Journal of Universal Computer Science, vol. 13, no. 9 (2007), 1287-1299
submitted: 12/11/06, accepted: 15/9/07, appeared: 28/9/07 © J.UCS

Improving the Performance of a Tagger Generator in an Information Extraction Application

José A. Troyano, Fernando Enríquez, Fermín Cruz
José M. Cañete-Valdeón, F. Javier Ortega
(Department of Computer Languages and Systems
University of Seville, Spain
{troyano,fenros,fcruz,canete,javierortega}@us.es)

Abstract: In this paper we present an experience in the extraction of named entities from Spanish texts using stacking. Named Entity Extraction (NEE) is a subtask of Information Extraction that involves the identification of groups of words that make up the name of an entity, and the classification of these names into a set of predefined categories. Our approach is corpus-based, we use a re-trainable tagger generator to obtain a named entity extractor from a set of tagged examples. The main contribution of our work is that we obtain the systems needed in a stacking scheme without making use of any additional training material or tagger generators. Instead of it, we have generated the variability needed in stacking by applying corpus transformation to the original training corpus. Once we have several versions of the training corpus we generate several extractors and combine them by means of a machine learning algorithm. Experiments show that the combination of corpus transformation and stacking improve the performance of the tagger generator in this kind of natural language processing applications. The best of our experiments achieves an improvement of more than six percentual points respect to the predefined baseline.

Key Words: Named Entity Extraction, Corpus Transformation, System Combination, Stacking
Category: I.2.6, I.2.7

1 Introduction

Named Entity Extraction [Borthwick 1999] is a subtask of Information Extraction [McCallum 2005] that involves the identification of words, or groups of words, that hold the nominal information in a natural language text. These names are also classified according to semantic categories (persons, localizations, organizations, etc.) that provide useful information for a subsequent processing of the text. Our approach to this task is corpus-based: given a corpus with a significant set of examples, we train a tagger generator to obtain a named entity extractor.

Our main goal is to improve the performance of a tagger generator given a training corpus. To achieve this we have applied stacking, a machine learning technique that combines the results of a first learning stage (in our case the tagger generator) with a second learner. A system combination technique, such as stacking, needs several taggers in order to have different views of the problem.

In a corpus-based approach, this variability is usually obtained by using several training corpus or several learners. Instead of these methods, we have applied corpus transformation to obtain several versions of the original training corpus. This way, we obtain the different systems needed in stacking without making use of additional data or software.

All the transformations included in this paper are quite easy to implement, so we obtain the new training material with a low cost. Experiments show that these transformations not only allow us to have new versions of the training corpus, but also achieve better results than those obtained when the tagger generator is trained with the original corpus.

We have developed two kinds of experiments. In the first ones, we measure the impact of stacking in the NEE task. In the second ones, we simplify the task by extracting the entities without classifying them into categories; we call this new task Named Entity Recognition (NER).

We have also evaluated the effects of using new phrases and already known phrases for building the database for the stacking technique. New phrases are the ones that have not been used previously to train the taggers of the first learning stage. The information given by a new phrase in the stacking phase is more valuable as it represents a similar situation to the one we will have to face in the real application of the recognizer. The disadvantage of using new phrases for stacking is that the corpus has to be divided in two parts (one for training the taggers and the other for the stacking phase) with the subsequent loss of learning capacity in both phases. In the experiments that make use of already known examples, we reuse the phrases that were employed to train the taggers in the creation of the database, to which new phrases are added. This way, we have the quality given by the new phrases and the quantity given by the already known phrases.

The organization of the rest of the paper is as follows. In the second section we define the NEE and NER tasks; and present the resources, measures and baselines used in our experiments. In section three we describe the transformations that we have applied to obtain the additional versions of the training corpus. Section four presents the experiments carried out with the stacking technique. Finally, in section five we draw the conclusions and point out some future work.

2 The NEE and NER tasks

As we have mentioned before, we have experimented with two different, but related, tasks: named entity recognition and named entity extraction. The first one deals with the identification of the boundaries of a named entity, while the second one also classifies the name into a predefined set of categories.

