Improving the Performance of a Named Entity Extractor by Applying a Stacking Scheme

José A. Troyano, Víctor J. Díaz, Fernando Enríquez, and Luisa Romero

Av. Reina Mercedes s/n 41012, Sevilla (Spain)
troyano@lsi.us.es

Abstract. In this paper we investigate the way of improving the performance of a Named Entity Extraction (NEE) system by applying machine learning techniques and corpus transformation. The main resources used in our experiments are the publicly available tagger TnT and a corpus of Spanish texts in which named entities occurrences are tagged with BIO tags. We split the NEE task into two subtasks 1) Named Entity Recognition (NER) that involves the identification of the group of words that make up the name of an entity and 2) Named Entity Classification (NEC) that determines the category of a named entity. We have focused our work on the improvement of the NER task, generating four different taggers with the same training corpus and combining them using a stacking scheme. We improve the baseline of the NER task ($F_{\beta=1}$ value of 81.84) up to a value of 88.37. When a NEC module is added to the NER system the performance of the whole NEE task is also improved. A value of 70.47 is achieved from a baseline of 66.07.

1 Introduction

Named Entity Extraction involves the identification of words that make up the name of an entity, and the classification of this name into a set of categories. For example, in the following text, the words "Juan Antonio Samaranch" are the name of a person, the word "COI" is an organization name, "Río de Janeiro" is the name of a place and, finally, "Juegos Olímpicos" is an event name:

> El presidente del *COI*, *Juan Antonio Samaranch*, se sumó hoy a las alabanzas vertidas por otros dirigentes deportivos en *Río de Janeiro* sobre la capacidad de esta ciudad para acoger unos *Juegos Olímpicos*.

In order to implement a system that extracts name entities from plain text we have to meet with two different problems, the recognition of a named entity and its classification. Named Entity Recognition (NER) is the identification of the word sequence that forms the name of an entity, and Named Entity Classification (NEC) is the subtask in charge of deciding which is the category assigned to a previously recognized entity.

There are systems that perform both subtasks at once. Other systems, however, make use of two independent subsystems to carry out each subtask sequentially. The second architecture allows us to choose the most suitable technique to each subtask. Named entity recognition is a typical grouping task (or chunking) while choosing its category is a classification problem. In practice, it has been shown [3] that the division into two separate subtasks is a very good option.

Our approach to the NEE problem is based on the separate architecture. In the development of the NER module we have followed the next steps:

– To eliminate from the corpus the information relating to the categories, leaving only the information about the identification of named entity boundaries.
– To apply three transformations to the recognition corpus. Thus we have different views of the same information which enable the tagger to learn in different ways.
– To train TnT with the four corpora available for the NER task (the original and the results of the three transformations).
– To combine the results of the four taggers in order to obtain a consensual opinion. This combination has been carried out applying a stacking scheme, where the results of the different models are used to generate a training database employed in a second stage of learning.

The NEC module has been implemented by a classifier induced from a training database. The database is obtained calculating a feature vector for each entity in the training corpus. The NEC classifier and the classifier used in the stacking scheme have been built with algorithms of the weka package [14].

Experiments show that the three transformations improve the results of the NER task, and that system combination achieves better results than the best of the participant models in isolation. This improvement in the NER task repels positively in the performance of the NEE task. When a NEC module is applied to the previously recognized entities, an improvement of more than four points is achieved with respect to the baseline defined for the NEE task.

The organization of the rest of the paper is as follows. The second section presents the resources, measures and baselines used in our experiments. In section three we show how to improve the NER subtask applying corpus transformations. In section four we describe how to combine the results of the four NER systems with a stacking scheme. Section five presents the NEC module and the results of its use in conjunction with the best NER system. Finally, in section six we draw the final conclusions and point out some future work.

2 Resources and Baselines

In this section we describe the main resources used in our experiments, and the baselines we have employed to measure the improvements achieved.

