# Theory: Overview of SpaCy



17 minutes 0 / 5 problems solved



As you may know, NLTK is one of the most popular NLP libraries. It supports tokenization, normalization, and other text processing procedures. In this topic, we will introduce **SpaCy**, another NLP library. While NLTK is mostly used by students and scholars, SpaCy is designed for developers of language-related projects. SpaCy includes fewer algorithms, but they provide more precise results. We will discuss what differs SpaCy from NLTK and highlight the main features of it.

## §1. Installation

You need to install SpaCy first; use *pip* for this:

```
pip install spacy
```

You also need to import it afterward.

```
import spacy
```

SpaCy (as well as NLTK) contains a good of various *models* in different languages: English, Spanish, German, Chinese, French, etc. The whole list is available on the official SpaCy site.

In this topic, we will work with the `en_core_web_sm` model that was trained on blogs, news, and comments. It is used for POS-tagging, named entity recognition, and other tasks. You can install the model from the command line, as shown below:

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

You can use one of the following ways to upload it to your program:

1. Import the model using the integrated loader.

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

2. Import it as a Python module.

```
import en_core_web_sm


en_sm_model = en_core_web_sm.load()
```

The model import process is basic, you can do it for any other models.

> SpaCy requires a great amount of free space, and sometimes it also requires other additional programs. You may encounter several problems with the installation. Unfortunately, we can't cover every one of them, but you can find answers on special forums.

## §2. NLTK vs SpaCy

Before we start working with SpaCy, it would be good to compare the basic features of these libraries.

| Criteria | NLTK | SpaCy |
|---|---|---|
| Multi-Language Support | Yes | Yes |
| Types of Inputs and Outputs | Strings | Objects |
| Word Vector Support | No | Yes |
| Performance | Slow | Fast |

Current topic:

Overview of SpaCy

Topic depends on:

- ✕ Intro to text representations
- ✕ Text normalization

Topic is required for:

- Introduction to Stanza

Table of contents:

| | | |
|---|---|---|
| Target Audience | Researchers | Developers |

To sum up, SpaCy is a modern alternative for traditional NLTK.

# §3. Getting Started

Let's overview the basic NLP procedures and how they are implemented. It is essential to remember that SpaCy turns processed text not into strings, but objects.

### 1. Tokenization

SpaCy automatically divides your document into tokens when you use a for-loop.

```
doc = en_sm_model("Microsoft News delivers news from the most popular and trusted publishers.")
for i in doc:
    print(i.text)

# Microsoft
# News
# delivers
# news
# from

# the

# most

# popular

# and

# trusted

# publishers

# .
```

First, we create a doc-object using the already installed model. Then, with the help of a for-loop, we print all the token objects of the sentence.

### 2. POS-tagging and Lemmatization

You can establish the lemma for each token as well as its part of speech. Use the `token.lemma_` method for lemmas and the `token.pos_` method for parts of speech.

```
doc = en_sm_model("Microsoft News delivers news from the most popular and trusted publishers.")
for i in doc:
    print("{0} - {1} - {2}".format(i.text, i.lemma_, i.pos_))

# Microsoft - Microsoft - PROPN
# News - News - PROPN
# delivers - deliver - VERB
# news - news - NOUN
# from - from - ADP

# the - the - DET

# most - most - ADV

# popular - popular - ADJ

# and - and - CCONJ

# trusted - trusted - ADJ

# publishers - publisher - NOUN

# . - . - PUNCT
```

As you can see, we can easily get the necessary information about the tokens. You can find more token attributes in the Linguistic Features sections of the official SpaCy documentation.

## §4. Other NLP Procedures

SpaCy also provides more ways to process texts for further analysis. Their basic overview is presented below.

