1. The Basics of spaCy

Dr. W.J.B. Mattingly
Smithsonian Data Science Lab and United States Holocaust Memorial Museum

August 2021

In this notebook, we will not be working with spaCy in code, rather in concept. This entire JupyterBook is designed around approaching spaCy top-down. By this I mean approaching the things that spaCy does and can do and then exploring how to implement that in code. I think this is necessary so that as you explore the smaller components of spaCy, such as the Lemmatizer, you will understand how it fits into the larger architecture of the spaCy framework.

1.1. What is spaCy?

A good way to begin is by exploring the question, "What is spaCy?" spaCy (yes, spelled with a lowercase "s" and uppercase "C" is a natural language processing framework. **Natural language processing**, or NLP, is a branch of linguistics that seeks to parse human language in a computer system. This field is generally referred to as computational linguistics, though it has far reaching applications beyond academic linguistic research.

NLP is used in every sector of industry, from academics who leverage it to aid in research to financial analysts who try and predict the stock market. Lawyers use NLP to help analyze thousands of legal documents in seconds to target their research and medical doctors use it to parse patient charts. NLP has been around for decades, but with the increased promise of deep learning, a subfield of machine learning, that NLP rapidly expanded. This is because, as we shall learn all too well throughout this book, language is inherently ambiguous. By this, I mean that language does not always make perfect sense. In some cases, it is entirely illogical. The double-negative in English is a good example of this. In some contexts, it can be an emphatic positive, as in, "I cannot stress this enough, I do not like pasta." This is, of course a lie. I love pasta, but you get my point. In other cases, the double negative can be an emphatic negative, as in, "I ain't not doing that!"

As humans, especially native speakers of a language, we can parse these complex illogical statements with ease, especially with enough context. For computers, this is not always easy.

Because NLP is such a complex problem for computers, it requires a complex solution. The answer has been found in artificial neural networks, or ANNs or neural nets for short. These are the primary areas of research for deep learning practitioners. As the field of deep learning (and machine learning in general) expand and advance, so too does NLP. New methods for training, such as transformer models, push the field further.

1.2. How to Install spaCy

In order to install spaCy, I recommend visiting their website, here: https://spacy.io/usage . They have a nice user-friendly interface. Input your device settings, e.g. Mac or Windows or Linux, and your language, e.g. English, French, or German. The web-app will automatically populate the commands that you need to execute to get started. Since this is a JupyterBook, we can install these with a "!" before in a cell to indicate that we want to run a terminal command. I will be installing spaCy and thee small English model, en_core_web_sm.

```
!pip install spacy
```

```
!python -m spacy download en_core_web_sm
```

Now that we've installed spaCy let's import it to make sure we installed it correctly.

```
import spacy
```

Great! Now, let's make sure we downloaded the model successfully with the command below.

```
nlp = spacy.load("en_core_web_sm")
```

Print to PDF

Excellent! spaCy is now installed correctly and we have successfully downloaded the small English model. We will pick up here with the code in the next notebook. For now, I want to focus on big-picture items, specifically spaCy "containers".

