**Assignment 4: CoNLL2002 Spanish Named Entity Recognition (due 14 April 2019, 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 two 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 an ***unconstrained*** 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 baseline named entity recognizer (NER) (**40 points**)
2. To experiment with features and algorithms to improve the performance of your NER (**20 points**)

The materials provided in this zip file are:

1. Skeleton code (ner.py)
2. Evaluation script (conlleval.py)

**Deliverables**

Here are the deliverables that you will need to submit by sending it via email to [wijaya@bu.edu](mailto:wijaya@bu.edu) and [rxtan@bu.edu](mailto:rxtan@bu.edu) :

* writeup.pdf
* code (the code is in Python3)
* README file on how to run your code. **The grader must be able to run your code for your submission to be graded**.
* Constrained results (in a file called constrained\_results.txt)
* Unconstrained results (in a file called unconstrained\_results.txt)

If you run on SCC, before running the code to handle unicode you need to run export LANG="en\_US.UTF-8" before running python. Load python package using module load python/3.6.2 To install NLTK, use pip install nltk --user

**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](https://www.aclweb.org/anthology/W02-2024) 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](https://www.clips.uantwerpen.be/conll2002/ner/) 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](http://www.nltk.org/install.html).

To get the dataset you can run:

python -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.

The output of your code must be **word** **gold** **pred**, 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):

# Phrase-based score

python conlleval.py results.txt

**Resources**

Here are some other NER frameworks which you can run in your ***unconstrained*** version:

[CogComp NER](https://github.com/CogComp/cogcomp-nlp/tree/master/ner)

[LSTM-CRF](https://github.com/glample/tagger), neural network tagger

[Stanford NER](https://nlp.stanford.edu/software/CRF-NER.shtml), Stanford’s tried and true tagger

[spaCy](https://spacy.io/usage/training)

[Brown clustering software](https://github.com/percyliang/brown-cluster). You might find it useful.

[Europarl corpora](http://www.statmt.org/europarl/), look for the English-Spanish parallel text

[Monolingual embeddings for Spanish words](https://github.com/uchile-nlp/spanish-word-embeddings)

[Multilingual embeddings and word-to-word translations](https://github.com/facebookresearch/MUSE) for Spanish words

Note: you are not allowed to use pre-trained NER models even in the ***unconstrained*** version. Running pre-existing models and simply returning their outputs as your unconstrained results will not be accepted. Please train your own. You are allowed to use pre-trained embeddings.

**Baselines, Unconstrained, and Write-up**

The skeleton code we have given you gets about 49% F1 (on development set) right out of the box. (**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 about 85%. If you manage to beat that, then look for conference deadlines and start writing, because you can publish it.

**(10 points)** You will get 10 points for at least an attempt on the unconstrained version beyond what is required for the baseline.

Demonstrate that you have thought about the problem carefully, and come up with solutions both for the baseline **(20 points)** and for your unconstrained version **(10 points)** in your write up. Extra credit **(5 points)** for top-3 performance on the test set.

**Helpful Readings:**

[Survey on NER systems and features](http://www.oegai.at/konvens2012/proceedings/17_tkachenko12o/17_tkachenko12o.pdf)