Report on How to Model NER

NER is the first step towards information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. The raw and structured text is taken and named entities are classified into persons, organizations, places, money, time, etc. Named entities are identified and segmented into various predefined classes.

NER systems are developed with various linguistic approaches, as well as statistical and machine learning methods. NER has many applications for project or business purposes.

The NER model first identifies an entity and then categorizes the entity into the most suitable class. Some of the common types of Named Entities will be:

1. Organisations: NASA, CERN, ISRO, etc
2. Places: Mumbai, New York, Kolkata.
3. Money: 1 Billion Dollars, 50 Great Britain Pounds.
4. Date: 15th August 2020
5. Person: Elon Musk, Richard Feynman, Subhas Chandra Bose.

An important thing about NER models is that their ability to understand Named Entities depends on the data they have been trained on.

NER can be used for content classification, the various Named Entities of a text can be collected, and based on that data, the content themes can be understood. In academics and research, NER can be used to retrieve data and information faster from a wide variety of textual information. NER helps a lot in the case of information extraction from huge text datasets.

The Algorithm:

An entity can be a single word or even a group of words that refer to the same category.

Step 1: Detect an entity. This can be a word or a group of words that refer to the same category.

'*Bond*' ➡️ an entity that consists of a single word

'*James Bond*' ➡️ an entity that consists of two words, but they are referring to the same category.

After detecting an entity, the next step in a NER task is to categorize the detected entity. The categories of an entity can be anything depending on our use case.

Step 2: categorize the detected entity. The categories can be anything depending on our use case.

Person: Elon Musk, Richard Feynman, James Bond, Bond

NER Models

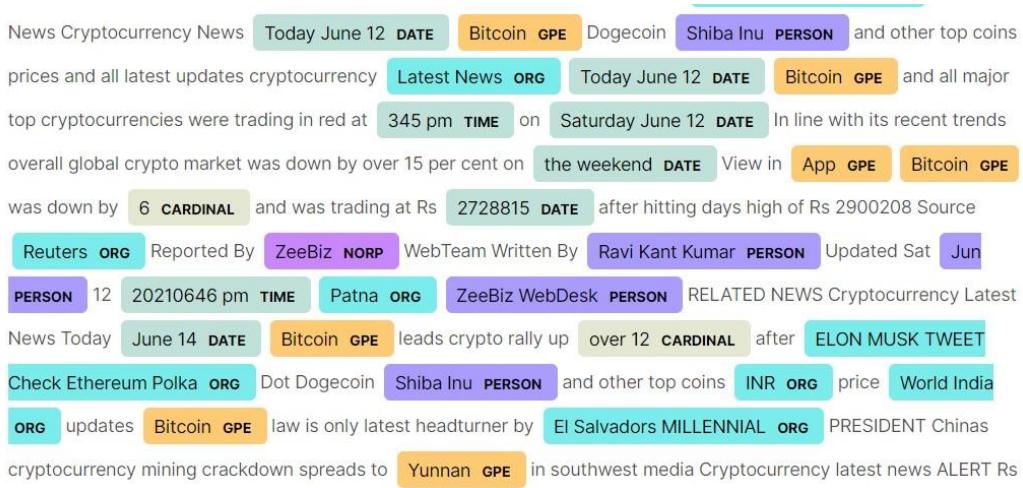
- Using spaCy
- Using ATLAS.ti 22
- bert-base-NER
- Stanford Named Entity Recognizer

NER using spaCy

Processing raw text intelligently is difficult: most words are rare, and it's common for words that look completely different to mean almost the same thing. That's exactly what spaCy is designed to do: you put in raw text and get back a Doc object, that comes with a variety of annotations.

spaCy is an open-source NLP library that can be used for various tasks. It has built-in methods for Named Entity Recognition. It has a fast statistical entity recognition system. spaCy can be used very easily for NER tasks. spaCy features an extremely fast statistical entity recognition system, that assigns labels to contiguous spans of tokens. The default trained pipelines can identify a variety of named and numeric entities, including companies, locations, organizations, and products. You can add arbitrary classes to the entity recognition system, and update the model with new examples.