The chosen language has been Spanish due to the special interest we have in

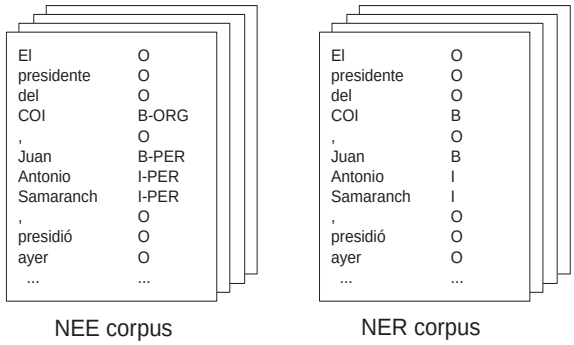

Figure 1: Original NEE corpus and simplified NER corpus.

developing tools to process this language. Nevertheless, the experiments can be reproduced for any other language if we have the corresponding tagged corpus.

Figure 1 shows a fragment of the original training NEE corpus, and its simplified version used in the NER task. IOB notation [Ramshaw 1995] is employed to define the limits of the multi-word names. In this notation, `B` tags denote the beginning of a name, `I` tags are associated to words inside names and `O` tags are reserved to words that appear outside of a name. In the NEE task, these tags are enriched with information about the categories. For example, `B-PER` tags are assigned to words that represent the beginning of the name of a person. There are four different categories: persons (PER), locations (LOC), organizations (ORG) and rest of entities (MISC). This way the number of different tags in the NER task is three, and nine in the NEE task. The training corpus has 9840 phrases, 316248 words, and 22352 entities, while the test corpus has 1915 phrases, 52923 words, and 4351 entities.

The other main resource we have used in our experiments is the tagger generator. We have chosen TnT [Brants 2000], one of the most widely used re-trainable taggers in NLP applications. It is based on 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 to deal with unknown words, assigning tag probabilities according to the word ending.

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

to both:

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

We will trust in $F_{\beta=1}$ measure for comparing the results of our experiments. It is a good performance indicator of a system and it is usually employed as comparison criterion. Table 1 shows the results obtained when TnT is trained with the original NER and NEE corpora; we adopt these results as baselines for further experiments in this paper.

	Precision	Recall	$F_{\beta=1}$
NER-baseline	81.61%	82.70%	82.15%
NEE-baseline	66.54%	66.45%	66.50%

Table 1: Baselines for NER and NEE tasks.

3 Improving NER and NEE through corpus transformation

In the following sections we will present the three transformations included in our experiments. Only one of them (change of tag set) is applicable to both tasks, NEE and NER. The other two are specific for NER (vocabulary reduction) and for NEE (specialization of tags), respectively.

Figure 2 shows how these transformations are applied to the training corpus and how TnT is trained with the resulting corpus in the NER task. We employ a similar schema with NEE transformations, using in this case the transformations defined for the NEE task.

3.1 Vocabulary reduction: NER-V

In this transformation we employ a technique similar to that used in [Rösler 2002], 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; basically we mitigate the lack of information of an unknown word with its typographic information. 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). The new version of the corpus is obtained by applying the following rules:

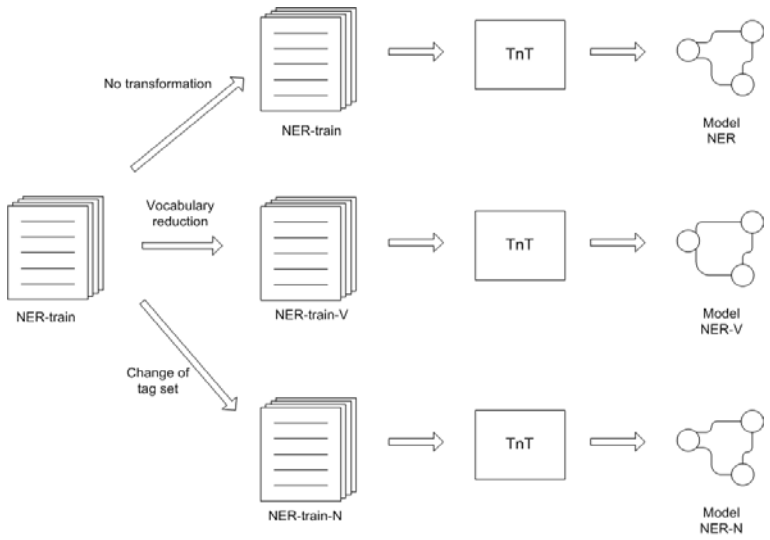

Figure 2: Obtaining different taggers from a single training corpus for the NER task

– Each word is replaced by a representative token, for example, *_upc_* for words that start with capital letters. These word patterns are identified by using a small set of regular expressions. Table 2 shows the regular expressions employed.

