



ABOUT THE PROJECT

MISSION

Build an entity extractor model to extract the store_number from the transaction_descriptor.

GOAL

Customized NER model is created using NER to find domain specific Keyword



ABOUT THE PROJECT

TOOL

- Spacy is an open-source software python library used in advanced natural language processing and machine learning. It will be used to build information extraction, natural language understanding systems, and to pre-process text for deep learning. It supports deep learning workflow in convolutional neural networks in parts-of-speech tagging, dependency parsing, and named entity recognition.
- 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.



Project steps

1

Data Preparation

2

Model Training

3

Testing

```
1 Mission = tasking;
2 fly(xtraight);
3 stay('TIME' < 5min);
4 circle_overivvariable =
5 % Define inputs
6 inputs.mach = [0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8];
7 inputs.altitude = [0, 10000, 20000, 30000, 40000];
8 alpha = database(db.download.alphasheet);
9 beta = database(db.installed.en);
10 %
11 % Optimize aerodynamic input
12 finned(inputs, alpha, beta)
```

Data Preparation

Data preparation is the main challenging task in NER. In this project , I was given a corpus of 301 records regrouped into 3 datasets; transaction_descriptor, store_number and dataset. The latter contains three sections: train, validate, and test.

To create a fixed training data in SpaCY, certain formatting of the data needs to be done, for it to be processed by the SpaCY NER model. As mentioned on <https://spacy.io/usage/training#basics>.

For the first part, I used the annotation tool : SpaCy NER annotation tool.

link: <http://agateteam.org/spacynerannotate/>

For the second part, the conversion of Data to .spacy format has to be made.

```
import pandas as pd
import os
from tqdm import tqdm
import spacy
from spacy.tokens import DocBin

nlp = spacy.load("en_core_web_lg") # Load other spaCy model

db = DocBin() # create a DocBin object

for text, annot in tqdm(TRAIN_DATA): # data in previous format
    doc = nlp.make_doc(text) # create doc object from text
    ents = []
    for start, end, label in annot["entities"]:
        span = doc.char_span(start, end, label=label, alignment_mode="contract")
        if span is None:
            print("Skipping entity")
        else:
            ents.append(span)
    doc.ents = ents # Label the text with the ents
    db.add(doc)

os.chdir(r'C:\Users\21263\Documents')
db.to_disk("./train.spacy") # save the docbin object
```

Model Training

The recommended way to train spaCy pipelines is via the spacy train command on the command line. It only needs a single config.cfg configuration file that includes all settings and hyperparameters. After saving the starter config to a file base_config.cfg, you can use the init fill-config command to fill in the remaining defaults. Training configs should always be complete and without hidden defaults, to keep your experiments reproducible.

python -m spacy init fill-config base_config.cfg config.cfg

You can now add your data and run train with your config.

```
C:\WINDOWS\system32\cmd.exe
(Snowflakes) C:\Users\z1263\OneDrive\Documents>python -m spacy init fill-config base_config.cfg config.cfg
Auto-filled config with all values
Saved config
config.cfg
You can now add your data and train your pipeline:
python -m spacy train config.cfg --paths.train ./train.spacy --paths.dev ./dev.spacy

(Snowflakes) C:\Users\z1263\OneDrive\Documents>
(Snowflakes) C:\Users\z1263\OneDrive\Documents>python -m spacy train config.cfg --output ./output --paths.train ./train.spacy --paths.dev ./train.spacy
Saving to output directory: output
Using CPU
===== Initializing pipeline =====
[2022-04-11 21:54:50,549] [INFO] Set up nlp object from config
[2022-04-11 21:54:50,557] [INFO] Pipeline: ['tok2vec', 'ner']
[2022-04-11 21:54:50,561] [INFO] Created vocabulary
[2022-04-11 21:54:50,561] [INFO] Finished initializing nlp object
[2022-04-11 21:54:50,693] [INFO] Initialized pipeline components: ['tok2vec', 'ner']
Initialized pipeline

===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.001
#   #      LOSS TOK2VEC LOSS NER ENTS_F ENTS_P ENTS_R SCORE
----- -----
0    0      0.00     68.83  0.00  0.00  0.00  0.00
0 200      3.94     643.00 100.00 100.00 100.00 1.00
116 400      0.00     100.00 100.00 100.00 100.00 1.00
187 600      0.00     100.00 100.00 100.00 100.00 1.00
287 800      0.00     100.00 100.00 100.00 100.00 1.00
387 1000     41.06    13.86 100.00 100.00 100.00 1.00
501 1200     0.00     0.00 100.00 100.00 100.00 1.00
701 1400     0.00     0.00 100.00 100.00 100.00 1.00
901 1600     0.00     0.00 100.00 100.00 100.00 1.00
1101 1800     0.00     0.00 100.00 100.00 100.00 1.00
Saved pipeline to output directory
output\model-last
```

