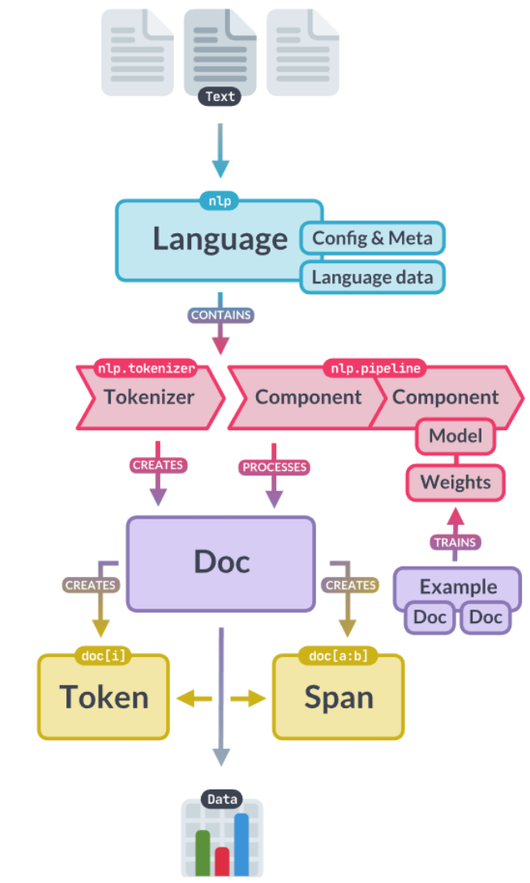
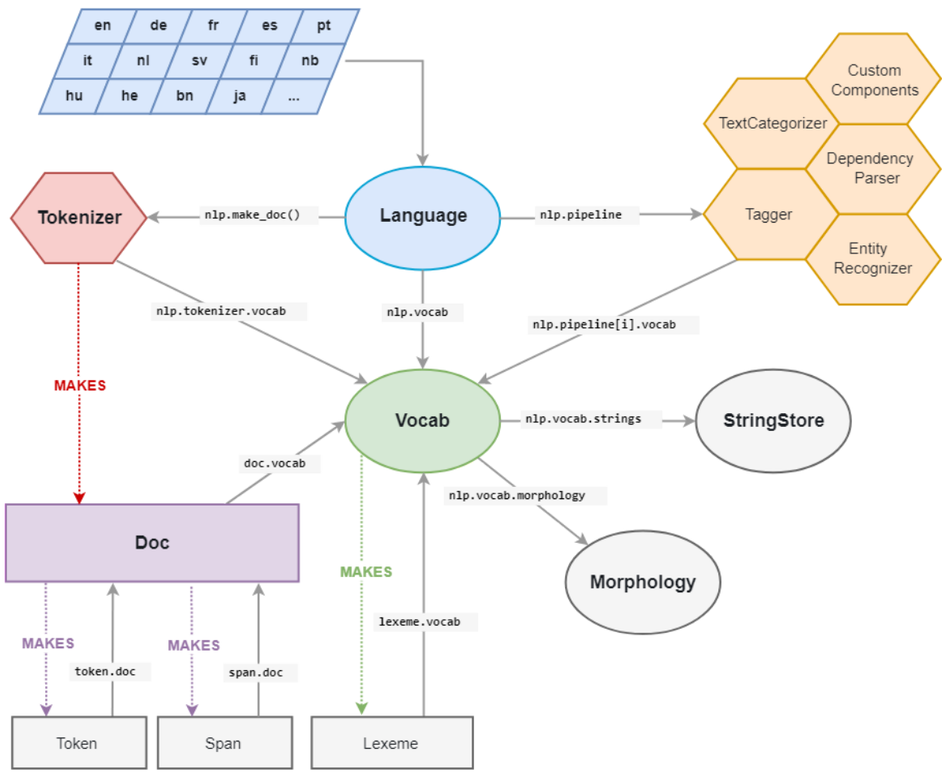
Spacy

