spaCy

Intro to Python NLP Library

SpaCy Intro

- Open-Source Python Library for Advanced NLP
- Made for Production Use
- For Multiple Purposes including
  - Information Extraction
  - Language Understanding
    - Sentiment Analysis (determining how someone felt based on text such as reviews)
    - Classification
    - Summarization

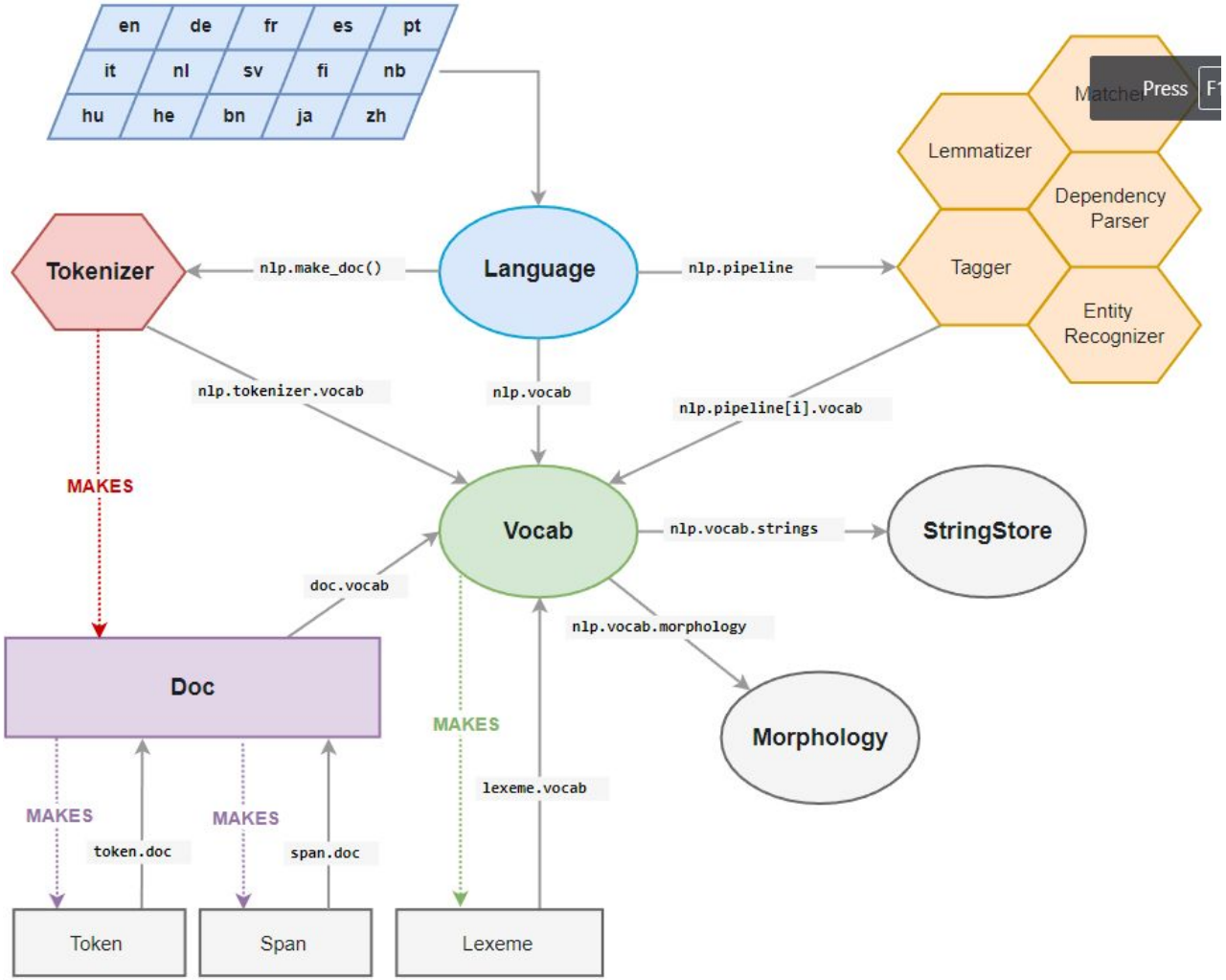
Features

Tokenization	Segmenting text into words, punctuations marks etc.
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.
Lemmatization	Assigning the base forms of words. For example, the lemma of "was" is "be", and the lemma of "rats" is "rat".
Sentence Boundary Detection (SBD)	Finding and segmenting individual sentences.
Named Entity Recognition (NER)	Labelling named "real-world" objects, like persons, companies or locations.

Features (2/2)

Similarity	Comparing words, text spans and documents and how similar they are to each other.
Text Classification	Assigning categories or labels to a whole document, or parts of a document.
Rule-based Matching	Finding sequences of tokens based on their texts and linguistic annotations, similar to regular expressions.
Training	Updating and improving a statistical model's predictions.
Serialization	Saving objects to files or byte strings.

Architecture



Container Objects

Doc	A container for accessing linguistic annotations.
Span	A slice from a Doc object.
Token	An individual token — i.e. a word, punctuation symbol, whitespace, etc.
