Open in app

Mohammed Terry-Jack

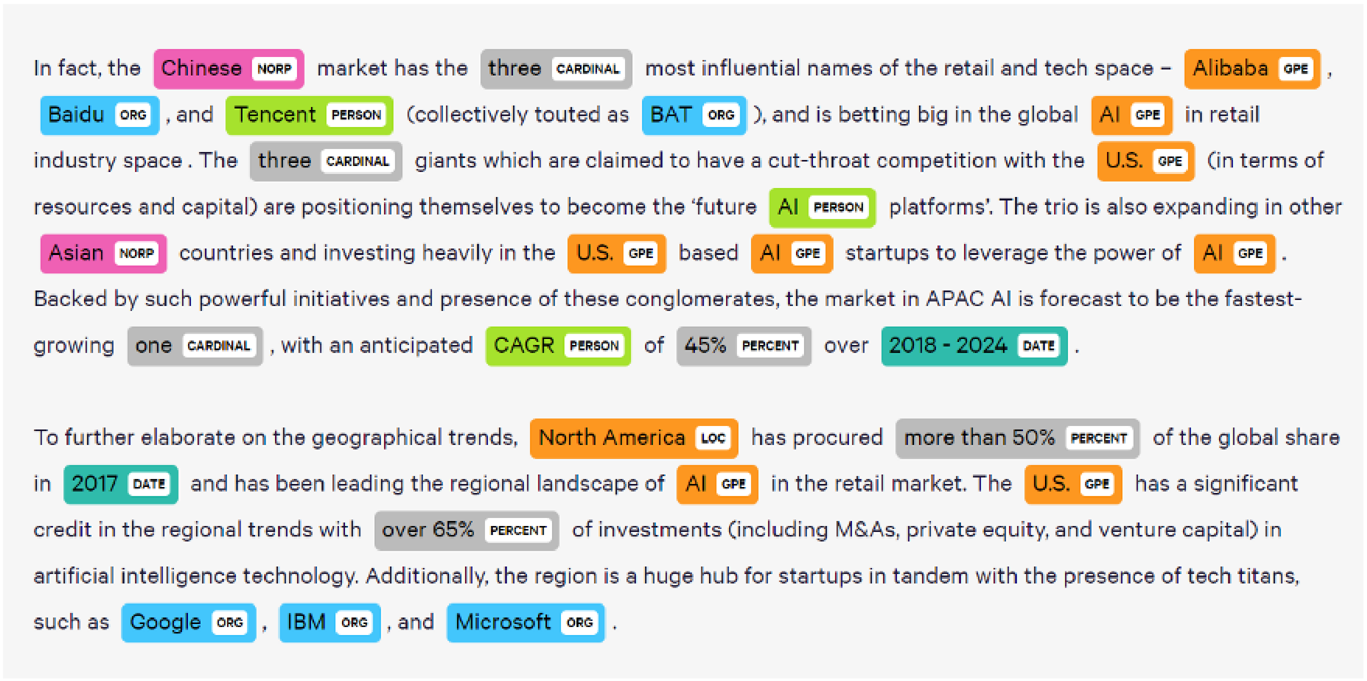
204 Followers · About

Follow

NLP: Pretrained Named Entity Recognition (NER)

Mohammed Terry-Jack May 4, 2019 · 6 min read

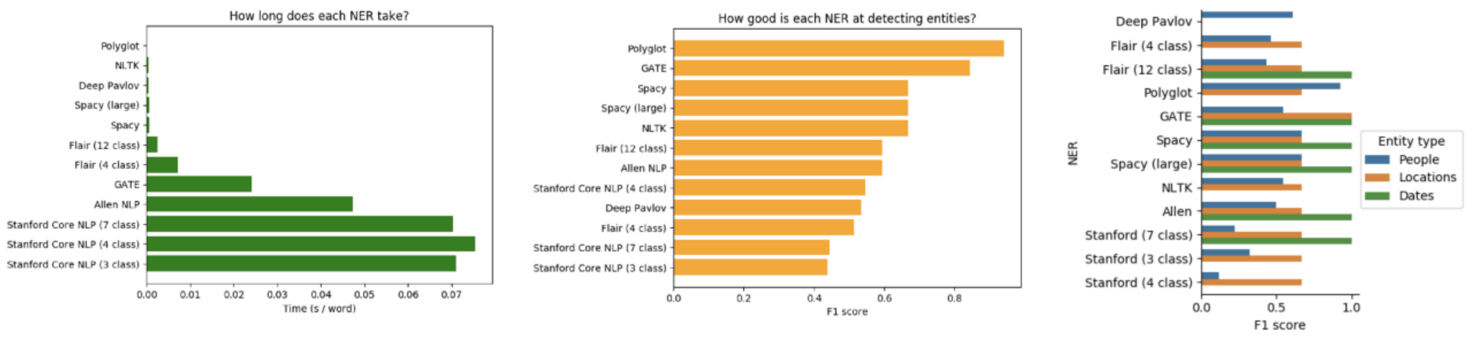
There are a good range of pre-trained **Named Entity Recognition** (NER) models provided by popular open-source NLP libraries (e.g. NLTK, Spacy, Stanford Core NLP) and some less well known ones (e.g. Allen NLP, Flair, Polyglot, Deep Pavlov) as well as the odd (free) API (e.g. GATE). This tutorial tests out a few and discusses their differing, underlying methods (from Rules-based and CRFs to Deep Neural Networks).



