# Spacy and Custom NER

Josh Smith <kognate@gmail.com>

May 23, 2018

## Outline

1. Natural Language Processing
2. Tools for NLP in Python
3. Demo of Basic Spacy
4. Named Entity Recognition in Spacy
5. Deploy the service

## Stop Words

- These are glue words and things that can be ignored sometimes.

```
doc = nlp(('These are glue words and'
           ' things that can be '
           'ignored sometimes.'))
[(x.lemma_, x.pos_, x.text)
 for x
 in [y
     for y
     in doc
     if y.is_stop]]
```

| | | |
|---|---|---|
| be | VERB | are |
| and | CCONJ | and |
| that | ADJ | that |
| can | VERB | can |
| be | VERB | be |
| sometimes | ADV | sometimes |

## Parsing Human Languages

But you need a language model (at minimum).

## Entity Recognition

This is knowing that Jason is a Person and not a Verb.

## NLTK

This is one of the most mature and widley used frameworks for NLP

## Spacy

This is the fastest. And has, IMHO, a great syntax

This is what we're going to talk about

## Installs from pip

```
pip install spacy
```

## don't forget to download at least one model

```
python -m spacy download en
```

## First, we must import and load

```python
import spacy
nlp = spacy.load('en')
```

## Now we can use it

```
import spacy
nlp = spacy.load('en')
doc = nlp("Hello everyone, I've some good news to give you"
cleaned = [y for y
           in doc
           if not y.is_stop and y.pos_ != 'PUNCT']
raw = [(x.lemma_, x.pos_) for x in cleaned]
print(raw)
raw
```

```
hello     INTJ
-PRON-    PRON
have      VERB
good      ADJ
news      NOUN
```

```python
import spacy
nlp = spacy.load('en')
to_analyze = ('Hello Code & Supply, '
              'my name is Josh and tonight '
              'we\'re in Pittsburgh')
doc = nlp(to_analyze)
ents = [(x.text, x.label_)
        for x in doc.ents]
print(ents)
ents
```

| | |
|---|---|
| Josh | PERSON |
| tonight | TIME |
| Pittsburgh | GPE |

- https://spacy.io/api/annotation

## gui tools:

- https://prodi.gy/ from the creators of spacy

## Train a model in Spacy

you can also create a vm for this:

```
gcloud compute instances create cands
     --image-family ubuntu-1804-lts
     --image-project ubuntu-os-cloud
     --machine-type n1-highcpu-16
```

## Some imports

```
import spacy
from spacy.matcher import PhraseMatcher
import plac
from pathlib import Path
import random
```

## Utility function

This function converts the output of the PhraseMatcher to something usable in training. The training data needs a string index of characters (start, end, label) while the matched output uses index of words from an nlp document.

```python
def offseter(lbl, doc, matchitem):
    o_one = len(str(doc[0:matchitem[1]]))
    subdoc = doc[matchitem[1]:matchitem[2]]
    o_two = o_one + len(str(subdoc))
    return (o_one, o_two, lbl)
```

## Load and setup

Here we load spacy and setup the pipes for training.

```
nlp = spacy.load('en')

if 'ner' not in nlp.pipe_names:
    ner = nlp.create_pipe('ner')
    nlp.add_pipe(ner)
else:
    ner = nlp.get_pipe('ner')
```

## Setup the phrase matches

This is to make our lives easier. Instead of setting this up by hand, we can use PhraseMatcher class from spacy to locate the text we want to label.

```
label = 'CIADIR'
matcher = PhraseMatcher(nlp.vocab)
for i in ['Gina Haspel', 'Gina', 'Haspel',]:
    matcher.add(label, None, nlp(i))
```

## What's that look like?

```
one = nlp('Gina Haspel was nomiated in 2018')
matches = matcher(one)
[match for match in matches]
```

```
1.7539557946531887(+19)   0   1
1.7539557946531887(+19)   0   2
1.7539557946531887(+19)   1   2
```

## Gather training data

```
res = []
to_train_ents = []
with open('gina_haspel.txt') as gh:
    line = True
    while line:
        line = gh.readline()
        mnlp_line = nlp(line)
        matches = matcher(mnlp_line)
        res = [offseter(label, mnlp_line, x)
               for x
               in matches]
        to_train_ents.append((line,
                              dict(entities=res)))
```

## Actually Train The Recognizer

```
optimizer = nlp.begin_training()

other_pipes = [pipe
               for pipe
               in nlp.pipe_names
               if pipe != 'ner']

with nlp.disable_pipes(*other_pipes):  # only train NER
    for itn in range(20):
        losses = {}
        random.shuffle(to_train_ents)
        for item in to_train_ents:
            nlp.update([item[0]],
                       [item[1]],
                       sgd=optimizer,
                       drop=0.35,
                       losses=losses)
```

## Build the Image

```
docker build -f Dockerfile-stemmer . -t gcr.io/codeandsuppl
```

## Gcloud

```
gcloud init
gcloud auth configure-docker
gcloud auth print-access-token
# then cut and paste that token
docker login -u oauth2accesstoken https://gcr.io
# you must have enabled the container registry before you ca
docker push gcr.io/codeandsupply/stemmer:latest
gcloud container clusters create experimental-aone
gcloud container clusters get-credentials experimental-aone
kubectl run stemmer-server
       --image gcr.io/codeandsupply/stemmer:latest
       --port 8000
kubectl expose deployment stemmer-server --type "LoadBalanc
kubectl get service stemmer-server
curl http://<HOSTNAME>:8000/

curl -s "http://127.0.0.1:8000/stemmer?source=Bill+is+a+nice
```