SPACY INSTALLATION AND BASIC OPERATIONS | NLP TEXT PROCESSING LIBRARY

What is spaCy

spaCy is an open-source Python library that parses and "understands" large volumes of text.

- spaCy is the best way to prepare text for deep learning.
- It interoperates seamlessly with Tensorflow, PyTorch, Scikit-learn, Gensim and the rest of Python's awesome AI ecosystem.
- With spaCy, you can easily construct linguistically sophisticated statistical models for a variety of NLP problems.
- spaCy excels at large-scale information extraction tasks.
- It's written from the ground up in carefully memory-managed Cython.
- Independent research in 2015 found spaCy to be the fastest in the world.
- If your application needs to process entire web dumps, spaCy is the library you want to be using.
- spaCy is designed to help you do real work — to build real products, or gather real insights.
- The library respects your time, and tries to avoid wasting it.
- It's easy to install, and its API is simple and productive.

Reference and for more details, refer (https://spacy.io/)

Installation and Setup¶

Create the virtual environment first

```
conda create -n spacyenv python=3
```

See the list of env available including the newly created spacyenv

```
conda info -e
```

Activate the env

```
conda activate spacyenv
```

Jupyter Notebook makes sure that the IPython kernel is available, but you have to manually add a kernel with a different version of Python or a virtual environment. First, you need to activate your virtual environment. Next, install ipykernel which provides the IPython kernel for Jupyter:

pip install --user ipykernel

Now, add your virtual environment to Jupyter
python -m ipykernel install --user --name=spacyenv

Now your env is created and added to jupyter notebook. Now to Install Spacy we have two options

1. From the command line or terminal:
conda install -c conda-forge spacy
or
pip install -U spacy

2. From Jupyter Notebook

Start the Jupyter Notebook from command line by typing
jupyter notebook

Now Install Spacy by typing and executing
!pip install -U spacy

Now download the language specific model. I will download the model for English language as below.

(You must run this as admin or use sudo)

From command line
python -m spacy download en

From Jupyter notebook
!python -m spacy download en

If successful, you should see a message like:
Linking successful
C:\Anaconda3\envs\spacyenv\lib\site-packages\en_core_web_sm -->
C:\Anaconda3\envs\spacyenv\lib\site-packages\spacy\data\en

you can now load the model via spacy.load('en')

Some basic operations to check if everything installed correctly

- This is a typical set of instructions for importing and working with spaCy.
- Don't worry, if it takes time - spaCy has large library to load:

In [2]:
```
# Import spaCy and load the language library
import spacy
nlp = spacy.load('en_core_web_sm')
```

In [3]:
```
# Create a Doc object
doc = nlp(u'Corona will go very soon. Do not get panic. Cases in U.S. have reduced in last 48 hours')

# Print each token separately
for token in doc:
    print(token.text, token.pos_, token.dep_)
```

```
Corona PROPN nsubj
will VERB aux
go VERB ROOT
very ADV advmod
soon ADV advmod
. PUNCT punct
Do VERB aux
not ADV neg
get VERB ROOT
panic NOUN dobj
. PUNCT punct
Cases NOUN nsubj
in ADP prep
U.S. PROPN pobj
have VERB aux
reduced VERB ROOT
in ADP prep
last ADJ amod
48 NUM nummod
hours NOUN pobj
```

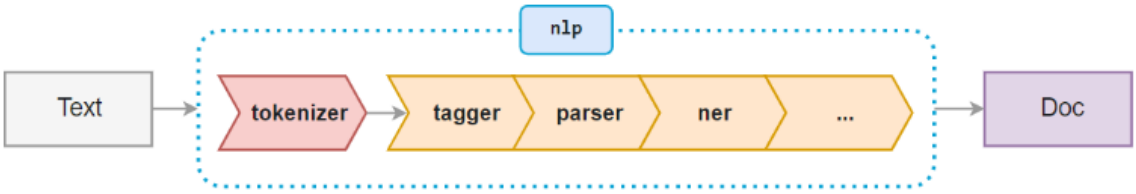
We can apply some formating on how the output is printed. However for now observe few things as below:

1. Corona is recognized as a Proper Noun, not just a word at the start of a sentence
2. U.S. is kept together as one entity (It is called 'token')

Later we will see what each of these abbreviations mean and how they're derived.

