**NLP in spaCy**

1) Loading and processing

What to load?

* Sentences, phrases, whole texts as strings and documents;

Processing?

* Processing raw text intelligently is difficult: most words are rare, and it's common for words that look completely different to mean almost the same thing. The same words in a different order can mean something completely different. Even splitting text into useful word-like units can be difficult in many languages.

2) Tokenization

Tokenization is the task of splitting a text into meaningful segments, called tokens.

How?

* During processing, first **tokenize** the text, i.e. segments it into words, punctuation and so on. This is done by applying rules specific to each language. For example, punctuation at the end of a sentence should be split off – whereas "U.K." should remain one token
* First, the raw text is split on whitespace characters, similar to text.split(' ') in Python. Then, the tokenizer processes the text from left to right. On each substring, it performs two checks:
* **Does the substring match a tokenizer exception rule?** For example, "don't" does not contain whitespace, but should be split into two tokens, "do" and "n't", while "U.K." should always remain one token.
* **Can a prefix, suffix or infix be split off?** For example, punctuation like commas, periods, hyphens or quotes.

If there's a match, the rule is applied, and the tokenizer continues its loop, starting with the newly split substrings.

**Algorithm summarization**

1. Iterate over space-separated substrings
2. Check whether we have an explicitly defined rule for this substring. If we do, use it.
3. Otherwise, try to consume a prefix.
4. If we consumed a prefix, go back to the beginning of the loop, so that special-cases always get priority.
5. If we didn't consume a prefix, try to consume a suffix.
6. If we can't consume a prefix or suffix, look for "infixes" — stuff like hyphens etc.
7. Once we can't consume any more of the string, handle it as a single token.

3) Filtering

Part-of-speech tagging

* This is where the statistical model comes in, which enables us to **make a prediction** of which tag or label most likely applies in this context. A model consists of binary data and is produced by showing a system enough examples for it to make predictions that generalize across the language – for example, a word following "the" in English is most likely a noun.
* Optimization: encode all strings coming from tokenizer as hash values.

English has a relatively simple morphological system. It can be handled using rules that can be keyed by the token, the part-of-speech tag, or the combination of the two. The system works as follows (spaCy):

1. The tokenizer consults a [mapping table](https://spacy.io/usage/adding-languages#tokenizer-exceptions) TOKENIZER\_EXCEPTIONS, which allows sequences of characters to be mapped to multiple tokens. Each token may be assigned a part of speech and one or more morphological features.
2. The part-of-speech tagger then assigns each token an **extended POS tag**. They express the part-of-speech (e.g. VERB) and some amount of morphological information, e.g. that the verb is past tense.
3. For words whose POS is not set by a prior process, a [mapping table](https://spacy.io/usage/adding-languages#tag-map) TAG\_MAP maps the tags to a part-of-speech and a set of morphological features.

Useful (extra info) that comes with tagging:

* Dependency tree: tags each word with the relationship it has with other words in a phrase. The final structure of all dependencies is named Dependency Tree
* Noun chunks:  "base noun phrases" – flat phrases that have a noun as their head. You can think of noun chunks as a noun plus the words describing the noun – for example, "the lavish green grass" or "the world’s largest tech fund".
* Named Entity:  "real-world object" that's assigned a name – for example, a person, a country, a product or a book title. spaCy can **recognise** [various types](https://spacy.io/api/annotation#named-entities) of named entities in a document, by asking the model for a **prediction**.

4) Extracting

Access

* Is done via the token object itself. Every word is a token, so it is pre-tagged accordingly with the processed information obtained by applying the learning and previous parsing algorithms.
* We can access the part-of-speech, position, the real-life object it depicts, and many more.

Use

* Further parsing of information, fetching and constructing new pieces of data filled with the extracted knowledge.