Named Entity Recognition Through Corpus Transformation and System Combination

José A. Troyano, Vicente Carrillo, Fernando Enríquez,
and Francisco J. Galán

Department of Languages and Computer Systems
University of Seville
Av. Reina Mercedes s/n 41012, Sevilla (Spain)
troyano@lsi.us.es

Abstract. In this paper we investigate the way of combining different taggers to improve their performance in the named entity recognition task. The main resources used in our experiments are the publicly available taggers TnT and TBL and a corpus of Spanish texts in which named entities occurrences are tagged with BIO tags. We have defined three transformations that provide us three additional versions of the training corpus. The transformations change either the words or the tags, and the three of them improve the results of TnT and TBL when they are trained with the original version of the corpus. With the four versions of the corpus and the two taggers, we have eight different models that can be combined with several techniques. The experiments carried out show that using machine learning techniques to combine them the performance improves considerably. We improve the baselines for TnT ($F_{\beta=1}$ value of 85.25) and TBL ($F_{\beta=1}$ value of 87.45) up to a value of 90.90 in the best of our experiments.

1 Introduction

Named Entity Extraction (NEE) is a subtask of Information Extraction (IE). It involves 1) the identification of words, or sequences of words, that make up the name of an entity and 2) the classification of this name into a set of categories. These categories are predefined and they conform what we call the domain taxonomy. For example, if the domain taxonomy contains the categories PER (people), ORG (organizations), LOC (places) and MISC (rest of entities), in the following text we find an example of each one of them:

> 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 a medio plazo unos *Juegos Olímpicos*.

The words "Juan Antonio Samaranch" conform the name of a person, the word "COI" is an organization name, "Río de Janeiro" is the name of place and, finally, "Juegos Olímpicos" is an event name classified into the category MISC.

If we want to implement a system that extracts named entities from plain text we would meet with two different problems, the recognition of named entities and their classification:

- Named Entity Recognition (NER) is the identification of the word sequence that conforms the name of an entity.
- 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. Therefore, chunking tools can be used to perform the first task, and classification tools for the second one. In practice, it has been shown [4] that the division into two separate subtasks is a very good option.

Our approach to the NEE problem is based on the separate architecture. We have focused the work presented in this paper on improving the performance of the first subtask of the architecture, the NER module. We have applied two main techniques:

- Corpus transformation: that allows us to train the taggers with different views of the original corpus, taking more advantage of the information contained in it.
- System combination: that takes into account the tags proposed by several systems to decide if a given word is part of a named entity.

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 describe the transformations that we have applied to obtain three additional versions of the training corpus. Section four describes the different methods that we have used to combine the tags proposed by each model. In section five we draw the final conclusions and point out some future work.

2 Resources and Baselines

In this section we describe the main resources used: CoNLL-02 corpus, TnT and TBL. We also define the baselines for our experiments with the results of TnT and TBL trained with the original corpus.

2.1 Corpus CoNLL-02

This corpus provides a wide set of named entity examples in Spanish. It was used in the Named Entity Recognition shared task of CoNLL-02 [13]. The distribution is composed of three different files:

– Training corpus with 264715 tokens and 18794 entities.
– Test-A corpus with 52923 tokens and 4315 entities. We have used this corpus as additional training material to estimate the parameters of some of the systems developed.
– Test-B corpus with 51533 tokens and 3558 entities. We have used it only to obtain the final experimental results.

The 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: 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}. We do not need the information about the category for recognition purposes, so we have simplified the tag set by removing the category information from the tags. Figure 1 shows a fragment of the original corpus, and its simplified version.

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

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

Fig. 1. Original corpus and corpus tagged only for the recognition subtask

2.2 Taggers

In order to have views as different as possible of the NER task we have chosen two taggers based upon radically different concepts, TnT and TBL. Both are publicly available and re-trainable.

TBL [3] is a transformation based learning technique that makes use of the knowledge provided by tagging errors. The basic idea of TBL consists of obtaining a set of rules that can transform an imperfect tagging into one with fewer errors. To achieve this goal, TBL implements an iterative process that starts with a naive tagging. This tagging is improved at each iteration learning rules that transform it into another one closer to the correct tagging. TBL has been successfully used in several Natural Language Processing (NLP) tasks like shallow parsing, POS tagging, text chunking or prepositional phrase attachment.

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. As a first step it computes the probabilities from a tagged corpus through maximum likelihood estimation, then it implements a linear interpolation smoothing method to manage the sparse data problem. It also incorporates a suffix analysis for dealing with unknown words, assigning tag probabilities according to the word ending.

