**Preprocessing of text data**

In any machine learning task, cleaning or preprocessing the data is as important as model building if not more. And when it comes to **unstructured** data like text, this process is even more important.

Some of the common text preprocessing / cleaning steps are:

* Lower casing
* Removal of Punctuations
* Removal of Stopwords
* Removal of Frequent words
* Removal of Rare words
* Stemming
* Lemmatization
* Removal of emojis
* Removal of emoticons
* Conversion of emoticons to words
* Conversion of emojis to words
* Removal of URLs
* Removal of HTML tags
* Chat words conversion
* Spelling correction

These are the different types of text preprocessing steps. But we do not need to do all of these all the times. We need to carefully choose the preprocessing steps based on our use case.

For example, in sentiment analysis, we do not remove the emojis or emoticons as it will convey some important information about the sentiment. Similarly we need to decide based on our use cases.

**Preprocessing with spacy:**

**python -m spacy download en\_core\_web\_sm**

First, you need to download the default model for the English language: en\_core\_web\_sm

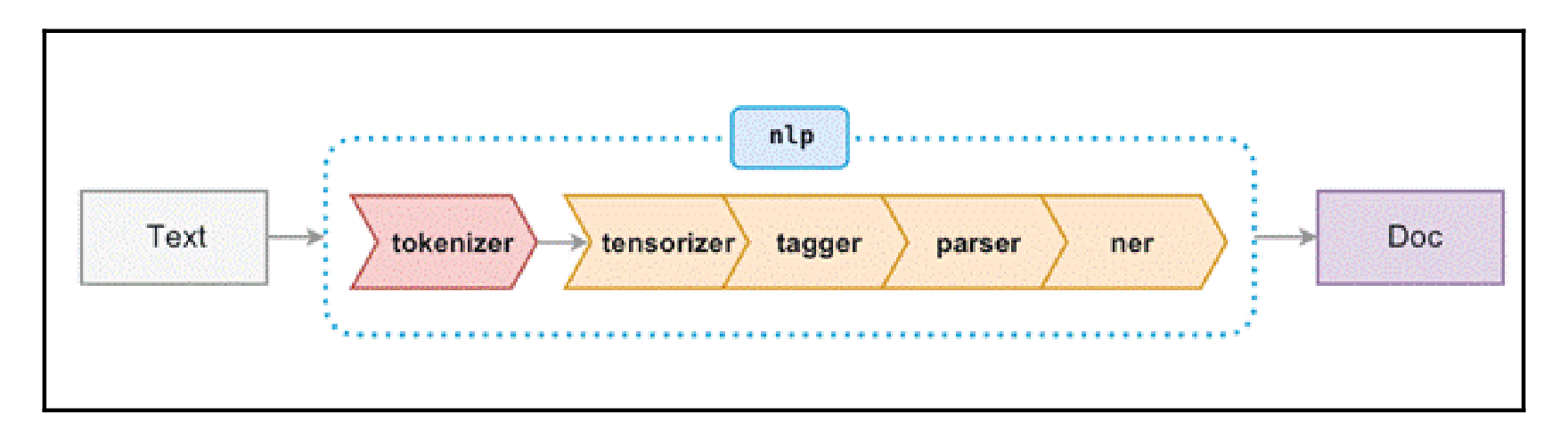
**import spacy**

**nlp = spacy.load("en\_core\_web\_sm")**

If these lines run without any errors, then it means that spaCy was installed and that the models and data were successfully downloaded.