– Not all words are replaced with its corresponding token: trigger words remain as they appear in the original corpus.

The results of the experiment *NER-V* are presented in Table 3, we can see that this transformation makes TnT improve from 82.15 to 83.72.

Reg. Expr.	Token	Definition
$[A-Z]+$"."	_abbrv_	Abbreviation
$\hat{}[A-Z].*$	_1word_	First word of a phrase (first letter in upper case)
$[A-Z].*$	_upc_	The first letter is in upper case
$[a-z].*$	_lwc_	The first letter is in lower case
$[A-Z]+$	_aUpc_	All letters are in upper case
$[a-z]+$	_aLwc_	All letters are in lower case

Table 2: Regular Expressions employed in the transformation NER-V.

The loss of information that this transformation causes is not a problem in the NER task because the typographical information is enough to build a good tagger. In the NEE task, however, we can not apply this transformation because we need the original words to decide the category of the entities.

3.2 Change of tag set: NER-N and NEE-N

In this case, 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` and changed slightly the semantic of the original `B` and `I` tags. The meaning of the tags assigned to words inside entities is now:

- `B`, assigned to words at 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 the last word.
- `E`, assigned to the last word of a multiple-word named entity.

We have also added more information to words outside entities, specializing `O` tags for 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.
- `BET`, for words that are between two entities.
- `O`, for words outside entities and not adjacent to entities

Both tasks, NER and NEE, improve with this transformation; results are shown in table 3 (experiments *NER-N* and *NEE-N*). In the case of *NER-N* we obtain the best individual result for this task (84.92).

3.3 Specialization of tags: NEE-PER, NEE-LOC, NEE-ORG and NEE-MISC

This transformation is based on a simple idea: an specific named entity extractor is built for each category. We have defined a transformation for each category in the NEE task. For example, the PER transformation replaces every `B` and `I` tags

of the categories LOC, ORG and MISC by the generic tags `B-XXX` and `I-XXX`, while `B-PER` and `I-PER` are not transformed.

This way we have four new corpora, each one specialized in the extraction of an specific type of entities (PER, LOC, ORG and MISC). None of these corpora solve the whole task, but each one gives us a different, and partial, view of the problem. These corpora arise simpler problems than the original NEE corpus, because we reduce the number of tags from nine to just five (`O`, `B-PER`, `I-PER`, `B-XXX`, and `I-XXX` in the *NER-PER* corpus, for example).

Table 3 shows the results obtained with the four new corpora. In this case results are not comparable to the NEE baseline because the tasks are different, however they are still interesting for combination purposes because we can use these partial results to construct a complete NEE extractor by means of system combination.

	Precision	Recall	$F_{\beta=1}$
NER-baseline	81.61%	82.70%	82.15%
NER-V	81.92%	85.59%	83.72%
NER-N	83.29%	86.60%	84.92%
NEE-baseline	66.54%	66.45%	66.50%
NEE-N	65.94%	68.52%	67.21%
NEE-PER	75.29%	75.74%	75.51%
NEE-LOC	73.26%	73.90%	73.58%
NEE-ORG	71.80%	72.36%	72.08%
NEE-MISC	77.93%	79.76%	78.83%

Table 3: Results of corpus transformation experiments.

