Introduction

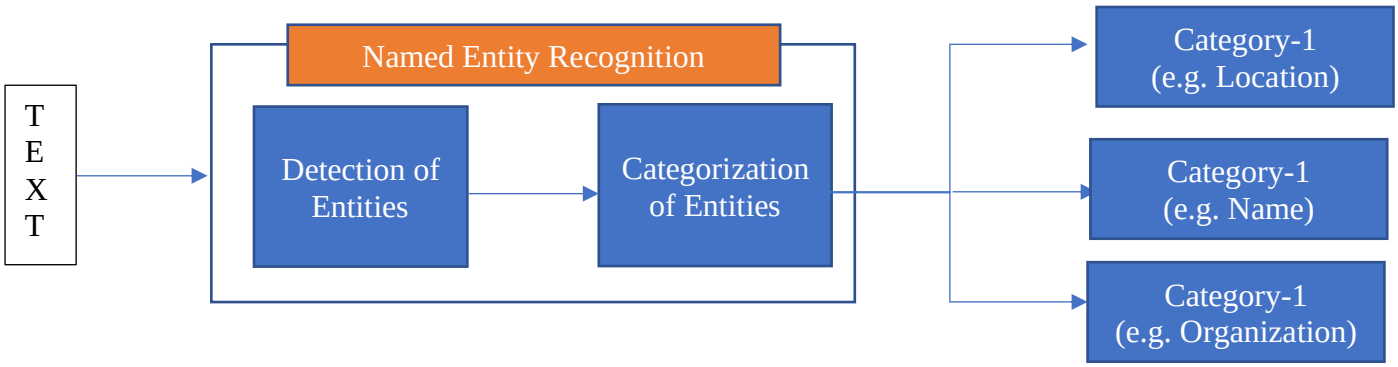
Named Entity Recognition **(NER)** is one of the types of Information Extraction task i.e. extracting structured information from unstructured and/or semi-structure documents and other electronically represented sources such as websites. Named Entity Recognition **(NER)** focuses on extracting information that classifies as "Named Entity" such as a person, location, organization, product, etc., that can be denoted with a proper name.

Named entity recognition is NLP technique that can scan text information source and pull out some fundamental entities and classify them into predefined categories.

For example, in the following sentence "Joseph Robinette Biden, Jr. was born in Scranton, Pennsylvania, the first of four children of Catherine Eugenia Finnegan Biden and Joseph Robinette Biden, Sr. In 1953", we can extract following named entities [Joseph Robinette Biden] person, [Scranton, Pennsylvania] location, [Catherine Eugenia Finnegan Biden] person, [Joseph Robinette] person etc.

Body

In this review, we will primarily discuss about usage of OpenNLP for NER. NER is a comprised of two step process that is Detecting the Entity and Categorizing them.



Models are building blocks for performing NER those are used for categorization. OpenNLP has provision to build and test model with help of "Name Finder Tool" although "Name Finder Tool" is not recommended in production since it loads model every time. A good approach is to load the models at start up to reduce latency. Solr/Elasticsearch implementations loads the model at process startup and use the Name Finder API embed in application itself.

Some pre-trained existing models are listed below

en	Name Finder	Date name finder model.
en	Name Finder	Location name finder model.
en	Name Finder	Money name finder model.
en	Name Finder	Organization name finder model.

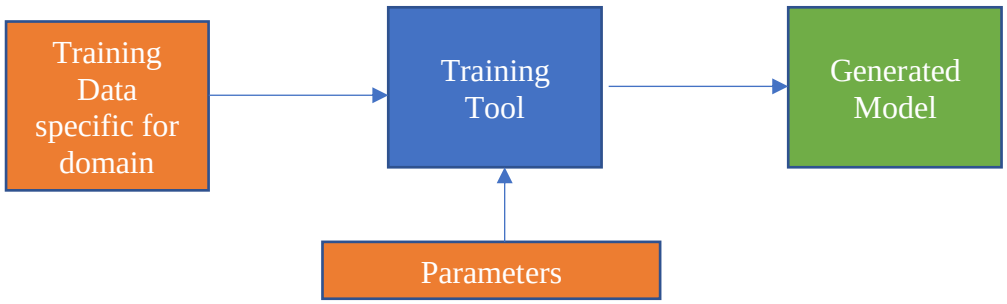
en	Name Finder	Percentage name finder model.
en	Name Finder	Person name finder model.
en	Name Finder	Time name finder model.