2.1 The Corpus and the Tagger

This corpus provides a wide set of named entity examples in Spanish. It was used in the NER task of CoNLL-02 [12]. The files are:

– Training corpus with 264715 tokens and 18794 entities
– Test corpus with 52923 tokens and 4315 entities

BIO notation is used to denote the limits of a named entity. The initial word of a named entity is tagged with a B tag, and the rest of words of a named entity are tagged with I tags. Words outside an entity are denoted with an O tag. There are four categories in the corpus taxonomy: PER (people), LOC (places), ORG (organizations) and MISC (rest of entities), so the complete set of tags is {B-LOC, I-LOC, B-PER, I-PER, B-ORG, I-ORG, B-MISC, I-MISC, O}.

The NER task does not need the category information, so we have simplified the tag set removing the category information from the tags. Figure 1 shows a fragment of the original corpus, and its simplified version used in the NER task.

Word	Tag
El	O
presidente	O
del	O
COI	B-ORG
,	O
Juan	B-PER
Antonio	I-PER
Samaranch	I-PER
,	O
se	O
sumó	O
...	...

Word	Tag
El	O
presidente	O
del	O
COI	B
,	O
Juan	B
Antonio	I
Samaranch	I
,	O
se	O
sumó	O
...	...

Fig. 1. Original Corpus and Corpus Tagged Only for the Recognition Subtask

We have choosen the tagger TnT as basis for developing the NER systems presented in this paper. TnT [1] is one of the most widely used re-trainable tagger in NLP tasks. It is based upon second order Markov Models, consisting of word emission probabilities and tag transition probabilities computed from trigrams of tags.

2.2 Measures and Baselines

The measures used in our experiments are, *precision*, *recall* and the overall measure $F_{\beta=1}$. These measures were originally used for Information Retrieval evaluation purposes, but they have been adapted to many NLP tasks.

Precision is computed according to the number of correctly recognized entities, and *recall* is defined as the proportion of the actual entities that the system has been able to recognize:

$$Precision = \frac{\text{correctly extracted entities}}{\text{extracted entities}}$$

$$Recall = \frac{\text{correctly extracted entities}}{\text{actual entities}}$$

Finally, $F_{\beta=1}$ combines recall and precision in a single measure, giving to both the same relevance:

$$F_{\beta=1} = \frac{2\, Precision\ Recall}{Precision + Recall}$$

We will trust in $F_{\beta=1}$ measure for analyzing the results of our experiments. It is a good performance indicator of a system and it is usually used as comparison criterion. Table 1 shows the results obtained when TnT is trained with the original corpus (*NEE_baseline*) and with its simplified version used in the NER subtask (*NER_baseline*), we will adopt these results as the baselines for further experiments in this paper.

Table 1. Baselines. NEE and NER Results with TnT

	Precision	Recall	$F_{\beta=1}$
NEE_baseline	66.28%	65.85%	66.07
NER_baseline	81.40%	82.28%	81.84

The NER baseline is much higher than the NEE baseline because the NER problem is simpler than the whole NEE task. In this paper we will take the approach of improving the NER subtask to build an NEE system (adding a NEC module) that improves the NEE baseline.

3 Improving NER Task Through Corpus Transformation

It seems logical to think that if we have more information before taking a decision we have more possibilities of choosing the best option. For this reason we have increased the number of models as a way of improving the performance of the NER task. There are two obvious ways of building new models: using new training corpora or training other taggers with the same corpus. We have tried a different approach, defining three transformations that give us three additional versions of the training corpus. Transformations can be defined to simplify the original corpus or to add new information to it. If we simplify the corpus, we reduce the number of possible examples and the sparse data problem will be smoothed. On the other hand if we enrich the corpus, the model can use new information to identify new examples not recognized by the original model.

3.1 Vocabulary Reduction

This transformation discards most of the information given by words in the corpus, emphasizing the most useful features for the recognition. We employ a technique similar to that used in [10] replacing the words in the corpus with tokens that contain relevant information for recognition. The goals pursued are:

– To stand out certain typographic features of words, like capitalization, that can help in deciding if a word is part of a named entity or not.
– To give more relevance to words that can help in the identification of a named entity.
– To group several words of the vocabulary into a single entry.

Apart from typographic information there are other features that can be useful in the identification of entities, for example non-capitalized words that frequently appear before, after or inside named entities. We call them trigger words and they are of great help in the identification of entity boundaries.

