Applying Stacking and Corpus Transformation to a Chunking Task

José A. Troyano, Víctor J. Díaz, Fernando Enríquez,
Vicente Carrillo, and Fermín Cruz

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 present an application of the stacking technique to a chunking task: named entity recognition. Stacking consists in applying machine learning techniques for combining the results of different models. Instead of using several corpus or several tagger generators to obtain the models needed in stacking, we have applied three transformations to a single training corpus and then we have used the four versions of the corpus to train a single tagger generator. Taking as baseline the results obtained with the original corpus ($F_{\beta=1}$ value of 81.84), our experiments show that the three transformations improve this baseline (the best one reaches 84.51), and that applying stacking also improves this baseline reaching an $F_{\beta=1}$ measure of 88.43.

1 Introduction

There are many tasks in natural language processing that consist in associating some kind of category to a group of words. Named Entity Extraction, Shallow Parsing, or Semantic Role Identification are three good examples. In this type of tasks, we can identify two subtasks: one that finds the boundaries of the group of words (chunk) and a second process that associates the correct tag to this group. In this paper we present a series of experiments on a clear example of chunking: the NER (Named Entity Recognition) problem. We show that corpus transformation and system combination techniques improve the performance in this task.

The NER task consists in the identification of the group of words that form a named entity. IOB notation is usually employed to mark the entities in a corpus. In this notation, the B tag denotes the beginning of a name, the I tag is assigned to those words that are within (or at the end of) a name, and the O tag is reserved for those words that do not belong to any named entity.

In the development of our experiments we have used a Spanish corpus tagged with NER information, and a re-trainable tagger generator based on Markov Models. In order to improve the performance of the NER task we have defined three transformations that give us modified versions of the training corpus, and we have trained the tagger generator with them to obtain different taggers.

Finally we have applied a stacking (machine learning) scheme to combine the results of the models.

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.

2 Resources, Evaluation and Baseline

The two main resources employed in our experiments are the corpus and the tagger generator. The corpus provides a wide set of named entity examples in Spanish. It was used in the Named Entity Recognition shared task of CoNLL-02 [14] and it is distributed in three different files, a train corpus, and two test corpus. We have used the additional test corpus in stacking experiments to generate the training database.

There are four categories in the corpus taxonomy: PER (people), LOC (places), ORG (organizations) and MISC (rest of entities). However, the NER task does not need the category information, so we have simplified the corpus by 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.

The other main resource is the tagger generator. We have chosen TnT [1], one of the most widely used re-trainable tagger in NLP applications. 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.

Word	*Tag*
La	O
Delegación	B-ORG
de	I-ORG
la	I-ORG
Agencia	I-ORG
EFE	I-ORG
en	O
Extremadura	B-LOC
transmitirá	O
hoy	O
...	...

NEE corpus

Word	*Tag*
La	O
Delegación	B
de	I
la	I
Agencia	I
EFE	I
en	O
Extremadura	B
transmitirá	O
hoy	O
...	...

NER corpus

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

Table 1. Baseline, TnT trained with *NER coprpus*

	Precision	Recall	$F_{\beta=1}$
Baseline	81.40%	82.28%	81.84

To evaluate our experiments, we have used the classical measures *precision*, *recall* and $F_{\beta=1}$. *Precision* is defined as the percentage of correctly extracted entities. *Recall* is defined as the proportion of entities that the system has been able to recognize from the total correct entities in the test corpus. The overall $F_{\beta=1}$ measure combines recall and precision, giving to both the same relevance:

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

We will use $F_{\beta=1}$ measure for comparing 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 *NER corpus* , we will adopt these results as the baseline for further experiments in this paper.

3 Corpus Transformation

In order to have different views of the NER problem, we have defined three transformations that applied to the original corpus give us three additional versions of it. This way, the tagger generator learns in four different ways and the resulting models can specialize in the recognition of named entities of different nature.

3.1 Vocabulary Reduction

In this transformation we employ a technique similar to that used in [12] replacing the words in the corpus with tokens that contain relevant information for recognition. One of the problems that we try to solve is the treatment of unknown words: the words that do not appear in the training corpus and, therefore, the tagger can not make any assumption about them. In the NER task, the lack of information of an unknown word can be mitigated with its typographic information because capitalization is a good indicator of the presence of a proper name. We also include in this transformation the knowledge given by 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 capitalized words. These word patterns are identified using a small set of regular expressions.

