Python for Data Science Cheat Sheet spaCy

Learn more Python for data science interactively at www.datacamp.com

DataCamp

About spaCy

spaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python. It's designed specifically for production use and helps you build applications that process and "understand" large volumes of text. **Documentation:** spacy.io

```
$ pip install spacy
```

```
import spacy
```

Statistical models

Download statistical models

Predict part-of-speech tags, dependency labels, named entities and more. See here for available models: spacy.io/models

Spans

Accessing spans

Span indices are **exclusive**. So `doc[2` token 2, up to – but not including! –

```
doc = nlp("This is a text")
span = doc[2:4]
span.text
# 'a text'
```

Creating a span manually

```
# Import the Span object
from spacy.tokens import Span
# Create a Doc object
doc = nlp("I live in New York")
# Span for "New York" with label
span = Span(doc, 3, 5, label="GP
span.text
# 'New York'
```

Linguistic features

Attributes return label IDs. For strin

Word vectors and similarity

To use word vectors, you need to install the larger models ending in `md` or `lg`, for example `en_core_web_lg`.

Comparing similarity

```
doc1 = nlp("I like cats")
doc2 = nlp("I like dogs")
# Compare 2 documents
doc1.similarity(doc2)
# Compare 2 tokens
doc1[2].similarity(doc2[2])
# Compare tokens and spans
doc1[0].similarity(doc2[1:3])
```

Accessing word vectors

```
# Vector as a numpy array
doc = nlp("I like cats")
# The L2 norm of the token's vector
doc[2].vector
doc[2].vector_norm
```

Pipeline components

Extension attributes

Custom attributes that are registere
`Token` and `Span` classes and becom

```
from spacy.tokens import Doc, Tok
doc = nlp("The sky over New York
```

Attribute extensions

```
# Register custom attribute on To
Token.set_extension("is_color", d
# Overwrite extension attribute w
doc[6]._.is_color = True
```

Property extensions

```
# Register custom attribute on Do
get_reversed = lambda doc: doc.te
Doc.set_extension("reversed", get
# Compute value of extension attr
doc._.reversed
# 'eulb si kroY weN revo yks ehT'
```

Method extensions