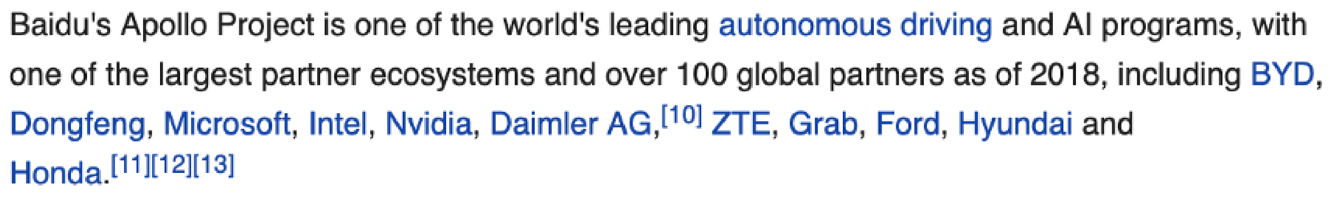
In fact, the Chinese NORP market has the three CARDINAL most influential names of the retail and tech space – Alibaba GPE , Baidu ORG , and Tencent PERSON (collectively touted as BAT ORG), and is betting big in the global AI GPE in retail industry space . The three CARDINAL giants which are claimed to have a cut-throat competition with the U.S. GPE (in terms of resources and capital) are positioning themselves to become the 'future AI PERSON platforms'. The trio is also expanding in other Asian NORP countries and investing heavily in the U.S. GPE based AI GPE startups to leverage the power of AI GPE . Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing one CARDINAL , with an anticipated CAGR PERSON of 45% PERCENT over 2018 - 2024 DATE .

To further elaborate on the geographical trends, North America LOC has procured more than 50% PERCENT of the global share in 2017 DATE and has been leading the regional landscape of AI GPE in the retail market. The U.S. GPE has a significant credit in the regional trends with over 65% PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as Google ORG , IBM ORG , and Microsoft ORG .

Automatically labelling entities in a document using an NER model well as the number of labels an NER model can classify (some classify 3 types of entities, others 4, others 12, etc) depends upon the public dataset used to train the model



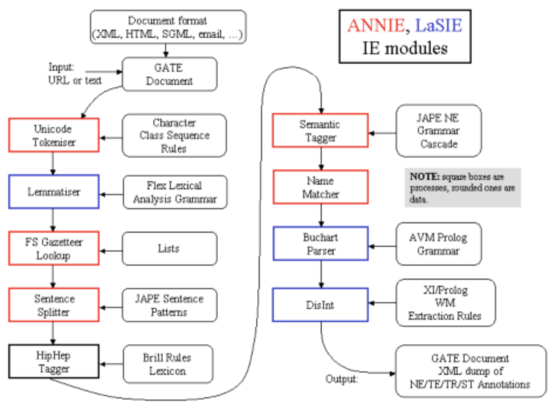
Speed of NER model predictions (Left), Quality of NER model using average F1 Score (Middle), F1 scores to measure the quality of entity labelling for three entity types: People, Locations, Dates (Right)


Baidu's Apollo Project is one of the world's leading autonomous driving and AI programs, with one of the largest partner ecosystems and over 100 global partners as of 2018, including BYD, Dongfeng, Microsoft, Intel, Nvidia, Daimler AG,[10] ZTE, Grab, Ford, Hyundai and Honda.[11][12][13]

The above text snippet from Wikipedia is given to the NER models shown below

GATE

This first one is an API, but I have included it here as it is free to use. It is called ANNIE and is rules-based (although it has rules that work on different layers of abstraction along the NLP pipeline).



```
[('2018', 'Date'),
 ('Ford', 'Location'),
 ('Apollo Project', 'Organization'),
 ('Microsoft', 'Organization'),
 ('Intel', 'Organization'),
 ('Nvidia', 'Organization'),
 ('Daimler AG', 'Organization'),
 ('Hyundai', 'Organization'),
 ('Honda', 'Organization')]
```