Word	*Tag*
La	O
Delegación	B
de	I
la	I
Agencia	I
EFE	I
en	O
Extremadura	B
transmitirá	O
hoy	O
...	...

NER corpus

Word	*Tag*
La	O
_starts_cap_	B
de	I
la	I
_starts_cap_	I
_all_cap_	I
en	O
_starts_cap_	B
transmitirá	O
lower	O
...	...

NER-V corpus

Word	*Tag*
La_det_	O
_starts_cap__noun_	B
de_prep_	I
la_det_	I
_starts_cap__noun_	I
_all_cap__noun_	I
en_prep_	O
_starts_cap__noun_	B
transmitirá_verb_	O
_lower__adv_	O
...	...

NER-P corpus

Fig. 2. Changing the words

– 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.

Figure 2 shows the result of applying vocabulary reduction (*NER-V corpus*). The results of the experiment *TnT-V* are presented in Table 2, we can see that this transformation makes TnT improve from 81.84 to 83.63.

3.2 Addition of Part-of-Speech Information

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 [4]. We make use of a compound tag in the substitution

Table 2. Results of corpus transformation

	Precision	Recall	$F_{\beta=1}$
Baseline	81.40%	82.28%	81.84
TnT-V	81.76%	85.59%	83.63
TnT-P	81.51%	84.79%	83.12
TnT-N	82.77%	86.33%	84.51

because the POS tag does not provide enough information to recognize an entity. We complete this information with the knowledge given by typographical features and trigger words. Figure 2 shows the result of the application of this transformation (*NER-P corpus*). Adding POS information also results in a performance improvement 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.

3.3 Changing the Tags

We replace the original IOB notation with a more expressive one that includes information about the position of words inside and around entities. In order to consider the position inside entities, we have added two new tags E and BE that are assigned, respectively, to words that end a multi-word named entity and to single-word named entities. The meaning of the tags assigned to words inside entities are now:

- 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.

We can also add more information to words outside entities, particularly we are interested in those words that appear just before or after an entity. We split the meaning of the non-informative O tag into four tags:

- BEF, that is assigned to those words that appear before an entity.
- AFT, assigned to words that appear after an entity.

Word	*Tag*
La	O
Delegación	B
de	I
la	I
Agencia	I
EFE	I
en	O
Extremadura	B
transmitirá	O
hoy	O
...	...

NER corpus

Word	*Tag*
La	BEF
Delegación	B
de	I
la	I
Agencia	I
EFE	E
en	BET
Extremadura	BE
transmitirá	AFT
hoy	O
...	...

NER-N corpus

Fig. 3. Changing the tags

– BET, for words that are between two entities.
– O, for words outside entities and not adjacent to entities

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, inside or around a named entity.

Figure 3 shows the result of applying this new tag set to a corpus fragment. 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 84.51, the best result of all the transformations studied.

4 System Combination

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 [10], parsing [8], noun phrase identification [13] 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 stacking behaves when the combined systems are obtained with transformed versions of the same training corpus.

4.1 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 use a training database. Each example in the training database includes the four tags proposed by the models for a given word and the actual tag. From this point of view, deciding the tag given the tags proposed by several models is a typical classification problem. Figure 4 shows a small database written in "arff" format, the notation employed by *weka* [16] 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.

An important advantage of using stacking as combining method is that we can include in the database heterogeneous information. Making use of this feature, we do not only include the tags of a given word in its register, but the tags assigned by the four models to its previous and following words are also included. This way, the registers of our database have twelve features instead of just four corresponding to the four tags of the word we are interested in.

We have used a corpus with new examples to generate the database, so we can ensure that de database used in stacking is independent of the models (training corpus) and it is also independent of the evaluation process (test corpus).

```
@relation combination
@attribute TnT            {O, B, I}
@attribute TnT-V          {O, B, I}
@attribute TnT-N          {O, B, I}
@attribute TnT-P          {O, B, I}
@data
I, I, I, B,      I
O, O, O, O,      O
B, B, B, B,      B
I, I, I, I,      I
O, I, I, I,      I
B, I, I, I,      I
O, O, O, O,      O
O, O, O, O,      O
B, B, B, O,      O
```

Fig. 4. A training data base. Each register corresponds to a word

Table 3. Results of stacking with a decision tree as learning technique

	Precision	Recall	$F_{\beta=1}$
Baseline	81.40%	82.28%	81.84
Decision Tree	87.96%	88.44%	88.20

Table 3 shows the results of the experiment *Decision Tree*, carried out using a decision tree [11] as stacking technique.

