Published in Quantrium.ai

Maria Philna Aruja Follow
Feb 12 · 7 min read · Listen

QUANTRIUM GUIDES

Top 3 Packages for Named Entity Recognition

Comparing SpaCy, NLTK and Flair — the top 3 NER models

By the time you finish reading this article, the amount of text on the internet will grow by over 2 million tweets, 15 papers in scientific journals, 4 Wikipedia articles, and countless news stories and blog posts.

To argue for the utility of such a seemingly infinite repository of text, the proponents of big-data will be quick to quip that *"data is the new oil"*, but even the richest of oil reserves require an array of tools that enable the extraction and purification of the oil to be of any use.

Named entity recognition is one such tool in the arsenal of natural language processing that allows us to tame large and unstructured text corpora. In this article, we elaborate on the inner workings of NER and discuss three widely used packages — NLTK, SpaCy, and Flair.

Named Entity Recognition

Named Entity Recognition is a two-step process that helps in extracting useful insights from data. It involves identifying keywords, i.e. the named entities, and then categorising them into broader classes.

For instance, in the sentence "The world's biggest chocolate cake will be made for this Christmas by the Royal cake house.", Chocolate cake, Christmas, and Royal cake house are the named entities in the sentence. Classifying them as say, "Food", "Occasion", "Organisation" is the next step in NER.

The Process

The process involves two stages.

1. **Extraction of entities** —In this step, the NER model scans the data and finds the words that can be treated as an entity. IOB tagging, Inside Outside Beginning tagging, can be used to mark the entities in the data. An NER model will be able to find the named entity in the model based on the named entities known to it.
2. **Classification of entities**—This is the stage when the entities are categorised into predefined classes, the named entity "chocolate cake" would be categorised as food for instance. The effectiveness of the classification of the entities will depend upon the relevance of

Spacy is a powerful Natural Language processing tool used to process large amounts of data. With support for over 64 languages and 63 trained pipelines for 19 languages, it is a handy tool in NLP.

It uses Bloom embedding and residual CNN's to identify the named entities. Here is an example of NER performed using SpaCy.

```
import spacy
import en_core_web_sm
nlp = en_core_web_sm.load()
article = nlp("Google LLC is a multinational technology company. \
               It was founded in September 1998 by Larry Page and Sergey Brin \
               while they were Ph.D. students at Stanford University in California")

for X in article.ents:
    print("Text:",X.text, "\tLabel:", X.label_)
```

Output

```
Text: Google LLC        Label: ORG
Text: September 1998    Label: DATE
Text: Larry Page        Label: PERSON
Text: Sergey Brin       Label: PERSON
Text: Ph.D.     Label: WORK_OF_ART
Text: Stanford University       Label: ORG
Text: California        Label: GPE
```

2. NLTK

Natural language Toolkit is a set of libraries used for NLP. It is widely used in research and for educational purposes. Written in Python, it has access to more than 50 text corpora across 7 languages.

One of the primary principles of NLTK, besides simplicity, consistency, and extensibility, is modularity. Modularity provides components that can be used independently. This might be especially useful for tuning only specific parts of the pipeline or even using third parties in conjunction with this toolkit.

NLTK does NER in two steps. The first step is POS (parts-of-speech) tagging or grammatical tagging, which is followed by chunking to extract the named entities. An example of NER performed in NLTK is given below.

```
import nltk
from nltk import word_tokenize,pos_tag
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
text = "Google LLC is a multinational technology company. \
        It was founded in September 1998 by Larry Page and Sergey Brin \
        while they were Ph.D. students at Stanford University in California"
for sent in nltk.sent_tokenize(text):
 for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sent))):
    if hasattr(chunk, 'label'):
       print(chunk.label(), ' '.join(c[0] for c in chunk))
```

Output

```
PERSON Sergey Brin
ORGANIZATION Stanford University
GPE California
```

3. Flair

Flair is a simple framework developed for NLP. Flair is built on top of PyTorch which is a powerful deep learning framework. Claimed to support over 250 languages, it is very useful in training small models.

