4.2 Named Entity Recognition with NLTK and SpaCy

Named entity recognition (https://en.wikipedia.org/wiki/Named-entity_recognition) (NER)is probably the first step towards information extraction that seeks to locate and classify **named entities** (https://en.wikipedia.org/wiki/Named_entity) in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. NER is used in many fields in **Natural Language Processing** (https://en.wikipedia.org/wiki/Natural_language_processing) (NLP), and it can help answering many real-world questions, such as:

- Which companies were mentioned in the news article?
- Were specified products mentioned in complaints or reviews?
- Does the tweet contain the name of a person? Does the tweet contain this person's location?

This article describes how to build named entity recognizer with **NLTK** (https://www.nltk.org/book/ch07.html) and **SpaCy** (https://spacy.io/usage/linguistic-features), to identify the names of things, such as persons, organizations, or locations in the raw text. Let's get started!

NLTK

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
```

Information Extraction

I took a sentence from **The New York Times** (https://www.nytimes.com/2018/07/18/technology/google-eu-android-fine.html), "European authorities fined Google a record $5.1 billion on Wednesday for abusing its power in the mobile phone market and ordered the company to alter its practices."

```
ex = 'European authorities fined Google a record $5.1 billion on Wednesday for abusing its power in the m
obile phone market and ordered the company to alter its practices'
```

Then we apply word tokenization and part-of-speech tagging to the sentence.

```
def preprocess(sent):
    sent = nltk.word_tokenize(sent)
    sent = nltk.pos_tag(sent)
    return sent
```

Let's see what we get:

```
sent = preprocess(ex)
sent
```

```
[('European', 'JJ'),
 ('authorities', 'NNS'),
 ('fined', 'VBD'),
 ('Google', 'NNP'),
 ('a', 'DT'),
 ('record', 'NN'),
 ('$', '$'),
 ('5.1', 'CD'),
 ('billion', 'CD'),
 ('on', 'IN'),
 ('Wednesday', 'NNP'),
 ('for', 'IN'),
 ('abusing', 'VBG'),
 ('its', 'PRP$'),
 ('power', 'NN'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('mobile', 'JJ'),
 ('phone', 'NN'),
 ('market', 'NN'),
 ('and', 'CC'),
 ('ordered', 'VBD'),
 ('the', 'DT'),
 ('company', 'NN'),
 ('to', 'TO'),
 ('alter', 'VB'),
 ('its', 'PRP$'),
 ('practices', 'NNS')]
```

Figure 1

We get a list of tuples containing the individual words in the sentence and their associated part-of-speech.

Now we'll implement noun phrase chunking to identify named entities using a regular expression consisting of rules that indicate how sentences should be chunked.

Our chunk pattern consists of one rule, that a noun phrase, NP, should be formed whenever the chunker finds an optional determiner, DT, followed by any number of adjectives, JJ, and then a noun, NN.

```
pattern = 'NP: {<DT>?<JJ>*<NN>}'
```

Chunking

Using this pattern, we create a chunk parser and test it on our sentence.

```
cp = nltk.RegexpParser(pattern)
cs = cp.parse(sent)
print(cs)
```

```
(S
  European/JJ
  authorities/NNS
  fined/VBD
  Google/NNP
  (NP a/DT record/NN)
  $/$
  5.1/CD
  billion/CD
  on/IN
  Wednesday/NNP
  for/IN
  abusing/VBG
  its/PRP$
  (NP power/NN)
  in/IN
  (NP the/DT mobile/JJ phone/NN)
  (NP market/NN)
  and/CC
  ordered/VBD
  (NP the/DT company/NN)
  to/TO
  alter/VB
  its/PRP$
  practices/NNS)
```

Figure 2

The output can be read as a tree or a hierarchy with S as the first level, denoting sentence. we can also display it graphically.

Figure 3

IOB tags have become the standard way to represent chunk structures in files, and we will also be using this format.

```
from nltk.chunk import conlltags2tree, tree2conlltags
from pprint import pprintiob_tagged = tree2conlltags(cs)
pprint(iob_tagged)
```

