**Exploring spaCy: Your one-stop library to build advanced NLP products**

*It’s a fast, seamless, and state-of-the-art library for natural language processing*

As Natural Language Processing or NLP becomes a staple to build modern AI-enabled products, open-source libraries prove a boon for their architects as they help cut down on the time and allow greater flexibility and seamless integration. spaCy is one such library for advanced NLP in the popular Python language. Today, we will explore spaCy, its features, and how you can get started with the free library to seamlessly build NLP products.

**What is spaCy?**

A free, open-source library, [spaCy](https://spacy.io/" \t "_blank) is suited for those working with a lot of text. It is designed for production use and allows you to build applications that have to deal with a large volume of text. You can use spaCy to build systems for information extraction, natural language understanding, or pre-process text for deep learning.

**What does it offer?**

spaCy offers a number of features and capabilities ranging from linguistic concepts to machine learning functionality. Some of its [features](https://spacy.io/usage/spacy-101#features) include:

***Tokenization***

Segmenting text into words, punctuation 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.

***Entity Linking (EL)***

Disambiguating textual entities to unique identifiers in a Knowledge Base.

***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.

**Other features include:**

● Support for 61+ languages

● 46 statistical models for 16 languages

● State-of-the-art speed

● Easy deep learning integration

● Syntax-driven sentence segmentation

● Built-in visualizers for syntax and NER

● Convenient string-to-hash mapping

● Export to numpy data arrays

● Easy model packaging and deployment

● Robust, rigorously evaluated accuracy

… and this is not even the complete list. However, as data scientists, we also need to look at all that spaCy isn’t. For instance, it is not an API or a platform to provide SaaS or web app. On the contrary, its library allows you to build NLP apps. It isn’t designed for chatbots either. It can, however, provide text processing capabilities as an underlying technology for chatbots.

**spaCy’s architecture**

The central [architecture](https://spacy.io/usage/spacy-101#architecture) of spaCy includes the Doc, which owns the sequence of tokens and all their annotations, and the Vocab object that owns a set of look-up tables that make common information available across documents. This allows for centralizing strings, word vectors, and lexical attributes, thereby avoiding storing multiple copies of this data and in turn, saving memory and ensuring there’s a single source of truth.

Similarly, the text annotations are designed — to allow a single source of truth. For this, the Doc object owns the data, and Span and Token point into it. It is constructed by the Tokenizer and then modified in place by the components of the pipeline. The Language object coordinates these components by taking raw text and sending it through the pipeline which returns an annotated document. It also orchestrates training and serialization.

**Getting started with spaCy**

Considered by many [as Ruby on Rails of NLP](https://www.kdnuggets.com/2018/05/getting-started-spacy-natural-language-processing.html), spaCy is easy to install and its API is simple and productive. You can get started with spaCy by first installing it along with its English language model. It is compatible with 64-bit CPython 2.7 / 3.5+ and runs on Unix/Linux, macOS/OS X and Windows.

Next, use a sample of text. After spaCy, import displaCy, which is used for visualizing some of spaCy’s modeling, and a list of English stop words. Then, load the English language model as a Language object and then call it in the sample text. This will return a processed Doc object.

This processed doc is split into individual words and annotated, but it contains all information of the original text. One can get the offset of a token into the original string, or reconstruct the original by joining the tokens and their trailing whitespace.

You can install the latest spaCy releases over pip and conda. I will try to show a few samples of usage

Initializing spaCy in python

import spacy  
nlp = spacy.load('en\_core\_web\_sm')

nlp("He went to play basketball")

Part-of-Speech (POS) Tagging using spaCy

import spacynlp = spacy.load('en\_core\_web\_sm')

doc = nlp("He went to play basketball")

for token in doc:  
 print(token.text, "-->", token.pos\_)

Output

He –> PRON  
went –> VERB  
to –> PART  
play –> VERB  
basketball –> NOUN

Dependency Parsing using spaCy

# dependency parsing

for token in doc:  
 print(token.text, "-->", token.dep\_)

Output

He –> nsubj  
went –> ROOT  
to –> aux  
play –> advcl  
basketball –> dobj

The dependency tag ROOT denotes the main verb or action in the sentence. The other words are directly or indirectly connected to the ROOT word of the sentence. You can find out what other tags stand for by executing the code below:

Named Entity Recognition using spaCy

doc = nlp("Indians spent over $71 billion on clothes in 2018")

for ent in doc.ents:  
 print(ent.text, ent.label\_)

Output

Indians NORP  
over $71 billion MONEY  
2018 DATE

Conclusion

Overall this library like spaCy kind of abstracts away all the complexity in terms of usage, and getting started with it is as simple as just writing a few lines of code. It really enables data engineers like me to quickly get onboarded and use it for unstructured textual data.