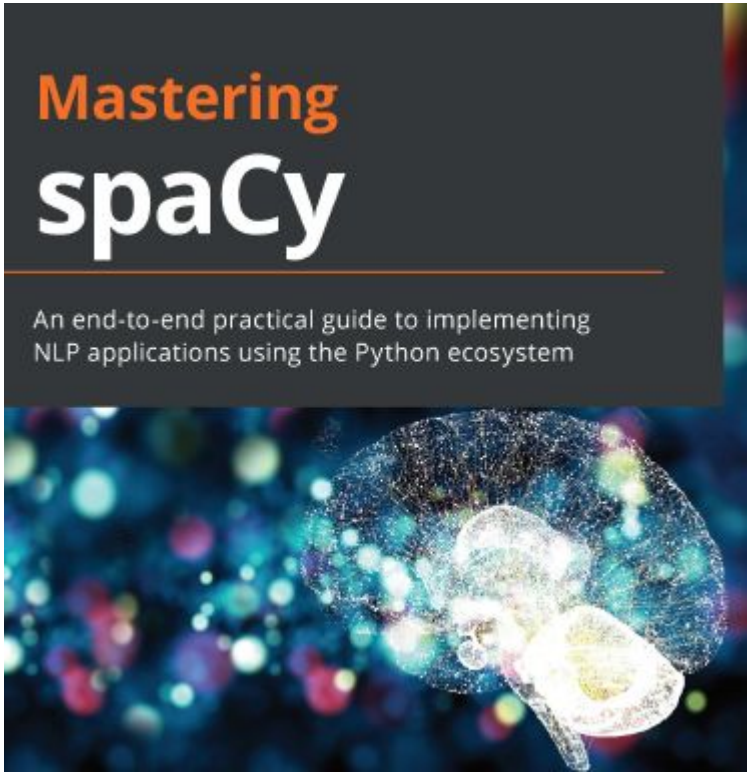
:)

Faster operation without GPU device

:(

Limited performances in Chinese models (unable to be customized)

Reference


Mastering
spaCy
An end-to-end practical guide to implementing
NLP applications using the Python ecosystem