Chapter 4

Named-Entity Recognition (NER)

Outline

- Introduction to Named Entity Recognition
- Introduction to SpaCy

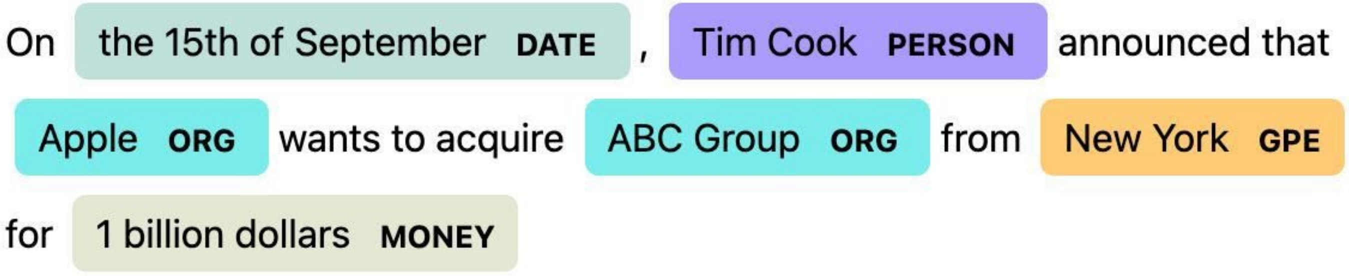
Introduction to Named Entity Recognition

What is Named Entity Recognition?

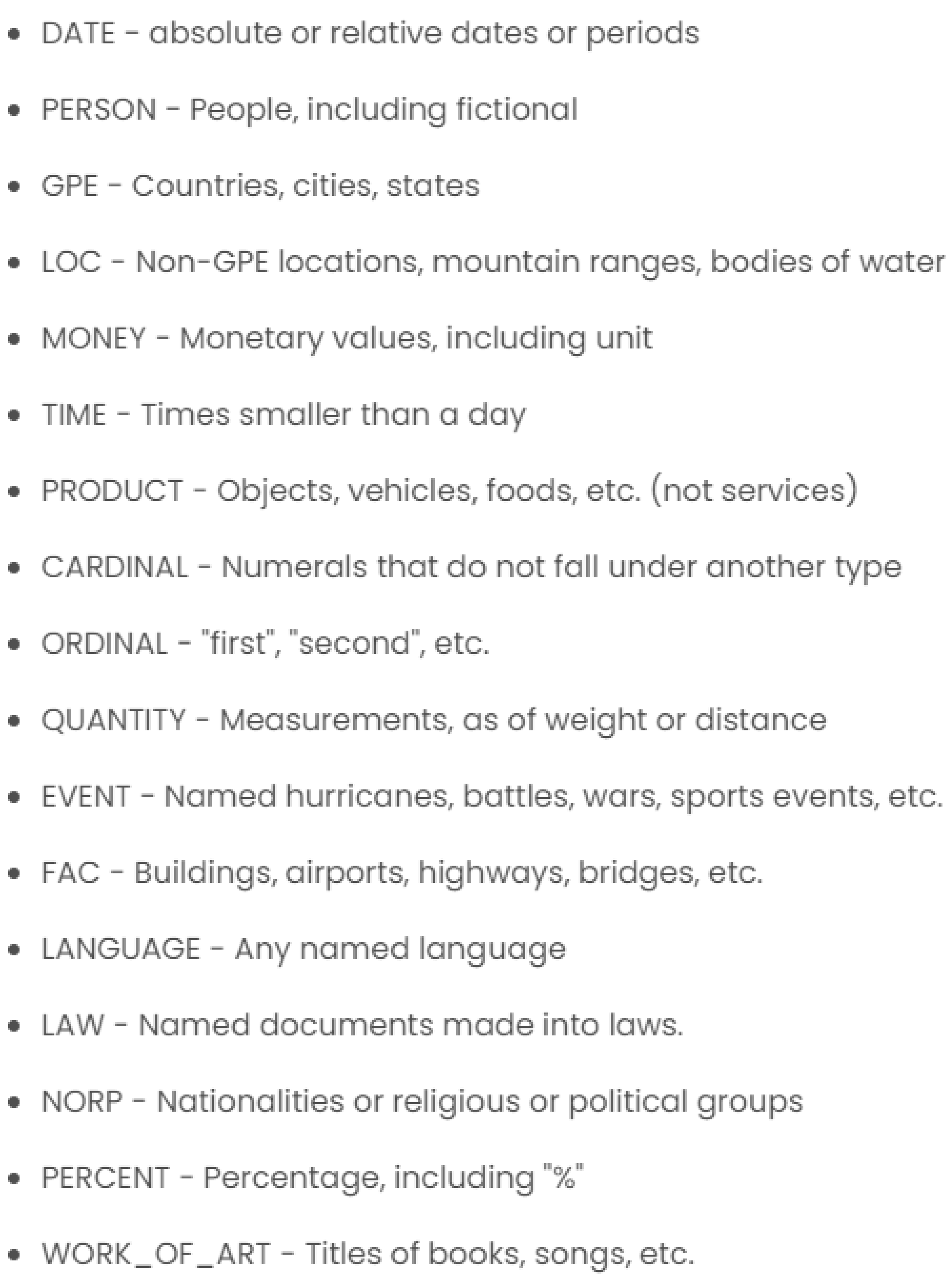
- NER is a natural language processing task used to identify important named entities in the text.
    - such as people, places, organizations, dates, states, and other categories depending on the used libraries and notation.
- NER can be used alongside topic identification
- NER can be used to
    - determine important items in a text and
    - answer basic NLU questions such as who? what? when? and where?

