HW2: Named Entity Recognition

Instructor: Dr. Malihe Alikhani
TA: Mert Inan
Additional Credits: Zhexiong Liu

Due: Mar 22, 2023, 11:59 PM (EST)

Named Entity Recognition is the task of finding and classifying named entities in text. This task is often considered a sequence tagging task, like part of speech tagging, where words form a sequence through time, and each word is given a tag. Unlike part of speech tagging however, NER 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. You will implement an entity tagger using scikit learn, filling out the stub that we give you.

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, with the features arguably being the more important of the two. 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?

Get a headstart on common NER features by looking at Figure 17.5 in the textbook, and reading Chapter 17.1, from an older edition of the textbook (use the link).

identity of w_i, identity of neighboring words
embeddings for w_i, embeddings for neighboring words
part of speech of w_i, part of speech of neighboring words
base-phrase syntactic chunk label of w_i and neighboring words
presence of w_i in a **gazetteer**
w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i is all upper case
word shape of w_i, word shape of neighboring words
short word shape of w_i, short word shape of neighboring words
presence of hyphen

Figure 17.5 Typical features for a feature-based NER system.

Source Code

Please use Canvas Files to download source code for this assignment:

- Code stub
- conlleval.py: evaluation script

The Data

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 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 helpful. We will use the Spanish corpus (although you are welcome to try out Dutch too).

The tagset is:

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

The data uses BIO encoding (called IOB in the textbook), 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.

We strongly recommend that you study the training and dev data (no one's going to stop you from examining the test data, but for the integrity of your model, it's best to not look at it). Are there idiosyncrasies in the data? Are there patterns you can exploit with cool 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.

You will be glad to hear that the data is a mercifully small download. See the NLTK data page for download options, but one way to get the conll2002 data is:

```
$ 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 get scores (assuming the above format is in a file called `results.txt`):

```
# Phrase-based score
$ python conlleval.py results.txt
```

Please create this output for the training set (as `train_results.txt`), development set as (`dev_results.txt`), and test set (as `test_results.txt`). You can retrieve the sentences with the following code:

```
train_sents = list(conll2002.iob_sents('esp.train'))
dev_sents = list(conll2002.iob_sents('esp.testa'))
test_sents = list(conll2002.iob_sents('esp.testb'))
```

(The python version of conlleval doesn’t calculate the token-based score, but if you really want it, you can use the original perl version. You would use the `-r` flag.)

Baselines

The version we have given you gets about 49% F1 right out of the box. We made some very simple modifications and got it to 60%. The threshold we ask you to beat is 68%, with partial credit available. The state of the art on the Spanish dataset is over 90%. If you manage to beat that, then look for conference deadlines and start writing, because you can publish it.

In order to earn full marks on this assignment, demonstrate that you have thought about the problem carefully, and come up with solutions beyond what was strictly required. This is very open-ended homework and we hope you take advantage of that to get out of your comfort zone and experiment.

Report

1. Explain the features you added for NER, why you expected them to help, and how they affected your performance. Include a table detailing the change in F1-score as a result of adding each feature or set of features.

2. Explain the different types of models you experimented with, how they performed, and which you chose for your final model. Include a table comparing the scores of different models. For each model, be sure to tune your parameters on the dev set (optimize your performance with regards to dev F1-score) and include tables recording the training F1-score and dev F1-score attained for each set of parameters.

3. Using your best performing model, do some error analysis (a necessary skill for any researcher to have!) and determine what types of mistakes your model seems to be making. Some things you can think about are in what cases the mistakes are typing issues (i.e. predicting ORG instead of LOC) vs. span issues (i.e. predicting B-LOC when it should be I-LOC), and whether those correlate with certain POS tags or contexts. A thoughtful analysis with cited examples should easily get full points for this part of the report.

Deliverables

Here are the deliverables that you will need to submit (**no compressed files are necessary**):

- Code (ner.py), as always, in Python 3.
- **Results for the best model** (in files called `train_results.txt`, `dev_results.txt`, `test_results.txt`)
- PDF Report (called writeup.pdf)

Recommended readings

1. Design Challenges and Misconceptions in Named Entity Recognition a very highly cited NER paper from a Penn professor.
2. Entity Extraction is a Boring Solved Problem – or is it?
3. Neural Architectures for Named Entity Recognition, a popular paper on… just read the title.
4. Introductory paper to CoNLL 2002 shared task
5. ner and pos when nothing is capitalized, a recent short-and-sweet Penn paper about the shortcomings of relying on capitalization.