Named Entity Extraction using AdaBoost

Xavier Carreras, Lluís Màrquez and **Lluís Padró**
TALP Research Center
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
{carreras,lluism,padro}@lsi.upc.es

1 Introduction

This paper presents a Named Entity Extraction (NEE) system for the CoNLL 2002 competition. The two main sub-tasks of the problem, recognition (NER) and classification (NEC), are performed sequentially and independently with separate modules. Both modules are machine learning based systems, which make use of binary AdaBoost classifiers.

For the NER module, we propose some variants in which local classifiers detect the beginning or the end of a named entity (NE), or determine for each word whether it belongs to a NE or not. Greedy and global inference methods are evaluated for combining the outcomes of such classifiers. The NEC module consists simply in a 4–class classification problem.

We use binary AdaBoost with confidence rated predictions as learning algorithm for the classifiers involved in the system. This algorithm has been applied, with significant success, to a number of problems in different areas, including NLP tasks (Schapire, 2002). See (Schapire and Singer, 1999) for details about the general algorithm, and (Carreras and Màrquez, 2001; Carreras et al., 2002) for particular applications to NLP domains. In our setting, the boosting algorithm combines several small fixed-depth decision trees. Each tree is learned sequentially by presenting the decision tree learning algorithm a weighting over the examples which depends on the previously learned trees.

Ortographic and semantic features evaluated in a shifting window are used for allowing a relational representation of the examples via many simple binary propositional features. The boosted decision trees construct conjunctions of such binary features, allowing the boosting classifier to work with complex and expressive rules.

2 Feature Representation

A window anchored in a word w represents the local context of w used by a classifier to make a decision on the word. In the window, each word around w is codified with a set of primitive features, together with its relative position to w. Each primitive feature with each relative position and each possible value forms a final binary feature for the classifier (e.g., *the word form at position -2 is* "calle"). The set of primitive features applied to each word in the window is the following:

- The **word form**, and **part–of–speech** (PoS), if available.
- **Ortographic Features.** These are binary and not mutually exclusive features that test whether the following predicates hold in the word: *initial-caps*, *all-caps contains-digits*, *all-digits alphanumeric*, *roman-number*, *contains-dots*, *contains-hyphen*, *acronym*, *lonely-initial*, *punctuation-mark*, *single-char*, *functional-word*, and *URL*. Functional words are determiners and prepositions which tipically appear in NEs
- **Word Type Patterns.** The type of a word is either *functional*, *capitalized*, *lowercased*, *punctuation mark*, *quote* or *other*. Each conjunction of types of contiguous words is a word type pattern. Patterns in the window which include the anchoring word are considered.
- **Bag–of–Words.** Form of the words in the window, without considering their position. (e.g. "banco" $\in$ *window*).
- **Trigger Words.** Class of trigger words in the window. Also a pattern of the words to the left of the anchoring word, with regard

to punctuation marks, prepositions, determiners, trigger words denoting person, location, organization, or other entities, and trigger words denoting geographical origin,

- **Gazetteer Features.** Class (geographical, first name, or surname) of words in the window appearing in the gazetteer.
- **Left Predictions.** The tags being predicted in the current classification. These features only apply to the words in the window to the left of the anchoring word w.

The NER module uses windows with form and PoS, ortographic features, word type patterns and left predictions.

The NEC module uses windows considering form and PoS, bag-of-words, trigger words and the gazetteer. It also uses the following features to represent the internal structure of the NE being classified: 1) Length (in words) of the entity being classified. 2) Pattern of the entity with regard to the types of the constituent words, and 3) For each class, two features indicating whether the whole NE or any word component appear in the gazetteer or not. 4) Suffixes (lengths 1 to 4) of each component and the whole NE .

3 The NER Module

The NE recognition task is performed as a combination of local classifiers which test simple decisions on each word in the text. We describe three variants of decision schemes for recognizing NEs by combination of classifiers.

BIO. In this well–known model, each word is tagged as either the beginning of a NE (B tag), a word inside a NE (I tag), or a word outside a NE (O tag). We use three binary classifiers for the tagging, one corresponding to each tag. When tagging, a sentence is processed from left to right, selecting for each word the tag with maximum confidence that is coherent with the current solution.

The three classifiers use a window to represent the context of the decision. All the words in the train set are used as training examples, applying a *one-vs-all* binarization.