It is pre-trained on an extremely large unlabelled text corpora. An example of NER performed in Flair is as given below.

```
from flair.data import Sentence
from flair.models import SequenceTagger
from segtok.segmenter import split_single
tagger = SequenceTagger.load('ner-ontonotes')
text = "Google LLC is a multinational technology company. \
        It was founded in September 1998 by Larry Page and Sergey Brin \
        while they were Ph.D. students at Stanford University in California"
sentence = [Sentence(sent, use_tokenizer=True) for sent in split_single(text)]
tagger.predict(sentence)

for sent in sentence:
 for entity in sent.get_spans('ner'):
     print(entity)
```

Output

```
Span [1,2]: "Google LLC"   [− Labels: ORG (0.9918)]
Span [5,6]: "September 1998"   [− Labels: DATE (0.9132)]
Span [8,9]: "Larry Page"   [− Labels: PERSON (0.9997)]
Span [11,12]: "Sergey Brin"   [− Labels: PERSON (0.9997)]
Span [16]: "Ph.D."   [− Labels: WORK_OF_ART (1.0)]
Span [19,20]: "Stanford University"   [− Labels: ORG (0.9826)]
Span [22]: "California"   [− Labels: GPE (0.9998)]
```

Factors of Consideration

The best package for an application can be selected based on several important factors — the availability of pre-trained model tags, the format of training data, ease of use, speed of execution, evaluation metrics, and visualisation capabilities of the package.

Pre-trained model tags

Pre-trained model tags indicate the capability and depth of a package. The more the pre-trained model tags, the more insightful the results will be. The available pre-trained NLTK model is only limited to 3 main tags which are PERSON, ORGANISATION, and GPE, while the available SpaCy and Flair models can identify up to 18 tags using the pre-trained model.

Format of training data

The three packages have different formats in which they accept the training data.

NLTK

The NLTK model for custom-named entity recognition can be developed with the help of the Stanford NER tagger, written in Java. The training data for NLTK looks like this:

```
in O
Google ORG
. O
```

Each document in the training data should be separated by a newline.

SpaCy

The training data for SpaCy version 2 is a list of tuples. Each tuple contains a text, start index, end index, and a label. For version 3, refer here.

For example

```
[
(“Quantrium is a company”,{“entities”:[(0,8,”ORG”)]}),
(“Ram is a good person”,{“entities”:[(0,2,”PER”)]})
]
```

Flair

To train an NER model using Flair, the training data should be of the following format. Each sentence in the training data should be separated using newlines. Each row has two columns. The first column is the word and the second column is the BIO-annotated NER tag.

For example

```
George B-PER
Washington I-PER
went O
to O
Washington B-LOC

Sam B-PER
Houston I-PER
stayed O
home O
```

The performance of trained models is better for SpaCy and Flair when compared to that of NLTK.

Ease of use

If one package allows you to do NER in one step, you may need to execute multiple steps to perform NER in other packages. In NLTK, POS tagging needs to be done as a first step. It is then followed by chunking. This will identify the entities in the document. In SpaCy the “ents” property of the doc object identifies the named entities whereas, in Flair, the predict function is used for the same.

Speed

Speed is very important when you’re dealing with large datasets but there is a trade-off between speed and accuracy. The speed of a package is measured using WPS, words per second. Of the three packages, NLTK is found to be the slowest while Spacy beats Flair. So if the processing involves a very large data set, then Spacy is the best option.

Visualisation

Visualising the entities is an effective way of understanding the data for which the three packages have their own methods. NER entities

Evaluation metric

A model can be evaluated using precision, recall and F1 scores. Both NLTK and Flair have built-in functions to evaluate the model whereas SpaCy uses a scorer package for evaluation.

The table below shows the precision, recall, F1 score, and accuracy score using the three packages for a CoNLL 2003 dataset:

Dataset	Package	Precision	Recall	F1 Score
CoNLL 2003	NLTK	0.88	0.89	0.89
CoNLL 2003	Spacy	0.88	0.86	0.87
CoNLL 2003	Flair	0.92	0.93	0.93

Selection of package

By comparing the performance matrix it is observed that Flair tops the list compared to the other two. In terms of speed, SpaCy beats the other two.

Each package offers unique advantages. While it is clear that Flair and SpaCy perform better than NLTK, a choice between SpaCy and Flair can be taken considering the nature of the dataset. For smaller datasets, Flair is a good option considering the accuracy. The time taken will be almost the same as that of SpaCy. SpaCy compromises its precision for its speed.

Limitations

There are limitations for all three packages. It is important to know these limitations for selecting the ideal package for the application. NLTK is slower when compared to the other two packages. It doesn't employ neural networks and splits the sentences without considering the semantics of the sentences.

SpaCy is huge, however, the package is not customisable and the internals are not opaque.

Flair is more ideal for smaller applications but its lower speed makes SpaCy a better candidate for large datasets.

References

1. Named-entity-recognition
2. SpaCy
3. NLTK
4. Evaluating an NER model
5. Training custom NER model using NLTK and SpaCy
6. Training custom NER model using Flair
7. NLTK for NLP