Lexeme	An entry in the vocabulary. It's a word type with no context, as opposed to a word token. It therefore has no part-of-speech tag, dependency parse etc.

Processing pipeline

Language	A text-processing pipeline. Usually you'll load this once per process as nlp and pass the instance around your application.
Pipe	Base class for processing pipeline components.
Tagger	Annotate part-of-speech tags on Doc objects.
DependencyParser	Annotate syntactic dependencies on Doc objects.
EntityRecognizer	Annotate named entities, e.g. persons or products, on Doc objects.
TextCategorizer	Assigning categories or labels to Doc objects.
Tokenizer	Segment text, and create Doc objects with the discovered segment boundaries.

Processing Pipeline (2/2)

Lemmatizer	Determine the base forms of words.
Morphology	Assign linguistic features like lemmas, noun case, verb tense etc. based on the word and its part-of-speech tag.
Matcher	Match sequences of tokens, based on pattern rules, similar to regular expressions.
PhraseMatcher	Match sequences of tokens based on phrases.

Other classes

Vocab	A lookup table for the vocabulary that allows you to access Lexeme objects.
StringStore	Map strings to and from hash values.
Vectors	Container class for vector data keyed by string.
GoldParse	Collection for training annotations.
GoldCorpus	An annotated corpus, using the JSON file format. Manages annotations for tagging, dependency parsing and NER.

Statistical Models

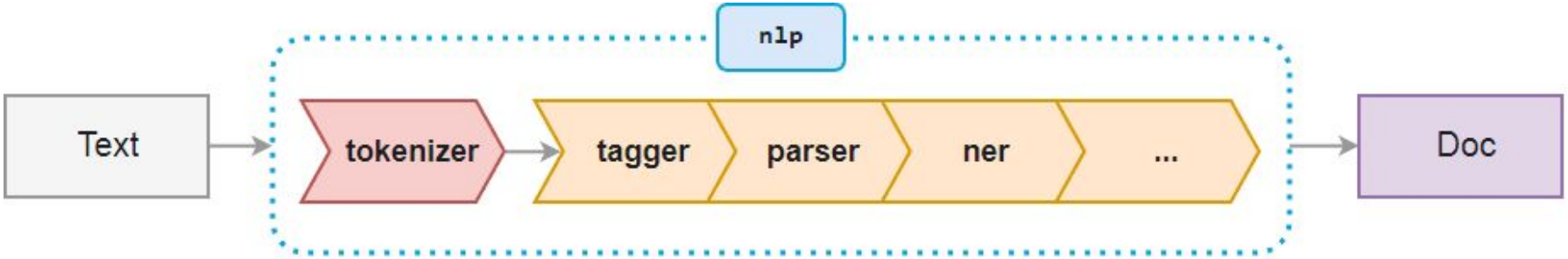
Binary weights for the part-of-speech tagger, dependency parser and named entity recognizer to predict those annotations in context.