Open-Close&I. In this scheme a NE is recognized by detecting the word which opens and the one which closes the NE. A sentence is processed left-to-right, greedily applying three classifiers: an open classifier searches for a NE start, and when detected, a close classifier searches the end of the NE. In order to robustify the search of the closing word, each word inside the current NE is tested by the I classifier of the BIO scheme, and if it is classified negatively, the NE is forced to close at the previous word.

The open classifier represents the context with a window from which the features are extracted. The close classifier makes use of two windows: the first, anchored in the current open word o, represents only the previous three words to o; the second, anchored in the current word c, represents the words from o to c, and the three following words to c. In this way, the whole current NE being closed is represented.

Training examples for the classifiers are generated by simulating the correct annotation of the train set: the open classifier takes words outside a NE as negative examples and beginnings of NEs as positive examples; the close classifier takes examples only from the words which form NEs in the data, considering positive examples the final words.

Global Open-Close. Again, this approach searches for opens and closes of NEs in the sentence, but global inference is made to produce the set of NEs for the whole sentence. First, each word in the sentence is independently classified by an open and a close classifier. After this, each pair of words (w_l, w_r), with $l \leq r$, is considered a NE candidate, taking the confidence of such candidate as the summation of the open confidence of w_l and the close confidence of w_r. The optimal set of NEs for the sentence is considered to be the coherent set which maximizes the summation of confidences of the NEs in the set. Using dynamic programming techniques, such optimal set can be found in quadratic time in the number of words.

The two classifiers involved in the scheme are trained considering as an example each word in the data, and making use of a window to represent each word.

4 The NEC Module

NEC is simply a classification task, consisting of assigning the NE type to each potential, and already recognized, NE. In this case, all the decisions are taken independently, and the classification of a certain NE does not influence the classification of the following ones.

The problem is binarized training up to ten diferent binary classifiers: The four possible *one-vs-all* classifiers, the three possible (non-symmetrical) combinations of *two-vs-two* classifiers, and three binary classifiers trained to distinguish between two categories ignoring the remaining two *(PER-vs-LOC*, *PER-vs-ORG*, and *LOC-vs-ORG)*. Further combinations were not used to keep computing cost at affordable levels.

Binary classifiers are combined using an Error Correcting Output Code (ECOC) with a loss–based decoding scheme (Allwein et al., 2000), that is, taking into account the confidence degree of each classifier instead of merely their binary output.

Error Correcting Output Codes consist of using redundant information in a binary code, in order to be able to correct noisy bits in a coded string. A well known application case is Hamming's distance. In our case, the redundant information is provided by the binary classifiers which decide on non-disjoint subsets of classes (e.g., *one-vs-all* for PER and *PER-vs-LOC* both use PER as positive examples).

5 Experiments

A list of functional words for each language was automatically constructed with the training set of the language. The lowercased words inside a NE that appeared more than 3 times were selected as functional words for the language. Similarly, a gazetter was constructed with the NEs in the training set. When training, only a random 40% of the entries in the gazetter were considered. Moreover, we used external knowledge for Spanish, namely a list of trigger words for NEs and an external gazetteer.

The exploration of parameters was carried out on the Spanish development set. For each binary classification problem we trained classifiers, with depth ranging from 1 to 5, and with up to 2,000 base trees per classifier. We tuned the depth and number of trees to combine by optimizing the F_1 on the development set.

The three NER models for the recognition were tested using different feature settings. We used windows of size ± 3, except for the variable non-global close windows. Trigger words, bag-of-words and gazetteer features did not help the recognition at all, and therefore were not used.

Spanish dev.	precision	recall	$F_{\beta=1}$
BIO	92.45%	90.88%	91.66
OPENCLOSE&I	90.50%	90.40%	90.45
gOPENCLOSE	91.58%	90.21%	90.89

Table 1: NER results on the development set.

	precision	recall	$F_{\beta=1}$
Spanish dev.	77.91%	76.59%	77.24%
Spanish test	79.27%	79.29%	79.28%

Table 2: Spanish results without making use of external knowledge.

Table 1 presents the results of NE recognition for the three presented variants. The BIO model, being the simplest, performed slightly better than the open-close approaches.

For the NEC module, the window size was set to ± 3, considering size ± 5 for the bag-of-words features. All the information turned to be relevant for the entity classification. Each single classifier was trained and tuned, and several ECOC combination matrices were tested, concluding that the best setting is the one combining all the classifiers.

