Two minutes NLP — SpaCy sheet

POS tagging, dependency parsing, NER, and senter

Fabio Chiusano · Follow
Published in NLPlanet · 3 min read · Jan 26, 2022

Photo by Shelbey Fordyce on Unsplash

SpaCy is a free, open-source library for advanced Natural Language Processing in Python. It is designed specifically for production use and helps build applications that process and understand large volumes of text. It can be used for a multitude of use cases, such as information extraction, natural language understanding systems or to pre-process text for deep learning.

List of spaCy tasks

Here's a list of NLP tasks that spaCy can perform.

NAME	DESCRIPTION
Tokenization	Segmenting text into words, punctuations marks etc.
Part-of-speech (POS) **Tagging**	Assigning word types to tokens, like verb or noun.
Dependency Parsing	Assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object.
Lemmatization	Assigning the base forms of words. For example, the lemma of “was” is “be”, and the lemma of “rats” is “rat”.
Sentence Boundary Detection (SBD)	Finding and segmenting individual sentences.
Named Entity Recognition (NER)	Labelling named “real-world” objects, like persons, companies or locations.
Entity Linking (EL)	Disambiguating textual entities to unique identifiers in a knowledge base.
Similarity	Comparing words, text spans and documents and how similar they are to each other.
Text Classification	Assigning categories or labels to a whole document, or parts of a document.
Rule-based Matching	Finding sequences of tokens based on their texts and linguistic annotations, similar to regular expressions.
Training	Updating and improving a statistical model’s predictions.
Serialization	Saving objects to files or byte strings.

List of tasks that spaCy can perform. Image from https://spacy.io/usage/spacy-101.

While some of spaCy’s features work independently, others require trained pipelines to be loaded. SpaCy currently offers trained pipelines for a variety of languages, which can be installed as individual Python modules. Here’s an example where we download the trained pipeline *en_core_web_sm*.

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

spacy_1.sh hosted with ❤ by GitHub view raw

The trained pipeline you choose always depends on your use case and the texts you’re working with. For a general-purpose use case, the small, default pipelines (i.e. the ones ending in `sm`) are always a good start.

Tokenization

Tokenization consists in segmenting text into words, punctuations marks, etc. This is done by applying rules specific to each language.

```python
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("The cat is on the table")
for token in doc:
    print(token.text)

# The
# cat
# is
# on
# the
# table
```

spacy_2.py hosted with ❤ by GitHub view raw

POS Tagging

POS (Part of Speech) Tagging refers to categorizing words in a text in correspondence with a particular part of speech, depending on the definition of the word and its context.

```python
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("The cat is on the table")
for token in doc:
    print(f"{token.text} --- POS: {token.pos_}, {token.tag_}")

# The --- POS: DET, DT
# cat --- POS: NOUN, NN
# is --- POS: AUX, VBZ
# on --- POS: ADP, IN
# the --- POS: DET, DT
# table --- POS: NOUN, NN
```

spacy_3.py hosted with ❤ by GitHub view raw

The `pos_` attribute contains the simple UPOS part-of-speech tag, whereas the `tag_` attribute contains the detailed POS tag.

Dependency Parsing

Dependency Parsing consists in assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object.

```python
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("The cat is on the table")
for token in doc:
    print(f"{token.text} --- dependency label: {token.dep_}")

# The --- dependency label: det
# cat --- dependency label: nsubj
# is --- dependency label: ROOT
# on --- dependency label: prep
# the --- dependency label: det
# table --- dependency label: pobj
```

spacy_4.py hosted with ❤ by GitHub view raw

Stopwords

Stopwords are the most common words of a language, which are often ignored in NLP tasks as they usually carry little meaning to the sentences.

```python
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("The cat is on the table")
for token in doc:
    print(f"{token.text} --- is stopword: {token.is_stop}")

# The --- is stopword: True
# cat --- is stopword: False
# is --- is stopword: True
# on --- is stopword: True
# the --- is stopword: True
# table --- is stopword: False
```

spacy_5.py hosted with ❤ by GitHub view raw

Lemmatization

Lemmatization assigns the base forms of words. For example, the lemma of “was” is “be”, and the lemma of “dogs” is “dog”.

```python
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("The cat is on the table")
for token in doc:
    print(f"{token.text} --- lemma: {token.lemma_}")

# The --- lemma: the
# cat --- lemma: cat
# is --- lemma: be
# on --- lemma: on
# the --- lemma: the
# table --- lemma: table
```

spacy_6.py hosted with ❤ by GitHub view raw

Named Entity Recognition (NER)

Named Entity Recognition refers to labeling named “real-world” objects in texts, like persons, companies, or locations.

```python
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Elon Musk cofounded the electronic-payment firm PayPal and formed SpaceX.")

for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)

# Elon Musk 0 9 PERSON
# PayPal 48 54 ORG
```

spacy_7.py hosted with ❤ by GitHub view raw

Word embeddings

A word embedding is a learned representation (usually a vector of numbers) for text where words that have the same meaning have a similar

representation.

