**spaCy framework (simple model)**

NLP: extract human language is different from NLU (Natural Language Understanding)

* Named Entity Recognition (NER); Part-of-Speech Tagging (POS); Syntactic Parsing…

**spaCy Containers**

Doc: Contains the metadata about the whole text (sentences)

Token: iterate through items in Docs, we will get Tokens 🡪 words, punctuation. marks (self-contained important values)

Span: A span can be a token, or multiple tokens (Berlin, or Martin Luther King)

**Linguistic Annotation**

1. **Token Attributes**
   1. .text (only the text)
   2. .left\_edge/.right\_edge (part of the multi-word token to make up a larger span)
   3. .ent\_type\_ (the type of entity of the word)
   4. .ent\_iob\_ (to see if the token is inside, outside, or beginning of an entity)
   5. .lemma\_ (root form of word; known 🡪 we get know)
   6. .morph 🡪 what's the word is morphologically **(important for extracting data)**
   7. .pos\_ (grammatical extraction)
   8. .dep\_ (dependency relation 🡪 role it plays in a sentence)
   9. .lang\_ (language)
2. **Part of speech, dependency sparser**

**Named Entity Recognition**

NER: Extracting information from text

🡪 Approximately identify the entities in the doc. Note that they are not always accurate, and will improve accuracy as the model grows bigger.

Graphical user interface, text

Description automatically generatedText

Description automatically generated

**Word Vectors and spaCy (medium model)**

Word vectors are word embedings - numerical representation of words in multi-dimensional

🡪 Computers use this to understand the meaning of the real word 🡪 number representations

**Pipelines**

Structure of the spaCy framework: pipelines. A **pipeline** is a sequence of different **pipes**, each pipe in a computer system perform some computation/action to a set of data as it goes through the pipeline.

Each pipe changes the data a bit so it benefits the later one.



*Sample spaCy Pipeline for NER*

1. Attribute Ruler

Dependency Parser, TextCategorizer, Tokenizer, Sentencizer, Morpholog, etc.

1. Matcher

How do you add pipes?

nlp = spacy.blank("en")

nlp.add\_pipe("sentencizer")

Graphical user interface, text, application, email

Description automatically generatedNOTE: the difference between medium and small model is that medium model does not activate all pipelines like the small one 🡪 way faster

**Rules-Based spaCy**

1. **spaCy EntityRuler**

2 ways where you can add in information to spaCy language pipeline: rule-based approach, and machine learning based approach

* Rule-based: generate a set of rules based on either a list of known things, or a set of rules that can be generated through regEx, code, or linguistic features
  + Identifying dates (finite) 🡪 spaCy already has this
* Machine-learning: don't know how to write out the rules
  + Names of people (infinite)

Graphical user interface, text, application

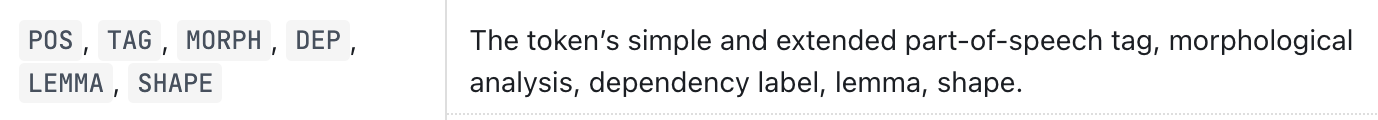
Description automatically generated

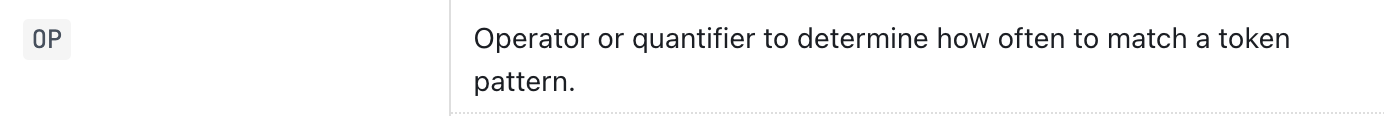
So here, we have been able to "help" the model when certain words show up. However, there is also a feature such that spaCy learns from the language pattern 🡪 **Matcher**

1. **spaCy Matcher**

The data is extracted differently from EntityRuler. Match is going to store the data within a vocab of the nlp model (nlp.vocab)

When working with a matcher, we work with things that are not necessarily an entity by itself.

FEATURE: Matching a sequence of characters, as well as a sequence of linguistic features



Text

Description automatically generated

// greedy="LONGEST" because the entities that are flagged by the model can be "January", "January 30st", "January 30st 2023" simutaneously, leading to redundant data. As a result, we only take the most significant one

However, we can see that this model may be limited, as it looks for specific sentence patterns that are not strictly enforced by human language.