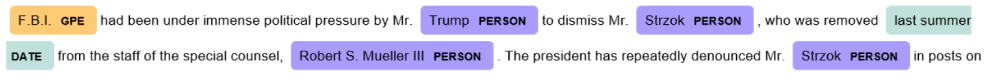
**Named Entity Recognition(NER):**

**WHAT?**

**NER** is a part of natural language processing (NLP) and information Extraction (IE).

**Task** –

To locate and classify named entity mentions in unstructured text into pre-defined categories such as the person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.



API: <https://explosion.ai/demos/displacy-ent>

**Problem Category** –

Sequence labelling task of pattern recognition. POS tagging is also one such problem which requires sequence labelling.

**NER Categories**

Three top-level:

**Entity Names**: represent the identity of an element.  
Ex: name of a person, title, organization, any living or nonliving thing etc.

**Temporal expression**: sequence of words with time related elements.  
Ex: calendar dates, times of day, durations etc.

**Numerical expression**: is a mathematical sentence involving only numbers and or operation symbols.   
Ex: financial numbers, tangible entities, mathematical expressions.

**WHY?**

NER is extensively used in question and answer systems, document clustering and various other text analytics applications.

For example,

For news and publishing houses knowing the relevant tags for each article would help in automatically categorizing and discovering the articles, recommending articles to user etc.

Automation of customer Support-Automatically tagged locations and product names can help smoothly route customer queries to right location and people in a company with multiple branches and many employees.

**Approaches to NER:**

* The classical approach is knowledge/rule based :  
  Rule based NER to extract date and time: <https://github.com/nltk/nltk_contrib/blob/master/nltk_contrib/timex.py>
* By using Supervised Machine Learning  
  🡪 Aided by sequence labelling methods like Hidden Markov Models(HMM) and Conditional Random Fields(CRF).  
  🡪 Neural Networks-RNNs and LSTMs.
* The combination of both.

**Hidden Markov Models(Generative Model):**<https://www.youtube.com/watch?v=kqSzLo9fenk>

<https://medium.freecodecamp.org/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24>

**Conditional Random Fields (Discriminative Model):**

<http://blog.echen.me/2012/01/03/introduction-to-conditional-random-fields/>

<https://medium.com/analytics-vidhya/pos-tagging-using-conditional-random-fields-92077e5eaa31>

**CRF packages in python**-python-crfsuite and sklearn-crfsuite.

<https://towardsdatascience.com/named-entity-recognition-and-classification-with-scikit-learn-f05372f07ba2>

**Nutshell:**

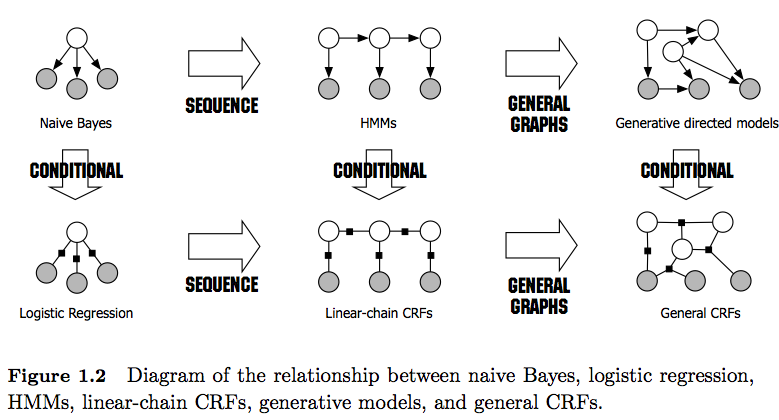
A CRF can be considered as a generalization of HMM or we can say that a HMM is a particular case of CRF where constant probabilities are used to model state transitions.