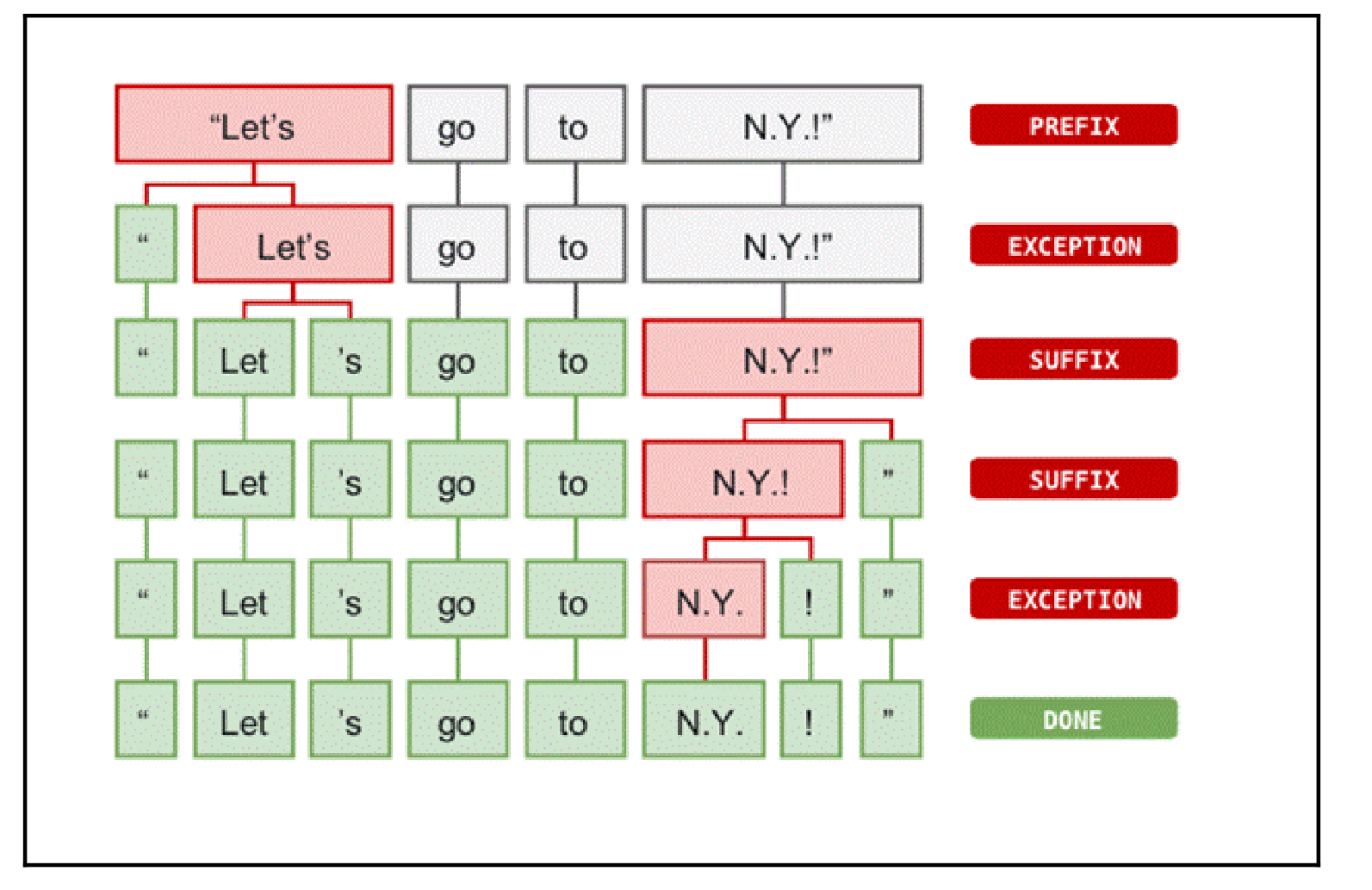
**doc = nlp(u'This is a sentence.')**

When you call nlp on Unicode text, spaCy first tokenizes the text to produce a Doc object. Doc is then processed in several different steps, what we also refer to as our pipeline.



**Tokenizing text**

Different languages will have different tokenization rules. Tokenization is the task of splitting a text into meaningful segments, called tokens. These segments could be words, punctuation, numbers, or other special characters that are the building blocks of a sentence.