Example of NER

- The text has been highlighted for different types of named entities.



On the 15th of September DATE , Tim Cook PERSON announced that Apple ORG wants to acquire ABC Group ORG from New York GPE for 1 billion dollars MONEY



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

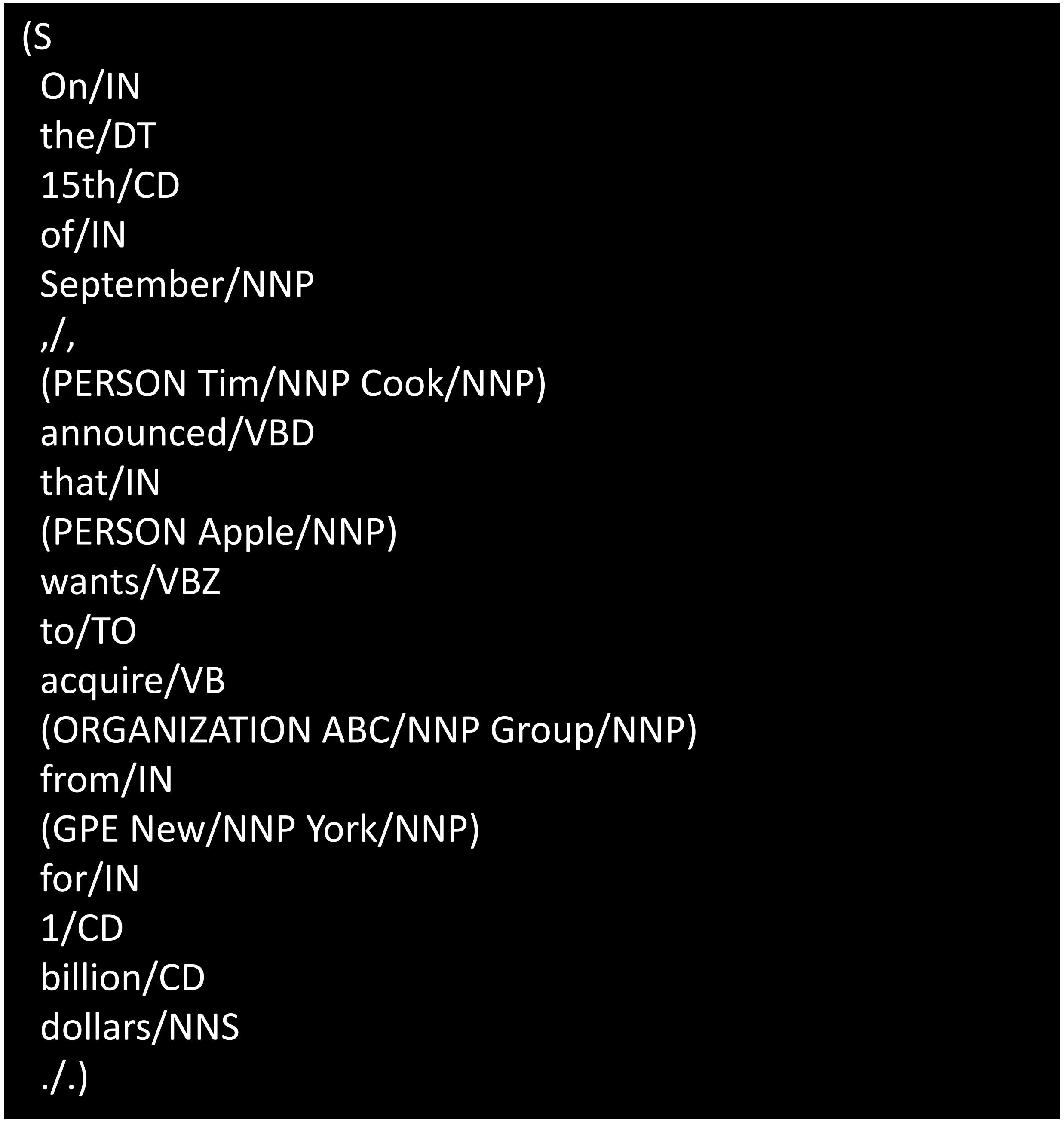
Using nltk for Named Entity Recognition

```
import nltk
#nltk.download('averaged_perceptron_tagger')
sentence = 'On the 15th of September, Tim Cook announced that Apple wants
to acquire ABC Group from New York for 1 billion dollars.'
tokenized_sent = nltk.word_tokenize(sentence)
tagged_sent = nltk.pos_tag(tokenized_sent)
```

```
[('On', 'IN'), ('the', 'DT'), ('15th', 'CD'), ('of', 'IN'), ('September', 'NNP'), (',', ','), ('Tim', 'NNP'), ('Cook',
'NNP'), ('announced', 'VBD'), ('that', 'IN'), ('Apple', 'NNP'), ('wants', 'VBZ'), ('to', 'TO'), ('acquire', 'VB'),
('ABC', 'NNP'), ('Group', 'NNP'), ('from', 'IN'), ('New', 'NNP'), ('York', 'NNP'), ('for', 'IN'), ('1', 'CD'),
('billion', 'CD'), ('dollars', 'NNS'), ('.', '.')]
```

NLTK POS Tags Examples : https://www.guru99.com/pos-tagging-chunking-nltk.html

```
print(nltk.ne_chunk(tagged_sent))
```



```
(S
  On/IN
  the/DT
  15th/CD
  of/IN
  September/NNP
  ,/,
  (PERSON Tim/NNP Cook/NNP)
  announced/VBD
  that/IN
  (PERSON Apple/NNP)
  wants/VBZ
  to/TO
  acquire/VB
  (ORGANIZATION ABC/NNP Group/NNP)
  from/IN
  (GPE New/NNP York/NNP)
  for/IN
  1/CD
  billion/CD
  dollars/NNS
  ./.)
```

- NLTK provides a classifier that has already been trained to recognize named entities, accessed with **nltk.ne_chunk**().

Let’s practice!