```
[('European', 'JJ', 'O'),
 ('authorities', 'NNS', 'O'),
 ('fined', 'VBD', 'O'),
 ('Google', 'NNP', 'O'),
 ('a', 'DT', 'B-NP'),
 ('record', 'NN', 'I-NP'),
 ('$', '$', 'O'),
 ('5.1', 'CD', 'O'),
 ('billion', 'CD', 'O'),
 ('on', 'IN', 'O'),
 ('Wednesday', 'NNP', 'O'),
 ('for', 'IN', 'O'),
 ('abusing', 'VBG', 'O'),
 ('its', 'PRP$', 'O'),
 ('power', 'NN', 'B-NP'),
 ('in', 'IN', 'O'),
 ('the', 'DT', 'B-NP'),
 ('mobile', 'JJ', 'I-NP'),
 ('phone', 'NN', 'I-NP'),
 ('market', 'NN', 'B-NP'),
 ('and', 'CC', 'O'),
 ('ordered', 'VBD', 'O'),
 ('the', 'DT', 'B-NP'),
 ('company', 'NN', 'I-NP'),
 ('to', 'TO', 'O'),
 ('alter', 'VB', 'O'),
 ('its', 'PRP$', 'O'),
 ('practices', 'NNS', 'O')]
```

Figure 4

In this representation, there is one token per line, each with its part-of-speech tag and its named entity tag. Based on this training corpus, we can construct a tagger that can be used to label new sentences; and use the nltk.chunk.conlltags2tree() function to convert the tag sequences into a chunk tree.

With the function nltk.ne_chunk(), we can recognize named entities using a classifier, the classifier adds category labels such as PERSON, ORGANIZATION, and GPE.

```
ne_tree = ne_chunk(pos_tag(word_tokenize(ex)))
print(ne_tree)
```

```
(S
  (GPE European/JJ)
  authorities/NNS
  fined/VBD
  (PERSON Google/NNP)
  a/DT
  record/NN
  $/$
  5.1/CD
  billion/CD
  on/IN
  Wednesday/NNP
  for/IN
  abusing/VBG
  its/PRP$
  power/NN
  in/IN
  the/DT
  mobile/JJ
  phone/NN
  market/NN
  and/CC
  ordered/VBD
  the/DT
  company/NN
  to/TO
  alter/VB
  its/PRP$
  practices/NNS)
```

Figure 5

Google is recognized as a person. It’s quite disappointing, don’t you think so?

SpaCy

SpaCy's named entity recognition (https://spacy.io/api/annotation#section-named-entities) has been trained on the OntoNotes 5 (https://catalog.ldc.upenn.edu/ldc2013t19) corpus and it supports the following entity types:

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type.

Figure 6 (Source: SpaCy)

Entity

```
import spacy
from spacy import displacy
from collections import Counter
import en_core_web_sm
nlp = en_core_web_sm.load()
```

We are using the same sentence, "European authorities fined Google a record $5.1 billion on Wednesday for abusing its power in the mobile phone market and ordered the company to alter its practices."

One of the nice things about Spacy is that we only need to apply nlp once, the entire background pipeline will return the objects.

```
doc = nlp('European authorities fined Google a record $5.1 billion on Wednesday for abusing its power in
 the mobile phone market and ordered the company to alter its practices')
pprint([(X.text, X.label_) for X in doc.ents])
```

```
[('European', 'NORP'),
 ('Google', 'ORG'),
 ('$5.1 billion', 'MONEY'),
 ('Wednesday', 'DATE')]
```

Figure 7

European is NORD (nationalities or religious or political groups), Google is an organization, $5.1 billion is monetary value and Wednesday is a date object. They are all correct.

Token

During the above example, we were working on entity level, in the following example, we are demonstrating token-level entity annotation using the BILUO tagging scheme to describe the entity boundaries.

TAG	DESCRIPTION
B EGIN	The first token of a multi-token entity.
I N	An inner token of a multi-token entity.
L AST	The final token of a multi-token entity.
U NIT	A single-token entity.
O UT	A non-entity token.

Figure 8 (Source: SpaCy)

```
pprint([(X, X.ent_iob_, X.ent_type_) for X in doc])
```

```
[(European, 'B', 'NORP'),
 (authorities, 'O', ''),
 (fined, 'O', ''),
 (Google, 'B', 'ORG'),
 (a, 'O', ''),
 (record, 'O', ''),
 ($, 'B', 'MONEY'),
 (5.1, 'I', 'MONEY'),
 (billion, 'I', 'MONEY'),
 (on, 'O', ''),
 (Wednesday, 'B', 'DATE'),
 (for, 'O', ''),
 (abusing, 'O', ''),
 (its, 'O', ''),
 (power, 'O', ''),
 (in, 'O', ''),
 (the, 'O', ''),
 (mobile, 'O', ''),
 (phone, 'O', ''),
 (market, 'O', ''),
 (and, 'O', ''),
 (ordered, 'O', ''),
 (the, 'O', ''),
 (company, 'O', ''),
 (to, 'O', ''),
 (alter, 'O', ''),
 (its, 'O', ''),
 (practices, 'O', '')]
```

Figure 9

`"B"` means the token begins an entity, `"I"` means it is inside an entity, `"O"` means it is outside an entity, and `""` means no entity tag is set.

Extracting named entity from an article

Now let's get serious with SpaCy and extracting named entities from a New York Times article, — "**F.B.I. Agent Peter Strzok, Who Criticized Trump in Texts, Is Fired (https://www.nytimes.com/2018/08/13/us/politics/peter-strzok-fired-fbi.html?hp&action=click&pgtype=Homepage&clickSource=story-heading&module=first-column-region&region=top-news&WT.nav=top-news)** ."

