Custom-trained Named Entity Recognition (NER) in Lorimer's *Gazetteer* with Spacy

Model setup.

Based on preliminary research, we identified two methods for creating the custom NER model: (i)creating a blank model and (ii) updating one of the existing pretrained models with custom Named Entities. In this project, since we are working with a sample of the Lorimer's Gazetteer data, we try both approaches to leave room for future improvements. We train models with the same set of features and compare the results from two models.

Pretrained model

We first load the pretrained 'en_core_web_sm' pipeline which we imported earlier. The pipeline is based on the English language corpus and 'sm' means small (we can also use the larger size corpora). Although based on English, in contrast to NLTK, this model can be further trained to recognize foreign and transliterated Named Entities.

Then, we retrieve existing ner components and display other components available in the pretrained pipeline - view code and output in figure 9.

```
#Load pretrained model loaded from spacy
nlp_pretrained = en_core_web_sm.load()
print("Loaded model : %s" % nlp_pretrained)
```

```
Loaded model : <spacy.lang.en.English object at 0x7fb04ad4bf90>
```

```
#Retrieve the existing ner component
ner_pretrained=nlp.get_pipe('ner')
#Print exisiting pipeline components
print("Pipeline objects: %s" % nlp_pretrained.pipe_names)
```

```
Pipeline objects: ['tagger', 'parser', 'ner']
```

Figure 9. Load the pretrained Spacy pipeline and retrieve the ner component.

Note that the pretrained 'en_core_web_sm' pipeline includes a tagger, parser and ner components. When we extract Named Entities using the nlp function (discussed further), the text is internally preprocessed in several steps by the components (parsing, tagging, tokenization, lemmatization) - each step returns a doc object to the next component in the pipeline (learn more here). This contrasts with the NLTK based NER system, where processing steps are conducted separately. Figure 10 illustrates the view of a sample pipeline.

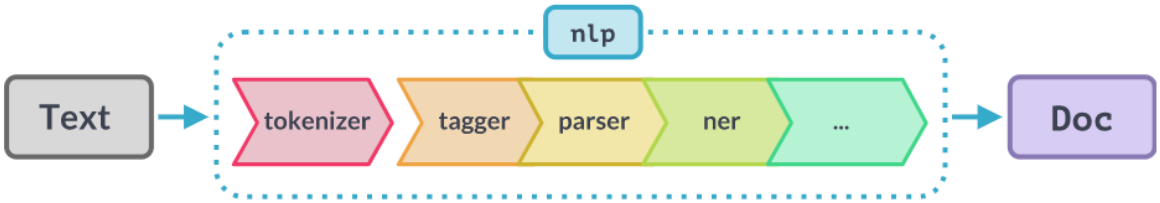

Figure 10. Spacy pretrained model pipeline overview. Source: Spacy.

Blank model

For the blank model, instead of loading an existing pipeline, we create a new blank pipeline. We specify language as English since we are working with both English and transliterated words (from Arabic to English). We also specify a demo vocabulary vector, which can be further optimized. Then, we create a new 'ner' component for our nlp_blank pipeline.
You can view the code and output in figure 11.

```
# Prepare an empty model to train
nlp_blank = spacy.blank('en')
# Assign name to the component
nlp_blank.vocab.vectors.name = 'demo'
print("Loaded model : %s" % nlp_blank)
```

```
Loaded model : <spacy.lang.en.English object at 0x7fb04ad59490>
```

```
#Create a new component 'ner' to the factory
ner_blank= nlp_blank.create_pipe('ner')
#Component ner is added last to the pipeline
nlp_blank.add_pipe(ner_blank, last = True)
print("Pipeline objects: %s" % nlp_blank.pipe_names)
```

```
Pipeline objects: ['ner']
```

Figure 11. Create a blank Spacy pipeline and ner component.

Note that, in contrast to the pretrained model, the blank model does not have any components except for the 'ner' component that we created.

