**Pipeline components**

Functions that take a `Doc` object, modify it and return it.

- `nlp`
- `tokenizer`
- `tagger`
- `parser`
- `ner`

**Pipeline information**

```python
custom_components = [nlp('I like cats')] # Function that modifies the doc and returns it
def custom_component(doc):
    print("Do something to the doc here!")
    return doc
nlp.add_pipe(custom_component, first=True)
```

**Comparing similarity**

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

**Accessing word vectors**

- `doc = nlp("The sky over New York is blue")`
- `doc[3:5].has_label("GPE")`
- `Span.set_extension("has_label", method=has_label)`
- `get_reversed = lambda doc: doc.text[::-1]`
- `Token.set_extension("is_color", default=False)`

**Extension attributes**

Custom attributes that are registered on the global `Doc`, `Token` and `Span` classes and become available as `. `_.

```python
from spacy.tokens import Doc, Token, Span
has_label = lambda span, label: span.label_ == label
Span.set_extension("has_label", method=has_label)
get_reversed = lambda doc: doc.text[::-1]
Token.set_extension("is_color", default=False)
```

**Attribute extensions**

- Register custom attribute on Token class
- Overwrite extension attribute with default value

```python
from spacy.tokens import Doc, Token, Span
has_label = lambda span, label: span.label_ == label
get_reversed = lambda doc: doc.text[::-1]
Token.set_extension("is_color", default=False)
```

**Property extensions**

- Register custom attribute on Doc class
- Compute value of extension attribute with getter

```python
get_reversed = lambda doc: doc.text[::-1]
```

**Method extensions**

- Register custom attribute on Span class
- Compute value of extension attribute with method

```python
get_reversed = lambda doc: doc.text[::-1]
```

**Rule-based matching**

**Token patterns**

- `"love cats", "loving cats", "loved cats"`
- `"10 people", "twenty people"`
- `"book", "a cat", "the sea"`
- `"DET", "OP": "?"`

**Operators and quantifiers**

- `-` Negate pattern and match exactly 0 times.
- `?` Make pattern optional and match 0 or 1 times.
- `+` Require pattern to match 1 or more times.
- `*` Allow pattern to match 0 or more times.

**Glossary**

- **Tokenization**: Segmenting text into words, punctuation etc.
- **Lemmatization**: Assigning the base forms of words, for example: "was" → "be" or "rats" → "rat".
- **Sentence Boundary Detection**: Finding and segmenting individual sentences.
- **Part-of-speech (POS) Tagging**: Assigning word types to tokens like verb or noun.
- **Dependency Parsing**: Assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object.
- **Named Entity Recognition (NER)**: Labeling named "real-world" objects, like persons, companies or locations.
- **Text Classification**: Assigning categories or labels to a whole document, or parts of a document.
- **Statistical model**: Process for making predictions based on examples.
- **Training**: Updating a statistical model with new examples.