官网介绍
模型下载地址
中文模型





两大核心对象+Vocab存储单元

Language（习惯上实例化为nlp这个名字），包含两个主要构件：

1.Tokenizer:作用是数据转换，把传进来的字符串，转换成一个一个Token，这个Token是Spacy自定义的一种核心类型
打个比方，python自己都有列表了，numpy怎么还弄出来一个ndarray出来，它们的数据从视图上看都是序列，ndarray可以近似看成列表的plus版本
2.Component：就是一堆神经网络模型，把刚刚转换好的数据进行一系列处理，比如命名体识别，词性分析等
这些神经网络组件可以自己删减，也可以增加自己的神经网络，不过你要到官网看它是怎么运行的
这些神经网络的权重，一般由它内置的语言数据预训练好的，你只要加载就行，一开始你下载的那个"en_core_web_sm"本质就是模型的权重
当然也可以拿你自己的数据（需要转换成spacy的Example对象格式）去训练这些模型

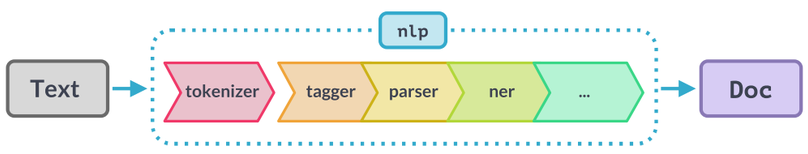
Doc（习惯上实例化为doc）：

相当于pandas里面的dataframe对象，就是存储上面token序列的一个壳子，dataframe本质就是存储一张矩阵(ndarray)的壳子。
并且可以通过切片方法访问，返回的类型就是Token，以及大一点的Span对象，反正对比dataframe去理解
上面讲的是doc对象最核心的data存储，这样就把外面的字符串通过Tokenizer转换成doc对象了，这个对象交给nlp的神经网络组件
进行各种处理，处理的结果放在doc对象的其他属性当中，比如sents，ents

Vocab：

这个东西可以理解为管理存储的地方，前面所有对象引用的就是它，第一张图里面其实也有画这个，但是为了更方便讲解，我把第一张图的Vocab删了

Pipline



使用步骤：

第一步：实例化nlp对象
第二步：将字符串传给nlp对象，它给你完成数据转换变成doc，并对doc进行分词，分句，命名体识别，结果存储在doc其他属性当中
第三步：取出这些处理结果，放到你的任务中去使用

模拟Spacy的架构设计

第一部分的设计思路参考常见神经网络模型架构设计：
①.Language对象相当于Transformer，是个模型架构
②.tokenizer以及各种component相当于Transformer的组件

第二部分的设计思路参考numpy，pandas的设计：
①.Vocab是存储数据的对象，Token和Span相当于把基本数据类型扩展为矩阵
②.Doc是处理好后的数据对象


```
class Language(nn.Module):
    def __init__(self):
        初始化各种组件
    def forward(self):
        doc=self.tokenizer()
        doc=self.components(doc)
        return doc
```

常见框架的设计

框架	基本数据类型	模块
sklearn	numpy，pandas	一个个独立的算法模块
NLTK	text，vocab，tree，corpus，annotation	一个个独立的算法模块

框架	基本数据类型	模块
Pytorch	tensor	基于Module设计的小组件，再拼成架构
Spacy	doc	基于Module改装的小组件，再拼成Language架构

NLTK是基于机器学习算法设计的，Spacy是基于深度学习框架设计的

简单上手使用

In [1]:
```
import spacy
```

第一步：加载模型参数，实例化Language

In [2]:
```
nlp =  spacy.load('en_core_web_sm')
```

第二步：把我的字符串交给nlp转换和处理，获得一个doc

In [3]:
```
text="The central data structures in spaCy are the Language class, the Vocab and the Doc object. The Language class is used to process a text and turn it into a Doc object. It's typically stored as a variable called nlp. The Doc object
```