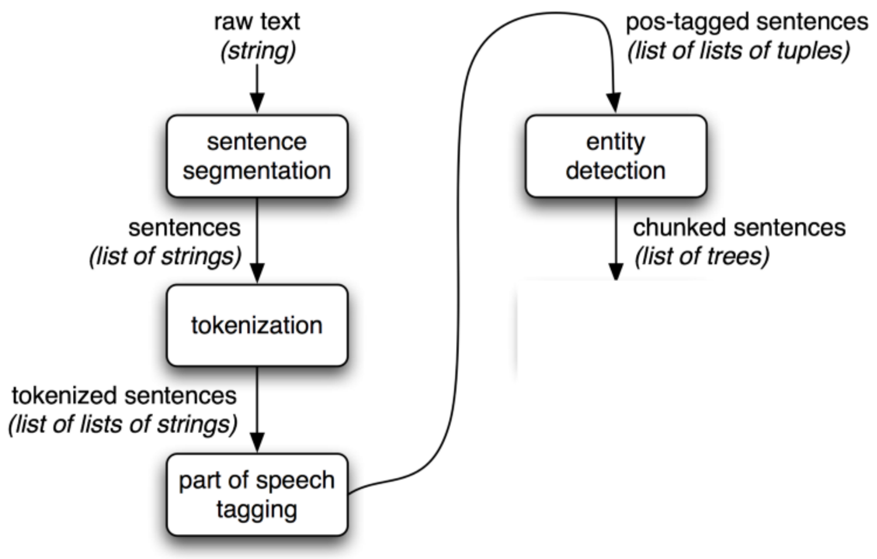
NER results from Gate's ANNIE

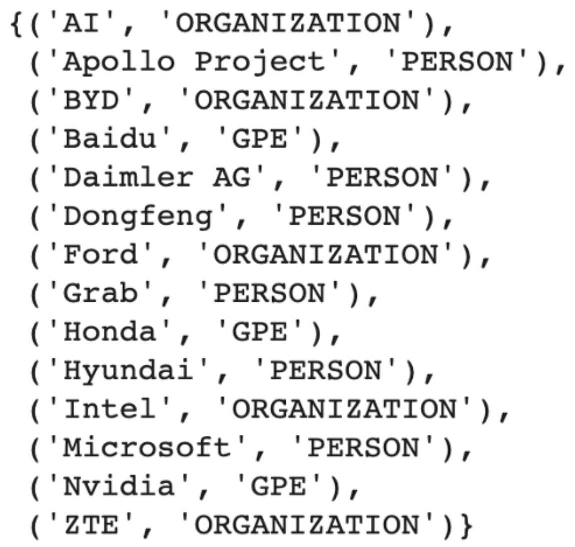
```
entity-recognizer"
headers = {'Content-Type': 'text/plain'}
response = requests.post(url, data=example_document,
headers=headers).json()

import json
print(json.dumps(response, indent=2))
```

NLTK

NLTK provides all the traditional NLP components to construct an NER pipeline to the one shown for GATE. The text is tokenised > the tokens are passed through a Part Of Speech (POS) tagger > a parser chunks the tokens based on their POS tags to find named entities.



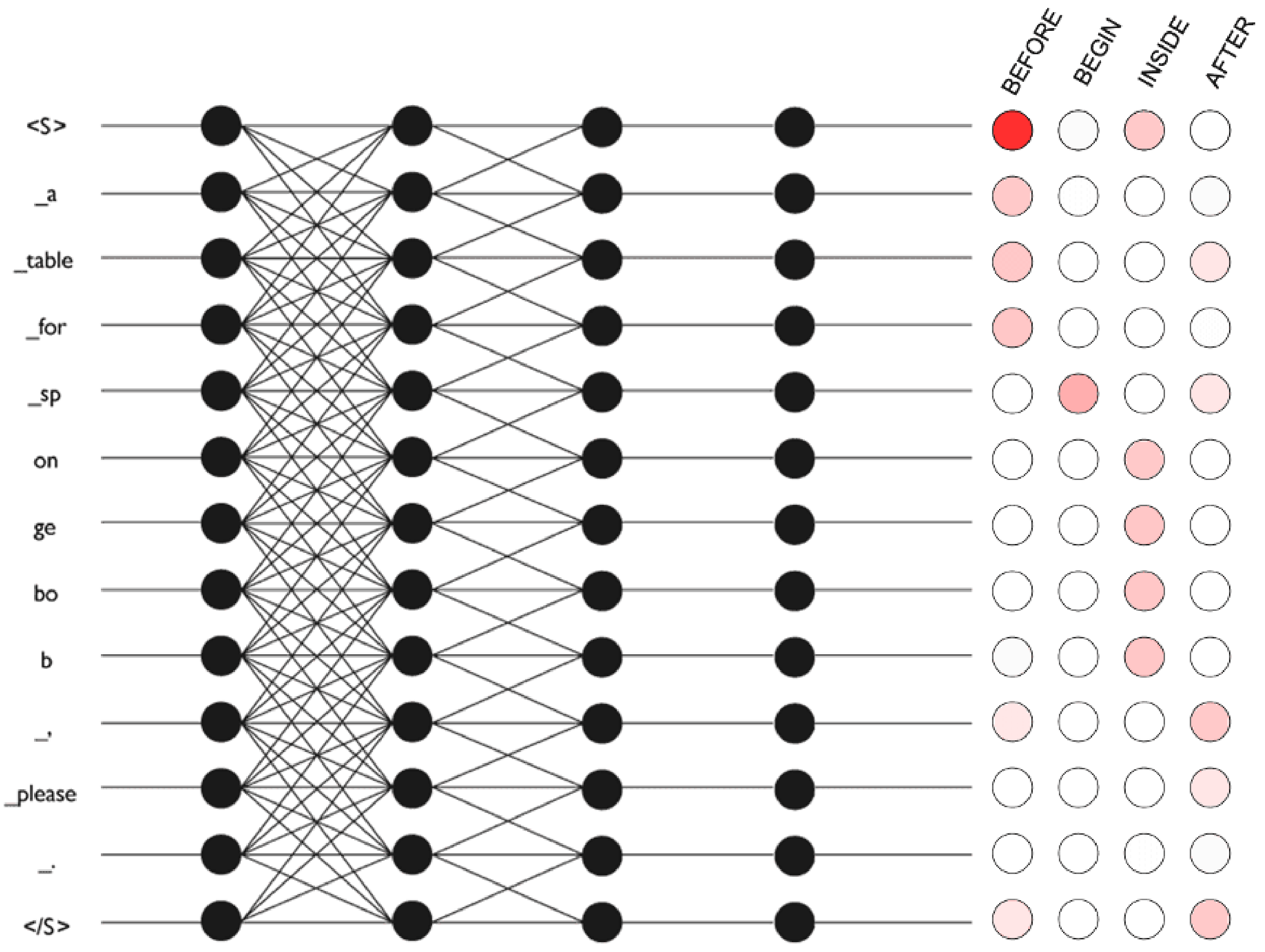


```
{('AI', 'ORGANIZATION'),
 ('Apollo Project', 'PERSON'),
 ('BYD', 'ORGANIZATION'),
 ('Baidu', 'GPE'),
 ('Daimler AG', 'PERSON'),
 ('Dongfeng', 'PERSON'),
 ('Ford', 'ORGANIZATION'),
 ('Grab', 'PERSON'),
 ('Honda', 'GPE'),
 ('Hyundai', 'PERSON'),
 ('Intel', 'ORGANIZATION'),
 ('Microsoft', 'PERSON'),
 ('Nvidia', 'GPE'),
 ('ZTE', 'ORGANIZATION')}
```