2.3 Measures

The measures used in our experiments are, *precision*, *recall* and the overall performance measure $F_{\beta=1}$. These measures were originally used for Information Retrieval (IR) 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:

$$Precision = \frac{\text{correct entities}}{\text{all recognized entities}} \qquad Recall = \frac{\text{correct entities}}{\text{actual entities}}$$

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.

2.4 Baselines

Table 1 shows the NER results obtained when TBL and TnT are trained with the CoNLL-02 corpus, we will adopt these results as baselines for the rest of experiments in this paper. TBL presents better results than TnT in the three measures, this will be a constant in the rest of experiments. In contrast, TBL is slower than TnT, while TnT trains in few seconds TBL needs several minutes to process the entire corpus.

Table 1. Baselines. NER results for TnT y TBL trained with the original version of CoNLL-02 corpus

	Precision	Recall	$F_{\beta=1}$
TnT	84.39%	86.12%	85.25
TBL	85.34%	89.66%	87.45

3 Corpus Transformations

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 decided to increase the number of models.

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 applied to the original corpus give us three additional versions of it. With four different views of the same information, the taggers learn in a different way and the resulting models can specialize in the recognition of named entities of different nature. 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 the added information to identify new examples not recognized in the original model. In the following subsection we describe the transformation explored in our experiments. Figure 2 show the results of applying these transformations to the example of Figure 1.

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

a) Vocabulary reduction.

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

b) Change of tag set.

Word	Tag
El_det_	O
presidente_noun_	O
del_prep_	O
_all_cap__noun_	B
,_punt_	O
_cap__noun_	B
_cap__noun_	I
_cap__noun_	I
...	...

c) Addition of POS information.

Fig. 2. Result of applying transformations to the corpus fragment showed in Figure 1

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 of named entities. We employ a technique similar to that used in [11] replacing the words in the corpus with tokens that contain relevant information for recognition.

One of the problems that we try to solve with this transformation is the treatment of unknown words. These are the words that do not appear in the training corpus, and therefore the tagger can not make any assumption about them. Handling unknown words is a typical problem in almost all corpus based applications, in the case of named entity recognition is even more important because unknown words are good candidates to be part of an entity. The lack

of information of an unknown word can be mitigated with its typographic information, because in Spanish (like many other languages) capitalization is used when writing named entities.

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, it _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, the 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.

Vocabulary reduction leads to an improvement in the performance of TnT and TBL. The results of the experiments (*TnT-V* and *TBL-V*) are presented in Table 2. TnT improves from 85.25 to 86.63 and TBL improves from 87.45 to 88.10.

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 have 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 named 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.

This new tag set give more relevance to the position of a word, forcing the taggers to learn which words appear more frequently at the beginning, at the end or inside a named entity.

Changing the tag set also leads to better results than those obtained with the original corpus. The results of the experiments (*TnT-N* and *TBL-N*) are showed in Table 2. TBL improves from 87.45 to 87.61 and TnT improves from 85.25 to 86.83, the best result achieved with TnT (with an error reduction of over 10%).

3.3 Addition of Part-of-Speech Information

Unlike the 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.

To obtain the POS tag of a word we have trained TnT with the Spanish corpus CLiC-TALP [5]. This corpus is a one hundred thousand word collection of samples of written language, it includes extracts from newspapers, journals, academic books and novels. It is completely tagged, each word has a lemma and a tag that indicates its part of speech and additional information like number, tense or gender. In our experiments we only have used the part of speech information.

We make use of a compound tag in the substitution because the POS tag does not provide enough information to recognize an entity. We would miss the knowledge given by typographical features. For this reason we decided to combine the POS tag with the tag resulting of the application of the vocabulary reduction transformation. The size of the new vocabulary is greater than the obtained with the first transformation, but it is still smaller than the size of the original vocabulary. So, besides the incorporation of the POS tag information, we still take advantage of the reduction of vocabulary in dealing with the unknown word and sparse data problems.

Adding part of speech information also implies an improvement in the performance of TBL and TnT. Table 2 presents the results of the experiments *TnT-P* and *TBL-P*. TnT improves from 85.25 to 86.69 and TBL improves from 87.45 to 89.22, the best result achieved with TBL (an error reduction of over 14%).