**Part-of-speech (POS) – tagging:**

POS-tagging is marking each token of the sentence with its appropriate part of speech, such as noun, verb, and so on. spaCy uses a statistical model to perform its POS-tagging.

**doc = nlp(u'John and I went to the park'')**

**for token in doc:**

**print((token.text, token.pos\_))**

This will give us the following output:

(u'John', u'PROPN')

(u'and', u'CCONJ')

(u'I', u'PRON')

(u'went', u'VERB')

(u'to', u'ADP')

(u'the', u'DET')

(u'park', u'NOUN')

(u'.', u'PUNCT')

('John', 'PROPN')

('and', 'CCONJ')

('I', 'PRON')

('went', 'VERB')

('to', 'ADP')

('the', 'DET')

('park', 'NOUN')

('.', 'PUNCT')

**Named entity recognition**

Named entities are available as the ents property of a Doc:

**doc = nlp(u'Microsoft has offices all over Europe.')**

**for ent in doc.ents:**

**print(ent.text, ent.start\_char, ent.end\_char, ent.label\_)**

**(u'Microsoft', 0, 9, u'ORG')**

**(u'Europe', 31, 37, u'LOC')**

spaCy has the following built-in entity types:

PERSON : People, including fictional ones

NORP : Nationalities or religious or political groups

FACILITY : Buildings, airports, highways, bridges, and so on

ORG : Companies, agencies, institutions, and so on

GPE : Countries, cities, and states

LOC : Non GPE locations, mountain ranges, and bodies of water

PRODUCT : Objects, vehicles, foods, and so on (not services)

EVENT : Named hurricanes, battles, wars, sports events, and so on

WORK\_OF\_ART : Titles of books, songs, and so on

LAW : Named documents made into laws

LANGUAGE : Any named language

**Rule-based matching**

SpaCy's default pipeline also performs rule-based matching. This further annotates tokens with more information and is valuable during preprocessing.

ORTH : The exact verbatim text of a token

LOWER, UPPER : The lowercase and uppercase form of the token

IS\_ALPHA : Token text consists of alphanumeric chars

IS\_ASCII : Token text consists of ASCII characters

IS\_DIGIT : Token text consists of digits

IS\_LOWER , IS\_UPPER , IS\_TITLE : Token text is in lowercase, uppercase, and title

IS\_PUNCT , IS\_SPACE , IS\_STOP : Token is punctuation, whitespace, and a stop word

LIKE\_NUM , LIKE\_URL , LIKE\_EMAIL : Token text resembles a number, URL, and email

POS , TAG : The token's simple and extended POS tag

DEP , LEMMA , SHAPE : The token's dependency label, lemma, and shape

**Stopwords**

You can access spacy stopwords using this:

**spacy\_stopwords = spacy.lang.en.stop\_words.STOP\_WORDS**

We can also add our own stop words to the list of stop words. For example:

**my\_stop\_words = [u'say', u'be', u'said', u'says', u'saying', 'field']**

**for stopword in my\_stop\_words:**

**lexeme = nlp.vocab[stopword]**

**lexeme.is\_stop = True**

We can also add words using this:

**from spacy.lang.en.stop\_words import STOP\_WORDS**

**print(STOP\_WORDS) # <- Spacy's default stop words**

**STOP\_WORDS.add("your\_additional\_stop\_word\_here")**

**Lemmatization**

A lemma is the base form of a token. The lemma of walking, walks, walked is walk. Lemmatization is the process of reducing the words to their base form or lemmas.

In spaCy, the lemmatized form of a word is accessed with the **.lemma\_ attribute.**

**Stemming**

Stemming refers to reducing a word to its root form. The stem does not have to be a valid word at all. Stemming algorithms remove affixes (suffixes and prefixes). For example, the stem of “university ”is “univers”.

**Homework:**

Write a function that performs the following preprocessing steps on a chosen text entry:

* Lower casing
* Removal of Punctuations
* Removal of Stopwords
* Removal of Frequent words
* Removal of Rare words
* Lemmatization

The text you use must be composed of several sentences.