NLTK NER results

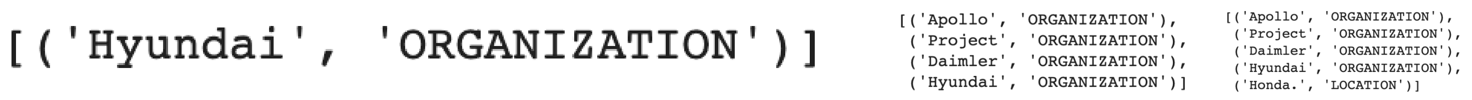
```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
{(' '.join(c[0] for c in chunk), chunk.label() ) for chunk in
nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(document))) if
hasattr(chunk, 'label') }
```

called a Conditional Random Field (CRF). This was the state of the art approach for a while (prior to more modern, deep learning NER models)



CRF with the Viterbi algorithm to decide the most probable sequence of tags

An older version of NLTK had an inbuilt wrapper which could access Stanford Core NLP and its pretrained models. Stanford Core NLP offers three pretrained NER models, containing 3, 4 and 7 entity types respectively.



```
[('Hyundai', 'ORGANIZATION')]
```

```
[('Apollo', 'ORGANIZATION'),
 ('Project', 'ORGANIZATION'),
 ('Daimler', 'ORGANIZATION'),
 ('Hyundai', 'ORGANIZATION')]
```

```
[('Apollo', 'ORGANIZATION'),
 ('Project', 'ORGANIZATION'),
 ('Daimler', 'ORGANIZATION'),
 ('Hyundai', 'ORGANIZATION'),
 ('Honda.', 'LOCATION')]
```

Results from Stanford Core NLP's 3class (left), 4class (centre) and 7class (right) NER models

