ABOUT THE PROJECT

MISSION

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

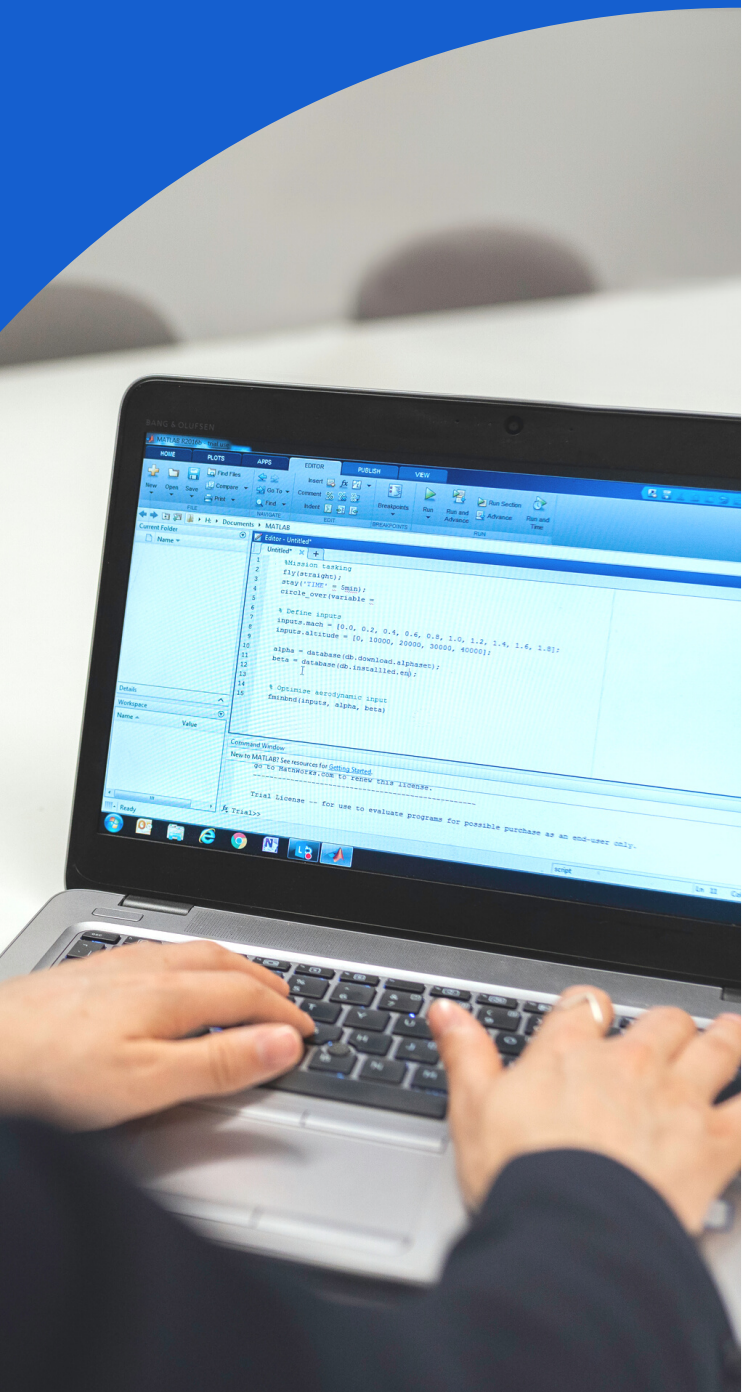
GOAL

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

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.

