NLP in Python: Unleashing the power of spaCy

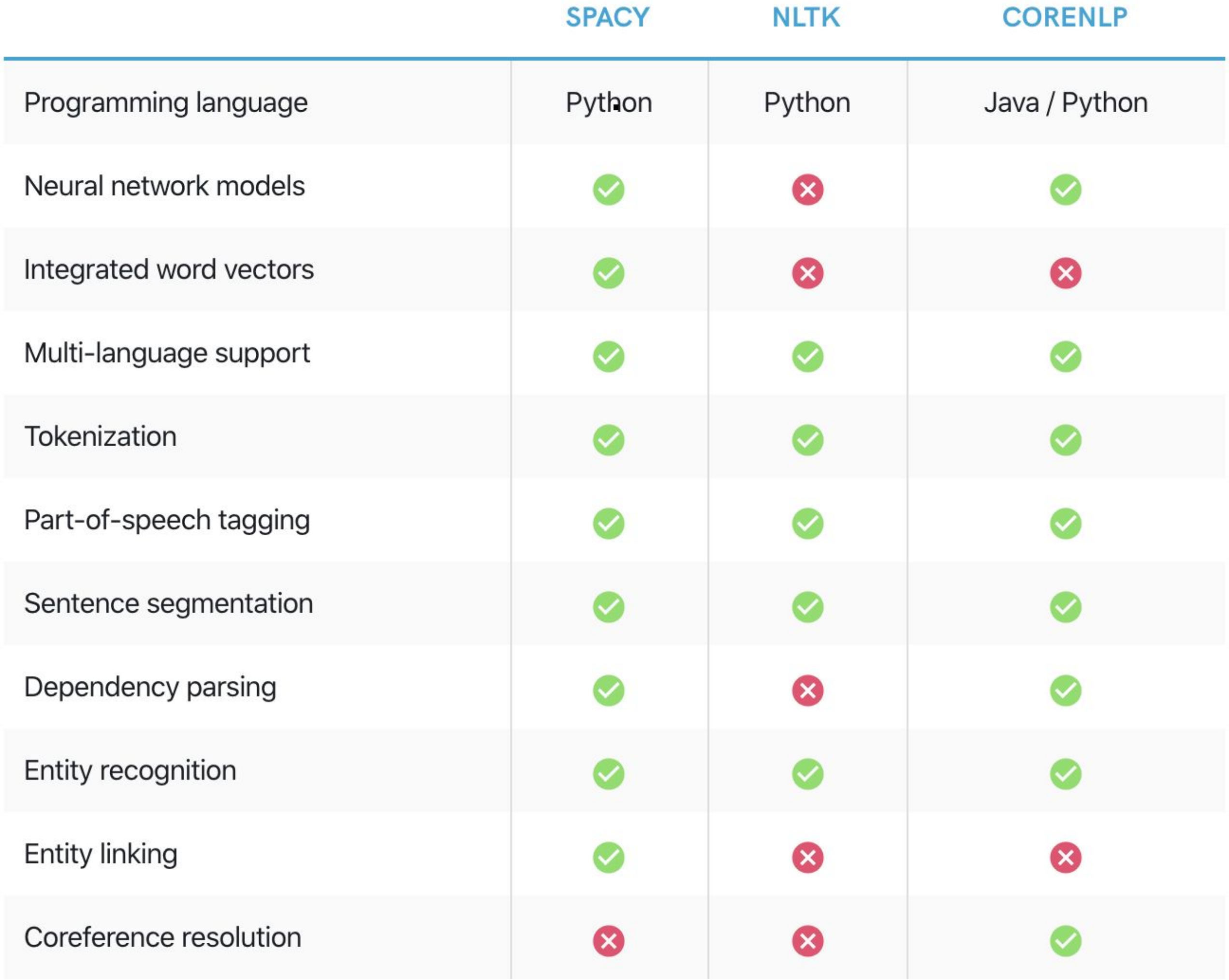
A practitioner’s adventure

Mario Garcia-Armas

FLIPBOARD

What is spaCy?