1.3. Containers

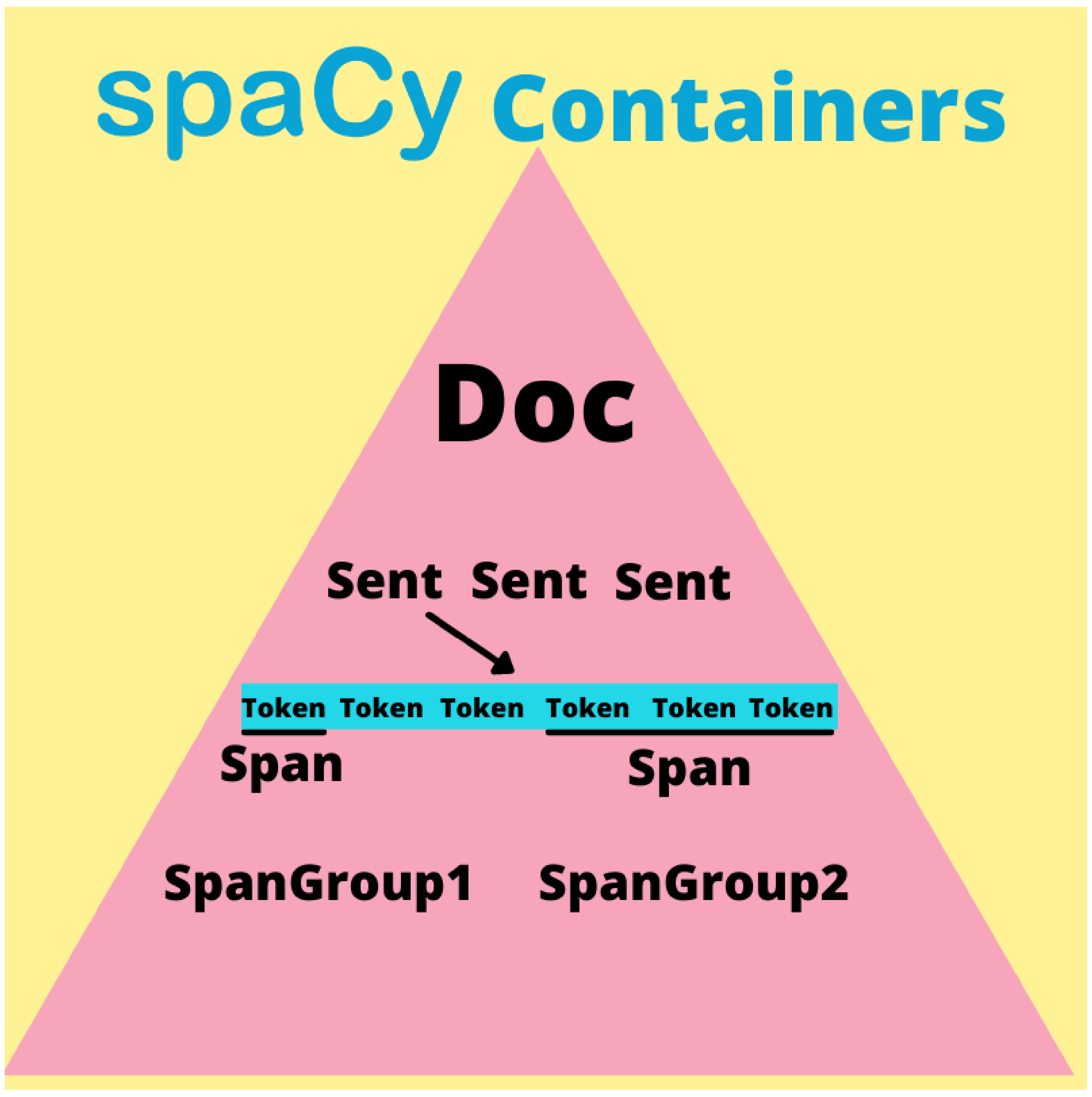
Containers are spaCy objects that contain a large quantity of data about a text. When we analyze texts with the spaCy framework, we create different container objects to do that. Here is a full list of all spaCy containers. We will be focusing on three (emboldened): Doc, Span, and Token.

- **Doc**
- DocBin
- Example
- Language
- Lexeme
- **Span**
- SpanGroup
- **Token**

I created the image below to show how I visualize spaCy containers in my mind. At the top, we have a Doc container. This is the basis for all spaCy. It is the main object that we create. Within the Doc container are many different attributes and subcontainers. One attribute is the Doc.sents, which contains all the sentences in the Doc container. The doc container (and each sentence generator) is made up of a set of token containers. These are things like words, punctuation, etc.

Span containers are kind of like token, in that they are a piece of a Doc container. Spans have one thing that makes them unique. They can cross multiple tokens.

We can give spans a bit more specificity by classifying them into different groups. These are known as SpanGroup containers.



If you do not fully understand this dynamic, do not worry. You will get a much better sense of this pyramid as we move forward throughout this book. For now, I recommend keeping this image handy so you can refer back to it as we progress through Part 1 of this book in which we explore the basics of spaCy. In the next chapter, we will start applying these concepts in code by creating a doc object and learning about the different attributes containers have as well as how to find the linguistic annotations.