Model Training

After training my model with the training data, I re-did it with my validation data to validate the model and improve it.

```
import pandas as pd
import os
from tqdm import tqdm
import spacy
from spacy.tokens import DocBin

nlp = spacy.blank("en") # Load a new spacy model
nlp = spacy.load(r"C:\Users\21263\OneDrive\Documents\output\model-best") # Load other spacy model

db = DocBin() # create a DocBin object

for text, annot in tqdm(EVALUATE_DATA): # data in previous format
    doc = nlp.make_doc(text) # create doc object from text
    ents = []
    for start, end, label in annot["entities"]:
        span = doc.char_span(start, end, label=label, alignment_mode="contract")
        if span is None:
            print("Skipping entity")
        else:
            ents.append(span)
    doc.ents = ents # label the text with the ents
    db.add(doc)

os.chdir(r'C:\Users\21263\Documents')
db.to_disk("./evaluate.spacy") # save the docbin object
```

```
C:\Windows\system32\cmd.exe
(snowflakes) C:\Users\21263\OneDrive\Documents>python -m spacy train config.cfg --output ./output1 --paths.train ./evaluate.spacy --paths.dev ./evaluate.spacy
3 Created output directory: output1
3 Saving to output directory: output1
3 Using CPU
=====
===== Initializing pipeline =====
2022-04-11 22:07:36,743 [INFO] Set up nlp object from config
2022-04-11 22:07:36,754 [INFO] Pipeline: ['tok2vec', 'ner']
2022-04-11 22:07:36,762 [INFO] Created vocabulary
2022-04-11 22:07:36,762 [INFO] Finished initializing nlp object
2022-04-11 22:07:36,873 [INFO] Initialized pipeline components: ['tok2vec', 'ner']
3 Initialized pipeline
=====
===== Training pipeline =====
3 Pipeline: ['tok2vec', 'ner']
3 Initial learn rate: 0.001
E   #      LOSS TOK2VEC LOSS NER ENTS_F ENTS_P ENTS_R SCORE
--- -----
0     0       0.00    67.67  0.00   0.00   0.00   0.00
49    200     6.59   619.35 100.00 100.00 100.00 1.00
109   400     0.00   0.00   100.00 100.00 100.00 1.00
176   600     0.00   0.00   100.00 100.00 100.00 1.00
248   800     0.00   0.00   100.00 100.00 100.00 1.00
368  1000     7.60   5.68   100.00 100.00 100.00 1.00
468  1200     0.00   0.00   100.00 100.00 100.00 1.00
544  1400    27.84   7.06   100.00 100.00 100.00 1.00
844  1600    27.16   4.86   100.00 100.00 100.00 1.00
1044 1800    49.93  10.14   100.00 100.00 100.00 1.00
3 Saved pipeline to output directory
output1\model-last
```

Model Testing

The next command in the workflow is the evaluate command which calculate the metrics using the best model.

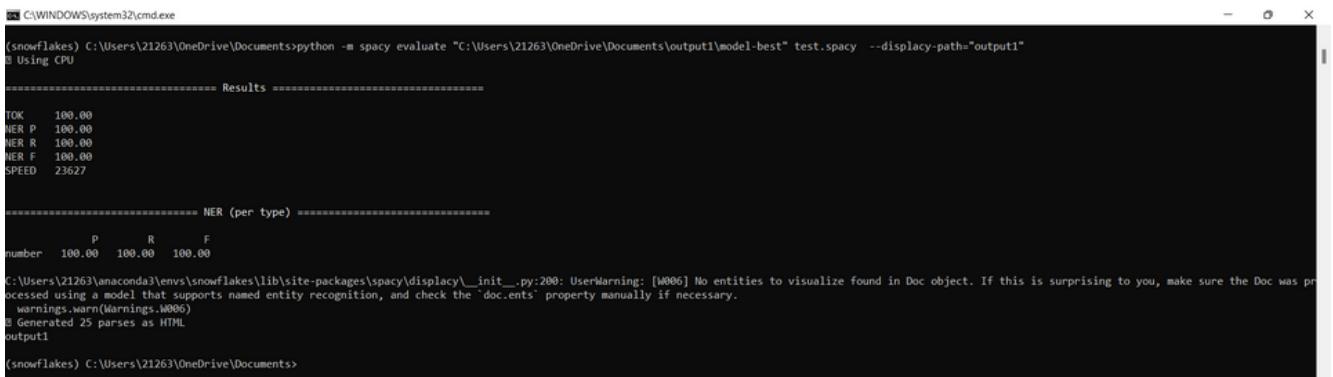
```
import pandas as pd
import os
from tqdm import tqdm
import spacy
from spacy.tokens import DocBin

nlp = spacy.blank("en") # Load a new spacy model
nlp = spacy.load(r"C:\Users\21263\OneDrive\Documents\output1\model-best") # Load other spacy model

db = DocBin() # create a DocBin object

for text, annot in tqdm(EVALUATE_DATA): # data in previous format
    doc = nlp.make_doc(text) # create doc object from text
    ents = []
    for start, end, label in annot["entities"]:
        span = doc.char_span(start, end, label=label, alignment_mode="contract")
        if span is None:
            print("Skipping entity")
        else:
            ents.append(span)
    doc.ents = ents # label the text with the ents
    db.add(doc)

os.chdir(r'C:\Users\21263\Documents')
db.to_disk("./test.spacy") # save the docbin object
```



The screenshot shows a Windows Command Prompt window with the following text:

```
C:\WINDOWS\system32\cmd.exe
(snowflakes) C:\Users\21263\OneDrive\Documents>python -m spacy evaluate "C:\Users\21263\OneDrive\Documents\output1\model-best" test.spacy --displacy-path="output"
Using CPU
=====
Results =====
TOK 100.00
NER P 100.00
NER R 100.00
NER F 100.00
SPEED 23627
=====
NER (per type) =====
number P R F
100.00 100.00 100.00
C:\Users\21263\anaconda3\envs\snowflakes\lib\site-packages\spacy\displacy\_init_.py:200: UserWarning: [W006] No entities to visualize found in Doc object. If this is surprising to you, make sure the Doc was processed using a model that supports named entity recognition, and check the `doc.ents` property manually if necessary.
  warnings.warn(warnings.W006)
Generated 25 parses as HTML
output1
(snowflakes) C:\Users\21263\OneDrive\Documents>
```

- p – denotes the precision value
- r – represents the recall value
- f – the f1-score

Conclusion

● Data Preparation:

Data annotation with agateam
Conversion to .spacy format

● Model Training:

Model Training with training data
Model Training with validation data

● Model Testing:

Model Evaluation