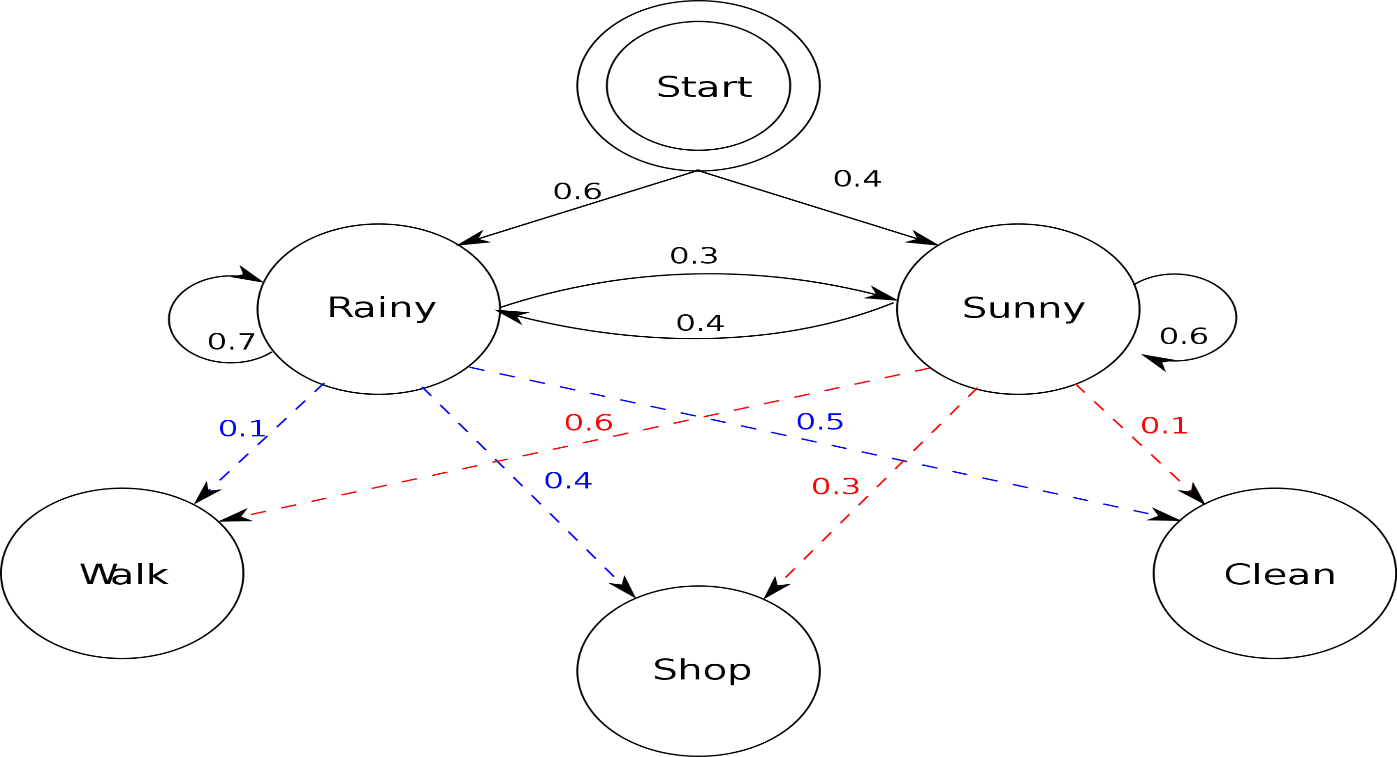
HMM are Generative Models whereas CRFs are Discriminative models.  
In General, A Discriminative model ‌models the decision boundary between the classes. A Generative Model ‌explicitly models the actual distribution of each class.

In final both of them is predicting the conditional probability P (Animal | Features). But Both models learn different probabilities.

A Generative Model ‌learns the joint probability distribution p(x,y). It predicts the conditional probability with the help of Bayes Theorem. A Discriminative model ‌learns the conditional probability distribution p(y|x).

<https://medium.com/@mlengineer/generative-and-discriminative-models-af5637a66a3>





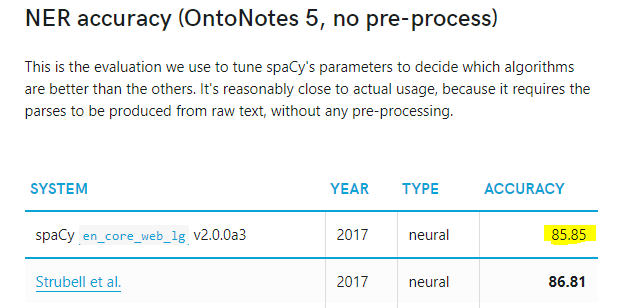
**Popular libraries in Python:**

* **Spacy** features fast statistical NER as well as an open-source named-entity visualizer. It has been trained on the [OntoNotes 5](https://catalog.ldc.upenn.edu/ldc2013t19) corpus.