A decision tree uses a binary tree to predict the value of a target variable from those of a set of predictor variables. The tree is built by successively splitting nodes according to an information gain criterion. A pruning criterion is also applied to confine the tree size to appropriate limits. This technique is one of the best and most commonly used learning algorithm in classification.

The $F_{\beta=1}$ measure is 88.20, which is better than the baseline (81.84) and also better than the best of participant models in the stacking experiment (*TnT-N* with 84.59).

4.2 Using Other Machine Learning Algorithms

Apart from allowing the use of heterogeneous information, the use of machine learning as combination method has another important advantage: it is possible to choose among a large variety of schemes and techniques to find the most suitable for a specific problem. We have experimented with several machine learning algorithms included in the *weka* package to compare their performance when they are trained with the database that we have created. Most of them are rule-based because this kind of classifiers behaves better with discrete databases:

– Bagging [2]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

Table 4. Results of stacking with different classifiers

	Precision	Recall	$F_{\beta=1}$
Baseline	81.40%	82.28%	81.84
Decision Table	86.52%	87.59%	87.05
Random Tree	86.43%	87.84%	87.13
Part	87.70%	87.84%	87.72
Bagging	88.20%	88.42%	88.31
Ripper	88.88%	87.98%	88.43

replacement the original database. Each new data set can be used to train a model and the answers of all 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. In our experiment we have used decision trees as base learner with this scheme.

- Decision Table [9] is a rule-based classifier. The model consists of a schema, in which only the most representative attributes of the database are included, and a body that has labelled instances of the database defined by the features of the schema.
- Part [15] is the rule-based version of decision trees, it uses a divide and conquer strategy, building a partial decision tree in each iteration and converting the best leaf of the tree into a rule.
- Ripper [5] applies an iterative and incremental pruning process to obtain an error reduction. At a first stage it generates a set of rules that is optimized by generating new rules with randomized data and pruning them.
- Random Tree [16] is an adaptation of decision tree in which every node consider only a subset of the attributes of the database, this subset is chosen randomly.

Table 4 shows the results of the experiments. All of them present good results, the best one is achieved with Ripper (88.43) improving more than six percent points the baseline. 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].

5 Conclusions and Future Work

In this paper we have shown that the combination of several taggers is an effective technique for improving a chunking task like named entity recognition. Taking as baseline the results obtained when a tagger generator (TnT) is trained with a corpus, we have investigated alternative methods for taking more advantage of the knowledge provided by the corpus. By means of corpus transformation we have obtained three different views of the training corpus, with them we have obtained three taggers that improve the results obtained with the original version of the corpus.

Once we had four different taggers we have applied stacking, combining them by generating a training database of examples and applying machine learning. We have experimented with several classifiers reaching a best result of 88.43 in the $F_{\beta=1}$ measure, more than six percent points better than the baseline (81.84). 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 [3].

Much future work remains. We are interested in applying the ideas of this paper in the recognition of entities in specific domains, and in the growth of corpus, using the jointly assigned tag as agreement criterion in co-training or active learning schemes.

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. Cohen, W. W.: Fast Effective Rule Induction. In *Proceedings of the 12th International Conference on Machine Learning*. Morgan Kaufmann (1995) 115–123
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. Kohavi, R.: The Power of Decision Tables. In *Proceedings of the European Conference on Machine Learning*. LNCS 914. (1995) 174–189.
10. Pedersen, T.: A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. In *Proceedings of NAACL00*. USA (2000) 63–69
11. Quinlan, J.R.: Induction of decision trees. In *Machine Learning* 1 (1986) 81–106.
12. 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
13. 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
14. 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
15. Witten, I.H., Frank, E.: Generating accurate rule sets without global optimization. In *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufman (1998). 144–151
16. Witten, I.H., Frank, E.: Data Mining. Machine Learning Algorithms in Java. Morgan Kaufmann Publishers (2000)