Pipeline

When we run nlp, our text enters a *processing pipeline* that first breaks down the text and then performs a series of operations to tag, parse and describe the data. Image source: https://spacy.io/usage/spacy-101#pipelines



We can see what operations are inside the pipeline using the code below

In [9]:
```
nlp.pipeline
```

Out[9]:
```
[('tagger', <spacy.pipeline.Tagger at 0x13c507d9ec8>),
 ('parser', <spacy.pipeline.DependencyParser at 0x13c507da588>),
 ('ner', <spacy.pipeline.EntityRecognizer at 0x13c507dab28>)]
```

In [10]:
```
nlp.pipe_names
```

Out[10]:
```
['tagger', 'parser', 'ner']
```

Tokenization

- The first step in processing text is to split up all the component parts (words & punctuation) into "tokens".
- These tokens are annotated inside the Doc object to contain descriptive information.

In [12]:
```
doc2 = nlp("Apple is looking at buying U.K. startup for $1 billion")
for token in doc2:
    print(token.text, token.pos_, token.dep_)
```

```
Apple PROPN nsubj
is VERB aux
looking VERB ROOT
at ADP prep
buying VERB pcomp
U.K. PROPN compound
startup NOUN dobj
for ADP prep
$ SYM quantmod
1 NUM compound
billion NUM pobj
```

In [15]:
```
doc3 = nlp("Apple isn't looking  at buying U.K. startup.")
for token in doc3:
    print(token.text, token.pos_, token.dep_)
```

```
Apple PROPN nsubj
is VERB aux
n't ADV neg
looking VERB ROOT
  SPACE
at ADP prep
buying VERB pcomp
U.K. PROPN compound
startup NOUN dobj
. PUNCT punct
```

- Notice how `isn't` has been split into two tokens.
- spaCy recognizes both the root verb `is` and the negation attached to it.
- Notice also that both the extended whitespace and the period at the end of the sentence are assigned their own tokens.

Doc3 also contains the original text. You can see it by executing the `doc3` object.

```
In [16]: doc3
Out[16]: Apple isn't looking  at buying U.K. startup.

In [17]: print(doc3[0])
         Apple

In [18]: doc3[1]
Out[18]: is
```

Part-of-Speech Tagging (POS)

- The next step after splitting the text up into tokens is to assign parts of speech.
- In the above example, `Apple` was recognized to be a ***proper noun***. Here some statistical modeling is required.
- For example, words that follow "the" are typically nouns.

For a full list of POS Tags visit https://spacy.io/api/annotation#pos-tagging

```
In [29]: doc4 = nlp(u"Apple isn't looking at buying U.K. startup.")

         for token in doc4:
             print(token.text, token.pos_)
```

```
Apple PROPN
is VERB
n't ADV
looking VERB
at ADP
buying VERB
U.K. PROPN
startup NOUN
. PUNCT
```

To see the full name of a tag use `spacy.explain(tag)`

```
In [22]: spacy.explain('PROPN')
Out[22]: 'proper noun'

In [23]: spacy.explain('nsubj')
Out[23]: 'nominal subject'
```

Other Important information which Spacy assign to tokens.

Will cover all these in detail in next articles under NLP Spacy Series.

Tag	Description	doc4[0].tag
.text	The original word text	Apple
.lemma_	The base form of the word	Apple
.pos_	The simple part-of-speech tag	PROPN/proper noun
.tag_	The detailed part-of-speech tag	NNP/noun, proper singular
.shape_	The word shape – capitalization, punctuation, digits	Xxxxx
.is_alpha	Is the token an alpha character?	True
.is_stop	Is the token part of a stop list, i.e. the most common words of the language?	False

Span (Slicing)

```
In [36]: doc5 = nlp(u"Apple isn't looking at buying U.K. startup.")
         sliced_text = doc5[4:7]
         sliced_text
Out[36]: at buying U.K.

In [37]: type(sliced_text)
Out[37]: spacy.tokens.span.Span
```

Sentences

Print sentence tokens instead of word tokens

```
In [35]: doc6 = nlp(u'This is the first sentence. This is another sentence. This is the last sentence.')
         for sent in doc6.sents:
             print(sent)

         This is the first sentence.
         This is another sentence.
         This is the last sentence.

In [38]: doc6[6].is_sent_start
Out[38]: True

In [42]: doc6[11].is_sent_start
Out[42]: True

In [44]: doc6[2].is_sent_start
```

Note:

This is the first article from the "**NLP SPACY Series**". Here I have covered only basic operations.

Next Article I will cover Text Preprocessing steps in detail.

Thank You

If you like my Posts on Machine Learning, Please connect with me on

LinkedIn: https://www.linkedin.com/in/ashutoshtripathi1/

Instagram: https://www.instagram.com/ashutosh_ai/