Add custom Named Entity labels to the models.

Pretrained model.

Firstly, before adding the new Named Entity types, we can review the existing default NE types - view figure 12.

```
# Existing entity labels for the model
pretrained_previous_ents = ner_pretrained.labels
print('[Pretrained model: existing entities]')
for name in ner_pretrained.labels[::-1]:
  print(name) # print starting from last element
```

```
[Pretrained model: existing entities]
WORK_OF_ART
TRIBE
TIME
QUANTITY
PRODUCT
PERSON
PERCENT
ORG
ORDINAL
NORP
MONEY
LOC_CUSTOM
LOC
LAW
LANGUAGE
GPE_CUSTOM
GPE
FAC
EVENT
DATE
CARDINAL
```

```
#Description of the Named Entity label
spacy.explain("GPE")
```

```
'Countries, cities, states'
```

Figure 12. Display of the existing Named Entity types for the pretrained model and explain function.

We can also view description for each of the labels using the command ‘spacy.explain(label)’:

Then, we add the new labels and display the updated list of Named Entity types for the pretrained model - view figure 13.

```
# Add new Named Entity labels to the NER pipeline for the blank model
for label in training_data["classes"]:
  ner_pretrained.add_label(label)
```

```
# Updated entity labels for the pretrained model
pretrained_new_ents = ner_pretrained.labels
print('[Pretrained model: new entities]')
for name in ner_pretrained.labels[::-1]:
  print(name) # print starting from last element
```

```
[Pretrained model: new entities]
WORK_OF_ART
TRIBE
TIME
QUANTITY
PRODUCT
PERSON
PERCENT
ORG
ORDINAL
NORP
MONEY
LOC_CUSTOM
LOC
LAW
LANGUAGE
GPE_CUSTOM
GPE
FAC
EVENT
DATE
CARDINAL
```

Figure 13. Adding new labels and display of the updated Named Entity types for the pretrained model.

Blank model.

In contrast to the pretrained model, the blank model does not have Named Entity types. We also add new labels and display them - view figure 14.

```
# Existing entity labels for the blank model
blank_previous_ents = ner_blank.labels
print('[Blank model: existing entities]')
if (len(ner_blank.labels))==0:
  print("No Named Entity labels")
for name in ner_blank.labels:
  print(name)
```

```
[Blank model: existing entities]
No Named Entity labels
```

```
# Add new Named Entity labels to the NER pipeline for the blank model
for label in training_data["classes"]:
  ner_blank.add_label(label)
```

```
# Updated entity labels for the pretrained model
blank_new_ents = ner_blank.labels
print('[Blank model: new entities]')
for name in ner_blank.labels:
  print(name)
```

```
[Blank model: new entities]
GPE_CUSTOM
LOC_CUSTOM
TRIBE
```

Figure 14. Adding new labels and display of the existing (none) and updated Named Entity types for the blank model.

Create an optimizer for training the model.

In Machine Learning, optimizers refer to algorithms or methods that are used to change the attributes of the neural network such as weights and learning rate to reduce the losses and achieve the highest accuracy. In Spacy, functions ‘begin_training’ and ‘resume_training’ create and return an Optimizer object that can be used for updating and training the model further; ; view code in figure 15.

For the blank model, we use the ‘begin_training’ function (called ‘initialize’ in version 3), which sets up vocabulary, tok2vec weights and initializes methods implemented by the pipeline components or tokenizer.

For the pretrained model, we use the ‘resume_training’ function, which continues training a trained pipeline and initializes the ‘rehearsal’ for any pipeline component that has a rehearse method.

```
#Create a new optimizer for the blank model
optimizer_blank = nlp_blank.begin_training()
#Train the existing optimizer for the pretrained model
optimizer_pretrained = nlp_pretrained.resume_training()
```

Figure 15. Create an optimizer for pretrained and blank models.

Training and updating the models.