Entities Supported: <https://spacy.io/api/annotation#section-named-entities>

Spacy’s deep learning implementation: <https://www.youtube.com/watch?v=sqDHBH9IjRU&feature=youtu.be>

**Accuracy**: <https://spacy.io/usage/facts-figures>



* **Stanford NER**- CRFClassifier. The software provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models.
* With the function **NLTK**.ne\_chunk(), we can recognize named entities such as PERSON, ORGANIZATION, and GPE.
* **Summary:**
* Stanford NER need extra implementation for entities with more than 1 word. Also, the performance of tagging is the slowest by comparing to other two libraries.
* spaCy seems like the easier library to get the entities labelled and no extra setup is needed.
* NLTK NE\_Chunk needs more setups (downloading pre-trained file) but it is just one-off. The result seems to be not good compared to others.

**NOTE:**

Named entities are often not simply singular words, but are chunks of text, e.g. "University of British Columbia" or "Bank of America". Therefore, some chunking and parsing prediction model is required to predict whether a group of tokens belong in the same entity.

In Machine Learning Approach to incorporate multi-word entity names (or chunks), our dataset needs to use IOB tags as labels to train the model. In IOB tags, each word is tagged with one of three special chunk tags, I (Inside), O (Outside), or B (Begin). A word is tagged as B if it marks the beginning of a chunk, subsequent words within the chunk are tagged I and all other words are tagged O.

<https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da>

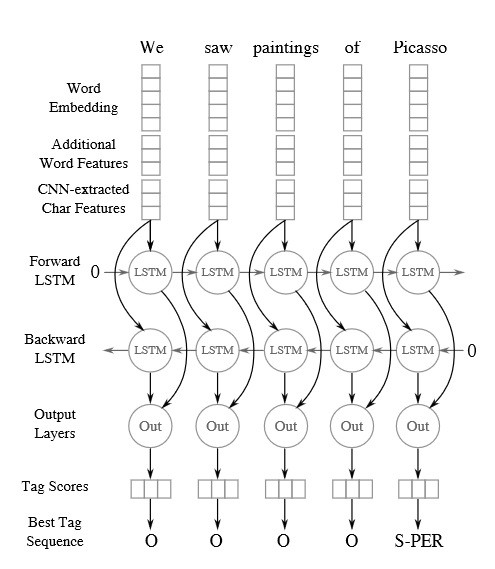
<https://towardsdatascience.com/named-entity-recognition-3fad3f53c91e>

**Deep Learning in NER:**

**State –of –the-art Deep learning approaches to NER and their accuracies:**

<http://nlpprogress.com/english/named_entity_recognition.html>

[**Bidirectional LSTM-CNNs**](https://arxiv.org/pdf/1511.08308.pdf)**:**



Source [*https://towardsdatascience.com/named-entity-recognition-ner-meeting-industrys-requirement-by-applying-state-of-the-art-deep-698d2b3b4ede*](https://towardsdatascience.com/named-entity-recognition-ner-meeting-industrys-requirement-by-applying-state-of-the-art-deep-698d2b3b4ede)

**Other Notable Implementations:**

* [Bidirectional LSTM-CNNS-CRF:](https://arxiv.org/pdf/1603.01354.pdf)
* NER using Neural nets and ELMo (Embedding from Language Models):  
  ELMo contextual embeddings are a real breakthrough for NER, boosting performances by 2.0 points in f-score on the CoNLL 2003 NER corpus, but at the cost of a 25-times slower prediction time.
  + If we consider the best performing NER system based on CRF, until very recently it was difficult to view neural NER systems as clear improvements for NER in case of relatively small corpus like CoNLL 2003 (averaged f-scores of neural NER not using ELMo are similar to the best reported f-score for CRF). The gain of neural NER was however clear for larger training data like [Ontonotes](https://catalog.ldc.upenn.edu/LDC2013T19) (around +3.0 on the f-score).
  + [ELMo](https://allennlp.org/elmo) contextual embeddings brought a very significant boost in performance.   
    The idea of contextual embeddings is that embeddings are generated dynamically based on the context of several words (the sentence in the case of ELMo) instead of fixed-word embeddings independent from the context (like with word2vec, GloVe, FastText, etc.).  
    ELMo relies on a bidirectional language model applied to a sequence (the sentence) to generate the vector representation of a word.
  + However, calculating embeddings in context is much more time consuming than simply performing the look-up of a static word embedding
* BERT (Bidirectional Encoder Representations from Transformers):

<https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>

**Interesting Read:**

<http://jalammar.github.io/illustrated-bert/>

**Implementations:**

<https://www.depends-on-the-definition.com/guide-sequence-tagging-neural-networks-python/>

**Further Reading:**

**Unsupervised and Semi supervised learning Approach for NER:** There is no Current Industry standard implementations for NER.