```
from bs4 import BeautifulSoup
import requests
import redef url_to_string(url):
    res = requests.get(url)
    html = res.text
```

```
    soup = BeautifulSoup(html, 'html5lib')
    for script in soup(["script", "style", 'aside']):
        script.extract()
    return " ".join(re.split(r'[\n\t]+', soup.get_text()))ny_bb = url_to_string('https://www.nytimes.com/
2018/08/13/us/politics/peter-strzok-fired-fbi.html?hp&action=click&pgtype=Homepage&clickSource=story-head
ing&module=first-column-region&region=top-news&WT.nav=top-news')
article = nlp(ny_bb)
len(article.ents)
```

188

There are 188 entities in the article and they are represented as 10 unique labels:

```
labels = [x.label_ for x in article.ents]
Counter(labels)
```

```
Counter({'CARDINAL': 5,
         'DATE': 29,
         'EVENT': 1,
         'GPE': 35,
         'LOC': 1,
         'NORP': 5,
         'ORDINAL': 1,
         'ORG': 26,
         'PERSON': 84,
         'WORK_OF_ART': 1})
```

Figure 10

The following are three most frequent tokens.

```
items = [x.text for x in article.ents]
Counter(items).most_common(3)
```

```
[('Strzok', 32), ('F.B.I.', 17), ('Trump', 10)]
```

Figure 11

Let’s randomly select one sentence to learn more.

```
sentences = [x for x in article.sents]
print(sentences[20])
```

```
Firing Mr. Strzok, however, removes a favorite target of Mr. Trump from the ranks of the F.B.I. and gives t
F.B.I. director, Christopher A. Wray, a chance to move beyond the president’s ire.
```

Figure 12

Let's run `displacy.render (https://spacy.io/api/top-level#displacy.render)` (https://spacy.io/api/top-level#displacy.render) to generate the raw markup.

```
displacy.render(nlp(str(sentences[20])), jupyter=True, style='ent')
```

Firing Mr. Strzok PERSON , however, removes a favorite target of Mr. Trump PERSON from the ranks of the F.B.I. GPE ar
PERSON and the F.B.I. GPE director, Christopher A. Wray PERSON , a chance to move beyond the president's ire.

Figure 13

One miss-classification here is F.B.I. It is hard, isn't it?

Using spaCy's built-in displaCy visualizer (https://spacy.io/usage/visualizers), here's what the above sentence and its dependencies look like:

```
displacy.render(nlp(str(sentences[20])), style='dep', jupyter = True, options = {'distance': 120})
```

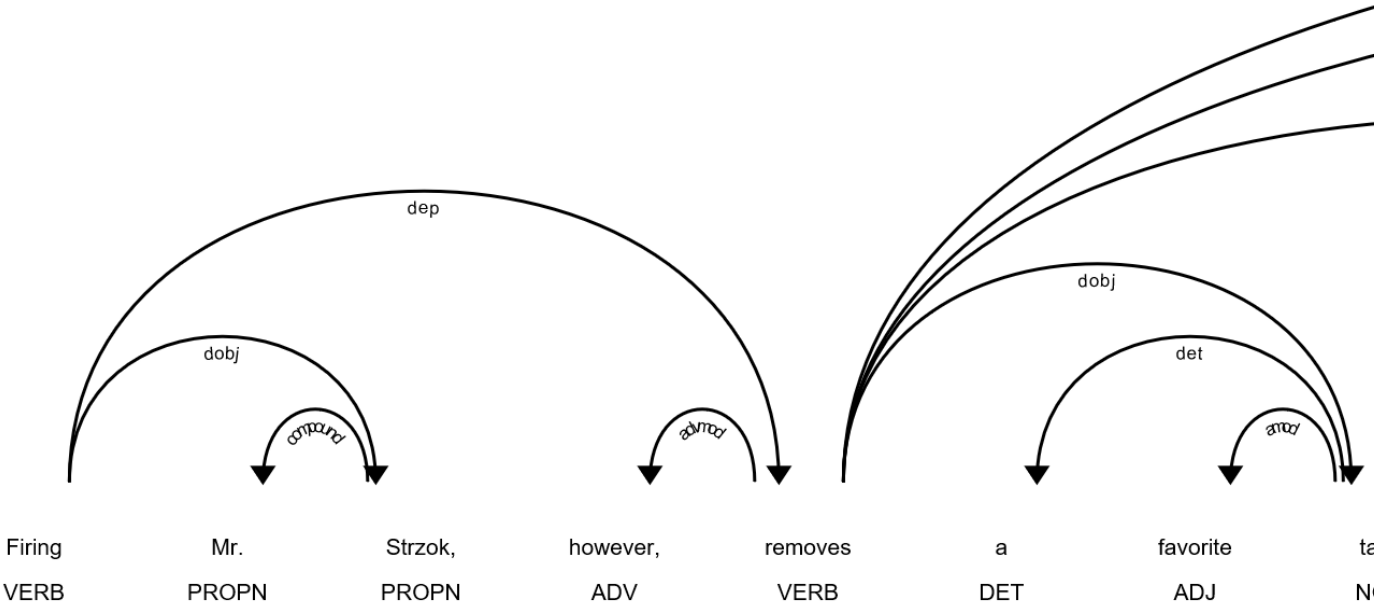

Figure 14

Next, we verbatim, extract part-of-speech and lemmatize this sentence.