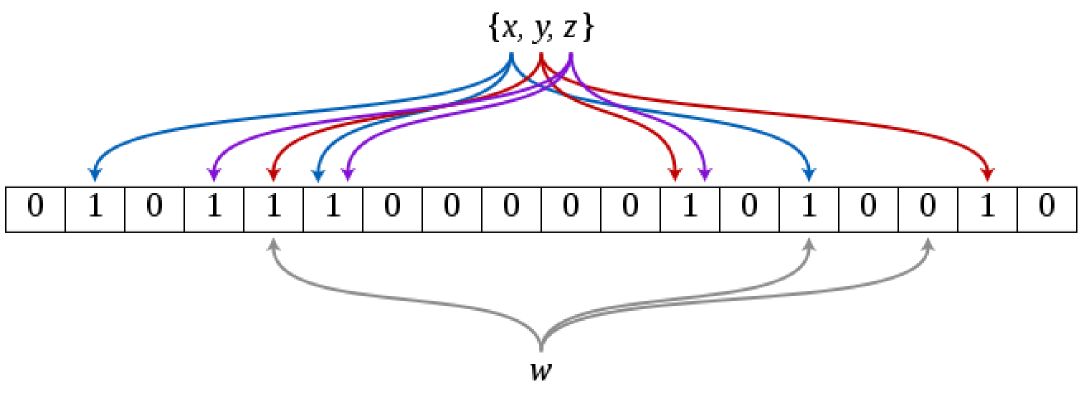
```
from nltk.tag.stanford import StanfordNERTagger
jar = "stanford-ner-2015-04-20/stanford-ner-3.5.2.jar"
model = "stanford-ner-2015-04-20/classifiers/"

st_3class = StanfordNERTagger(model +
"english.all.3class.distsim.crf.ser.gz", jar, encoding='utf8')
st_4class = StanfordNERTagger(model +
"english.conll.4class.distsim.crf.ser.gz", jar, encoding='utf8')
st_7class = StanfordNERTagger(model +
"english.muc.7class.distsim.crf.ser.gz", jar, encoding='utf8')

st_3class.tag(document.split())
st_4class.tag(document.split())
st_7class.tag(document.split())
```

Spacy

Spacy's NER model is a simple classifier (e.g. a shallow feedforward neural network with a single hidden layer) that is made powerful using some clever feature engineering. Before the input features are fed into the classifier, a stack of weighted bloom embedding layers merge neighbouring features together. This gives each word a unique representation for each distinct context it is in.



```
{('2018', 'DATE'),
 ('Apollo Project', 'ORG'),
 ('BYD', 'ORG'),
 ('Baidu', 'ORG'),
 ('Daimler AG', 'ORG'),
 ('Dongfeng', 'PERSON'),
 ('Ford', 'ORG'),
 ('Honda', 'ORG'),
 ('Hyundai', 'ORG'),
 ('Intel', 'ORG'),
 ('Microsoft', 'ORG'),
 ('Nvidia', 'ORG'),
 ('ZTE', 'ORG'),
 ('over 100', 'CARDINAL')}
```

bloom embeddings merge multiple, neighbouring features (x,y,z) into a single fixed-length vector

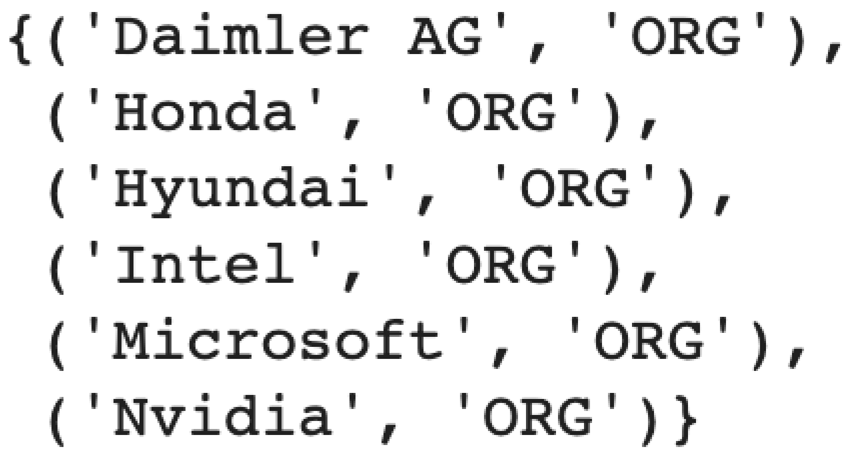
```
!python3 -m spacy download en_core_web_lg
import spacy
sp_lg = spacy.load('en_core_web_lg')
{(ent.text.strip(), ent.label_) for ent in sp_lg(document).ents}
```

> *"Successful approaches to address NER rely on supervised learning …they require human annotated datasets which are scarce."*

Rather, it uses huge unlabelled datasets (like Wikipedia) with automatically inferred entity labels (via features such as hyperlinks).

> *"We use the internal links embedded in Wikipedia articles to detect named entity mentions. When a link points to an article identified by Freebase as an entity article, we include the anchor text as a positive training example."*

By cleverly addressing the supervised learning labelling limitation, Polyglot has been able to leverage a massive multilingual corpus to train even a simple classifier (e.g. a feedforward neural network) to become a very robust, competitive NER model.