Both pieces of information, trigger words and typographic clues, are extracted from the original corpus through the application of the following rules:

– Each word is replaced by a representative token, for example, *_starts_cap_* for words that start with capital letters, *_lower_* for words that are written in lower case letter, *_all_cap_* if the whole word is upper case, etc. These word patterns are identified using a small set of regular expressions.
– Not all words are replaced with its corresponding token, trigger words remain as they appear in the original corpus. The list of trigger words is computed automatically counting the words that most frequently appear around or inside an entity.

Figure 2 shows the result of applying these rules to the corpus fragment of Figure 1. Vocabulary reduction leads to an improvement in the performance of the NER subtask. The results of the experiment *TnT-V* are presented in Table 2, we can see that TnT improves from 81.84 to 83.63.

3.2 Change of Tag Set

This transformation does not affect to words but to tags. The basic idea is to replace the original BIO notation with a more expressive one that includes information about the words that usually end a named entity. The new tag set has five tags, the three original (although two of them change slightly their semantic) plus two new tags:

– B, that denotes the beginning of a named entity with more than one word.
– BE, that is assigned to a single-word named entity.
– I, that is assigned to words that are inside of a multiple-word name entity, except to the last word.
– E, assigned to the last word of a multiple-word named entity.
– O, that preserves its original meaning: words outside a named entity.

The new tag set gives more relevance to the position of a word, forcing the tagger to learn which words appear more frequently at the beginning, at the end or inside a named entity. Figure 2 shows the result of applying this new tag set to the corpus fragment of Figure 1. Changing the tag set also leads to better results in the NER task than those obtained with the original corpus. The results of the experiment *TnT-N* are showed in Table 2. In this case, TnT improves from 81.84 to 84.59, the best result of the three transformations studied.

3.3 Addition of Part-of-Speech Information

Unlike previous corpus transformations, in this case we will make use of external knowledge to add new information to the original corpus. Each word will be replaced with a compound tag that integrates two pieces of information:

– The result of Applying the First Transformation (Vocabulary Reduction).
– The part of speech (POS) tag of the word.

In order to obtain the POS tag of a word we have trained TnT with the tagged corpus CLiC-TALP [4]. This corpus is a one hundred thousand word collection of samples of written language, it includes extracts from newspapers, journals, academic books and novels. Figure 2 shows the result of the application of this transformation to the corpus fragment of Figure 1. Adding part of speech information also implies an improvement in the performance of TnT in the NER task. Table 2 presents the results of the experiment *TnT-P*, in this case TnT reaches an $F_{\beta=1}$ measure of 83.12.

Word	Tag
El	O
presidente	O
del	O
_all_cap_	B
,	O
_starts_cap_	B
_starts_cap_	I
_starts_cap_	I
,	O
se	O
lower	O
...	...

a) Vocabulary reduction.

Word	Tag
El	O
presidente	O
del	O
COI	BE
,	O
Juan	B
Antonio	I
Samaranch	E
,	O
se	O
sumó	O
...	...

b) Change of tag set.

Word	Tag
El_det_	O
presidente_noun_	O
del_prep_	O
_all_cap__noun_	B
,_punt_	O
_starts_cap__noun_	B
_starts_cap__noun_	I
_starts_cap__noun_	I
,_punt_	O
se_pron_	O
_lower__verb_	O
...	...

c) Addition of POS information.

Fig. 2. Result of Applying Transformations to the Corpus Fragment Showed in Figure 1

4 Improving the NER Task Through System Combination

The three transformations studied cause an improvement in the performance of the NER task. But we still have room for improvement if instead of applying the transformations separately we make them work together. A superficial analysis of the texts tagged by each model shows that not all the models make the same mistakes. There are some "very difficult" examples that are not recognized by any model, but many of the mistakes can be corrected taking into account the tag proposed by other models. System combination is not a new approach in

NLP tasks, it has been used in several problems like part of speech tagging [6], word sense disambiguation [8], parsing [7] and noun phrase identification [11].

