A Simple Named Entity Extractor using AdaBoost

Xavier Carreras and **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-2003 shared task competition. As in the past year edition (Carreras et al., 2002a), we have approached the task by treating the two main sub–tasks of the problem, recognition (NER) and classification (NEC), sequentially and independently with separate modules. Both modules are machine learning based systems, which make use of binary and multiclass AdaBoost classifiers.

Named Entity recognition is performed as a greedy sequence tagging procedure under the well–known BIO labelling scheme. This tagging process makes use of three binary classifiers trained to be *experts* on the recognition of B, I, and O labels, respectively. Named Entity classification is viewed as a 4–class classification problem (with LOC, PER, ORG, and MISC class labels), which is straightforwardly addressed by the use of a multiclass learning algorithm.

The system presented here consists of a replication, with some minor changes, of the system that obtained the best results in the CoNLL-2002 NEE task. Therefore, it can be considered as a benchmark of the state–of–the–art technology for the current edition, and will allow also to make comparisons about the training corpora of both editions.

2 Learning the Decisions

We use AdaBoost with confidence rated predictions as learning algorithm for the classifiers involved in the system. More particularly, the basic binary version has been used to learn the I, O, and B classifiers for the NER module, and the multiclass multilabel extension (namely AdaBoost.MH) has been used to perform entity classification.

The idea of these algorithms is to learn an accurate strong classifier by linearly combining, in a weighted voting scheme, many simple and moderately–accurate base classifiers or rules. Each base rule is learned sequentially by presenting the base learning algorithm a weighting over the examples, which is dynamically adjusted depending on the behavior of the previously learned rules.

AdaBoost has been applied, with significant success, to a number of problems in different areas, including NLP tasks (Schapire, 2002). We refer the reader to (Schapire and Singer, 1999) for details about the general algorithms (for both the binary and multiclass variants), and (Carreras and Màrquez, 2001; Carreras et al., 2002b) for particular applications to NLP domains.

In our setting, the boosting algorithm combines several small fixed–depth decision trees, as base rules. Each branch of a tree is, in fact, a conjunction of binary features, allowing the strong boosting classifier to work with complex and expressive rules.

3 Feature Representation

A window W anchored in a word w represents the local context of w used by a classifier to make a decision on that 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 **street**"). The kind of information coded in those features may be grouped in the following kinds:

- **Lexical**: Word forms and their position in the window (e.g., $W(3)$="bank"). When available, word lemmas and their position in the window.
- **Syntactic**: Part-of-Speech tags and Chunk tags.
- **Orthographic**: Word properties with regard to how is it capitalized (*initial-caps*, *all-caps*), the kind of characters that form the word (*contains-digits*, *all-digits*, *alphanumeric*, *roman-number*), the presence of punctuation marks (*contains-dots*, *contains-hyphen*, *acronym*), single character patterns (*lonely-*

initial, *punctuation-mark*, *single-char*), or the membership of the word to a predefined class (*functional-word*[1]), or pattern (*URL*).

- **Affixes**: The prefixes and suffixes of the word (up to 4 characters).
- **Word Type Patterns**: Type pattern of consecutive words in the context. The type of a word is either *functional* (`f`), *capitalized* (`C`), *lowercased* (`l`), *punctuation mark* (`.`), *quote* (`'`) or *other* (`x`). For instance, the word type pattern for the phrase "John Smith payed 3 euros" would be `CClxl`.
- **Left Predictions**: The {B,I,O} tags being predicted in the current classification (at recognition stage), or the predicted category for entities in left context (at classification stage).
- **Bag-of-Words**: Form of the words in the window, without considering positions (e.g., "bank"$\in W$).
- **Trigger Words**: Triggering properties of window words. An external list is used to determine whether a word may trigger a certain Named Entity (NE) class (e.g., "president" may trigger class PER).
- **Gazetteer Features**: Gazetteer information for window words. An external gazetteer is used to determine possible classes for each word.

4 The NER Module

The Named Entity recognition task is performed as a combination of local classifiers which test simple decisions on each word in the text.

According to a BIO labelling scheme, 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. All the words in the train set are used as training examples, applying a *one-vs-all* binarization. When tagging, the sentence is processed from left to right, greedily selecting for each word the tag with maximum confidence that is coherent with the current solution (e.g., O tags cannot be followed by I tags). Despite its simplicity, the greedy BIO tagging performed very well for the NER task. Other more sophisticated representations and tagging schemes, studied in the past edition (Carreras et al., 2002a), did not improve the performance at all.

The three classifiers use the same information to codify examples. According to the information types introduced in section 3, all the following features are considered for each target word: **lexical**, **syntactic**, **orthographic**, and **affixes** in a {-3,+3} window; **left_predictions** in a {-3,-1} window; and all the **word_type_patterns** that cover the 0 position in a {-3,+3} window.

The semantic information represented by the rest of features, namely **bag-of-words**, **trigger_words**, and **gazetteer_features**, did not help the recognition of Named Entities, and therefore was not used.

5 The NEC Module

NEC is regarded as a classification task, consisting of assigning the NE type to each already recognized NE. In contrast to the last year system, the problem has not been binarized and treated in an ECOC (error correcting output codes) combination scheme. Instead, the multiclass multilabel AdaBoost.MH algorithm has been used. The reason is that although ECOC provides slightly better results, its computational cost is also much higher than the required for AdaBoost.MH.

The algorithm has been employed with different parameterizations, by modeling NEC either as a three-class classification problem (in which MISC is selected only when the entity is negatively classified as PER, ORG and LOC) or as a four-class problem, in which MISC is just one more class. The latter turned out to be the best choice (with very significant differences).

The window information described in section 3 is used in the NEC module computing all features for a {-3,+3} window around the NE being classified, except for the **bag-of-words** group, for which a {-5,+5} window is used. Information relative to **orthographic**, **left_predictions**, and **bag-of-words** features is straightforwardly coded as described above, but other requires further detail:

- **Lexical features**: Apart from word form and lemma for each window position, two additional binary features are used: One is satisfied when the focus NE form and lemma coincide exactly, and the other when they coincide after turning both of them into lowercase.
- **Syntactic features**: Part-of-Speech (PoS) and Chunk tags of window words (e.g., $W(3)$.PoS=NN). PoS and Chunk pattern of the NE (e.g., NNPS_POS_JJ for the NE "People_'s_Daily")
- **Affix features**: Prefixes and suffixes of all window words. Prefixes and suffixes of the NE being classified and of its internal components (e.g., considering the entity "People_'s_Daily", "ly" is taken as a suffix of the NE, "ple" is taken as a suffix of the first internal word, etc.).
- **Trigger Words**: Triggering properties of window words (e.g., $W(3).trig$=PER). Triggering properties of components of the NE being classified (e.g., for

[1]Functional words are determiners and prepositions which typically appear inside NEs.

the entity “Bank_of_England” we could have a feature $NE(1).trig$=ORG). Context patterns to the left of the NE, where each word is marked with its triggering properties, or with a functional–word tag if appropriate (e.g., the phrase “the president of United States”, would produce the pattern f_ORG_f for the NE “United_States”, assuming that the word “president” is listed as a possible trigger for ORG).

- **Gazetteer Features**: Gazetteer information for the NE being classified and for its components (e.g., for the entity “Bank_of_England”, features $NE(3)$.gaz=LOC and NE.gaz=ORG would be activated if “England” is found in the gazetteer as LOC and “Bank_of_England” as ORG, respectively.

- Additionally, binary features encoding the length in words of the NE being classified are also used.

6 Experimental Setting

The list of *functional words* for the task has been automatically constructed using the training set. The lowercased words inside a NE that appeared more than 3 times were selected as functional words for the language.

Similarly, a gazetteer was constructed with the NEs in the training set. When training, only a random 40% of the entries in the gazetteer were considered. Moreover, we used external knowledge in the form of a list of trigger words for NEs and an external gazetteer. These knowledge sources are the same that we used in the last year competition for Spanish NEE. The entries of the trigger–word list were linked to the Spanish WordNet, so they have been directly translated by picking the corresponding synsets of the English WordNet. The gazetteer has been left unchanged, assuming interlinguality of most of the entries. The gazetteer provided by the CoNLL-2003 organization has not been used in the work reported in this paper.

In all cases, a preprocess of attribute filtering was performed in order to avoid overfitting and to speed–up learning. All features that occur less than 3 times in the training corpus were discarded.

For each classification problem we trained the corresponding AdaBoost classifiers, learning up to 4,000 base decision trees per classifier, with depths ranging from 1 (decision stumps) to 4. The depth of the base rules and the number of rounds were directly optimized on the development set. The set of unlabelled examples provided by the organization was not used in this work.

7 Results

The described system has been applied to both languages in the shared task, though German and English environments are not identical: The German corpus enables the use of lemma features while English does not. Also, the used trigger word list is available for English but not for German.

The results of the BIO model for the NER task on the development and test sets for English and German are presented in table 1. As will be seen later for the whole task, the results are systematically better for English than for German. As it can be observed, the behaviour on the development and test English sets is quite different. While in the development set the NER module achieves a very good balance between precision and recall, in the test set the precision drops almost 4 points, being the F_1 results much worse. On the contrary, development and test sets for German are much more similar. In this case, recall levels obtained for the language are much lower compared to precision ones. This fact is indicating the difficulty for reliably detecting the beginnings of the Named Entities in German (all common and proper nouns are capitalized). Probably, a non–greedy tagging procedure would have the chance to improve the recognition results.