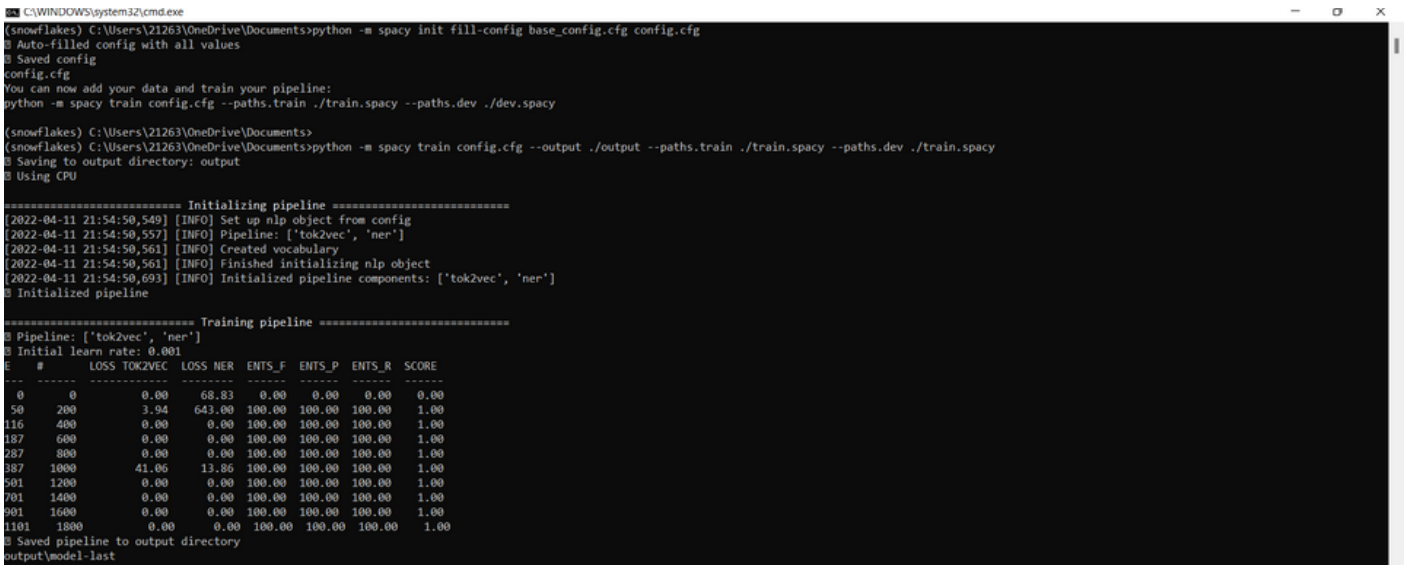
```
import pandas as pd
import os
from tqdm import tqdm
import spacy
from spacy.tokens import DocBin

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"]: # add character indexes
        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

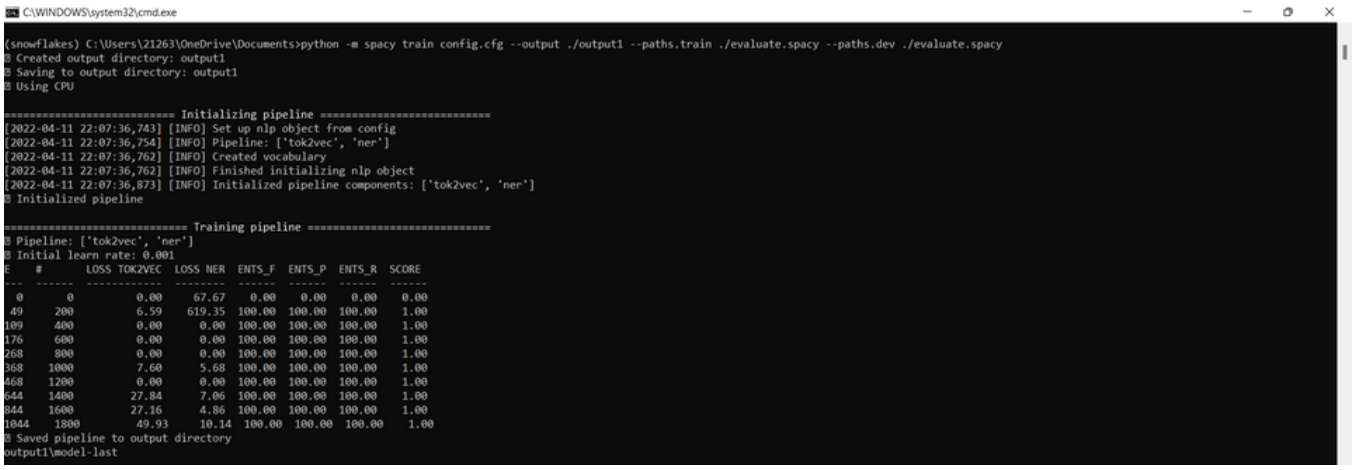
You can now add your data and run train with your config.



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"]: # add character indexes
        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
```



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"]: # add character indexes
        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
```

- 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