# Assignment 4: CoNLL2002 Spanish Named Entity Recognition (due 21 April 2020, 11.59 pm EST)]

The goal of this assignment is to implement a Named Entity Recognition (NER) system, which is the task of finding and classifying named entities in text. This task is like part of speech tagging, where words form a sequence through time, and each word is given a tag. NER however usually uses a relatively small number of tags, where the vast majority of words are tagged with the 'non-entity' tag, or O tag.

Your task is to implement your own named entity recognizer. There will be three versions of this task: the first, the ***constrained*** version, is the required entity tagger that you implement using **scikit-learn**, filling out the skeleton code that we give you. The second is the ***HMM*** version, where you will have to use an HMM library that is compatible with sklearn called `hmmlearn` (that you can find here[1]) to implement a named entity recognizer. The third is the ***sky-is-the-limit*** version where you use whatever tool, technique, or feature you can get your hands to get the best possible score on the dataset.

As with nearly all NLP tasks, you will find that the two big points of variability in NER are (a) the features, and (b) the learning algorithm. The point of this assignment is for you to think about and experiment with both of these. Are there interesting features you can use? What latent signal might be important for NER? What have you learned in the class so far that can be brought to bear?

The goals of this assignment are:

(1) To implement your own named entity recognizer (NER), to play around with features and algorithms that will help NER performance.

(2) To get a better understanding of HMMs by interpreting NER as a sequence tagging task and using an HMM to tackle it.

(3) To get a taste of choosing between and implementing state-of-the-art models for a specific task.

The materials provided in this zip file are:

```
|-- ner.py          # Skeleton code
|-- conlleval.py    # Evaluation script
```

## Deliverables

Here are the deliverables that you will need to submit.

```
|-- writeup.pdf
|-- README.txt
|-- code/
|   |-- ...
|-- constrained_results.txt       # Output of running your code for the constrained version.
|-- hmm_results.txt               # Output of running the HMM version.
|-- sky-is-the-limit_results.txt  # Output of running the sky-is-the-limit version.
```

[1] https://hmmlearn.readthedocs.io/en/latest/

The `...results.txt` files should make more sense after reading the rest of the assignment.

## Dataset

The data we use comes from the Conference on Natural Language Learning (CoNLL) 2002 shared task of named entity recognition for Spanish and Dutch. The introductory paper[2] to the shared task will be of immense help to you, and you should definitely read it. You may also find the original shared task page[3] helpful. We will use the **Spanish** corpus.

The tag set is:

- **PER**: for Person
- **LOC**: for Location
- **ORG**: for Organization
- **MISC**: for miscellaneous named entities

The data uses BIO encoding, which means that each named entity tag is prefixed with a B-, which means beginning, or an I-, which means inside. So, for a multiword entity, like “James Earle Jones”, the first token “James” would be tagged with “B-PER”, and each subsequent token is “I-PER”. The O tag is for non-entities.

You should study the training (esp.train) and development (esp.testa) data (for the integrity of your model, it’s best to not look at the test data (esp.testb)). Are there idiosyncrasies in the data? Are there patterns you can exploit as features? Are there obvious signals that identify names? For example, in some Turkish writing, there is a tradition of putting an apostrophe between a named entity and the morphology attached to it. A feature of `isApostrophePresent()` goes a long way. Of course, in English and several other languages, capitalization is a hugely important feature. In some African languages, there are certain words that always precede city names.

The data is packaged nicely from NLTK. Get installation instructions here: installing NLTK[4].

To get the dataset you can run:

```
python3 -m nltk.downloader conll2002
```

## Evaluation

There are two common ways of evaluating NER systems: phrase-based, and token-based. In phrase-based, the more common of the two, a system must predict the entire span correctly for each name. For example, say we have text containing “James Earle Jones”, and our system predicts “[PER James Earle] Jones”. Phrase-based gives no credit for this because it missed “Jones”, whereas token-based would give partial credit for correctly identifying “James” and “Earle” as B-PER and I-PER respectively. We will use phrase-based to report scores.

Each line of the output of your code must be `word gold pred`, separated by tab characters. As in:

```
La B-LOC B-LOC
Coruña  I-LOC   I-LOC
,   O   O
23  O   O
may O   O
(   O   O
EFECOM  B-ORG   B-ORG
)   O   O
.   O   O
```

Here’s how to score your output (assuming the above format is in a file called results.txt):

[2] https://www.aclweb.org/anthology/W02-2024
[3] https://www.clips.uantwerpen.be/conll2002/ner/
[4] http://www.nltk.org/install.html

```
python conlleval.py results.txt
```

## Resources

Here are some other NER frameworks which you can run in your ***sky-is-the-limit*** version:

- BERT NER[5]: You should be able to get really good scores using this.
- CogComp NER[6]
- LSTM-CRF[7]: neural network tagger.
- Stanford NER[8]: Stanford's tried and true tagger.
- spaCy[9].
- Brown clustering software[10]: You might find it useful.
- Europarl corpora[11]: look for the English-Spanish parallel text.
- Monolingual embeddings for Spanish words[12].
- Multilingual embeddings and word-to-word translations[13] for Spanish words.

Note: you are not allowed to use models pretrained with NER data even in the ***sky-is-the-limit*** version. Running pre-existing models and simply returning their outputs as your sky-is-the-limit results will not be accepted. Please train your own. You are allowed to use pre-trained embeddings, as well as models such as BERT that are pretrained with some objective with the intention that they will later be finetuned for a specific downstream task(for example, NER).

## A break down of the points

The skeleton code we have given you gets about 49% F1 (on development set) right out of the box.

- ***Constrained version***:
  - (**20 points**) Create a baseline that achieves more than or equal to 60% F1 on the development set. The state of the art on the Spanish dataset is below 90%. If you manage to beat that, then look for conference deadlines and start writing, because you can publish it.
  - **(20 points)** Your write up regarding the ***constrained*** version will earn you up to 20 points based upon how well you demonstrate the thoughtfulness you have put into your model. For example, you can discuss why you chose certain features, which models from `scikit-learn` worked well, and why you think that is, . . . etc.
- ***HMM*** version:
  - **(20 points)** For the modelling named entity recognition as a sequence tagging problem and using the `hmmlearn` library to train an HMM model and predict using it.
- ***sky-is-the-limit*** version:
  - **(10 points)** You will get 10 points for any attempt here.

[5] https://github.com/huggingface/transformers/tree/master/examples/ner
[6] https://github.com/CogComp/cogcomp-nlp/tree/master/ner
[7] https://github.com/glample/tagger
[8] https://nlp.stanford.edu/software/CRF-NER.shtml
[9] https://spacy.io/usage/training
[10] https://github.com/percyliang/brown-cluster
[11] http://www.statmt.org/europarl/
[12] https://github.com/uchile-nlp/spanish-word-embeddings
[13] https://github.com/facebookresearch/MUSE

  - **(10 points)** Similarly to the ***constrained*** version, you will get points for the thoughtfulness demonstrated in the README regarding this section.

- **(5 points)** for top-3 performance on the test set. This is for extra credit.

## Helpful Readings

- Survey on NER systems and features[14]

[14] http://www.oegai.at/konvens2012/proceedings/17_tkachenko12o/17_tkachenko12o.pdf