To make them compact and fast, spaCy's small pipeline packages (all packages that end in `sm`) don't ship with word vectors and only include context-sensitive tensors. This means you can still use the `similarity()` methods to compare sentences and words, but the result won't be as good, and individual tokens won't have any vectors assigned. So in order to use real word vectors, you need to download a larger pipeline package.

```
python -m spacy download en_core_web_md
```

spacy_8.sh hosted with ❤ by GitHub view raw

This is how you get word embeddings with spaCy.

```
import spacy

nlp = spacy.load("en_core_web_md")
tokens = nlp("cat is on the aofafgag")

vectors = []
for token in tokens:
    print(token.text, token.has_vector, token.is_oov)
    vectors.append(token.vector)

# The True False
# cat True False
# is True False
# on True False
# the True False
# aofafgag False True

print(vectors[0])

# [ 2.7204e-01 -6.2030e-02 -1.8840e-01  2.3225e-02 -1.8158e-02  6.7192e-03
# -1.3877e-01  1.7708e-01  1.7709e-01  2.5882e+00 -3.5179e-01 -1.7312e-01
#  4.3285e-01 -1.0708e-01  1.5006e-01 -1.9982e-01 -1.9093e-01  1.1871e+00
#  ...
```

spacy_9.py hosted with ❤ by GitHub view raw

Sentence similarity

With spaCy, you can compute similarities between sentences. This is done by averaging the word embeddings of the words in each sentence and then computing similarity with a similarity measure.

```python
import spacy

nlp = spacy.load("en_core_web_md")  # make sure to use larger package!
doc1 = nlp("I like salty fries and hamburgers.")
doc2 = nlp("Fast food tastes very good.")
doc3 = nlp("Where is the cat.")

# Similarity of doc1 and doc2
print(doc1.similarity(doc2))

# 0.7799485853415737

# Similarity of doc1 and doc3
print(doc1.similarity(doc3))

# 0.6210606690259671
```

spacy_10.py hosted with ❤️ by GitHub view raw

Thank you for reading! If you are interested in learning more about NLP, remember to follow NLPlanet on Medium, LinkedIn, and Twitter!

NLPlanet related posts

Awesome NLP — 21 popular NLP libraries of 2022

The landscape of NLP libraries

medium.com

Two minutes NLP — A Taxonomy of Tokenization Methods

Word-level, Character-level, BPE, WordPiece, and SentencePiece

medium.com

Two minutes NLP — Sentence Transformers cheat sheet

Sentence Embeddings, Text Similarity, Semantic Search, and Image Search

medium.com

NLP Artificial Intelligence Data Science Machine Learning Spacy

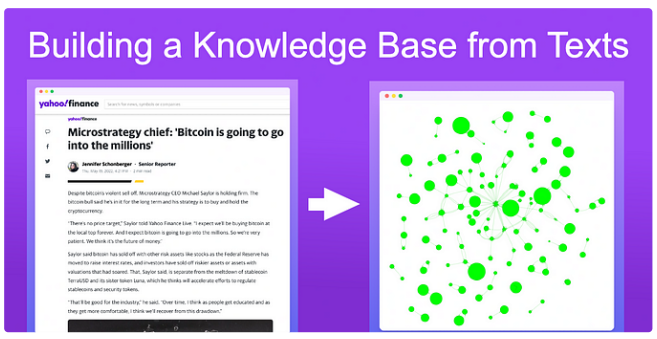
Written by Fabio Chiusano

5.2K Followers · Editor for NLPlanet

Follow

Freelance data scientist — Top Medium writer in Artificial Intelligence

More from Fabio Chiusano and NLPlanet



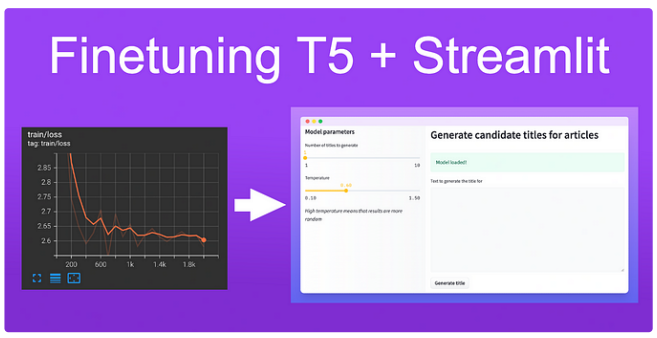
Fabio Chiusano in NLPlanet

Building a Knowledge Base from Texts: a Full Practical Example

Implementing a pipeline for extracting a Knowledge Base from texts or online articles

12 min read · May 25, 2022

1.1K 10



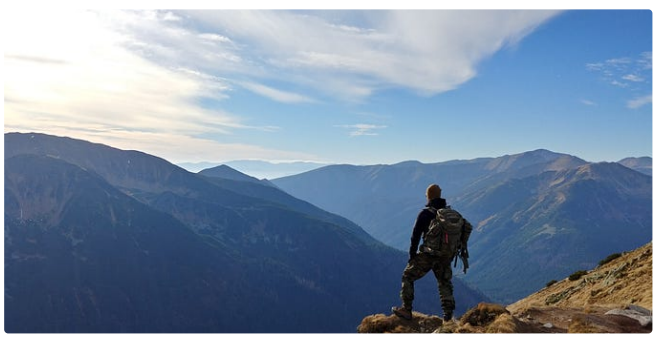
Fabio Chiusano in NLPlanet

A Full Guide to Finetuning T5 for Text2Text and Building a Demo...

All you need to know to build a full demo: Hugging Face Hub, Tensorboard, Streamlit,...

15 min read · May 17, 2022

440 10



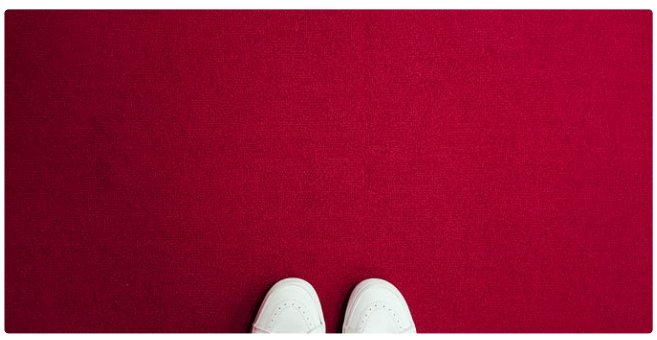
Fabio Chiusano in NLPlanet

Two minutes NLP—Perplexity explained with simple probabilities

Language models, sentence probabilities, entropy

5 min read · Jan 27, 2022

190 3



Fabio Chiusano in NLPlanet

Two minutes NLP—Learn the ROUGE metric by examples

ROUGE-N, ROUGE-L, ROUGE-S, pros and cons, and ROUGE vs BLEU

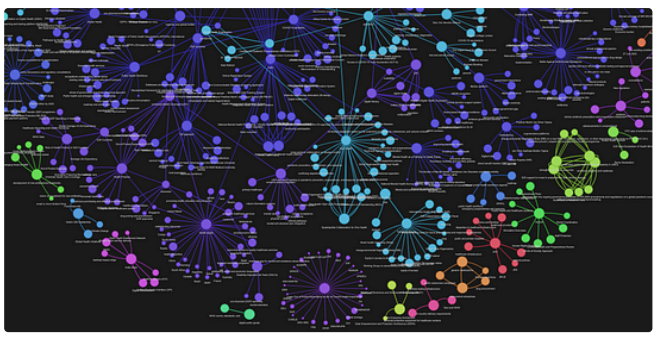
5 min read · Jan 19, 2022

132 2

See all from Fabio Chiusano

See all from NLPlanet

Recommended from Medium



Rahul Nayak in Towards Data Science

How to Convert Any Text Into a Graph of Concepts

A method to convert any text corpus into a Knowledge Graph using Mistral 7B.

12 min read · Nov 10

4.4K 42

Youssef Hosni in Towards AI

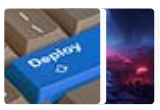
Andrej Karpathy LLM Paper Reading List for LLM Mastery

LLM Paper Recommendation from World Leading AI Researcher

✦ · 10 min read · Dec 7

443 1

Lists



Predictive Modeling w/ Python

20 stories · 690 saves

Natural Language Processing

986 stories · 475 saves

Practical Guides to Machine Learning

10 stories · 785 saves

data science and AI

38 stories · 4 saves

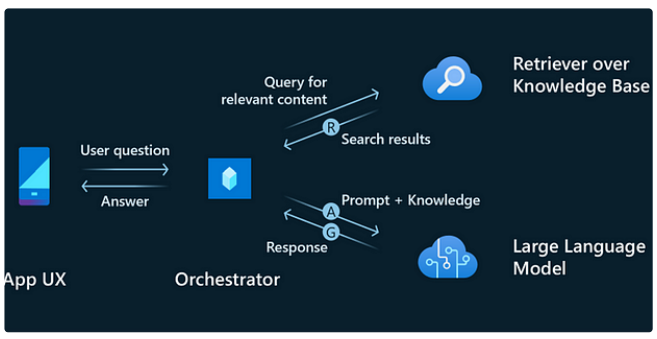
Gursev Pirge in John Snow Labs

Text Preprocessing: Splitting texts into sentences with Spark NLP

Using Sentence Detection in Spark NLP for Text Preprocessing

8 min read · Jul 21

2



James Nguyen

Forget RAG: Embrace agent design for a more intelligent grounded...

The Retrieval Augmented Generation (RAG) design pattern has been commonly used to...

6 min read · Nov 18

314 4

Chetankumar Khadke

Information extraction with Zephyr 7B

Information extraction with LLM uses advanced language models like Zephyr to...

11 min read · Nov 27

122

Dilip Kashyap

Google’s new AI Model Gemini now available in Bard, here is how to use

Google recently introduced Gemini AI, its latest large language model (LLM), built to...

3 min read · Dec 7

598 13

See more recommendations