Table 2. Results of transformation experiments

	Precision	Recall	$F_{\beta=1}$
TnT	84.39%	86.12%	85.25
TnT-V	85.19%	88.11%	86.63
TnT-N	86.21%	87.47%	86.83
TnT-P	85.33%	88.09%	86.69
TBL	85.34%	89.66%	87.45
TBL-V	87.72%	88.48%	88.10
TBL-N	86.78%	89.07%	87.91
TBL-P	89.14%	89.29%	89.22

4 System Combination

The three transformations studied cause an improvement in the performance of the NER task. This proves that the two techniques employed, adding information

and removing information, can produce good versions of the original corpus through different views of the same text.

But we still have room for improvement if instead of applying the transformations separately we make them work together. We can take advantage of discrepancies among models to choose the most suitable tag for a given word. In the following sections we present the experiments carried out by combining the results of the eight models using different combination schemas. All of these schemas achieve better values for $F_{\beta=1}$ than the best of the participant models in isolation.

System combination is not a new approach in NPL tasks, it has been used in several problems like part of speech tagging [7], word sense disambiguation [9], parsing [8], noun phrase identification [12] and even in named entity extraction [6]. The most popular techniques are voting and stacking (machine learning methods), and the different views of the problem are usually obtained using several taggers or several training corpora. In this paper, however, we are interested in investigate how these methods behave when the combined systems are obtained with transformed versions of the same training corpus.

4.1 Voting

The most obvious way of combining different opinions about the same task is voting. Surprisingly, and despite its simplicity, voting gives very good results, better even than some of the more sophisticated methods that we will present in further subsections. We have carried out two experiments based on this combination scheme:

- *Voting*: one model, one vote. The opinion of each model participant in the combination is equally important.
- *Voting-W*: giving more importance to the opinion of better models. The vote of a model is weighted according to its performance in a previous evaluation.

Table 3 shows the results for these experiments, both achieved better values for $F_{\beta=1}$ than the best of the participant models (*TBL-P* with 89.22). *Voting* reached 89.97 and *Voting-W* 90.02.

4.2 Stacking

Stacking consists in applying machine learning techniques for combining the results of different models. The main idea is to build a 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.

In order to be able to learn the way in which every model is right or wrong, we need a set of examples, known as *training database* in machine learning terminology. Each example in the training database includes the eight tags proposed by the models for a given word (we call them features) and the actual tag (we call it class). From this point of view, deciding the tag given the tags proposed by several models is a typical classification problem.

Figure 3 shows a small database written in "arff" format, the notation employed by *weka* [14] to represent training databases. *Weka* is a collection of machine learning algorithms for data mining tasks, and is the tool that we have used in our stacking experiments.

```
@relation colaboration
@attribute TnT            {O, B, I}
@attribute TnT-VOC-RED    {O, B, I}
@attribute TnT-NEW-TAGS   {O, B, I}
@attribute TnT-POS        {O, B, I}
@attribute TBL            {O, B, I}
@attribute TBL-VOC-RED    {O, B, I}
@attribute TBL-NEW-TAGS   {O, B, I}
@attribute TBL-POS        {O, B, I}
@attribute ACTUAL-TAG     {O, B, I}
@data
I, I, I, B, B, B, I, I,     I
O, O, O, O, O, O, O, O,     O
B, B, B, B, B, B, B, B,     B
I, I, I, I, I, I, I, I,     I
O, I, I, I, I, I, O, O,     I
B, I, I, I, I, I, B, B,     I
O, O, O, O, O, O, O, O,     O
O, O, O, O, O, O, O, O,     O
B, B, B, O, O, B, B, B,     O
```

Fig. 3. A training data base. Each register corresponds to a word

Most of the examples in figure 3 can be resolved with a voting scheme, because the majority opinion agrees with the actual tag. However the last example presents a different situation, six of the eight models assign the tag B to the word in question, while only two (TnT-POS and TBL) assign the correct tag O. If this example is not an isolated one, it would be interesting to learn it and assign the tag O to those words that present this answer pattern.

The number of examples in the training database has a considerable influence in the learning process, using more examples usually leads to a better performance of the classifier. We have used the other evaluation corpus (test A) to generate them, it is independent of the models and it is also independent of the evaluation process. This corpus, with 52923 tokens, provides enough examples to learn a good classifier.

Table 3 shows the results of the experiment *Tree*, carried out using a decision tree [10] as stacking technique. The $F_{\beta=1}$ measure is 89.72, better than the best of participant models (*TBL-P* with 89.22) but worse than the value obtained by voting (90.02). This does not mean that stacking is a worse technique than voting, we will see in the following experiments that stacking achieves better results than voting. The fact is that the generalization process carried out to induce the tree does not cover all the examples, but there is still a feature

of stacking that can compensate this phenomenon, the possibility of merging heterogeneous information.

4.3 Adding Contextual Information

As we have mentioned before one of the advantages of machine learning techniques is that they can make use of information of different nature. While in the voting scheme we only can take into account the tags proposed by the eight models, in a training database we can include as many features as we consider important for taking the correct decision.

One of the most valuable information is the knowledge of the tags assigned to the words around the word in question. To add this information we only have to extend the number of features of each example in the database. Now, besides the tags assigned by each model to a word we will include in the database the tags assigned by the models to the surrounding words. We have carried out two experiments varying the number of words included in the context:

- Tree-1: We only include the tags of the previous and the following words. So each example has twenty four features.
- Tree-2: The tags of the two previous words and the two following words are included. So each example has now, forty features.

In both experiments decision tree is the technique employed to learn the classifier. Table 3 shows the results of the experiments *Tree-1* and *Tree-2*, both improve the results of voting and stacking without contextual information. *Tree-1* got a value of 90.23 in $F_{\beta=1}$ measure, and *Tree-2* got 90.48.

Bigger values of the context increase considerably the number of features in the database and do not lead to better results.

4.4 Bagging

Apart from allowing the use of heterogeneous information, machine learning have 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 this schemas, 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. Generally, bagging leads to better results than those obtained with a single classifier. The price to pay is that this kind of combination methods increase the computational cost associated to learning.

Table 3 shows the results of the experiment *Bagging*. In this experiment we apply this scheme using a decision tree as base learner. With this method we obtain the best result (90.90), with an error reduction of over 38% and 27% with respect to the baselines given, respectively, by *TnT* and *TBL* experiments.

Table 3. Results of combination experiments

	Precision	Recall	$F_{\beta=1}$
Voting	89.67%	90.28%	89.97
Voting-W	89.43%	90.62%	90.02
Tree	88.93%	90.53%	89.72
Tree-1	89.54%	90.92%	90.23
Tree-2	90.18%	90.78%	90.48
Bagging	90.69%	91.12%	90.90

5 Conclusions and Future Work

In this paper we have shown that the combination of several taggers is an effective technique for named entity recognition. Taking as baselines the results obtained when TnT and TBL are trained with a corpus annotated with named entity tags, we have investigate alternative methods for taking more advantage of the knowledge provided by the corpus. We have experimented with three corpus transformations, adding or removing information to the corpus. The three of them improve the results obtained with the original version of the corpus.

The four versions of the corpus, and the two different taggers allow us to build eight models that can be combined using several techniques. All the proposed combination techniques improve the results of the best of the participant models in isolation. We have experimented with voting, stacking using a decision tree as learning technique, and stacking using bagging as learning scheme. Our best experiment achieved an $F_{\beta=1}$ measure of 90.90 what means an error reduction of 38.30% and 27.49% in relation to the baselines given by TnT and TBL. This performance is similar to state of the art NER systems, with comparable results to those obtained by the best system in the CoNLL-02 competition [4] that achieved an $F_{\beta=1}$ value of 91.66 in the recognition task.

We have developed our systems for recognizing named entities in Spanish texts because we are specially interested in this language, but it would be easy to reproduce the experiments in other languages having the corresponding corpus.

Much future work remains. We are interested in applying the ideas of this paper in the recognition 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. Brill, E.: Transformation-based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics 21* (1995) 543–565
4. Carreras, X., L. Màrquez y L. Padró: Named Entity Extraction using AdaBoost. In *CoNLL02 Computational Natural Language Learning*. Taiwan (2002) 167–170
5. Civit, M.: Guía para la anotación morfosintáctica del corpus CLiC-TALP. *X-TRACT Working Paper WP-00/06*. (2000)
6. Florian, R., Ittycheriah, A., Jing, H., Zhang, T.: Named Entity Recognition through Classifier Combination. In *Proceedings of CoNLL-2003*. Canada (2003) 168–171
7. 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
8. 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
9. Pedersen, T.: A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. In *Proceedings of NAACL00*. USA (2000) 63–69
10. Quinlan, J.R.: Induction of decision trees. In *Machine Learning* 1 (1986) 81–106.
11. 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
12. 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
13. 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
14. Witten, I.H., Frank, E.: Data Mining. Machine Learning Algorithms in Java. Morgan Kaufmann Publishers (2000)