OpenNLP has a provision to train and use a custom model for custom use cases. The custom model fits in for the use cases where the need cannot be fulfilled using models provided by OpenNLP such as detecting a different entity type or working in another domain.
The model can be trained by two means listed below

1. Training Tool
2. Training API

We will discuss both of them briefly

Training Tool takes training data and custom parameters as input and generate the model that can be used in Name Finder API.



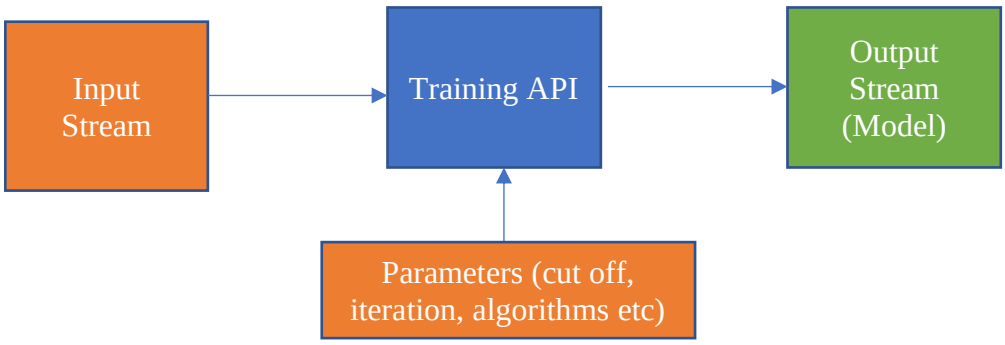
Sample Training Data format is as below

```
<START:person> Pierre Vinken <END> , 61 years old , will join the board
as a nonexecutive director Nov. 29 .
Mr . <START:person> Vinken <END> is chairman of Elsevier N.V. , the
Dutch publishing group
```

The training data has defined structure and rules that are listed below

- Entities are surrounded by tags
- Spacing between tags (START/END) and labeled data is required
- A line end depicts end of document
- A line is treated as a sentence by OpneNLP
- At least 15000 sentences should be there in training data for model to perform well

Alternatively **Training API** can be used to train a custom model embedded within an Application.

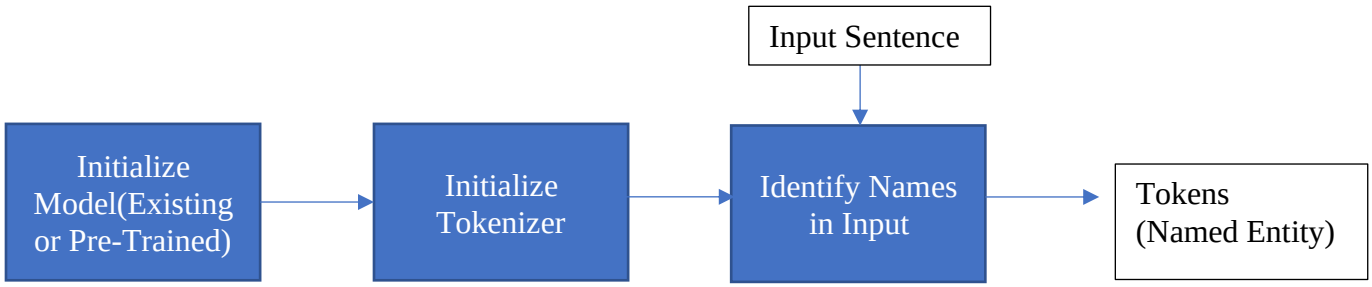


Following github link has sample files for Parameter tuning
https://github.com/apache/opennlp/tree/master/opennlp-tools/lang/ml

The Training Tool extracts features and inputs them to the machine learning algorithm(s) for document classification. A feature is nothing but token string or number. In OpenNLP, such features are generated by feature generators. OpenNLP has provision to configure the default feature generators and their parameters. OpenNLP is quite extensible i.e., OpenNLP model can be trained for custom features for domain specific use cases.

Using Model

Following flow diagram explains how OpenNLP extracts Named Entity.