NER with nltk

- Uses **nltk** to find the named entities in this article.
- Import **nltk, sent_tokenize** and **word_tokenize** from **nltk.tokenize**.
- Read TXT file (**tim_cook.txt**) save to **article**
- Tokenize **article** into sentences.

```
# Tokenize the article into sentences: sentences
sentences = ____._______________(article)
```

- Tokenize each sentence in **sentences** into words using a list comprehension.

```
# Tokenize each sentence into words: token_sentences
token_sentences = [____._______________(sent) for sent in sentences]
```

NER with nltk

- Inside a list comprehension, tag each tokenized sentence into parts of speech using **nltk.pos_tag()**.

```
# Tag each tokenized sentence into parts of speech: pos_sentences
pos_sentences = [____.__________(sent) for sent in token_sentences]
```

- Chunk each tagged sentence into named-entity chunks using **nltk.ne_chunk_sents()**. Along with **pos_sentences**, specify the additional keyword argument **binary=True**.

```
# Create the named entity chunks: chunked_sentences
chunked_sentences = ____.______________(pos_sentences, binary=True)
```

NER with nltk

- Loop over each sentence and each chunk, and test whether it is a named-entity chunk by testing if it has the attribute **label**, and if the **chunk.label()** is equal to **"NE"**. If so, print that chunk.

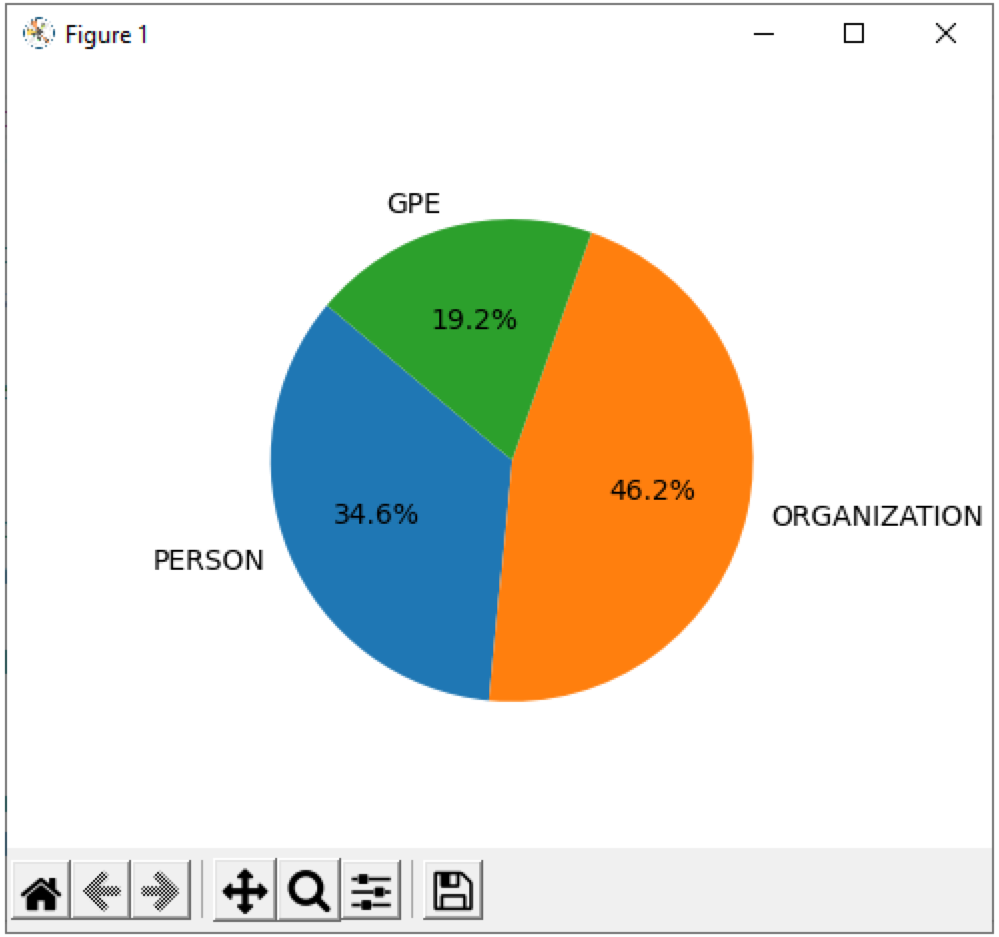
```
# Test for stems of the tree with 'NE' tags
for sent in chunked_sentences:
    for chunk in sent:
        if hasattr(chunk, "label") and chunk.label() == "NE":
            print(chunk)
```

```
(NE Tim/NNP Cook/NNP)
(NE Apple/NNP)
(NE CEO/NNP)
(NE Tim/NNP)
(NE Apple/NNP)
(NE Tim/NNP)
(NE Corporate/NNP)
(NE Compaq/NNP)
(NE Compaq/NNP)
(NE Tim/NNP)
(NE Reseller/NNP Division/NNP)
(NE Intelligent/NNP Electronics/NNP)
(NE Tim/NNP)
(NE IBM/NNP)
(NE North/JJ)
(NE American/JJ)
(NE North/NNP)
(NE Latin/NNP America/NNP)
(NE Tim/NNP)
(NE Duke/NNP University/NNP)
(NE Fuqua/NNP Scholar/NNP)
(NE Science/NNP)
(NE Industrial/NNP Engineering/NNP)
(NE Auburn/NNP University/NNP)
```

Charting practice

- Use some extracted named entities and their groupings from articles to chart the diversity of named entity types in the articles.
- Create a **defaultdict** called **ner_categories**, with the default type set to **int**.

```
# Create the defaultdict: ner_categories
ner_categories = defaultdict(int)
```



Charting practice

- Fill up the dictionary with values for each of the keys. Remember, the keys will represent the **label()**.
  - In the outer **for** loop, iterate over **chunked_sentences**, using **sent** as your iterator variable.
  - In the inner **for** loop, iterate over **sent**. If the condition is true, increment the value of each key by 1.

```
# Create the nested for loop
for sent in chunked_sentences:
    for chunk in sent:
        if hasattr(chunk, 'label'):
            ner_categories[chunk.label()] += 1
```

Charting practice

- For the pie chart labels, create a list called **labels** from the keys of **ner_categories**, which can be accessed using **.keys()**.

```
# Create a list from the dictionary keys for the chart labels: labels
labels = list(________________._____())
```

- Use a list comprehension to create a list called **values**, using the **.get()** method on **ner_categories** to compute the values of each label **v**.

```
# Create a list of the values: values
values = [________________.____(__) for v in labels]
```

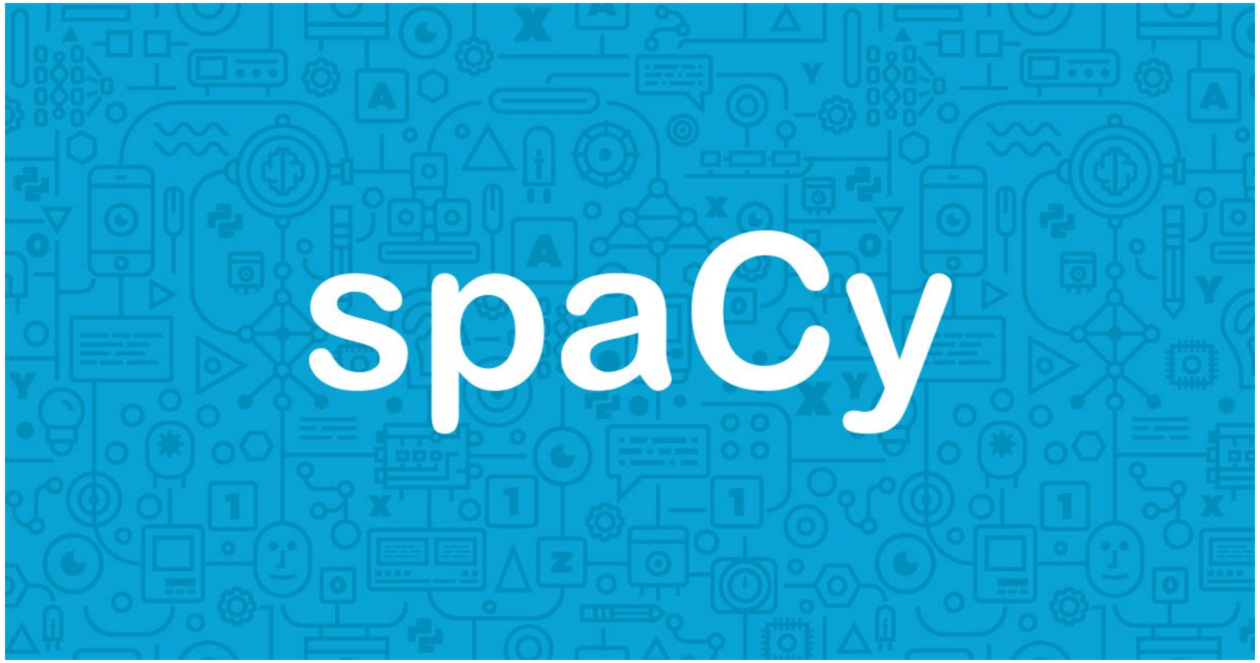
Charting practice

- Use **plt.pie()** to create a pie chart for each of the NER categories. Along with **values** and **labels=labels**, pass the extra keyword arguments **autopct='%1.1f%%'** and **startangle=140** to add percentages to the chart and rotate the initial start angle.

```
# Create the pie chart
plt.pie(values, labels=labels, autopct='%1.1f%%', startangle=140)
```

- Display the pie chart.

```
# Display the chart
plt.________()
```

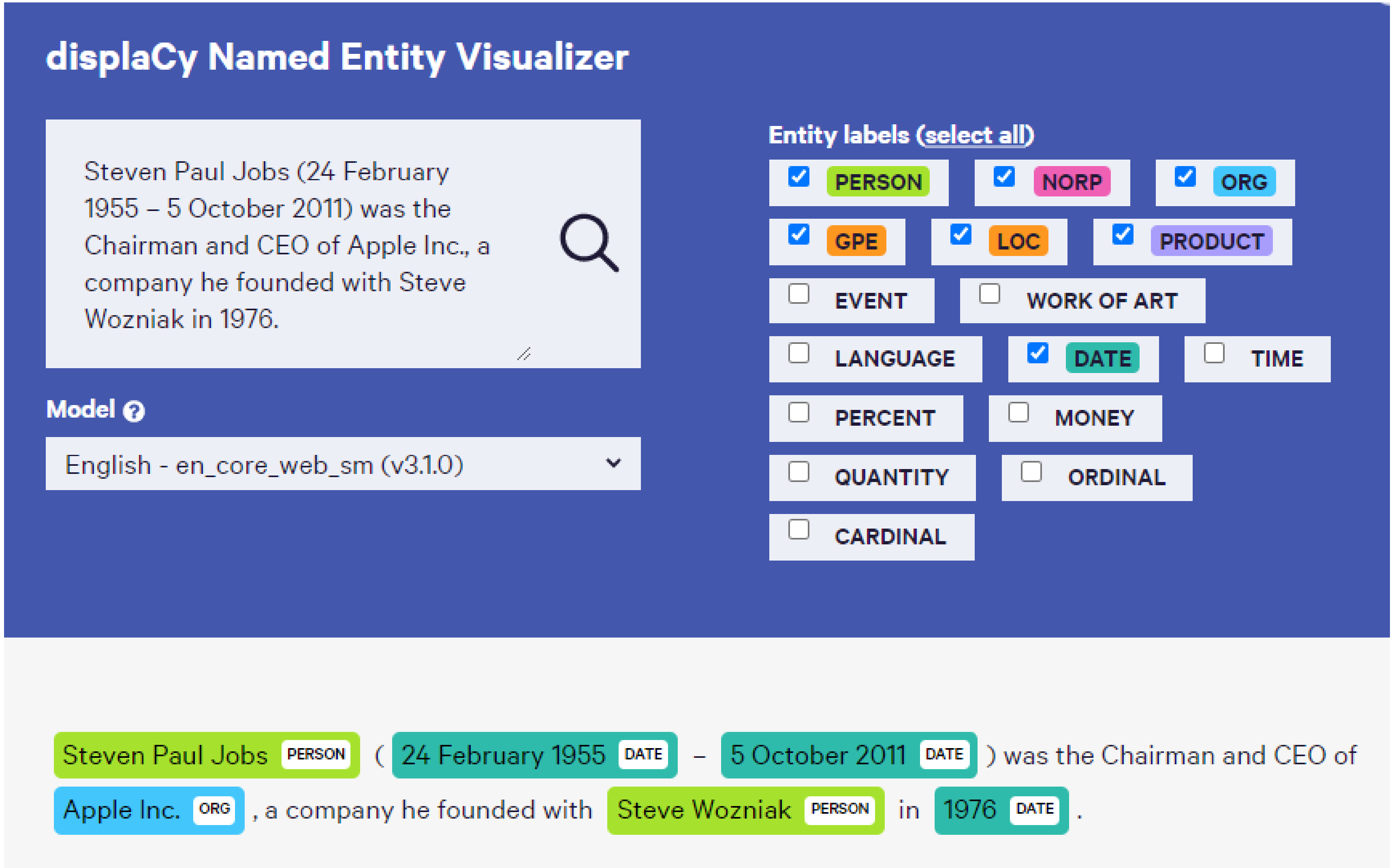


Introduction to SpaCy

What is SpaCy?

- SpaCy is a great NLP library similar to Gensim, but with different implementations.
- Focus on creating NLP pipelines to generate models and corpora.
- SpaCy is open-source and has extra libraries and tools.
  - Displacy - a visualization tool for viewing parse trees.

Displacy entity recognition visualizer



displaCy Named Entity Visualizer

Steven Paul Jobs (24 February 1955 – 5 October 2011) was the Chairman and CEO of Apple Inc., a company he founded with Steve Wozniak in 1976.

Model

English - en_core_web_sm (v3.1.0)

Entity labels (select all)

☑ PERSON ☑ NORP ☑ ORG
☑ GPE ☑ LOC ☑ PRODUCT
□ EVENT □ WORK OF ART
□ LANGUAGE ☑ DATE □ TIME
□ PERCENT □ MONEY
□ QUANTITY □ ORDINAL
□ CARDINAL

Steven Paul Jobs PERSON (24 February 1955 DATE – 5 October 2011 DATE) was the Chairman and CEO of Apple Inc. ORG , a company he founded with Steve Wozniak PERSON in 1976 DATE .

- https://explosion.ai/demos/displacy-ent

SpaCy NER

- To start using spacy for Named entity recognition,
- To install it and download all the appropriate pre-trained word vectors.
- You can also train vectors yourself and load them; but the pretrained ones let us get started immediately.

- Why use SpaCy for NER?
  - Spacy comes with informal language corpora, allowing you to more easily find entities in documents like Tweets and chat messages.