When porting the system to Dutch, the same setting than in Spanish turned to be the optimal. This time, no external resources were used, and part-of-speech features were considered for both tasks. In NER, the BIO approach was significantly better than open-close models.

Table 3 presents the results on the NEE task obtained by pipelining the NER and NEC modules. The NER module used the BIO approach. The NEC module used an ECOC of all learned binary classifiers. For Spanish sets, features included external knowledge (gazetteer and list of trigger words). In order to allow fair comparison with other systems, table 2 presents the overall results achieved on the Spanish sets without using external knowledge.

6 Discussion

Interestingly, the simplest BIO scheme achieves the best results for the NER task. However, since it is a straightforward strategy, there is no room for global inference. Regarding the OpenClose scheme, the global inference turns to be slightly better than the greedy approach. Other OpenClose schemes based on learning specific classifiers for predicting the scoring assigned to potential NEs did not improve the performance over the basic scoring based on summation.

The results for the complete NEE task are rather lower than those of NER(Table 3). The performance of the NEC system evaluated on the output of a perfect NER system scored from 80% to 86% on the four sets. Thus, the chaining of the two modules causes the propagation of errors and the degradation of performance on the main task. The MISC category turns to be the most hard to predict. This may be caused either by the difficulty of characterising a so general category, or by some inconsistencies in the data regarding segmentation and classification of the MISC named entities.

It is also remarkable that the use of additional knowledge sources, such as features codifying the occurrence of words contained in external gazetteers and lists of trigger words yield and improvement of 2% in the NEC system.

A failed attempt to improve the performance consisted in labelling the whole data set in two steps: In the first step, the recognized NEs were stored in a temporal gazetteer, used in the second step to produce the final labelling. The idea behind was to take advantage of the common repetitions of a particular NE in a text, but no improvement was achieved.

The system performs significantly better in Spanish than in Dutch, even when no external resources are used for the former (Table 2). In our opinion, the design of our system, in terms of strategy and feature space, is language independent. A main line of improvement, therefore, would be to add language-depandant processing to the system based on a linguistic study. Besides, these particular Dutch data may be more difficult to predict than the Spanish data.

An open line of research to be addressed is the simultaneous approach of NER and NEC tasks, so each decision may take advantage of the synergy between both knowledge levels.

Acknowledgments

Research partially funded by the European Commission (Meaning, IST-2001-34460) and the Spanish Research Dept. (Hermes, TIC2000-0335-C03-02; Petra - TIC2000-1735-C02-02). Xavier Carreras holds a grant by the Catalan Government Research Department.

Spanish dev.	precision	recall	$F_{\beta=1}$
LOC	79.04%	80.00%	79.52%
MISC	55.48%	54.61%	55.04%
ORG	79.57%	76.06%	77.77%
PER	87.19%	86.91%	87.05%
overall	79.15%	77.80%	78.47%

Spanish test	precision	recall	$F_{\beta=1}$
LOC	85.76%	79.43%	82.47%
MISC	60.19%	57.35%	58.73%
ORG	81.21%	82.43%	81.81%
PER	84.71%	93.47%	88.87%
overall	81.38%	81.40%	81.39%

Dutch devel.	precision	recall	$F_{\beta=1}$
LOC	69.71%	80.25%	74.61%
MISC	80.50%	73.59%	76.89%
ORG	76.79%	69.66%	73.05%
PER	77.73%	77.40%	77.57%
overall	76.52%	74.82%	75.66%

Dutch test	precision	recall	$F_{\beta=1}$
LOC	78.54%	80.67%	79.59%
MISC	81.03%	70.51%	75.41%
ORG	70.03%	72.75%	71.36%
PER	80.71%	82.25%	81.47%
overall	77.83%	76.29%	77.05%

Table 3: Final results for Spanish and Dutch.

References

E. L. Allwein, R. E. Schapire, and Y. Singer. 2000. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, 1:113–141.

X. Carreras and L. Màrquez. 2001. Boosting Trees for Clause Splitting. In *Proceedings of the 5th CoNLL*, Tolouse, France.

X. Carreras, L. Màrquez, V. Punyakanok, and D. Roth. 2002. Learning and Inference for Clause Identification. In *Proceedings of the 14th European Conference on Machine Learning, ECML*, Helsinki, Finland.

R. E. Schapire and Y. Singer. 1999. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3).

R. E. Schapire. 2002. The boosting approach to machine learning. an overview. In *Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA.