towardsdatascience.com

Training Custom NER - Towards Data Science

Nishanth N

3–4 minutes

This article explains, how to train and get the custom-named entity from your training data using spacy and python.

tds

> *The article explains what is spacy, advantages of spacy, and how to get the named entity recognition using spacy. Now, all is to train your training data to identify the custom entity from the text.*

What is spaCy?

SpaCy is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython. The library is published under the MIT license and its main developers are Matthew Honnibal and Ines Montani, the founders of the software company Explosion.

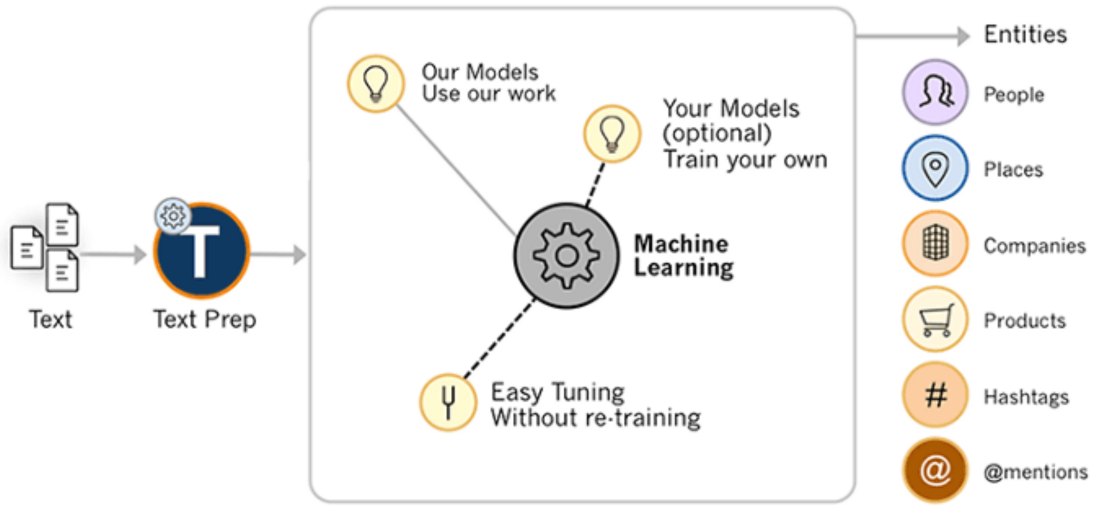
> *Unlike NLTK, which is widely used for teaching and research, spaCy focuses on providing software for production usage. As*

> *of version 1.0, spaCy also supports deep learning workflows that allow connecting statistical models trained by popular machine learning libraries like Tensor Flow, PyTorch, or MXNet through its machine learning library Thinc.*

Why not NLTK?

While NLTK provides access to many algorithms to get something done, spaCy provides the best way to do it. It provides the fastest and most accurate syntactic analysis of any NLP library released to date. It also offers access to larger word vectors that are easier to customize.

Custom -NER



An example of NER in action

Step: 1 Installation instructions

pip

After installation, you need to download a language model

conda

Step: 2 Model Training

You can start the training once you completed the first step.

→ Initially, import the necessary packages required for the custom creation process.

→ Now, the major part is to create your custom entity data for the input text where the *named entity* is to be identified by the model during the testing period.

→ Define the variables required for the training model to be processed.

→ Next, load a blank model for the process to carry out the NER action and set up the pipeline with only NER using *create_pipe* function.

→ Here, we want to train the recognizer by disabling the unnecessary pipeline except for NER. The *nlp_update* function can be used to train the recognizer.

Output: Training losses

```
100%|██████████| 3/3 [00:00<00:00, 32.00it/s]{'ner':
10.165919601917267}100%|██████████| 3/3
[00:00<00:00, 30.38it/s]{'ner':
8.44960543513298}100%|██████████| 3/3
[00:00<00:00, 28.11it/s]{'ner':
7.798196479678154}100%|██████████| 3/3
[00:00<00:00, 33.42it/s]{'ner':
6.569828731939197}100%|██████████| 3/3
[00:00<00:00, 29.20it/s]{'ner': 6.784278305480257}
```

→ To test the trained model,

→ Finally, save the model to your path which stored in the *output_dir* variable.

Once you saved the trained model you can load the model using

The full source code available on GitHub.

Conclusion

I hope you have now understood how to train your own NER model on top of the spaCy NER model. Thanks for reading!