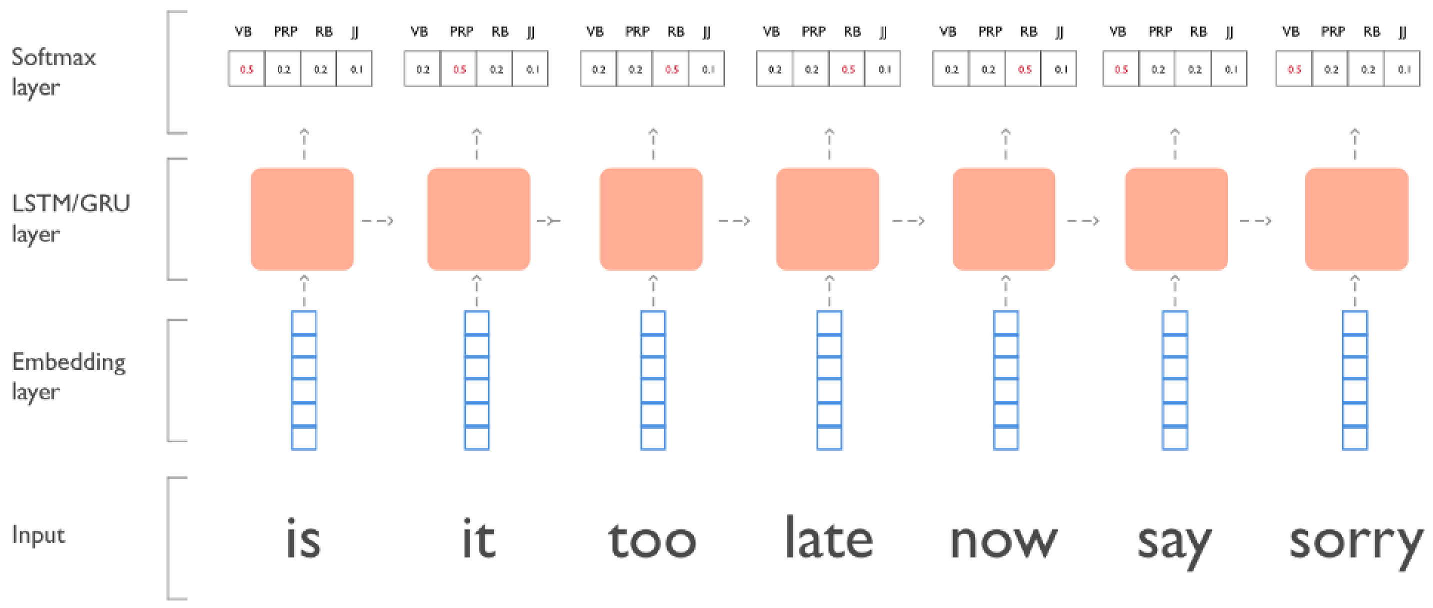
```
{('Daimler AG', 'ORG'),
 ('Honda', 'ORG'),
 ('Hyundai', 'ORG'),
 ('Intel', 'ORG'),
 ('Microsoft', 'ORG'),
 ('Nvidia', 'ORG')}
```

Results of Polyglot's NER model

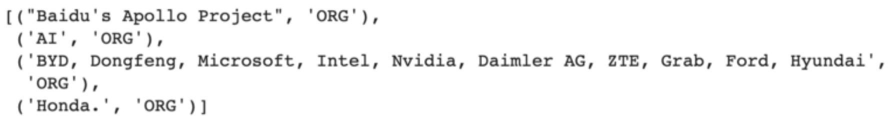
```
!pip3 install -U git+https://github.com/aboSamoor/polyglot.git@master
!polyglot download embeddings2.en ner2.en
from polyglot.text import Text
Text(document).entities
```

Flair

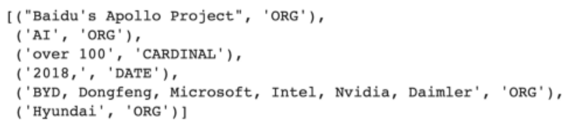
different). Whereas Spacy adds a feature engineering step to encode the context of neighbouring words into its NER model at the word embedding layer, Flair uses a deep learning model to do such feature engineering implicitly (an LSTM *remembers* previous words which have appeared in the sentence).



An LSTM with a word embedding layer



```
[("Baidu's Apollo Project", 'ORG'),
 ('AI', 'ORG'),
 ('BYD, Dongfeng, Microsoft, Intel, Nvidia, Daimler AG, ZTE, Grab, Ford, Hyundai',
  'ORG'),
 ('Honda.', 'ORG')]
```