```
[(x.orth_,x.pos_, x.lemma_) for x in [y
                                      for y
                                      in nlp(str(sentences[20]))
                                      if not y.is_stop and y.pos_ != 'PUNCT']]
```

```
[('Firing', 'VERB', 'fire'),
 ('Mr.', 'PROPN', 'mr.'),
 ('Strzok', 'PROPN', 'strzok'),
 ('removes', 'VERB', 'remove'),
 ('favorite', 'ADJ', 'favorite'),
 ('target', 'NOUN', 'target'),
 ('Mr.', 'PROPN', 'mr.'),
 ('Trump', 'PROPN', 'trump'),
 ('ranks', 'NOUN', 'rank'),
 ('F.B.I.', 'PROPN', 'f.b.i.'),
 ('gives', 'VERB', 'give'),
 ('Mr.', 'PROPN', 'mr.'),
 ('Bowdich', 'PROPN', 'bowdich'),
 ('F.B.I.', 'PROPN', 'f.b.i.'),
 ('director', 'NOUN', 'director'),
 ('Christopher', 'PROPN', 'christopher'),
 ('A.', 'PROPN', 'a.'),
 ('Wray', 'PROPN', 'wray'),
 ('chance', 'NOUN', 'chance'),
 ('president', 'NOUN', 'president'),
 ('’s', 'PART', '’s'),
 ('ire', 'NOUN', 'ire')]
```

Figure 15

```
dict([(str(x), x.label_) for x in nlp(str(sentences[20])).ents])
```

```
{'Bowdich': 'PERSON',
 'Christopher A. Wray': 'PERSON',
 'F.B.I.': 'GPE',
 'Strzok': 'PERSON',
 'Trump': 'PERSON'}
```

Figure 16

Named entity extraction are correct except “F.B.I”.

```
print([(x, x.ent_iob_, x.ent_type_) for x in sentences[20]])
```

```
[(Firing, 'O', ''), (Mr., 'O', ''), (Strzok, 'B', 'PERSON'), (,, 'O', ''), (however, 'O', ''), (,, 'O', ''),
'', (a, 'O', ''), (favorite, 'O', ''), (target, 'O', ''), (of, 'O', ''), (Mr., 'O', ''), (Trump, 'B', 'PERSON
'', (the, 'O', ''), (ranks, 'O', ''), (of, 'O', ''), (the, 'O', ''), (F.B.I., 'B', 'GPE'), (and, 'O', ''), (
(Mr., 'O', ''), (Bowdich, 'B', 'PERSON'), (and, 'O', ''), (the, 'O', ''), (F.B.I., 'B', 'GPE'), (director, 'O
'', (Christopher, 'B', 'PERSON'), (A., 'I', 'PERSON'), (Wray, 'I', 'PERSON'), (,, 'O', ''), (a, 'O', ''), (ch
(to, 'O', ''), (move, 'O', ''), (beyond, 'O', ''), (the, 'O', ''), (president, 'O', ''), ('s, 'O', ''), (ire,
'O', '')]
```

Figure 17

Finally, we visualize the entity of the entire article.

F.B.I. Agent Peter Strzok PERSON , Who Criticized Trump PERSON in Texts, Is Fired GPE - The New York Times ORG Se contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON . 13 CARDINAL , 2018WASHINGTON CARDINAL — PERSON , the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messa oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzo said Monday DATE .Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON , who rose DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months inquiry.Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal ema F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON , who was remo DATE from the staff of the special counsel, Robert S. Mueller III PERSON . The president has repeatedly denounced Mr. Strzok P Twitter EVENT , and on Monday DATE expressed satisfaction that he had been sacked.Mr. Trump's ORG victory traces back when Mr. Strzok PERSON 's conduct was laid out in a wide-ranging inspector general's report on how the F.B.I. GPE handled the Hillary Clinton's PERSON emails in the run-up to the 2016 DATE election. The report was critical of Mr. Strzok PERSON 's conc

Figure 18