### 1. Stopword Removal

Some tasks may require stopword removal. To implement this feature in SpaCy, you can import the built-in stopwords.

```
from spacy.lang.en.stop_words import STOP_WORDS
```

Let's have a look at them. It will produce a set:

```
print(STOP_WORDS)

# {'afterwards', 'would', 'others', 'thence', 'itself', 'besides', 'five' ...}
```

So, the example given in the previous section can be processed in the following way:

```
doc = en_sm_model("Microsoft News delivers news from the most popular and trusted publishers.")
for i in doc:
    if i.lemma_ not in STOP_WORDS:
        print(i.lemma_)

# Microsoft
# News
# deliver
# news

# popular

# trusted

# publisher

# .
```

Be careful with lemmatization. If you don't lemmatize your tokens in advance, some stopwords won't be recognized.

All articles, conjunctions, and some frequent adverbs are omitted. Mind the dot; the punctuation marks **are not** included in the stop words.

### 2. Named Entity Recognition

A **named entity** is an object of the real world, for example, a person, an organization, etc. SpaCy can recognize different named entities in a document by forecasting it with the help of a built-in model. The standard way of named entity recognition (NER) is `doc.ents`.

```
doc = en_sm_model("Microsoft News delivers news from the most popular and trusted publishers.")
for i in doc.ents:
    print(i.text)

# Microsoft News
```

The output is a line with the company name.

### 3. Dependency Tree

You can also see a syntactic structure of sentences. A dependency tree reflects syntactic relations between words in a sentence. In the example below, we will print a head element and a child (dependent) element of each phrase.

```
doc = en_sm_model("Microsoft News delivers news from the most popular and trusted publishers.")
for i in doc:
  print("{0} --> {1}".format(i.head.text, i.text))

# News --> Microsoft
# delivers --> News
# delivers --> delivers
# delivers --> news
# news --> from

# publishers --> the

# popular --> most

# publishers --> popular

# popular --> and

# popular --> trusted

# from --> publishers

# delivers --> .
```

The first element of the output is the head element, the second one is a child. Mind the verb that is the head of the sentence. It doesn't depend on anything. In the tree representation, the head element is dependent on itself, so the line `delivers --> delivers` is correct.

You can also visualize your syntactic structure using the `displacy` library. More information on this library is available in the displaCy section of the official SpaCy documentation. The result is a picture of visualized relations between words.

In the picture, you can also find different types of dependencies. For example, `amod` stands for an **adjectival modifier**. You can learn more about types of dependencies in the Annotations Specifications of the official documentation.

## §5. Word Vectors

Word similarity can be determined by comparing **word vectors** that are also known as word embeddings. Some models in SpaCy provide built-in vectors. Unfortunately, `en_core_web_sm` cannot deal with vectors. In the example below, we use `en_core_web_lg`, which is the biggest model for the English language.

```
en_lg_model = en_core_web_lg.load()
doc = en_lg_model("Cats don't like dogs.")
for token1 in doc:
    for token2 in doc:
        print(token1.text, token2.text, token1.similarity(token2))

# Cats Cats 1.0
# Cats do 0.34179398
# Cats n't 0.3543607

# Cats like 0.3767576

# Cats dogs 0.83117634

# ...
```

The words represented in the sentence are common in English, so we use them to demonstrate how the built-in vectors work in SpaCy. In our case, the model's predictions are to the point. A "*cat*" is very similar to "*dog*", but "*cat*" has a lower percentage of similarity with the word "*like*". The same words are 100% similar to each other.

# §6. Conclusion

In this topic, we have covered the main aspects of working with SpaCy. So far, we have learned:

- how to install SpaCy and download modules;
- the common and distinct features of NLTK and SpaCy;
- how to implement the basic NLP procedures: tokenization, lemmatization, and POS-tagging;
- how to use the built-in stopword set for text processing;
- how to implement NER;
- how to create a syntactic tree of a sentence;
- how to work with word vectors.

Of course, SpaCy has more possibilities, the official documentation can help you with them.

Report a typo

**9** users liked this theory. **0** didn't like it. **What about you?**

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion