Abdishakur · Follow

Nov 16, 2020 · 5 min read ★ · Listen

How to Extract Locations from Text with Natural Language Processing

Quick and Simple Location Extraction with Spacy and Python.

Named Entity Recognition — Image created with Canva by the Author.

Natural Language Processing (NLP) is one of the most researched fields in AI. We generate a lot of unstructured data through our daily activities, and therefore, the popularity of NLP has increased over the past five to ten years.

For spatial data science, NLP can help in understanding which locations are present in a

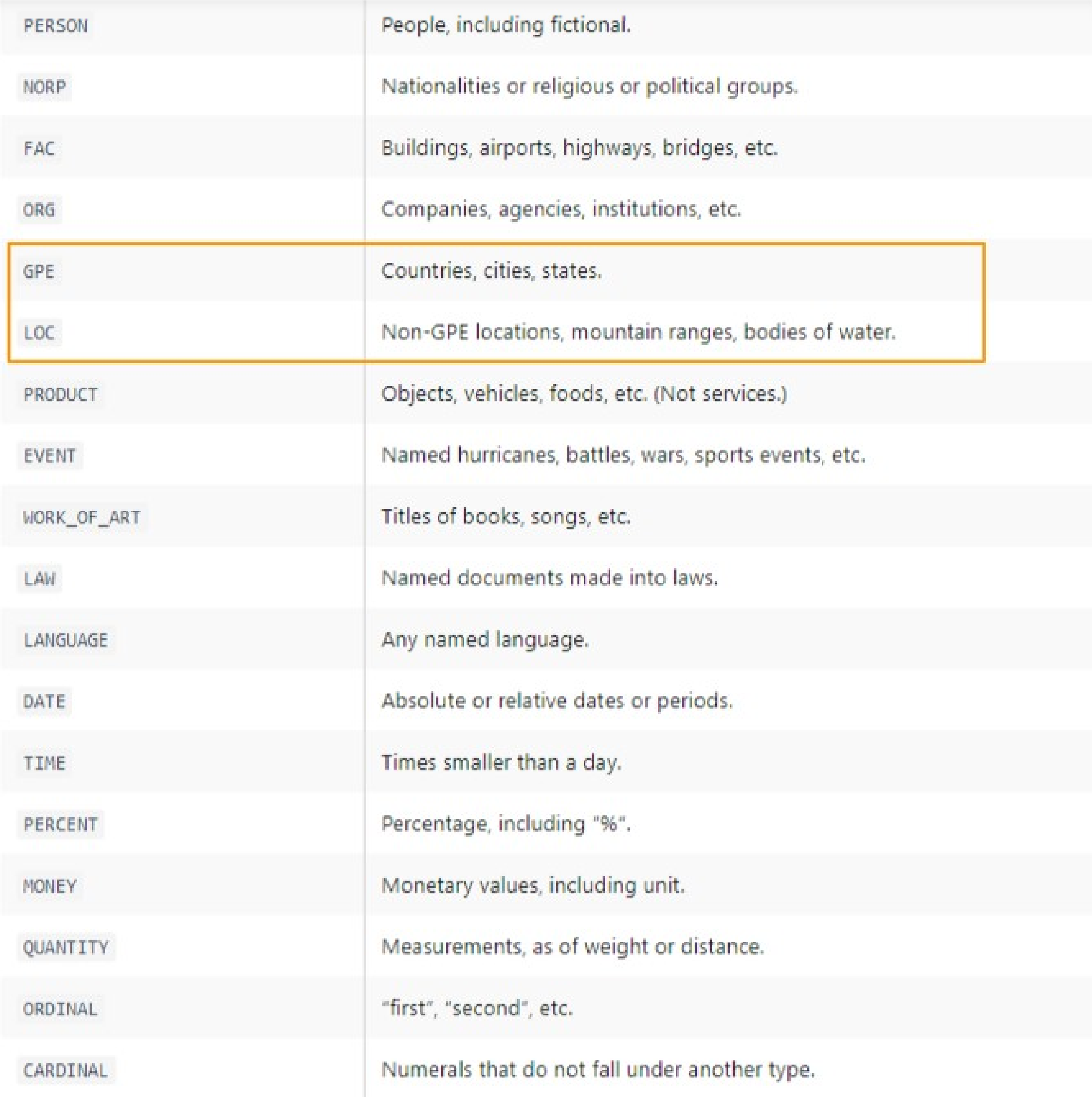
This tutorial will show you how to extract location features from text documents using Named Entity Recognition (NER) in spaCy. The first section of the article shows you how to obtain locations from one single sentence and the features available with spaCy NER. In the last section of the article, we will extract places from a body of text files and then geocode the addresses to display them on a map.

Named Entity Recognition (NER)

> ***Named-entity recognition*** *(NER) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations,* ***locations****, medical codes, time expressions, quantities, monetary values, percentages, etc. — Source.*

Named Entity Recognition can be accomplished with different methods, and a simple regular expression would be a good option. However, some other techniques use statistical models or neural networks to extract the entities. Spacy is one of the most used Python libraries for Natural language processing.

With Spacy's Named Entity Recognition, you can extract two types of location features: Geopolitical Entity(GPE)and Non-GPE locations.



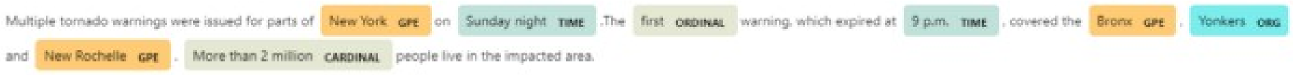
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, 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.

Named Entity Recognition types — Spacy.

Let us see a simple example of using Spacy to extract location data from a news article.

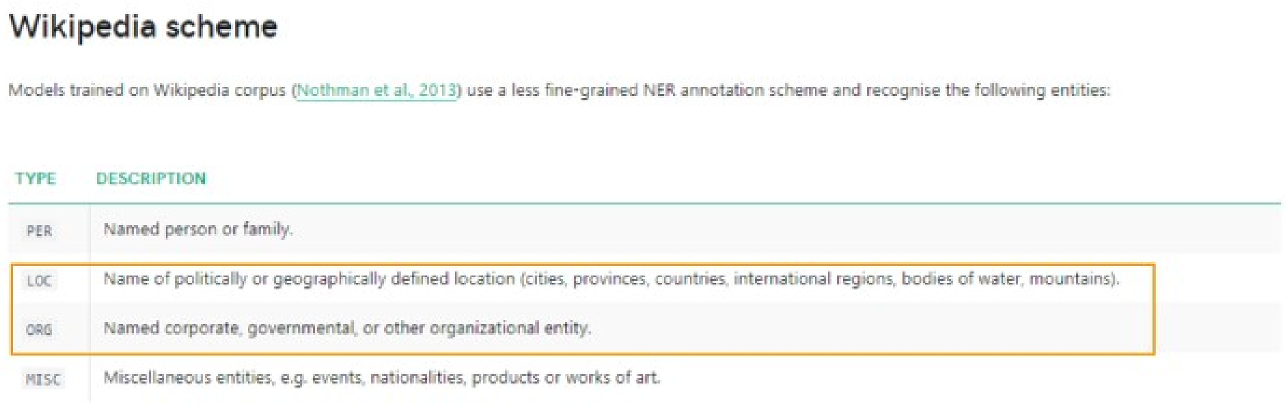
```
import spacy
from spacy import displacy

nlp = spacy.load(‘en_core_web_sm’)

# Text with nlp
doc = nlp(" Multiple tornado warnings were issued for parts of New
York on Sunday night.The first warning, which expired at 9 p.m.,
```

The output is an annotated text where SPacy highlights Entities. Here we see it extracted New York, Bronx, Yonkers, and New Rochelle as GPE.



Multiple tornado warnings were issued for parts of New York GPE on Sunday night TIME .The first ORDINAL warning, which expired at 9 p.m. TIME , covered the Bronx GPE , Yonkers ORG and New Rochelle GPE . More than 2 million CARDINAL people live in the impacted area.

For less fine-grained NER annotations, you can use spaCy's Wikipedia Scheme.



Wikipedia scheme

Models trained on Wikipedia corpus (Nothman et al., 2013) use a less fine-grained NER annotation scheme and recognise the following entities:

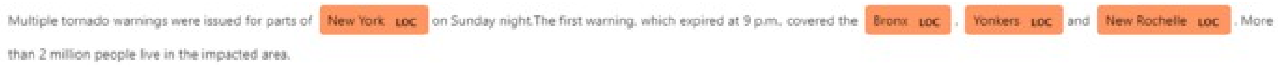
TYPE	DESCRIPTION
PER	Named person or family.
LOC	Name of politically or geographically defined location (cities, provinces, countries, international regions, bodies of water, mountains).
ORG	Named corporate, governmental, or other organizational entity.
MISC	Miscellaneous entities, e.g. events, nationalities, products or works of art.

Spacy Wikipedia NER scheme

Let us see how this scheme performs with the same example above.

```
nlp_wk = spacy.load('xx_ent_wiki_sm')

doc = nlp_wk(" Multiple tornado warnings were issued for parts of New
York on Sunday night.The first warning, which expired at 9 p.m.,
covered the Bronx, Yonkers and New Rochelle. More than 2 million
people live in the impacted area.")

displacy.render(doc, style="ent")
```

And the output for the Wikipedia scheme extracts all those places as locations.



Multiple tornado warnings were issued for parts of New York LOC on Sunday night.The first warning, which expired at 9 p.m., covered the Bronx LOC , Yonkers LOC and New Rochelle LOC . More than 2 million people live in the impacted area.

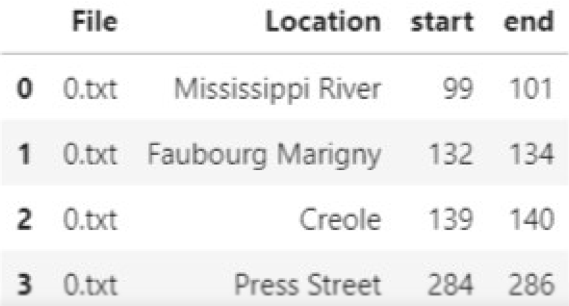
simple. We need to loop through the folder and read the text files to pass it to a Name Entity Recognition(NER) algorithm.

I use here the Geowebnews dataset. It has anotations, but we will not use the annotations, since we are only interested in extracting locations quickly. Ofcourse, this is not comprehensive but gives you an overall pattern of the locations available in the text documents.

Let us a create a simple loop to read the files and run Spacy's NER algorithm. Note here that we will use the Wikipedia schema since it offers more location context.

```
import os
import pandas as pd
DIR = ‘GeoWebNews/text’

locations = []

for fn in os.listdir(DIR):
 with open(f’{DIR}/{fn}’,encoding=’utf-8') as f:
 doc = nlp_wk(f.read())
 locations.extend([[fn, ent.text, ent.start, ent.end] for ent in
doc.ents if ent.label_ in [‘LOC’]])

df = pd.DataFrame(locations, columns=[‘File’, ‘Location’,
‘start’,’end’])
df.head()
```

When we run the NER process with each file, we also append all locations present in the documents into a list. Then we create a pandas Dataframe from the list. In total, the locations found in these documents are 2155 and here is the first few rows of the dataset.



	File	Location	start	end
0	0.txt	Mississippi River	99	101
1	0.txt	Faubourg Marigny	132	134
2	0.txt	Creole	139	140
3	0.txt	Press Street	284	286

Now this only extracts the location names and therefore we need to geocode those locations. We can use the Openstreetmap free geocoding service to get the coordinates of this places.

```
import pandas as pd
import geopandas as gpd
import geopy
import matplotlib.pyplot as plt
from geopy.extra.rate_limiter import RateLimiter

locator = geopy.geocoders.Nominatim(user_agent=”mygeocoder”)
geocode = RateLimiter(locator.geocode, min_delay_seconds=1)

df[“address”] = df[“Location”].apply(geocode)
```

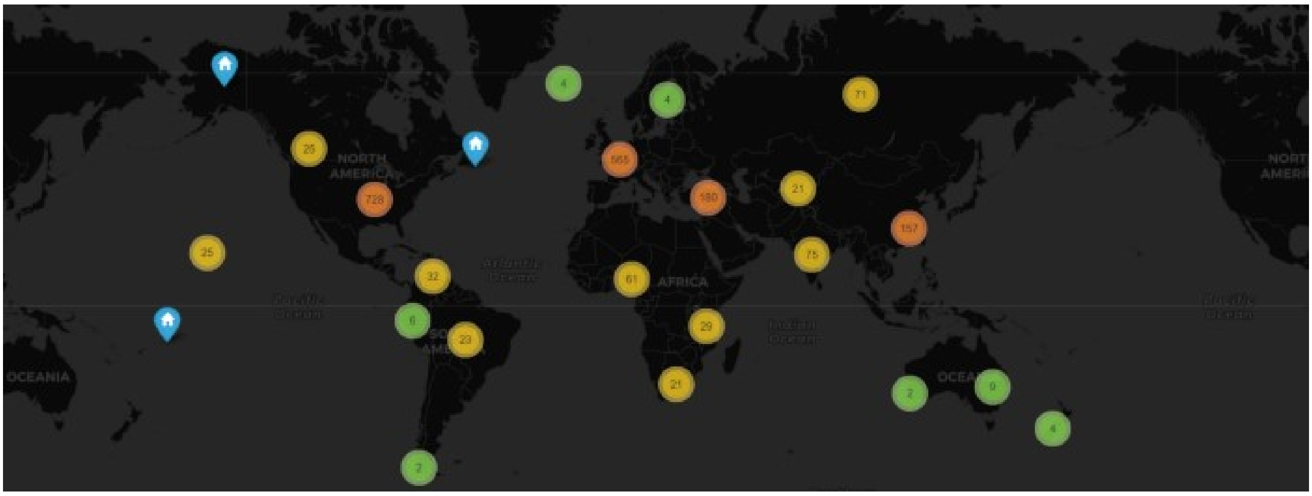
After we geocde the location names, we end up in a dataframe where we have also latitude,longitude, alitude and other geographic features in a new column. Now we will split those strings into new columns and remove all rows without a latitude value.

```
df[‘coordinates’] = df[‘address’].apply(lambda loc: tuple(loc.point)
if loc else None)

df[[‘latitude’, ‘longitude’, ‘altitude’]] =
pd.DataFrame(df[‘coordinates’].tolist(), index=df.index)

df.latitude.isnull().sum()

df = df[pd.notnull(df[“latitude”])]
```

We can then plot these geocoded values into a map. I use here Folium but feel free to use any other library of your choice.

```
import folium
from folium.plugins import FastMarkerCluster

folium_map = folium.Map(location=[59.338315,18.089960],
 zoom_start=2,
 tiles=’CartoDB dark_matter’)
```

And as you can see from the map below, the documents have locations of almost all places around the world.



NER Extracted locations display with Folium Map — Image by the author.

Final Thoughts

In this tutorial, we have started with simple sentence and extracted locations present in the text. Next, we have used GeoWebNews dataset stored in text documents and create a loop that goes through each document and extracts locations using Spacy. Finaly, we have used free Geocoding services from OpenStreetMap to geocode the locations and we end up with a map of all locations present in the documents.

Geoparsing with Python and Natural Language Processing

Extracting place names and assigning coordinates — Tutorial

towardsdatascience.com