Named Entity Recognition as Structured Prediction

Joachim Daiber
University of Groningen
2397331
jodaiber@gmx.de

Carmen Klaussner
University of Groningen
2401541
Carmen@wordsmith.de

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1 Application: Named Entity Recognition

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 names outside the classic *enamax* type. Apart from the above, there is the *timex* type, which covers date & time expressions and *numex* which is for monetary values & percent. Table 1 gives an overview of the different NE types.

In considering named entities, it is important to distinguish between named entities and the mention of entities that are in fact non-specific. The difference lies in the reference of the entity. A named entity refers to a specific, unique person/time/location, whereas in mere mentions of entites, these can refer to multiple events, such as the time expression: *In June*, referring to any possible year. Also, there could be a reference to a person, as in: *the prof*, which in itself is not a specific entity. However, it would be a deictic reference and point back to the mention of a real *NE*, as in the following coreference chain:

Named Entity Type	**Example**
Person Names	Greta Garbo, John, Mary
Organisations	Benetton, Coca-Cola Gmbh
Locations	Paris, Australia, Bristol
Miscellaneous	World Cup 2014
Date Expression	01-02-2012, January, 6th, 1998
Time Expression	10 p.m.
Monetary Value	\$15, £100
Percent	30%

Table 1: NE Examples

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

Named Entities can consist of more than one syntactic type. Syntactic types feature in syntactic rules that define, how sentences can be build. Typical rules are:
$S \rightarrow NPVP$ [1]
$VP \rightarrow V(NP)$ [2]

The usual overall type for a named entity is

[1] A sentence S breaks down into a noun phrase NP and a verb phrase VP

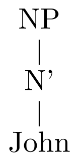
[2] A verb phrase VP consists of a verb followed by an optional NP

$NP \rightarrow NP\ PP$
$PP \rightarrow P\ NP$

Figure 1: Recursive Noun Phrase

noun phrase, which can be both simple and complex, meaning it can consist of a only a noun or also additional other syntactic types. An example for a simple homogenous entity is displayed in syntactic tree representation in 1.

1.

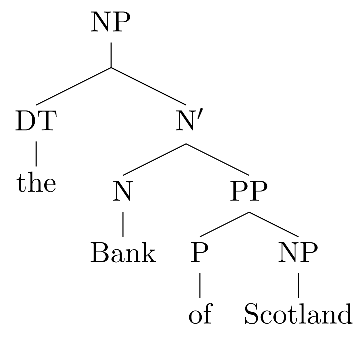


N' signals an intermediate state in the syntactic tree where adjuncts can be joined to the noun phrase. The closer the entity is joined to a tree, the more it usually contributes to the meaning. Most languages allow for a recursive definition of, for instance, a noun phrase that in theory can grow infinitely big. A **NP** can consist of a determiner, such as *the* and a common noun or simply a propernoun. **PP** stands for prepositional phrase (PP) and consists of a preposition and a noun phrase. The syntactic rules for PPs are displayed in 1. These rules allow for constructions such as in sentence (2).

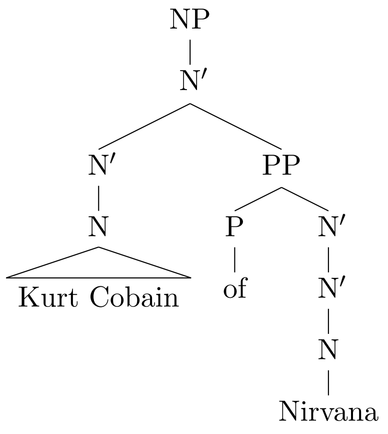
(2) The Bank of Scotland is in the brown building with the brown bricks on its roof...

We distinguish between PP-complements and PP-modifiers, where modifiers simply add to the description of the noun phrase, and complements are more inherent to the meaning of the entity. The PP *of Scotland* is semantically defining for the noun *bank*, whereas *with the brown bricks on its roof* just adds to the description of *brown building*. Regarding named entities, one wants to differentiate between a PP that defines the entity an one that is not inherent to its meaning. The phrases in 1 and 1 give examples of how a syntactic tree defines semantic connection. In *Bank of Scotland*, the PP *of Scotland* is attached directly, whereas in 1 both *Kurt Cobain* and *Nirvana* constitute own concepts and are joined at a later level to express this distance. Thus, in connection with Named Enitites, we want to be able to capture types of the first example as a NE, but not of the second.

1. (a)



(b)



1.1 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.*

1.2 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

1.3 CoNLL Data and Baseline Performance

The training and test data for Spanish & Dutch was taken from the *CoNLL Shared Task 2002* (**?**) and the *CoNLL Shared Task 2003* (Tjong Kim Sang and De Meulder, 2003) provided the one for English & German. The ConNLL shared tasks are organised challenges, where the organisers propose a task and provide the participants with the training and test data. In 2002 and 2003, the task was Named Entity Recognition. In both years, the baseline rate for the individual 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). Table 2 shows the baseline for the four different languages. The data is annotated with part-of-speech tags and named-entity tags, as well as Bio-tags that indicate the position in the Named Entity. Our approach is based on the system that won the challenge for English and German in 2003.

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 $\pm$ 1.2%
German	31.86%	28.89%	30.30 $\pm$ 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).

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 Implementation

3.1 Learning and Decoding

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

As the predicted structure, we use the labels for every token in the full sentence. If a token is not a Named Entity, it is assigned the label `O`.

Learning is implemented using the Structured Perceptron algorithm. To avoid overfitting, we average the parameters after the last iteration (Collins, 2002). For finding the best sequence of labels for a sentence, we use a modified version

(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: O O O 0 ORG O O O O O O PER PER O

Figure 2: Input and predicted structure for the Named Entity task.

of the Viterbi algorithm. The algorithm finds the sequence $\hat{y}$, such that:

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

This sequence can be computed efficiently in $O(N^2|\mathbf{x}|)$ via dynamic programming. 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:

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

This function $\delta_i(a)$ can be defined recursively as:

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

3.2 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 Node features are features that only depend on the current token (node): 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 ⇒ produce

Table 3: Node Features

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

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

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 and 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	**Test A**			**Test B**		
	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`.

[Collins2002] M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

[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.