What is an entity:

- Words and text that have a specific name (label for text)

Extracting NERs is useful for:

- Naming Entities for medical notes (drugs, dosages, disease names, proteins, etc.)
- Naming Entities for journal articles (author, companies, topic, algorithm/method)
- Regex cannot distinguish Madoff as a person rather than a firm, also can't predict a new ML method, but may be able to identify a new one...
- Named entity linking

Creating NERs is difficult because:

- Research progress has been slow; Many pre-trained NERs are old – language changes
- There exist very few tools (Spacy/Prodigy being one, others?)
- It is knowledge intensive (you need to know the entities!) and need to train on at least 200 examples (manually tagged)
- Mix of easy and hard cases
- Hard to show your ideas are working (more work on challenging cases doesn't always result in much increase in performance whereas putting more care into other labeling may)

Using Spacy Prodigy for a custom NER: (*include old commands and new commands*)

Prodigy (go over basic commands

PRODIGY_HOST = 8090 PRODIGY_LOGGING=basic

```
db-in
db-out
dataset stats
stats -l
stats -ls
stats dams_ner
stats
```

1. Seed patterns (*prodigy sense2vec.teach*)
   Seeds cannot be formatted as "random forest", but instead "random_forest". Not all seeds are possible, think of synonyms.
2. Label using web interface (*prodigy ner.manual*). Be sure to include a few REJECT examples as well as all the ACCEPT labels. IGNORE is meant for very specific examples you want to skip, otherwise, you should reject the text if there are no entities [4]
3. Pretrain (*spacy pretrain*)
   Worth using if not using a transformer or have a small dataset. Pretraining initializes the model with information from the text and forces the network to model something about word cooccurrence. Pretraining allows the model to focus more on the important specific contextual modeling in the training process. [1]
4. Train (*prodigy ner.batch-train*)
   Batch-train is a little different than spaCy train, which was developed for larger datasets. Prodigy batch-train supports incomplete annotations (binary decisions from the web interface and example biased by score. Use prodigy batch-train for prototype models from annotations and spacy train for production model with a large number of annotations. [2]
5. Improve the model (*ner.make-gold*)
   This labeling is the 'gold standard' – all named entities are labeled and all tokens that are not labeled are not an entity.[3] gold-standard only uses ACCEPT, nothing to gain from REJECT here.

I WAS UNLABELING THE CORRECT LABEL THEN HITTING REJECT WHEN IT LABELED ALL OF THEM IN MAKE-GOLD, HMMMM….

6. **Produces a model loaded directly in Spacy** Python
   Nlp = space.load("drugs-model")
   Doc = nlp.(u'fentanyl is dangerous')
   [(ent.text, ent.label_) for ent in doc.ents]

How Spacy Prodigy works:

- Start as a state machine with no output attached, all the words in the sentence ahead in the buffer, look at next word, have an action that starts an entity (begin move), fixes the label at the start of the entity (instead of end)
- Use BILUO tagging scheme
- O = outside of an entity
- Block out the actions and build into the framework
- Statistical model for predictions (how NN is structures). Framework:
    - Embed: Come up with the list of words
        - Norm, prefix, suffix, shape
        - No fixed vocabulary due to how they are stored in table (nothing is out of dictionary)
        - 128 vectors
    - Encode: Find them in context
        - Go from context-independent vectors to context-sensitive matrix
        - Uses Convolutional Neural Network, trigram CNN layer takes a window on either side of the word (relearn what this word means by its neighbors and output a new vector)
    - Attend: Summary vector
    - Predict:
        - MLP
        -

[1] https://spacy.io/usage/embeddings-transformers
[2] https://support.prodi.gy/t/prodigy-ner-batch-train-vs-spacy-train/1665
[3] https://support.prodi.gy/t/training-of-annotated-dataset-with-ner-make-gold/1867/2
[4] https://support.prodi.gy/t/ignored-sentences-for-text-classification/1183