4.1 Stacking

Stacking consists in applying machine learning techniques for combining the results of different models. The main idea is to build a combining system that learns the way in which each model is right or makes a mistake. In this way, the final decision is taken according to a pattern of correct and wrong answers.

We need a training database to be able of learning the way in which every model is right or wrong. Each register in the training database includes all the tags proposed by the participant models for a given word and the actual tag. In order to enrich the training database we have included in the registers the tags of the two previous and the two following words, this way we take into account not only the information associated to a word but also information about its context. We make the database independent of the training and test corpus using an additional corpus of 51533 words to generate the registers. Figure 3 presents an extract of the generated database. For each model there are five tags that correspond, respectively, to the two previous words, the word in question and the two following words. The last tag of each register is the actual tag of the word represented by the register.

TnT	TnT-V	TnT-N	TnT-P	Tag
OOOOB	OOOOB	OOOOB	OOOOB	O
OOOBO	OOOB I	OOOBO	OOOBO	O
OOBOB	OOB I I	OOBOB	OOBOB	B
OBOBO	OB I I O	OBOBO	OBOBO	O
BOBOO	B I I OO	BOBOO	BOBOO	B
OBOOO	I I OOO	OBOOO	OBOOO	O

Fig. 3. Extract of the Generated Database

Apart from allowing the use of heterogeneous information, machine learning has another important advantage over voting: it is possible to choose among a great variety of schemes and techniques to find the most suitable one to each problem. *Bagging* [2] is one of these schemes, it provides a good way of handling the possible bias of the model towards some of the examples of the training database. Bagging is based on the generation of several training data sets taking as base a unique data set. Each new version is obtained by sampling with replacement the original database. Each new data set can be used to train a model and the answers of all the models can be combined to obtain a joint answer. The joint answer is usually obtained by voting. In the experiment *TnT-Stack* we have applied a bagging scheme (using decision trees [9] as base classifier) to combine the results given by *TnT-V*, *TnT-N* and *TnT-P*. Table 2 shows the results of

Table 2. Results of NER Experiments

	Precision	Recall	$F_{\beta=1}$
NER_baseline	81.40%	82.28%	81.84
TnT-V	81.76%	85.59%	83.63
TnT-N	85.04%	84.15%	84.59
TnT-P	81.51%	84.79%	83.12
TnT-Stack	88.68%	88.05%	88.37

this experiment. With this system we obtain the best result (88.37), with an improvement of 6.53 points with respect to the baseline for the NER subtask. There are other authors that also propose stacking as a way of improve the performance of NEE systems, for example [5] and [15]. In both cases they use several taggers to obtain the different opinions combined through the stacking scheme. In this sense, the main contribution of our work is the use of corpus transformation to obtain the different models needed to apply stacking. This way we have the variability necessary to apply system combination without using several training corpora or several taggers.

5 The NEC Module

After the NER stage, we have a text in which possible entities have been identified without specifying the class they belong. At this point, the named entity extractor is completed with a NEC module implemented by a classifier induced from a training database. The database is obtained by calculating a feature vector for each entity in the training corpus. The features used to generate the vectors are the following:

1. Orthographic: we check if an entity contains words that begin with capital letters, or if they contain digits, Roman numbers, quotes, etc.
2. Length and Position: we use as feature the length in words of an entity, as well as the relative position within the phrase.
3. Suffixes: frequent suffixes for each category are calculated from the training corpus.
4. Contexts: relevant words for each category are calculated in a window of three words around the entities found in the training corpus.
5. Content Words: for each category, the set of significant words is calculated eliminating stop words and those in small letters from the examples of the training corpus.

We have used a Support Vector Classifier [13] to implement this classification task. With this method a binary classifier is defined through an hyperplane that optimally divides the classes. Multi-class classifiers can be built combining binary classifiers with a pairwise ensemble scheme. It has been shown that the optimal hyperplane is determined by only a small fraction of the data points, the so-called

Table 3. Results of NEE Experiments

	Precision	Recall	$F_{\beta=1}$
NEE_baseline	66.28%	65.85%	66.07
NEC_alone	78.22%	78.22%	78.22
TnT-Stack-NEC	70.72%	70.22%	70.47

support vectors. The support vector classifier training algorithm is a procedure to find these vectors.