SpaCy NER

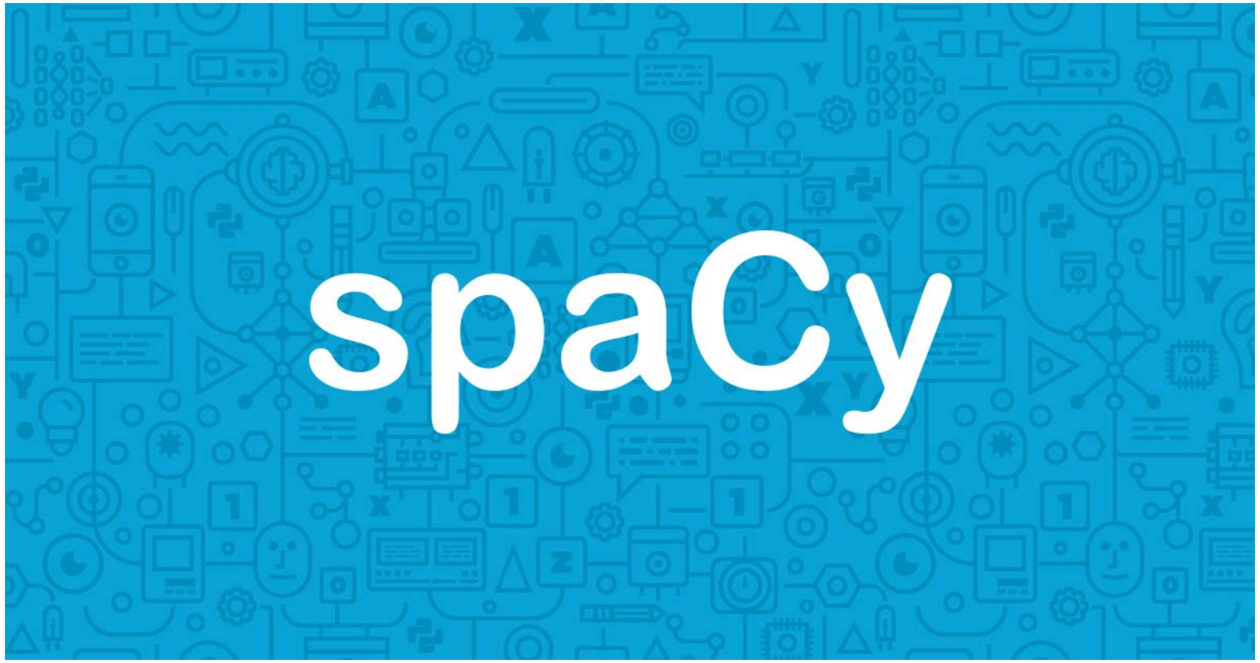
- pip install spacy
- python -m spacy download en_core_web_sm

```
import spacy
nlp = spacy.load('en_core_web_sm')

doc = nlp("Berlin is the capital of Germany")
doc.ents

print(doc.ents[0],doc.ents[0].label_)
```

```
Berlin GPE
```

Note:: "GPE“ is geo-political entities such as city, state/province, and country.

“en_core_web_sm” is a small English pipeline trained on written web text (blogs, news, comments), that includes vocabulary, syntax and entities.



Let’s practice!

Comparing NLTK with spaCy NER

- Import **spacy**.

```
import spacy
```

- Load the **'en_core_web_sm'** model using **spacy.load()**.

```
# Instantiate the English model: nlp
nlp = _______._______(_________________)
```

Comparing NLTK with spaCy NER

- Create a **spacy** document object by passing **article** into **nlp()**.

```
# Create a new document: doc
doc = ______(article)
```

- Using **ent** as your iterator variable, iterate over the entities of **doc** and print out the labels (**ent.label_**) and text (**ent.text**).

```
# Print all of the found entities and their labels
for _____ in doc.ents:
    print(____._______, _____._______)
```

```
PERSON Tim Cook
PERSON Tim Cook
ORG Apple
DATE August 2011
PERSON Tim
CARDINAL ™
PRODUCT ™
ORG Macintosh
ORG Apple
PERSON Tim
ORG Corporate Materials for Compaq
ORG Compaq
PERSON Tim
ORG the Reseller Division at Intelligent Electronics
PERSON Tim
DATE 12 years
ORG IBM
ORG North American Fulfillment
ORG Personal Computer Company
GPE North
LOC Latin America
PERSON Tim
ORG Duke University
ORG Industrial Engineering
```

Questions

Reference: https://app.datacamp.com/learn