Lexical entries in the vocabulary, i.e. words and their context-independent attributes like the shape or spelling.

Word vectors, i.e. multi-dimensional meaning representations of words that let you determine how similar they are to each other.

Configuration options, like the language and processing pipeline settings, to put spaCy in the correct state when you load in the model.

Language Processing Pipeline



- Takes Text as Input
- Processes Text through a configured set of tasks that analyze the text
- Output a Doc with various annotations on tokens, sequences of tokens, etc.
- Allow for a flexible and efficient way of defining how the text should be processed that is callable.

Pipeline (2/3)

NAME	COMPONENT	CREATES	DESCRIPTION
tokenizer	`Tokenizer`	`Doc`	Segment text into tokens.
tagger	`Tagger`	`Doc[i].tag`	Assign part-of-speech tags.
parser	`DependencyParser`	`Doc[i].head`, `Doc[i].dep`, `Doc.sents`, `Doc.noun_chunks`	Assign dependency labels.
ner	`EntityRecognizer`	`Doc.ents`, `Doc[i].ent_iob`, `Doc[i].ent_type`	Detect and label named entities.
textcat	`TextCategorizer`	`Doc.cats`	Assign document labels.
…	custom components	`Doc._.xxx`, `Token._.xxx`, `Span._.xxx`	Assign custom attributes, methods or properties.

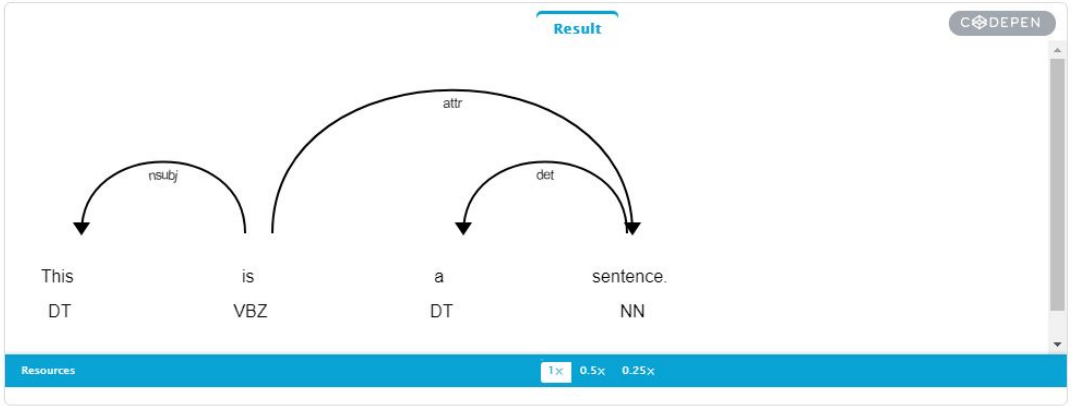
Pipeline (3/3)

"pipeline": ["tagger", "parser", "ner"]

`meta.json` - Configuration file with components for pipelines as well as other configurations such as language, etc.

Visualizers (official with 2.0 release)

```
import spacy

from spacy import displacy



nlp = spacy.load('en')

doc = nlp(u'This is a sentence.')

displacy.serve(doc, style='dep')
```



Spacy.io and other links

- https://spacy.io/
- https://spacy.io/usage/#section-quickstart
- https://spacy.io/usage/
- https://spacy.io/usage/models
- https://spacy.io/usage/spacy-101
- https://spacy.io/usage/examples
-
- https://nlpforhackers.io/complete-guide-to-spacy/
- https://www.kaggle.com/enerrio/scary-nlp-with-spacy-and-keras

Demo Examples