In [4]:
```
doc=nlp(text)
```

第三步：从doc里面拿出我想要的东西

doc参考 主要看init中包含哪些属性，基本就是一些重要的处理结果，想了解哪个点哪个
token参考 就是分词，也是基本的数据单元，可以随便看看有什么属性和方法

1.按切片访问数据

In [5]:
```
doc[3],type(doc[3])
```

Out[5]:
```
(structures, spacy.tokens.token.Token)
```

In [6]:
```
doc[:10],type(doc[:10])
```

Out[6]:
```
(The central data structures in spaCy are the Language class,
 spacy.tokens.span.Span)
```

2.获得句子(是一个generator)

In [7]:
```
for sentence in doc.sents:
    print(sentence)
print("放句子的数据结构是一个：",type(doc.sents))
```

```
The central data structures in spaCy are the Language class, the Vocab and the Doc object.
The Language class is used to process a text and turn it into a Doc object.
It's typically stored as a variable called nlp.
The Doc object owns the sequence of tokens and all their annotations.
By centralizing strings, word vectors and lexical attributes in the Vocab, we avoid storing multiple copies of this data.
This saves memory, and ensures there's a single source of truth.
Text annotations are also designed to allow a single source of truth: the Doc object owns the data, and Span and Token are views that point into it.
The Doc object is constructed by the Tokenizer, and then modified in place by the components of the pipeline.
The Language object coordinates these components.
It takes raw text and sends it through the pipeline, returning an annotated document.
It also orchestrates training and serialization.
放句子的数据结构是一个： <class 'generator'>
```

3.词性

In [8]:
```
i=0
for token in doc:
    if i==5:
        break
    print(f"{token}-{token.pos_}")
    i+=1
```

```
The-DET
central-ADJ
data-NOUN
structures-NOUN
in-ADP
```

4.命名体

In [9]:
```
for ent in doc.ents:
    print(ent,ent.label_)
```

```
Language LOC
Vocab ORG
Language LOC
Vocab ORG
Span ORG
Token ORG
Tokenizer ORG
Language LOC
```

5.文本相似度计算

In [10]:
```
doc1=nlp("i am a dog")
doc2=nlp("he is a pig")
doc1.similarity(doc2)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_12824\1275351301.py:3: UserWarning: [W007] The model you're using has no word vectors loaded, so the result of the Doc.similarity method will be based on the tagger, parser and NER, wh
ich may not give useful similarity judgements. This may happen if you're using one of the small models, e.g. `en_core_web_sm`, which don't ship with word vectors and only use context-sensitive tensors. You can always add your own word v
ectors, or use one of the larger models instead if available.
  doc1.similarity(doc2)
```

Out[10]:
```
0.8220974469759822
```

5.可视化展示

点击上面进入官网直接查看文档，注意官网右边还有例子展示，很友好
除了displacy，还有别的可视化工具，点我
从里面找到了一个不错的在线小工具，点我

In [11]:
```
spacy.displacy.render(doc,style="ent",jupyter=True)
```

The central data structures in spaCy are the Language LOC class, the Vocab ORG and the Doc object. The Language LOC class is used to process a text and turn it into a Doc object. It's typically stored as a variable called nlp. The Doc object owns the sequence of tokens and all their annotations. By centralizing strings, word vectors and lexical attributes in the Vocab ORG , we avoid storing multiple copies of this data. This saves memory, and ensures there's a single source of truth.Text annotations are also designed to allow a single source of truth: the Doc object owns the data, and Span ORG and Token ORG are views that point into it. The Doc object is constructed by the Tokenizer ORG , and then modified in place by the components of the pipeline. The Language LOC object coordinates these components. It takes raw text and sends it through the pipeline, returning an annotated document. It also orchestrates training and serialization.

小结：
这里只是入门简单使用，这些结果的更加深入的细节就不探讨了，
包括pipline中各种模型的使用与修改，加入你自己的数据，这个在自然语言处理
课程中专门细学的时候再回头查询相关属性和方法的使用