	Precision	Recall	$F_{\beta=1}$
English devel.	95.65%	95.51%	95.58
English test	91.93%	94.02%	92.96
German devel.	88.15%	71.55%	78.99
German test	85.87%	72.61%	78.68

Table 1: Results of the BIO recognizer for the NER task

Regarding NEC task, optimal feature selection is different for each language: Chunk information is almost useless in English (or even harmful, when combined with PoS features), but useful in German. On the contrary, although the use of left predictions for NEC is useful for English, the lower accuracy of the German system renders those features harmful (they are very useful when assuming perfect left predictions). Table 2 presents NEC accuracy results assuming perfect recognition of entities.

English		German	
features	accuracy	features	accuracy
basic	91.47%	basic	79.02%
basic+P	92.14%	basic+P	79.29%
basic+C	91.60%	basic+C	79.04%
basic+PC	92.12%	basic+PC	79.91%
basic+Pg	93.86%	basic+PCg	81.54%
basic+PG	95.05%	basic+PCG	**85.12**%
basic+PGT	**95.14**%		

Table 2: NEC accuracy on the development set assuming a perfect recognition of named entities

The *basic* feature set includes all lexical, orthographic, affix and bag–of–words information. P stands for Part-of-

Speech features, C for chunking–related information, T for trigger–words features and g/G for gazetteer–related information[2]. In general, more complex features sets yield better results, except for the C case in English, as commented above.

Table 4 presents the results on the NEE task obtained by pipelining the NER and NEC modules. The NEC module used both knowledge extracted from the training set as well as external sources such as the gazetteer or trigger word lists.

Almost the same conclusions extracted from the NER results apply to the complete task, although here the results are lower due to the cascade of errors introduced by the two modules: 1) Results on English are definitely better than on German; 2) Development and test sets present a regular behaviour in German, while for English they are significantly different. We find the latter particularly disappointing because it is indicating that no reliable conclusions can be extracted about the generalization error of the NEE system constructed, by testing it on a 3,000 sentence corpus. This may be caused by the fact that the training set is no representative enough, or by a too biased learning of the NEE system towards the development set.

Regarding particular categories, we can see that for English the results are not extremely dissimilar (F_1 values fall in a range of 10 points for each set), being LOC and PER the most easy to identify and ORG and MISC the most difficult. Comparatively, in the German case bigger differences are observed (F_1 ranges from 52.58 to 80.79 in the test set), e.g., recognition of MISC entities is far worse than all the rest. Another slight difference against English is that the easiest category is PER instead of LOC.

In order to allow fair comparison with other systems, table 3 presents the results achieved on the development set without using external knowledge. The features used correspond to the *basic* model plus Part-of-Speech information (plus Chunks for German), plus a gazetteer build with the entities appearing in the training corpus.

	Precision	Recall	$F_{\beta=1}$
English devel.	90.34%	90.21%	90.27
English test	83.19%	85.07%	84.12
German devel.	74.87%	60.77%	67.09
German test	74.69%	63.16%	68.45

Table 3: Overall results using no external knowledge

Acknowledgments

This research has been 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.

[2] g refers to a gazetteer containing only entities appearing in the training set while G includes also external knowledge

English devel.	Precision	Recall	$F_{\beta=1}$
LOC	95.33%	94.39%	94.86
MISC	89.94%	83.41%	86.55
ORG	86.98%	88.14%	87.56
PER	91.79%	94.68%	93.21
Overall	91.51%	91.37%	91.44

English test	Precision	Recall	$F_{\beta=1}$
LOC	88.14%	90.41%	89.26
MISC	82.02%	75.36%	78.54
ORG	78.40%	80.43%	79.41
PER	86.36%	91.65%	88.93
Overall	84.05%	85.96%	85.00

German devel.	Precision	Recall	$F_{\beta=1}$
LOC	75.72%	73.67%	74.68
MISC	72.34%	42.48%	53.52
ORG	76.89%	63.82%	69.75
PER	83.84%	68.88%	75.63
Overall	77.90%	63.23%	69.80

German test	Precision	Recall	$F_{\beta=1}$
LOC	70.31%	70.92%	70.61
MISC	64.91%	44.18%	52.58
ORG	71.70%	54.08%	61.65
PER	87.59%	74.98%	80.79
Overall	75.47%	63.82%	69.15

Table 4: Final results for English and German

References

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

X. Carreras, L. Màrquez, and L. Padró. 2002a. Named Entity Extraction using AdaBoost. In *Proceedings of the 6th CoNLL*, Taipei, Taiwan.

X. Carreras, L. Màrquez, V. Punyakanok, and D. Roth. 2002b. 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.