The **displaCy** visualizer lets you explore an entity recognition model's behavior interactively. If you're training a model, it's very useful to run the visualization yourself. To help you do that, spaCy comes with a visualization module. You can pass a Doc or a list of Doc objects to displaCy and run `displacy.serve` to run the web server, or `displacy.render` to generate the raw markup. Using spaCy's built-in displaCy visualizer, here's what our example sentence and its dependencies look like:



Morphology: Morphological features are stored in the `MorphAnalysis` under `Token.morph`, which allows you to access individual morphological features. Inflectional morphology is the process by which a root form of a word is modified by adding prefixes or suffixes that specify its grammatical function but do not change its parts of speech. We say that a **lemma** (root form) is **inflected** (modified/combined) with one or more **morphological features** to create a surface form.

Example:

CONTEXT	**SURFACE**	**LEMMA**	**POS**	**MORPHOLOGICAL FEATURES**
I was reading the paper	reading	read	VERB	`VerbForm=Ger`

Statistical morphology: spaCy's statistical `Morphologizer` component assigns the morphological features and coarse-grained part-of-speech tags as `Token.morph` and `Token.pos`.

Rule-based morphology: For languages with relatively simple morphological systems like English, spaCy can assign morphological features through a rule-based approach, which uses the **token text** and **fine-grained part-of-speech tags** to produce coarse-grained part-of-speech tags and morphological features.

- The part-of-speech tagger assigns each token a fine-grained part-of-speech tag. In the API, these tags are known as `Token.tag`. They express the part of speech (e.g., verb) and some amount of morphological information, e.g., that the verb is past tense (e.g., `VBD` for a past tense verb in the Penn Treebank).
- For words whose coarse-grained POS is not set by a prior process, a mapping table maps the fine-grained tags to coarse-grained POS tags and morphological features.

In [1]:
```
import spacy
from spacy import displacy

NER = spacy.load("en_core_web_sm")
```

In [2]:
```
raw_text="The Indian Space Research Organisation or is the national space agency of India, headquartered in Bengaluru. It ope
rates under Department of Space which is directly overseen by the Prime Minister of India while Chairman of ISRO acts as exec
utive of DOS as well."
```

In [3]:
```
text1= NER(raw_text)
```

In [4]:
```
for word in text1.ents:
    print(word.text,word.label_)
```

```
The Indian Space Research Organisation ORG
the national space agency ORG
India GPE
Bengaluru GPE
Department of Space ORG
India GPE
ISRO ORG
DOS ORG
```

In [5]:
```
spacy.explain("ORG")
```

Out[5]:
```
'Companies, agencies, institutions, etc.'
```

In [6]:
```
spacy.explain("GPE")
```

Out[6]:
```
'Countries, cities, states'
```

In [7]:
```
displacy.render(text1,style="ent",jupyter=True)
```

The Indian Space Research Organisation ORG or is the national space agency ORG of India GPE, headquartered in Bengaluru GPE. It operates under Department of Space ORG which is directly overseen by the Prime Minister of India GPE while Chairman of ISRO ORG acts as executive of DOS ORG as well.

```
#use of Beautiful Soup for web scraping
from bs4 import BeautifulSoup
import requests
import re
URL="https://www.zeebiz.com/markets/currency/
news-cryptocurrency-news-today-june-12-bitcoin-dogecoin-shiba-inu-and-oth
er-top-coins-prices-and-all-latest-updates-158490"
html_content = requests.get(URL).text
soup = BeautifulSoup(html_content, "lxml")

#body content
body=soup.body.text
#use of regex to clean the text
body= body.replace('n', ' ')
body= body.replace('t', ' ')
body= body.replace('r', ' ')
body= body.replace('xa0', ' ')
body=re.sub(r'[^ws]', '', body)
body[1000:1500]

#a visual to display NE directly in the text
text3= NER(body)
displacy.render(text3,style="ent",jupyter=True)


```

NER using ATLAS.ti 22

ATLAS.ti is a powerful workbench for the qualitative analysis of larger bodies of textual, graphical, audio, and video data.

It offers a variety of tools for accomplishing the tasks associated with any systematic approach to unstructured data, i.e., data that cannot be meaningfully analyzed by formal, statistical approaches.

