spaCy Cheat Sheet

by Nuozhi via cheatography.com/122797/cs/22963/

Init

```
from spacy.lang.en import English
nlp = English()
```

Basic

```
doc = nlp("SOME TEXTS")
span = doc[i:j]
token = doc[i]
```

Pre-trained Model

```
nlp = spacy.load('en_core_web_sm')
doc = nlp(MY_TEXT)
```

Name entity

```
doc.ents
```

.text
.label_

Matcher

```
matcher = spacy.matcher.Matcher(nlp.vocab)
matches = matcher(doc)
```

[(id, start, end)]

Add pattern to matcher

```
pattern = [ { key: value } ]
matcher.add("PATTERN_NAME", None, pattern)
```

Two types of key:
1. regex pattern
2. label (i.e. POS, entity)

spacy.tokens

Doc	`Doc(nlp.vocab, words=words, spaces = spaces)`
Span	`Span(doc, i, j, label="PERSON")`

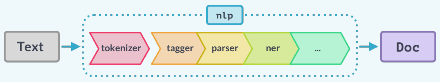
index: i, j
words: a collection of words
spaces: a collecture of booleans

Similarity

word vector	token.vector
Doc similarity	`doc1.similarity(doc2)`
Span similarity	`span1.similarity(span2)`
Token similarity	`token1.similarity(token2)`
Doc-Token similarity	`doc.similarity(token)`

return a similarity score 0~1
NOT for small model
cosine similarity by default

Pipeline



```
nlp.pipe_names
nlp.pipeline
```

Add pipeline component

```
def fn(doc):
    # function body
    return doc
nlp.add_pipe(fn, last, first, before, after)
```

Set custom attributes

add metadata	`doc._.ATTR = "ATTRIBUTE NAME"`
register globally	`Doc.set_extension("ATTR", default=None)`

set to doc, tokens, spans
access property via ._

Extension attribute types

attribute	`Token.set_extension("ATTR", defaut=Bool)`
property	`Span.set_extension("PROP", getter=fn)`
method	`Doc.set_extension("METHOD", method=fn)`

Boost up

```
nlp.pipe(DATA)
```

Passing in context

```
data = [ ("SOME TEXTS", {"KEY": "VAL"}), (...), ]
# Method 1
for doc, ctx in nlp.pipe(data, as_tuple=True):
    print( doc.ATTR, ctx[KEY] )
# Method 2
Doc.set_extension("KEY", default=None)
for doc, ctx in nlp.pipe(data, as_tuples=True):
    doc._.KEY = ctx["KEY"]
```

Using tokenizer only

```
# Method 1
doc = nlp.make_doc("SOME TEXTS")
# Method 2
with nlp.disable_pipes("tagger", "parser"):
    doc = nlp(text)
```