To update the existing pretrained and blank models, we firstly need to disable other components in each of the models so that we train only the Named Entity recognizer. Disabling is optional for the blank model since, as mentioned in the ‘Setup’ section, blank model has only one component (ner); however, disabling is necessary for the pretrained model.

To train the ner component for each of the models, the model has to be iterated a sufficient number of times; based on preliminary research, we iterate the model 50 times.

Before each iteration, it is advised to shuffle the text segments from the training data, which ensures that the model will not make generalizations based on the order of the text segments.

During the training, we skip segments that were not annotated. For each text segment and its list of annotated entities, we update the blank and pretrained model by providing the corresponding optimizer and loss function.

Testing the blank and pretrained models on a sample file.

```
#Disable other pipelines to train only the ner pipeline
other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']
with nlp.disable_pipes(*other_pipes):  # only train NER component
#Iterate through several instances to have more training data
  for iter in range(50):
    losses={}
    random.shuffle(training_data["annotations"])
    #For each text and annotations
    for text, annotations in training_data["annotations"]:
      #Skip the segments of the text that were not annotated--skip empty entities
        if len(text) > 0:
          # Update model parameters with text and annotations for that text
          # Text, annotations are required and optimizer,losses are optional parameters
            nlp_blank.update([text], [annotations], sgd=optimizer_blank, losses=losses)
            nlp_pretrained.update([text], [annotations], sgd=optimizer_pretrained, losses=losses)
            print(losses)
```

```
{'ner': 84.56919939412988, 'parser': 0.0, 'tagger': 1.182168220561846e-06}
{'ner': 84.56921263235245, 'parser': 0.0, 'tagger': 1.2015042839789319e-06}
{'ner': 84.5698020181679, 'parser': 0.0, 'tagger': 1.2026231800658254e-06}
{'ner': 84.56980214508131, 'parser': 0.0, 'tagger': 1.2040938766303571e-06}
{'ner': 90.88876082549916, 'parser': 0.0, 'tagger': 1.2207872806228198e-06}
{'ner': 90.8951227155014, 'parser': 0.0, 'tagger': 1.2210851113551185e-06}
{'ner': 90.92848881654027, 'parser': 0.0, 'tagger': 1.2334091606917169e-06}
{'ner': 90.92848886708866, 'parser': 0.0, 'tagger': 1.2614985723772448e-06}
{'ner': 90.92975637750763, 'parser': 0.0, 'tagger': 1.2628170155096923e-06}
{'ner': 90.92975698529611, 'parser': 0.0, 'tagger': 1.2628539158605228e-06}
{'ner': 5.685899795972525e-08, 'parser': 0.0, 'tagger': 1.5374027606540608e-09}
{'ner': 1.973512291929716e-06, 'parser': 0.0, 'tagger': 8.633176507011342e-09}
{'ner': 1.975988073316009e-06, 'parser': 0.0, 'tagger': 9.02673974456647e-09}
{'ner': 3.080870042166885, 'parser': 0.0, 'tagger': 2.1468662791956916e-08}
{'ner': 7.1333946783394735, 'parser': 0.0, 'tagger': 3.112116381265295e-08}
{'ner': 7.133395108240384, 'parser': 0.0, 'tagger': 3.986336100103571e-08}
{'ner': 7.133395108959267, 'parser': 0.0, 'tagger': 4.1012980289689693e-08}
{'ner': 7.133395273409728, 'parser': 0.0, 'tagger': 4.104930862933176e-08}
{'ner': 7.133395883590258, 'parser': 0.0, 'tagger': 4.1067421756649236e-08}
{'ner': 7.13339694688132, 'parser': 0.0, 'tagger': 4.2200952026297256e-08}
{'ner': 7.1333970327246226, 'parser': 0.0, 'tagger': 1.0868589579889065e-07}
{'ner': 9.40948549233599, 'parser': 0.0, 'tagger': 1.5022594009213297e-07}
```

Figure 16. Train the blank and pretrained models.