- Advanced NLP library written in Python / Cython.
- Supports multiple languages (53+ at the time of this writing).
- Features part of speech (POS) tagging, dependency parsing, named entity recognition (NER), noun chunking, and pretrained word vectors.
- It’s open source and very actively developed by a great community. Repo can be found at https://github.com/explosion/spaCy.



	SPACY	NLTK	CORENLP
Programming language	Python	Python	Java / Python
Neural network models	✓	✗	✓
Integrated word vectors	✓	✗	✗
Multi-language support	✓	✓	✓
Tokenization	✓	✓	✓
Part-of-speech tagging	✓	✓	✓
Sentence segmentation	✓	✓	✓
Dependency parsing	✓	✗	✓
Entity recognition	✓	✓	✓
Entity linking	✓	✗	✗
Coreference resolution	✗	✗	✓

spaCy

More details available at
https://spacy.io/usage/facts-figures

FLIPBOARD

Basic spaCy usage

```
from spacy.lang.en import English

nlp = English()

doc = nlp('PyData is the world’s best meetup!')

print(*(f'{token.text} {"✔️" if token.is_stop else "❌"}'
        for token in doc), sep='|')
# OUTPUT:

# PyData ❌|is ✔️|the ✔️|world ❌|'s ✔️|best ❌|meetup ❌|! ❌
```

Basic spaCy usage (cont.)

What if we want to do more...? For example, say we wanted to figure out where the nouns are in that sentence. To achieve this, we need to unleash the full power of spaCy pipelines and statistical models!

```
print(f"Current spaCy pipeline: {nlp.pipe_names}")

# OUTPUT:
# Current spaCy pipeline: []
```

Introduction to spaCy models

https://spacy.io/models

- Current support for 11 languages (English, French, German, Spanish, etc.).
- One-line installation: e.g., `python -m spacy download en_core_web_sm`.
- Models are installed as pip packages for convenience.
- Supports fine-tuning the models to your own data.

Introduction to spaCy models (cont.)

```
import spacy

nlp = spacy.load('en_core_web_sm')

doc = nlp('PyData is the world\'s best meetup!')

print(*(f'{token.text}' for token in doc if

        token.pos_ in {'NOUN', 'PROPN'}), sep=' | ')

# OUTPUT

# PyData | world | meetup
```

Customizing spaCy: best-in-class tokenizer

How does spaCy tokenization work? 🤔 Here is a rough idea of the algorithm:

- Split text at whitespace characters and iterate over terms.
- Match prefixes / suffixes while possible and handle those first (e.g., open brackets, question marks, etc.).
- Match special tokens that should never be separated.
- Match hardcoded exceptions (e.g., “don’t” splits into (“do”, “not”)).
- Match non-whitespace separators, such as hyphens.

Customizing spaCy: best-in-class tokenizer (cont.)

Is the default tokenizer (for English) the best money can buy? The answer clearly depends on your use case.

Goal: Build a tokenizer that does not split hyphenated words.

How do we go about doing this...?

Customizing spaCy: best-in-class tokenizer (cont.)

Welcome to what I call “the blurse of open source libraries”:

- Pros: You have access to the source code!
- Cons: You have access to the source code!

Let’s see this in practice in the live demo! 🤣

Customizing spaCy: simpler noun chunker

SpaCy pipelines are very flexible and they allow for fine-tuned customization.

Goal: Build a simple noun chunker that tags consecutive nouns, optionally preceded by a single adjective.

Let's jump straight into the live demo!

Some references

- https://spacy.io/ (of course)
- https://course.spacy.io/ (highly recommended)
- https://www.youtube.com/channel/UCFduT4kW_eLDbEW6XoA5F0A (Explosion channel on Youtube -- lots of great videos)
- https://explosion.ai/blog/ (tons of cool NLP/spaCy stuff)
- Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing (for anything classical NLP)