4 Stacking

Stacking consists of applying machine learning techniques to combine 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. Therefore the final decision is made 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 tags proposed by the constituent models for a given word together with the right tag. From this point of view, deciding the tag given the tags proposed by several models is a typical classification problem. We generate the database in "arff"

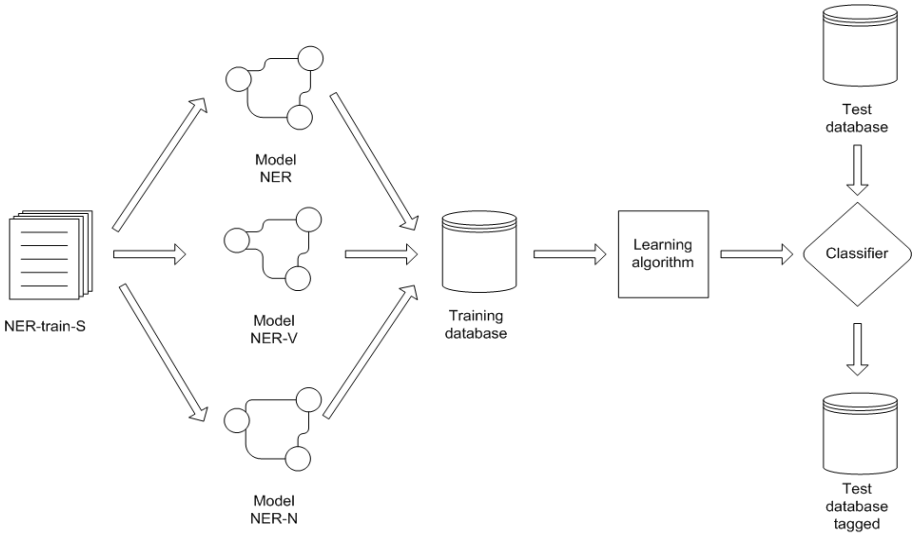

Figure 3: Stacking process for the NER task

format, the notation employed by *weka* [Witten and Frank 2000] to represent training databases.

It would be interesting to have a new set of sentences to generate the database, so we can ensure that the database used in stacking is independent of the models (training corpus) and it is also independent of the evaluation process (test corpus). In order to achieve this independence, we have split the train corpus into two corpus: *train-T* that is used to train the models, and *train-S* that contains fresh frases for building the training database.

Figure 3 shows the way in which we combine several systems in the NER stacking experiments (the scheme for the NEE task is similar). The corpus *NER-train-S* is tagged by the different models obtained with the transformations, and the tags proposed by each system are combined to build the training database that we use in the second learning phase. A similar process is applied to the test corpus in order to generate the test database.

We can include heterogeneous information in the database. Making use of this feature, we do not only include the tags of a given word in its register, we also add the tags assigned by the four models to its previous and following words. In the NEE experiments, we can also combine the tags provided by full systems, like *NEE-V*, with less informative tags (`B-XXX` or `I-XXX`) obtained from specialized systems like *NEE-PER*.

We have made experiments with different partitions of the original corpus, to test the effect of using more or less data dedicated to the tagger training and

to the creation of the stacking database. In all of them we have used *decision tables* [Kohavi 1995] as the learning algorithm due to its good performance with discrete databases like ours.

Stacking (and System combination in general) is not a new approach in NPL tasks, it has been used in several problems like part of speech tagging [Halteren et al. 2001], word sense disambiguation [Florian and Yarowsky 2002], parsing [Henderson and Brill 1999], noun phrase identification [Tjong et al. 2000] and even named entity extraction [Florian et al. 2003]. We can not directly compare the results because the tasks and the corpora employed are different. However, we can use the improvement with respect of the best participant system as an indicative measure. In these approaches the improvement varies from 0.35 percentual points [Halteren et al. 2001] to three percentual points [Florian et al. 2003], and all of them need several learners to obtain the different views of the problems needed in stacking. In our experiments we achieve similar results, with an improvement of 6 percentual points in the NER task and 3 points in the NEE task. We also get to these results without needing additional training material and not even different taggers to create the variability needed for the stacking process.

In the following sections we show the results of our experiments with stacking using two different strategies to create the training database.

4.1 Using new sentences for building the database

The first strategy consists of following the intuition of using always new sentences for creating the training database. We have called these experiments *stack-new* and only the sentences of the *train-S* partition are used to create the database, while the sentences of the *train-T* partition are kept back solely for training the tagger. In figure 4 it has been emphasized with a dash border the only portion of the corpus that is used in these stacking experiments.

