Python für NLP

8 Python für NLP
- NLTK
- spaCy

Was ist spaCy?

- Python-Module für NLP-Entwicklung
- Fokus auf Geschwindigkeit und State-of-the-Art-Performanz
- Konzentration auf wenige essentielle Aufgaben
- für jede Aufgabe nur ein Algorithmus implementiert
- unterstützte Sprachen: Englisch, Deutsch, Französisch, Spanisch

Ein Beispiel

```
>>> import spacy
>>> nlp = spacy.load("en")
>>> doc = nlp(u"I love computational linguistics!")
>>> for token in doc:
...     print token, token.tag_, token.prob

I PRP -4.064180850982666
love VBP -7.974212169647217
computational JJ -19.579313278198242
linguistics NNS -19.579313278198242
! . -5.515825748443603 5
```

NLTK vs spaCy

NLTK:

- gut geeignet für Lehre
- enthält viele Algorithmen und Schnittstellen
- gut dokumentiert

spaCy:

- Idee: “getting things done”
- schneller und bessere Ergebnisse als NLTK
- kann sehr gut im Pre-Processing verwendet werden (auch für Deutsch)!

Dokumentation

- API-Dokumentation, Beispiele und Tutorials:
  `https://spacy.io/docs/`

Daten in spaCy

```
>>> doc = nlp(u"I love computational linguistics!")
```

- Annotationen gespeichert in Objekt der `Doc`-Klasse
- einzelne Wörter: Objekte der `Token`-Klasse
- Zugriff auf Wörter/Sequenzen: `doc[i]`, `doc[(i, j)]`
- annotiert: Tokenisierung, Satz-Splitting, POS-Tags, Dependenzbäume, Named Entity Recognition, Sentiment
- weitere Informationen: Wahrscheinlichkeit nach Language Model, Wort-Vektoren

Informationen auf Wortebene

```
>>> doc = nlp(u"I love computational linguistics!")
>>> print doc[1].tag_
```

Alle Attribute: `https://spacy.io/docs/api/token`

Informationen auf Dokumentenebe

```
>>> doc = nlp(u"I love computational linguistics!")
>>> print doc.vector
```

Alle Attribute: `https://spacy.io/docs/api/doc`

Aufbau I

- man muss zunächst die Pipeline laden:

```
>>> import spacy
>>> nlp = spacy.load("en")
```

- die Pipeline erwartet Text(e) als Unicode-Strings:

```
>>> doc = nlp(u"""I love computational linguistics!
Green frogs are green""")
>>> docs = nlp.pipe(list_of_texts)
```

Aufbau II

- dann kann man über Dokumente, Sätze und Token iterieren:

```
>>> for token in doc:
...     print(token.lemma_)
```

```
>>> other_doc = nlp(u"I hate geometry.")
>>> for sentence in doc.sents:
...     print sentence, sentence.similarity(other_doc)
I love computational linguistics! 0.839319268367
Green frogs are green. 0.646078181155
```

Vorhandene Annotationen

- schon vorhandene Annotation zu nutzen ist auch möglich:

```
>>> import spacy
>>>
>>> nlp = spacy.load("en")
>>>
>>> tokens = [u"Just", u"some", u"example"]
>>> doc = spacy.tokens.Doc(nlp.vocab, words=tokens)
>>> nlp.tagger(doc)
>>> for tok in doc:
...     print tok.pos_, tok.tag_
```

https://spacy.io/docs/usage/processing-text

Übung 12