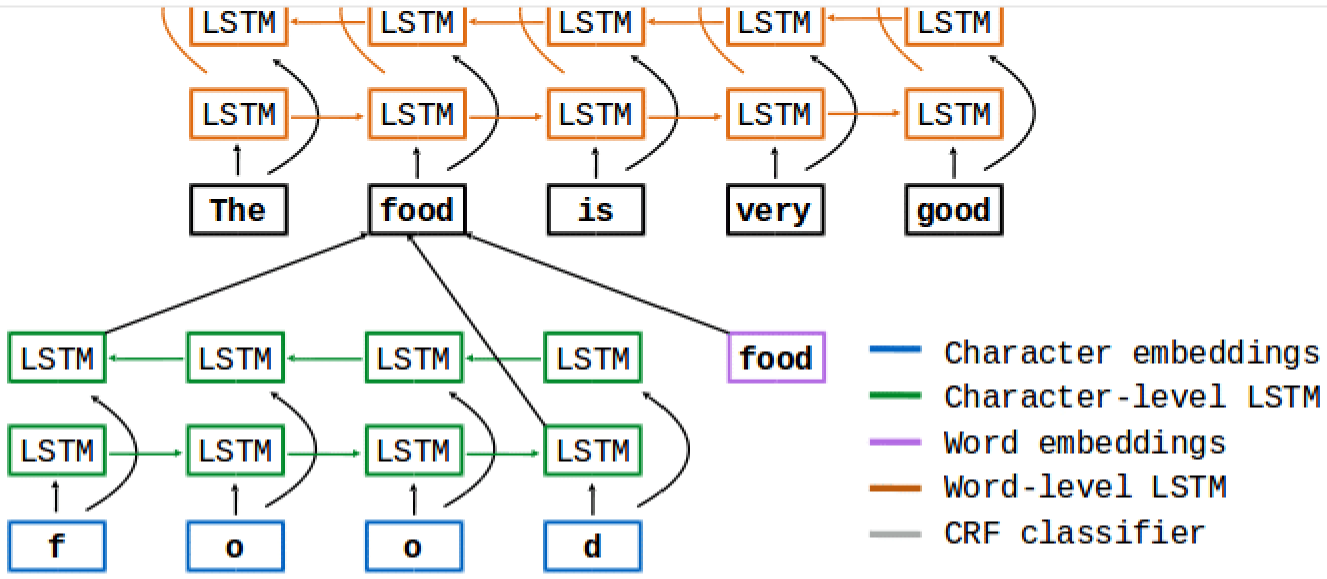
```
[("Baidu's Apollo Project", 'ORG'),
 ('AI', 'ORG'),
 ('over 100', 'CARDINAL'),
 ('2018,', 'DATE'),
 ('BYD, Dongfeng, Microsoft, Intel, Nvidia, Daimler', 'ORG'),
 ('Hyundai', 'ORG')]
```

(left) results from flair's 4 class 'ner-ontonotes-fast' NER model. (right) results from flair's 12 class 'ner' NER model

```
!pip3 install flair
from flair.models import SequenceTagger
model = SequenceTagger.load('ner-ontonotes-fast') #.load('ner')
from flair.data import Sentence
s = Sentence(document)
model.predict(s)
s.to_dict(tag_type='ner')
```

Deep Pavlov

DeepPavlov uses a slightly newer variant of Flair's deep neural architecture known as *a Hybrid Bi-LSTM-CRF* model.



biLSTM CRF model

It is a bi-directional LSTM on top of word and character-level embedding layers (as before). However, it combines an additional Conditional Random Fields (CRF) layer to the output of the model

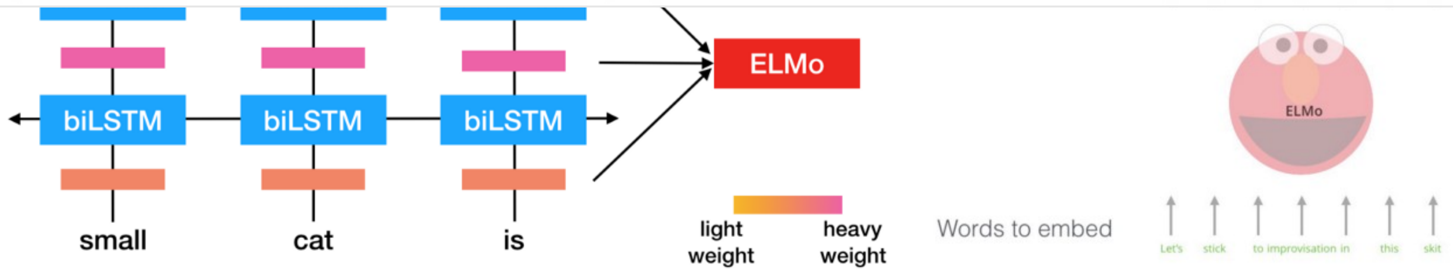
```
{('Apollo Project', 'ORG'), ('Daimler AG', 'ORG'), ('over 100', 'CARDINAL')}
```

Results from Deep Pavlov's NER model

```
!pip3 install deeppavlov
!python3 -m deeppavlov install ner_ontonotes
from deeppavlov import configs, build_model
deeppavlov_ner = build_model(configs.ner.ner_ontonotes,
download=True)
deeppavlov_ner([sentence])
```