The curves with dash lines in graphs of figures 5 and 6 show for the NER task and the NEE task, respectively, the results for this experiment with different partitions of the training corpus. At each point, a percentage of the corpus is used to train the models (*train-T*) and the rest of the sentences to create the database for stacking (*train-S*).

The results prove that the application of the stacking technique is positive. The best result obtained for the NER task is 87.97%, while for the NEE task is 67.33%. In both cases, the best results of the previous experiments are surpassed, with an improvement of more than five points respect to the baseline in the NER task (82.15%) and close to one point for the NEE task (66.50%). This smaller improvement in the NEE task is because it represents a more difficult problem, and therefore, it needs more training data. In the following section we will see

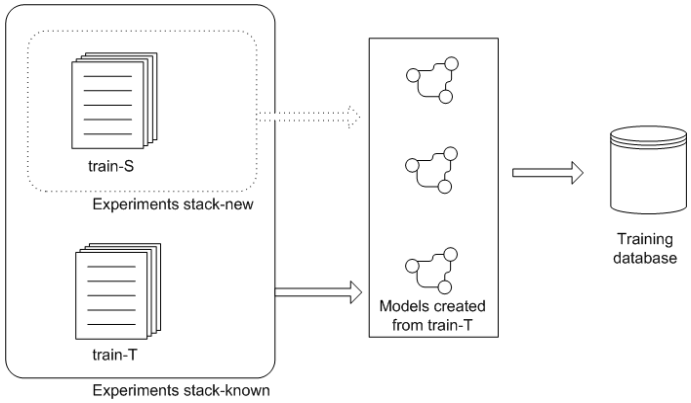

Figure 4: Using new or known examples to build the training database

how the NEE task is benefited by a database with more examples although some of them are of less quality.

In regards to the effect of the percentage of data dedicated to the training of the tagger (*train-T*) and to the stacking phase (*train-S*), there are different behaviours observed for the two tasks. In the NER task, the maximum is reached when little more than the 50% of the data is dedicated to train the models, and the performance falls from that point. This proves that the first training phase does not need all the data to reach an optimum performance and that leaves many sentences available for achieving a better performance from stacking. In the NEE task, however, the maximum is not reached until the 85% of the sentences is used for the first training phase. As it is a more difficult task (there are more tags involved), more sentences are needed so the taggers can offer an acceptable performance, leaving therefore less data available for stacking. In the case of the NEE task, the corpus is not big enough to solve the problem correctly and that is the reason why the stacking technique achieves a smaller improvement.

4.2 Using new and known sentences for building the databases

In these experiments (that we have called *stack-known*), the training database is built from the new sentences of *train-S* and from the sentences of *train-T* already used to train the taggers. In this case, all the elements of the figure 4 are involved. This way we obtain a bigger database, as all the words of the training corpus are used, but the quality is lower because the words of *train-T* have already been used in the first learning phase and they offer less information.

The results of the experiments for the NER and NEE tasks are shown with the solid curves of the graphs in figures 5 and 6. The best result obtained for the

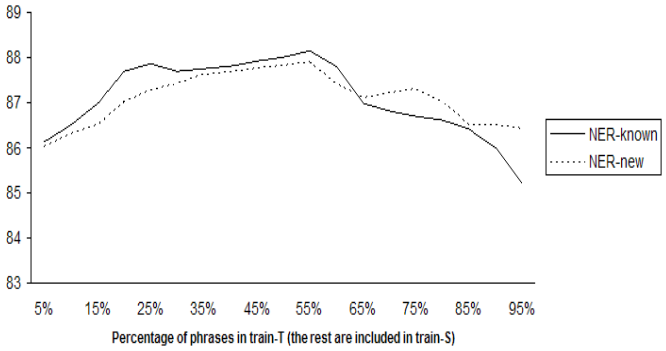

Figure 5: Different partitions of train corpus in NER experiments ($F_{\beta=1}$)

NER task is 88.22%, while in the NEE task it is 69.48%. This results are the best improvements respect to the baselines, surpassing in more than 6 percentual points the NER one and 3 points the NEE one.

