

### 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_
```

### 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 collection of booleans

### 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)

### Phrase matching

```
matcher =
spacy.matcher.PhraseMatcher(nlp.vocab)
pattern = nlp("Golden Retriever")
matcher.add("DOG", None, pattern)
for match_id, start, end in matcher(doc):
    span = doc[start:end]
```

### Similarity

**word vector** token.vector

**Doc** doc1.similarity(doc2)

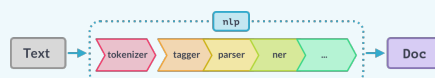
**Span** span1.similarity(span2)

**Token** token1.similarity(token2)

**Doc by Token** 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** doc.\_\_ATTR = "ATTRIBUTE"

**metadata** NAME"

**register** Doc.set\_extension("ATTR", globally default=None)

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

### Extension attribute types

**attribute** Token.set\_extension("ATTR", default=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("tag-
ger", "parser"):
    doc = nlp(text)
```