Table 3 shows the standalone performance of the NEC module (assuming no NER errors) and the results of its application to the best NER system studied in this paper (experiment *TnT-Stack-NEC*). An improvement of more than four points with respect to the baseline defined for the NEE is obtained.

6 Conclusions and Future Work

In this paper we have shown that the performance of a named entity extraction system can be improved by applying a stacking scheme. We have split the NEE task into two subtasks: recognition and classification. Our work has been focused on demonstrating that the performance of the NER task can be improved by combining different NER systems that have been obtained with only one tagger (TnT) and one training corpus. The variability necessary to be able to apply system combination has been achieved applying transformations to the training corpus. The baseline for the NER task is improved from 81.84 to 88.37. This performance is similar to state of the art recognizers, with comparable results to those obtained by one of the best NER systems for Spanish texts [3].

As additional conclusion we have demonstrated that the improvement of the NER task also entails an improvement in the complete extraction task. The experiments demonstrate that the baseline defined for the NEE task is surpassed with clarity, improving from 66.07 to 70.47.

Much future work remains. Recognition results are very good but the classification ones are still poor, we have implemented a very simple NEC module and there are still room for improvement in this aspect including new features and experimenting with new learning methods. We are also interested in applying the ideas of this paper to the extraction of entities in specific domains. In this kind of tasks the knowledge about the domain could be incorporated to the system via new transformations. We also plan to take advantage of system combination to help in the construction of annotated corpus, using the jointly assigned tag as agreement criterion in co-training or active learning schemes.

Acknowledgments

We would like to thank the groups CLiC of the University of Barcelona, and TALP of the Polytechnic University of Cataluña, for letting us use the corpus CLiC-TALP.

References

[1] Brants, T.: TnT. A statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference (ANLP00)*. USA (2000)224–231

[2] Breiman, L.: Bagging predictors. In *Machine Learning Journal* 24(1996) 123–140

[3] Carreras, X., L. Màrquez y L. Padró: Named Entity Extraction using AdaBoost. In *CoNLL02 Computational Natural Language Learning*. Taiwan (2002) 167–170

[4] Civit, M.: Guía para la anotación morfosintáctica del corpus CLiC-TALP. *X-TRACT Working Paper WP-00/06*. (2000)

[5] Florian, R.: Named entity recognition as a house of cards: Classifier stacking. In *CoNLL-2002. Computational Natural Language Learning*. Taiwan (2002) 175–178

[6] Halteren, v. H., Zavrel, J. , Daelemans, W.: Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics 27* (2001) 199–230

[7] Henderson, J. C., Brill, E.: Exploiting diversity in natural language processing. Combining parsers. In *1999 Joint Sigdat Conference on Empirical Methods in Natural Language Processing and Very Large Corpora. ACL*. USA (1999) 187–194

[8] Pedersen, T.: A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. In *Proceedings of NAACL00*. USA (2000) 63–69

[9] Quinlan, J.R.: Induction of decision trees. In *Machine Learning* 1 (1986) 81–106.

[10] Rössler, M.: Using Markov Models for Named Entity recognition in German newspapers. In *Proceedings of the Workshop on Machine Learning Approaches in Computational Linguistics*. Italy (2002)29–37

[11] Tjong Kim Sang, E.F., Daelemans, W., Dejean, H., Koeling, R., Krymolowsky, Y., Punyakanok, V., Roth, D.: Applying system combination to base noun phrase identification. In *Proceedings of COLING00*. Germany (2000) 857–863

[12] Tjong Kim Sang, E.F.: Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2002*. Taiwan (2002) 155–158

[13] Vapnik V. N.: *The Nature of Statistical Learning Theory*. Springer. (1995)

[14] Witten, I.H., Frank, E.: Data Mining. Machine Learning Algorithms in Java. Morgan Kaufmann Publishers (2000)

[15] Wu, D., Ngai, G., Carpuat, M.: A stacked, voted, stacked model for named entity recognition. In *CoNLL-2003. Computational Natural Language Learning*. Edmonton (2003) 200–203