It offers tools to manage, extract, compare, explore, and reassemble meaningful pieces from large amounts of data in creative, flexible, yet systematic ways.

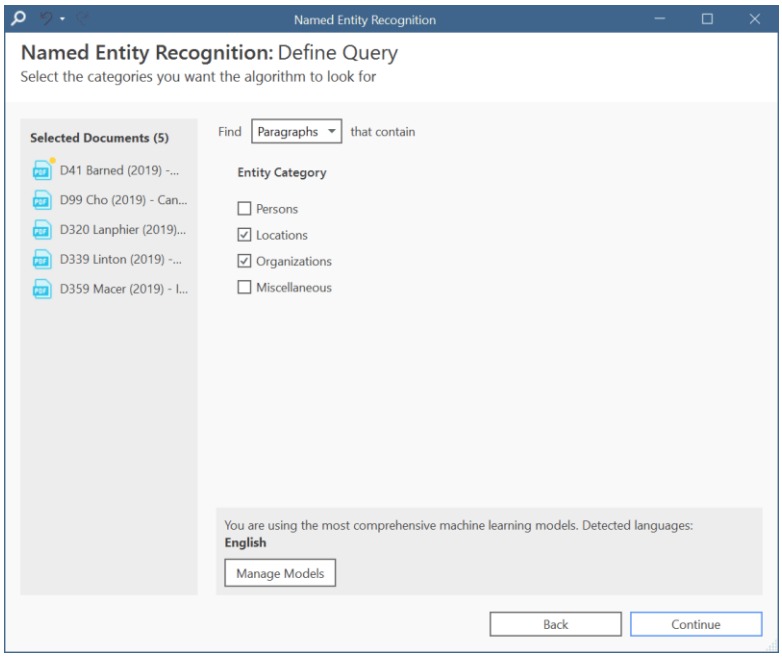
As ATLAS.ti is a tool for qualitative data analysis, the process is not fully automated. Before coding the data, you can review all results, make modifications or decide not to code certain findings.

You can think of it like a special auto-coding procedure, where you as the user do not enter a search term. Instead, ATLAS.ti goes through your data and finds entities for you.

You can select which entities you want to search for:

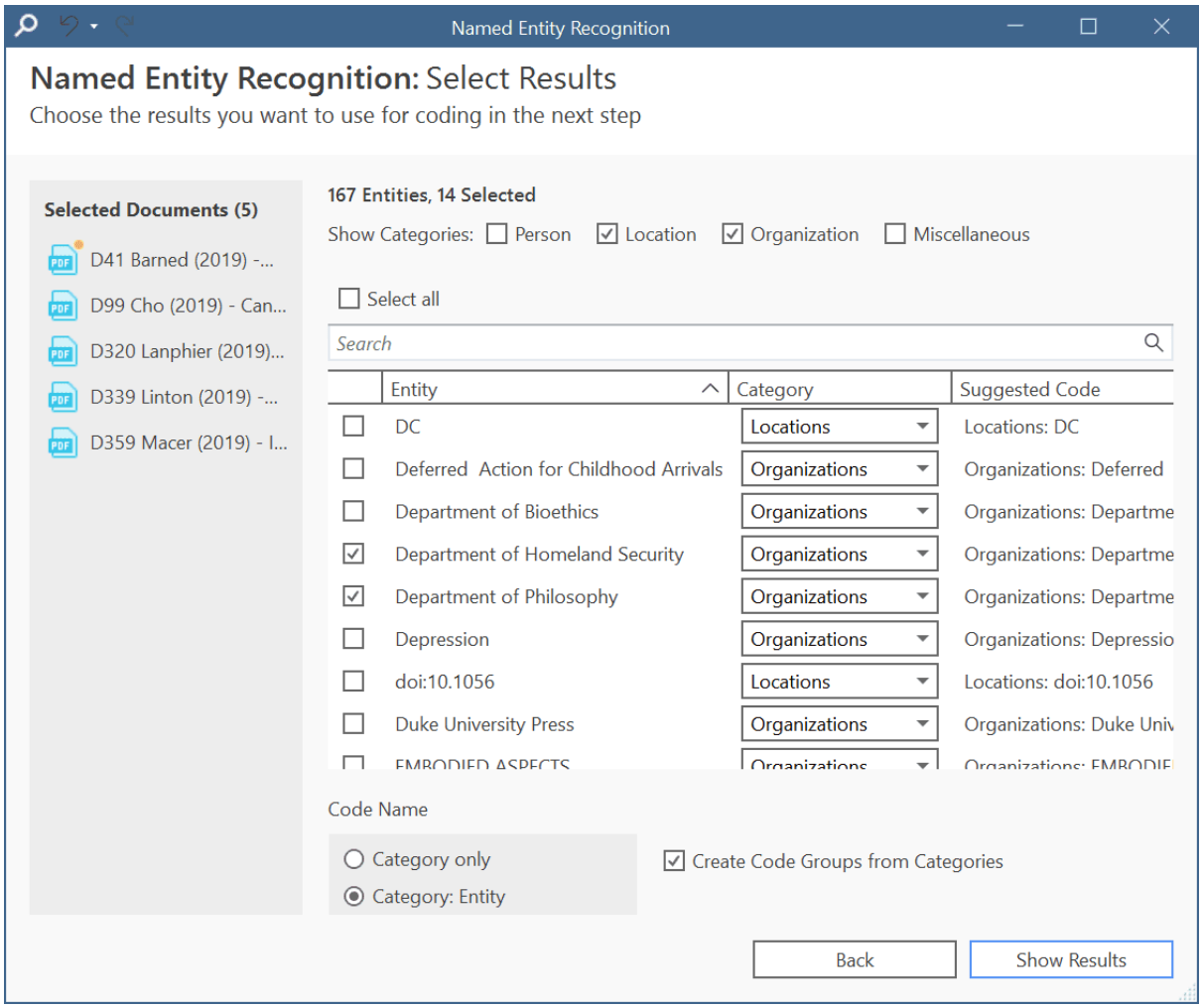
- person
- organization
- location
- miscellaneous (works of art, languages, political parties, events, titles of books, etc.)