Sample code reference for NER can be found in following link
https://www.tutorialspoint.com/opennlp/opennlp_named_entity_recognition.html

It is possible to determine the probability associated with a particular name that has been identified. It is useful when we want to filter out some of the

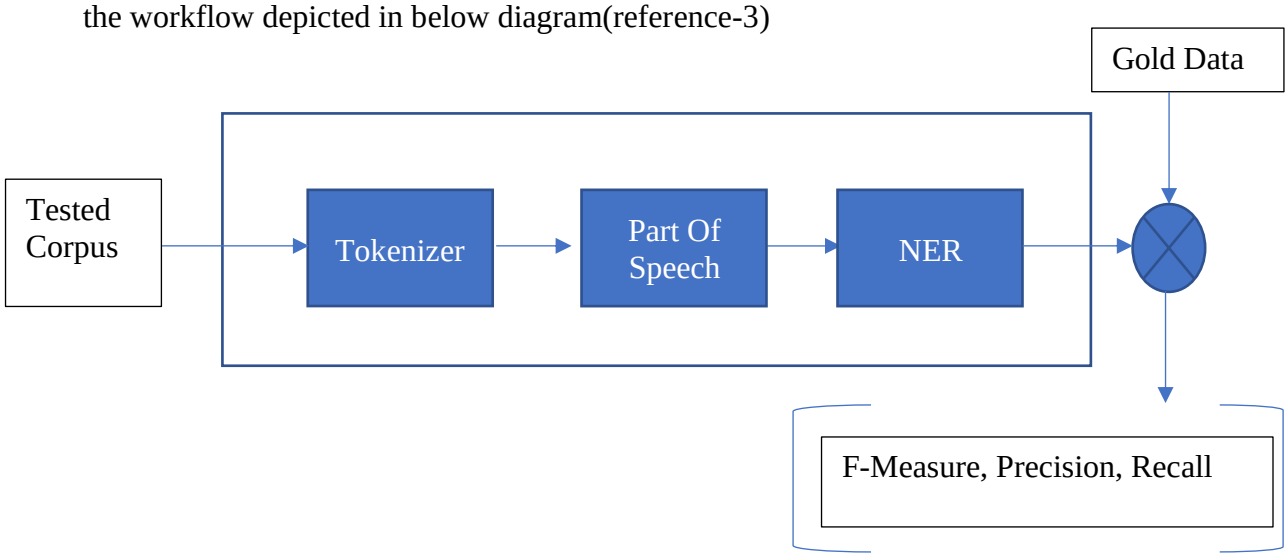
names returned by the name finder, which might be categorized by mistakes.

Major NER platforms apart from OpenNLP are listed below

- Stanford Named Entity Recognizer (SNER): It is JAVA based standard library for NER. Based on Conditional Random Fields (CRF) , it provides pre-trained models for performing NER.
- SpaCy: It is a python based framework with default trained pipelines
- Natural Language Toolkit (NLTK): Python based framework for NLP

Evaluation of NER platforms

The way to evaluate the performance of any given NER software often follows the workflow depicted in below diagram(reference-3)



As inputs, both a corpus of reference and the associated “gold data” are used in order to evaluate the extent to which the identification/classification resulting from the NER software matches with the gold data.

There are various studies performed for comparing various NER platform. The detailed comparison of NER platform is beyond scope of this technical review and some reference paper is already available(reference-3).Based on features discussed so far, we can clearly conclude that OpenNLP is one of the good choice for performing NER.

Conclusion

Identifying entities and their classification can be great advantage for various application that involve text processing and knowledge representation. So far, we discussed about OpenNLP features for performing Named Entity Recognition. We described how OpenNLP performs this task and examined how to customize

models for domain specific use cases using “Training Tool” and “Training API”. There are various alternative to OpenNLP and those were listed previously. Finally, we discussed about idea of comparing various NER platform. We could have discussed more about result for comparison of various NER platform and can be taken up as enhancement to this tech review.

References

1. https://opennlp.apache.org/docs/1.9.2/manual/opennlp.html#tools.namefind
2. https://www.tutorialkart.com/opennlp/ner-training-in-opennlp-with-name-finder-training-java-example/
3. https://www.researchgate.net/publication/337977695_A_Replicable_Comparison_Study_of_NER_Software_StanfordNLP_NLTK_OpenNLP_SpaCy_Gate