Named Entity Recognition using Structured Prediction

Joachim Daiber
University of Groningen
jodaiber@gmx.de

Carmen Klaussner
University of Groningen
Carmen@wordsmith.de

1 Introduction

Named Entity Recognition (NER) is an information extraction task and has first been introduced as part of the 6th MUC (Sixth Message Understanding Conference) that was focusing on the extraction of structured information from unstructured text, such as company names from newspaper articles (Nadeau and Sekine, 2007). The most prominent types are generally known under the name of *enamex* types, which are comprised of `Person Names`, `Organisations` and `Locations`. An additional type `Miscellaneous` captures person names outside the classic *enamax* type. Apart from these there is *timex*, which covers date & time expressions and *numex* which is for monetary values & percent. Table 1 gives an overview of the different NE types.

Named Entity Type	**Example**
Person Names	Greta Garbo, John, Mary
Organisations	Benetton, Coca-Cola Gmbh
Locations	Paris, Australia, Bristol
Miscellaneous	World Cup
Date Expression	01-02-2012, January, 6th, 1998
Time Expression	10 p.m.
Monetary Value	$15, 100 Euro
Percent	30%

Table 1: NE Examples

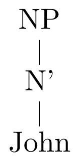
In considering named entities, it is important to distinguish between mention of entities that are non-specific as time expressions, such as *In June*, referring to any possible year or *the prof* as a person name, which itself is not a specific entity, but a deictic reference, which may point back to the mention of a real *NE*, as in the following coreference chain:

$NP \rightarrow NPPP$
$PP \rightarrow PNP$

Figure 1: Recursive Noun Phrase

(1) $Prof.\ Bateman \Rightarrow he \Rightarrow the\ prof$

Named Entities can consist of more than one syntactic type. The usual overall type for a named entity is noun phrase, which can be both simple and complex. An example for a simple heterogenous entity is displayed in syntactic tree represenation in (1).



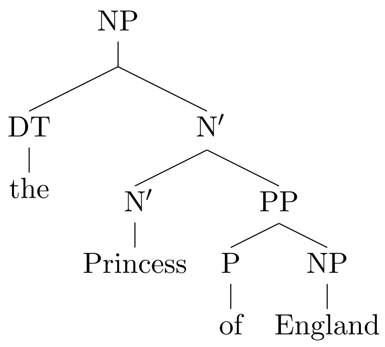
Most languages allow for a recursive definition of, for instance, a noun phrase that in theory can grow infinitely big. An **NP** can consist of a determiner, such as *the* and a common noun or simply a propernoun. **PP** stands for prepositional phrase and consists of a preposition and a noun phrase. The syntactic rules for PP-attachment are displayed in 1. These rules allow for constructions such as in sentence (2).

(2) The princess of Scotland is in the castle in England with the brown bricks on its roof...

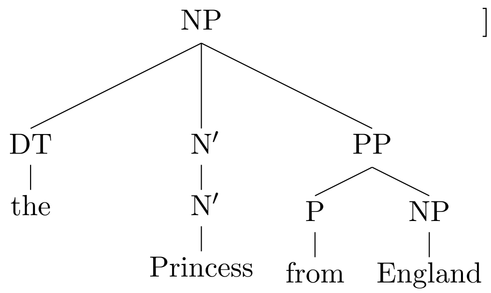
We distinguish between PP-complements and PP-modifiers, where modifiers simply add to the description of the noun phrase, maybe to set it apart and complements are more inherent to the meaning of the entity. The PP-attachment *of*

Scotland is semantically defining for the noun *princess*, whereas *with the brown bricks on its roof* just adds to the description of *the castle in England*. Regarding named entities, one wants to differentiate between a PP-attachment that defines the entity an one that is irrelevant.

1. (a)



(b)



]

Issues in Named Entity Recognition

Among the most common problems for NER is the issue of ambiguity, that is in particular *Polysemy* (Nadeau and Sekine, 2007), the property of some lexical representation having more than one possible meaning, and *Metonymy*, which refers to the concept of the part-whole/whole-part relation between two expressions. *Polysemy* could become an issue for NER when the lexical representation of an item could point to two different *NE* types. This is quite frequent with `Person Names` and `Locations`, since many people are named after cities, such as *Paris* or *Georgia*. Often the context will not be disambiguating as in (3).

(3) *Paris is beautiful.*

Metonymy is frequently an issue in literary texts and news data (which is often used in NER), where two items that are in a part-whole relationship, are substituted for each other respectively. Example (4) shows an instance of *whole-part*, where *London* is supposedly substituted for the **Goverment in London**.

(4) ***London*** *decided to increase the 1200 military personnel involved in Olympic security.*

Methods employed for NER

For *NER*, there exist both rule-based and statisical approaches. Rule-based methods make use of the underlying rules governing languages to extract named entities. However, this approach is high maintenance, quite time-consuming and requires extensive work of computational linguists (Nadeau and Sekine, 2007) and although the results are often high in precision, lacks considerably in recall.

In regard to statistical approaches, supervised-learning is the most common method applied in *NER*. Although, there are unsupervised approaches, their performance is not as high as for SL applications yet.

Prominent algorithms in NER include maximum entropy models,

neural network

CoNLL Data Set & Baseline Performance

The training and test dataset was taken from the *CoNLL Shared Task 2002* (**?**) and the *CoNLL Shared Task 2003* (Tjong Kim Sang and De Meulder, 2003). for Spanish & Dutch and English & German respectively. The ConNLL shared tasks are organised challenges, where the organiser propose a task and provide the training and test data. For 2002/2003 it was Named Entity Recognition. Also we base our system on the approach that won the challenge for English and German in 2003. The baseline rate for all languages was produced by a system, that had a unique class in the training data. In case that a phrase was part of more than one entity, the system would choose the longest one (Tjong Kim Sang and De Meulder, 2003). The data is annotated with POS-tags and NE-tags, as well as Bio-tags that indicate the position in the Named Entity.

The baseline for *Named Entity Recognition* in the four different languages is visualised in table

Language	Precision	Recall	F_1-measure
Spanish	26.27 %	56.48 %	35.86 %
Dutch	64.38 %	45.19 %	53.10 %
English	71.91%	50.90 %	59.61 ± 1.2 %
German	31.86 %	28.89 %	30.30 ± %1.3

Table 2: NER Baseline

2 Structured Prediction

Structured Prediction (Carreras, 2012) is a supervised-learning approach that maps an input $\mathbf{x}$ to an output $\mathbf{y}$:

$$x \rightarrow y.$$

Structured and Non-structured prediction differ in the form of their output. Non-structured output is atomic; it is binary prediction for a two-class problem and may corresponds to more than one of more than two possible labels for a multiclass problem. Structured Prediction gives back the prediction for the whole sequence. For the sentence in (5), it would return the corresponding labels combination in example (6).

(5) *The motor company 'Ford' was founded and incorporated by Henry Ford.*

(6) **x:** The motor company 'Ford' was founded and incorporated by Henry Ford.
y: 0 0 ORG 0 0 0 0 0 PER PER.

Also, Structured Prediction is different from similar approaches through its combination of features and label interactions instead of concentrating mainly on label interactions like HMM or a rich set of features like local classifiers.

3 Our Implementation

Our NER system for the languages English, German, Dutch and Spanish is trained and tested on the *CoNLL 2003* and *CoNLL 2002* data sets respectively.

Learning Labels of the whole sentence (0) for zero entity

Structure Structured Perceptron with Averaging

Decoding In this system we are using viterbi decoding, thus we are looking for the prediction $\hat{y}$, which is the sequence with the highest overall score:

$$\hat{y} = \arg\max_{\mathbf{y}\in\mathcal{Y}^n} \mathbf{w}\cdot \boldsymbol{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

This sequence can be computed efficiently in $O(N^2|\mathbf{x}|)$ using the Viterbi algorithm. As a first step, the trellis has to be constructed, then we have to find the $a \in \mathcal{Y}$ in the last column of the trellis with maximal score $\delta_n(a)$. From this, the sequence can be recovered via back-tracking trough the trellis. The score of the best sequence ending in a is:

$$\delta_i(a) = \max_{\mathbf{y}\in\mathcal{Y}^n, y_n=a} \sum_{j=1}^{n} \mathbf{w}\cdot \boldsymbol{f}(\mathbf{x}, j, y_{j-1}, y_j)$$

And $\delta_i(a)$ can be calculated recursively:

$$\delta_1(a) = \mathbf{w}\cdot \boldsymbol{f}(\mathbf{x}, 1, \emptyset, a)$$
$$\delta_i(a) = \max_{b\in\mathcal{Y}} \delta_{i-1}(b) + \mathbf{w}\cdot \boldsymbol{f}(\mathbf{x}, i, b, a)$$

Features

The features employed in the system can be divided into three categories: node, label and gazetteer features. We describe each of the three groups in the following.

Node features These are only present on the word in question: the *Token*, suffix and prefix, capitalisation.

Table 3 shows the various node features with an example respectively.

Label Interaction Features These features register which label has been assigned to the previous token and takes into account the most likely sequence. Thus, for the sequence: "Jack London went to New York" the NE tag combination of *PER* and *PER* in example (7)is more likely than*PER* and *LOC* as presented in (8). .

Feature	**Example**
Token	**“Amsterdam”**
Suffix	Amster**dam**
Prefix	**San** Sebastian
Captitalized	**B**enetton
Number Pattern	
UPPERCASE	**BENETTON**
POS-tag	Benetton **NNP**
Lemma	produced *Rightarrow* produce

Table 3: Node Features

(7) Jack London went to New York.
PER PER 0 0 LOC LOC.

(8) Jack London went to New York.
PER LOC 0 0 LOC LOC.

Gazetteer Features In order to create gazetteer lists for the more common named entities, we designed a *SPARQL* query, that would retrieve entries from *DBPedia* for all languages. The reliability of the respective list is learnt be the perceptron.

4 Experiments/ Evaluation

In the following section we present our experiments and the evalution of our system.

Experiments

Evaluation

For the evaluation of the system we used *Precision* and *Recall* as shown in (9) and (10) respectively. The general formula for the *F-Score* is shown in (11). Since we rate both *Precision* and *Recall* evenly, we use the harmonic mean as shown in (12).

(9) $Precision = \frac{gold\ tag \bigcap predicted}{predicted}$

(10) $Recall = \frac{gold\ tag \bigcap predicted}{gold\ tag}$

(11) $F_\beta = (1+\beta^2) * \frac{precision*recall}{\beta^2*precision+recall}$

(12) $F_1 = 2 * \frac{precision*recall}{precision+recall}$

Language	**testA**			**testB**		
—	Precision	Recall	F_1	Precision	Recall	F_1
Spanish						
Dutch						
English						
German						

Table 4: NER Structured Prediction Results

5 Discussion of Results

Some Challenges

6 Future Work & Conclusion

References

[Carreras2012] Xavier Carreras. 2012. Learning structured predictors. Lecture at Lisbon Machine Learning School 2012, `http://lxmls.it.pt/strlearn.pdf`.

[Nadeau and Sekine2007] D. Nadeau and S. Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

[Tjong Kim Sang and De Meulder2003] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL ’03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.