After the search is completed, ATLAS.ti shows you what it found, and you can make corrections. In the next step, you can review the results in context and code all results with the suggested codes, or decide for each hit whether to code it or not.



Reviewing Search Results:

If you selected to search for all entity types (persons, location, organization, and miscellaneous), you can review them all together, or just focus on one entity at a time. To do so, deactivate all other entity types.



The Search Engine Behind NER:

ATLAS.ti uses spaCy as its natural language processing engine.

Input data gets processed in a pipeline - one step after the other to improve upon the derived knowledge of the prior step.

The first step is a tokenizer to chunk a given text into meaningful parts and replace ellipses etc. For example, the sentence:

“I should’ve known (didn't back then).” will get tokenized to: “I should have known (did not back then).
“

The tokenizer uses a vocabulary for each language to assign a vector to a word. This vector was pre-learned by using a corpus and represents a kind of similarity in usage in the used corpus.

bert-base-NER

BERT - Bidirectional Encoder Representations from Transformers [2018] It is a language model that's a transformer-based machine learning technique for natural language processing pre-training developed by Google.

When it comes to dealing with NLP problems, BERT oftentimes comes up as a machine learning model that we can count on in terms of its performance. The fact that it's been pre-trained on more than 2,500M words and its bidirectional nature to learn information from a sequence of words makes it a powerful model to use.

BERT is an open-source machine learning framework for natural language processing (NLP). Bert-base-NER is a fine-tuned BERT model that is ready to use for NER and achieves state-of-the-art performance for the same. It is trained to recognize four types of entities:

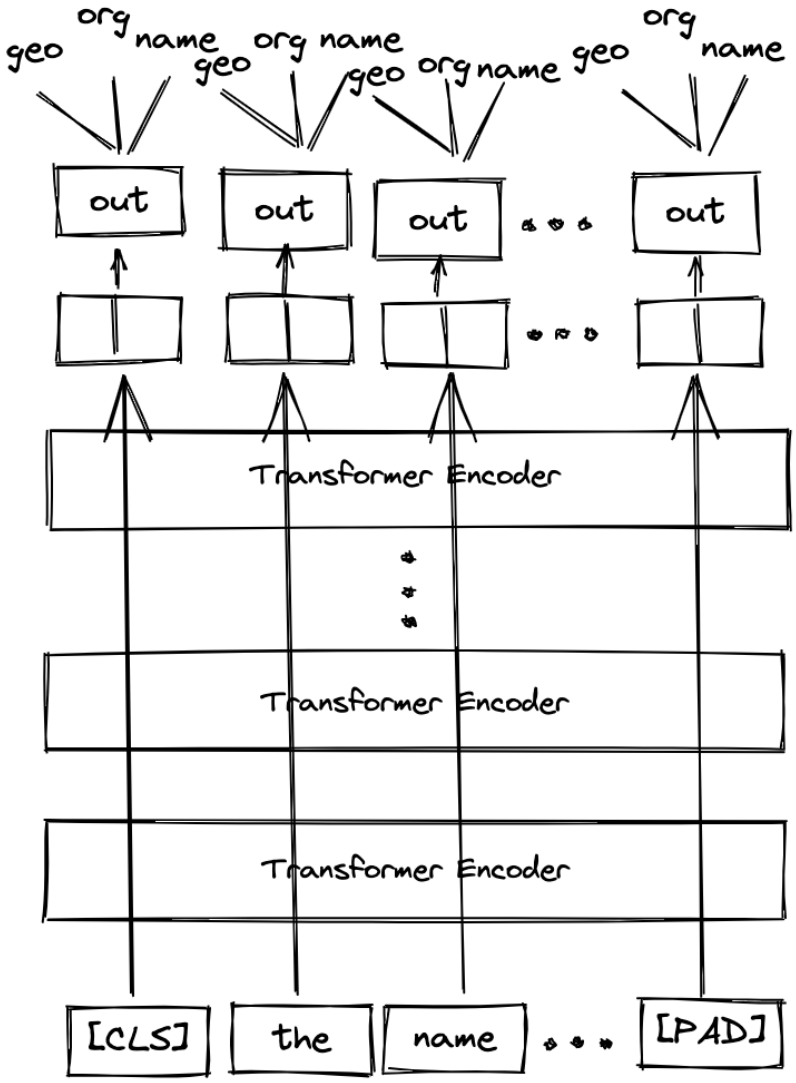
- location (LOC),
- organizations (ORG),
- person (PER) and
- Miscellaneous (MISC)

This model was fine-tuned on the English version of the standard CoNLL-2003 Named Entity Recognition dataset. CoNLL-2003 English Dataset has been derived from the Reuters corpus which consists of Reuters news stories.

For a text classification task, we only use the embedding vector output from the special [CLS] token. For NER tasks, we need to use the embedding vector output from all of the tokens.

In total, there are 9 entity categories, which are:

- `geo` for geographical entity
- `org` for the organization entity
- `per` for person entity
- `gpe` for geopolitical entity
- `tim` for time indicator entity
- `art` for artifact entity
- `eve` for event entity
- `nat` for natural phenomenon entity
- is assigned if a word doesn't belong to any entity.



How to use this model?

You can use this model with the Transformers pipeline for NER.

A Transformer pipeline describes the flow of data from origin systems to destination systems and defines how to transform the data along the way

Training Data: English version of the standard CoNLL-2003 Named Entity Recognition dataset

Training procedure: This model was trained on a single NVIDIA V100 GPU with recommended hyperparameters from the original BERT paper which trained & evaluated the model on the CoNLL-2003 NER task.

```python
from transformers import AutoTokenizer, AutoModelForTokenClassification
from transformers import pipeline

tokenizer = AutoTokenizer.from_pretrained("dslim/bert-base-NER")
model = AutoModelForTokenClassification.from_pretrained("dslim/bert-base-NER")

nlp = pipeline("ner", model=model, tokenizer=tokenizer)
example = "My name is Wolfgang and I live in Berlin"

ner_results = nlp(example)
print(ner_results)
```

Limitations and bias of bert-base-NER

This model is limited by its training dataset of entity-annotated news articles from a specific period. This may not generalize well for all use cases in different domains.

Furthermore, the model occasionally tags sub-word tokens as entities, and post-processing of results may be necessary to handle those cases.