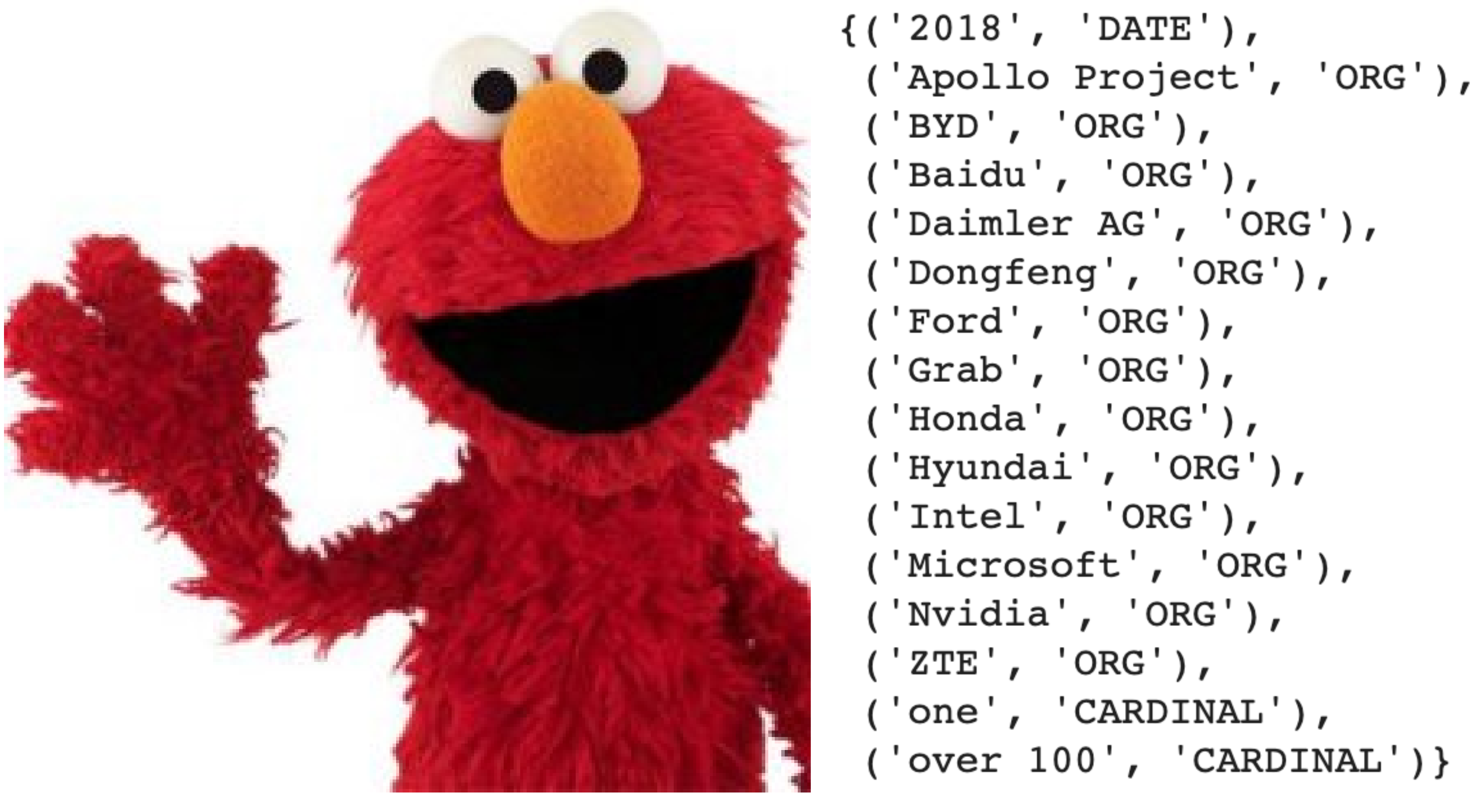
Allen NLP

Allen NLP offers two NER models with differt architectures. The smaller model uses a Gated Recurrent Unit (GRU) Network to embed words at the character level and another GRU to encode phrases from words embedding using Glove (like LSTMs, GRUs are able to remember sequences of words to add context to the representation).



Elmo language model can be used to embed words

The second, “fine-grained” NER model uses a bi-LSTM-CRF model (like DeepPavlov) but also includes a pre-trained bi-lstm network known as “Elmo” (the State-Of-The-Art language model prior to BERT and GPT2, etc) as a better word embedding layer.



```
{('2018', 'DATE'),
 ('Apollo Project', 'ORG'),
 ('BYD', 'ORG'),
 ('Baidu', 'ORG'),
 ('Daimler AG', 'ORG'),
 ('Dongfeng', 'ORG'),
 ('Ford', 'ORG'),
 ('Grab', 'ORG'),
 ('Honda', 'ORG'),
 ('Hyundai', 'ORG'),
 ('Intel', 'ORG'),
 ('Microsoft', 'ORG'),
 ('Nvidia', 'ORG'),
 ('ZTE', 'ORG'),
 ('one', 'CARDINAL'),
 ('over 100', 'CARDINAL')}
```

Results from Allen NLP’s NER model

```
!pip3 install allennlp
from allennlp.predictors import Predictor
al = Predictor.from_path("https://s3-us-west-
2.amazonaws.com/allennlp/models/fine-grained-ner-model-elmo-
```

Google Colaboratory

Edit description

colab.research.google.com

Don't forget to check out the related NLP tutorial on Entity Grounding

Machine Learning Ner Named Entity Recognition NLP Naturallanguageprocessing