The fact that the curves of the *stack-known* experiments surpass the *stack-new* experiments (except for the final stretch of the NER) proves that a bigger stacking database has a positive effect even when part of the examples used does not offer a valuable information.

It is specially significant the improvement observed in the NEE task. In the *stack-new* experiments the contribution of the stacking phase was smaller because of the complexity of the task and the lack of examples to solve it correctly. Now, with the insertion of the known examples, the lack of data is corrected and we have enough examples to exploit the possibilities of the stacking phase.

5 Conclusions

We have investigated how to use a stacking scheme in an Information Extraction domain, applying corpus transformation to obtain the taggers employed in the first learning phase. We have applied this idea to two NLP tasks, Named Entity Extraction and Named Entity Recognition. In both tasks we obtain satisfactory results. The main conclusions of our work are:

- The transformations studied improve the results of the original corpus. The improvement is specially significant in the NER task, where we reach an $F_{\beta=1}$ measure of 84.92 (from the baseline 81.84) in the *NER-N* experiment.
- We have managed to improve the performance of the tagger by using the stacking technique. Despite of what is done in other proposals that also em-

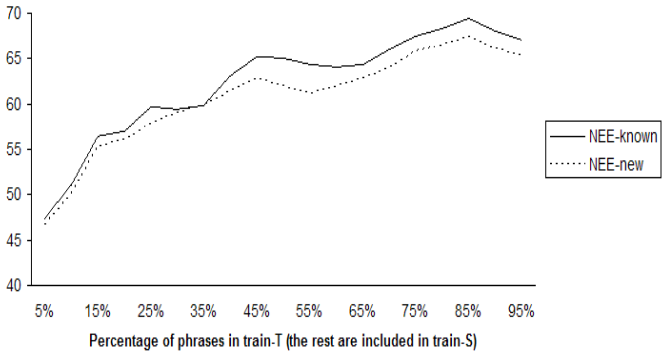

Figure 6: Different partitions of train corpus in NEE experiments ($F_{\beta=1}$)

ploy system combination techniques, we have made it without using neither any additional training material nor different taggers.

- The improvement achieved using the stacking technique reaches 6 percentual points in the NER task and 3 points in the NEE task.
- We have evaluated the effect of using only new data for creating the database for stacking and we have compared it with a strategy in which the sentences already used in creating the model are added to those new sentences. In both tasks (NER and NEE) the second strategy achieved better results, proving that more training data is always welcome, even when it is not *fresh data*.
- We have completed a research on how different partitions of the training corpus in the fragment dedicated to train the tagger (*train-T*) and to generate the stacking database (*train-S*) affect the global performance.

We are interested in applying the ideas of this paper to the recognition of entities in specific domains. In this kind of tasks the knowledge about the domain could be incorporated into 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.

References

[Borthwick 1999] Borthwick, A., 1999. A Maximum Entropy Approach to Named Entity Recognition *PhD thesis*, New York University.

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

[Florian and Yarowsky 2002] Florian, R., Yarowsky, D., 2002. Modeling Consensus: Classifier Combination for Word Sense Disambiguation. In *Proceedings of EMNLP'02*, Philadelphia, pp 25–32.

[Florian et al. 2003] Florian, R., Ittycheriah, A., Jing, H., Zhang, T., 2003. Named Entity Recognition through Classifier Combination. In *Proceedings of CoNLL-2003*, Canada, pp 168–171.

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

[Henderson and Brill 1999] Henderson, J.C., Brill, E., 1999. 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, pp 187–194.

[Kohavi 1995] Kohavi, R., 1995. The Power of Decision Tables. In *Proceedings of the European Conference on Machine Learning*, LNCS 914, pp 174–189.

[McCallum 2005] MCCallum, A., 2005. Information extraction: distilling structured data from unstructured text. In *ACM Queue*, Vol. 3, Iss. 9, pp 48–57.

[Ramshaw 1995] Ramshaw,L. A., Marcus, M.P., 1995. Text Chunking using Transformation-based Learning. In *proceedings of the Third ACL Workshop on Very Large Corpora*, Cambridge, USA, pp 82–94.

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

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

[Witten and Frank 2000] Witten, I.H., Frank, E., 2000. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers