# Improving Term Extraction by System Combination Using Boosting

Jordi Vivaldi[1], Lluís Màrquez[2], and Horacio Rodríguez[2]

[1]Institut Universitari de Lingüística Aplicada, Universitat Pompeu Fabra
La Rambla, 30–32. E-08002, Barcelona, Catalonia
jorge.vivaldi@info.upf.es
[2]TALP Research Center, Universitat Politècnica de Catalunya
Jordi Girona Salgado 1–3. E-08034, Barcelona, Catalonia
{lluism,horacio}@lsi.upc.es

**Abstract.** Term extraction is the task of automatically detecting, from textual corpora, lexical units that designate concepts in thematically restricted domains (e.g. medicine). Current systems for term extraction integrate linguistic and statistical cues to perform the detection of terms. The best results have been obtained when some kind of combination of simple base term extractors is performed [14]. In this paper it is shown that this combination can be further improved by posing an additional learning problem of how to find the best combination of base term extractors. Empirical results, using AdaBoost in the metalearning step, show that the ensemble constructed surpasses the performance of all individual extractors and simple voting schemes, obtaining significantly better accuracy figures at all levels of recall.

## 1 Introduction

As most scientific disciplines evolve in an increasingly faster manner, the creation of new terms grows continuously and their timelife decreases. This context can explain the growing interest in Automatic Term Extraction (TE) Systems. Terms, like words, are lexical units that designate concepts in a thematically constrained domain. Terms cannot be distinguished from general language words just by looking at their forms. It has been empirically shown (see [15]) that term structure coincides with that of words and, often, terms take complex forms, i.e., they are made up of more than one lexical unit. Both terms and words are created and manipulated according to the same linguistic rules. Sometimes, however, a word can be considered a term only in some of its forms.

Usually, the construction of large term repositories has been carried out by terminologists. This task demands a massive manual intervention, which is unfeasible in accordance with today's requirements. Hence, there is a strong need to react quickly (and in a standardized way) so as to comply with current requirements of information. Many techniques have been applied to TE but none of them has proved to be fully successful in isolation. Differences in source texts,

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 515–526, 2001.

text specialization levels, end-user profiles and purposes, and level of automation account for this fact.

**Current Status**. Most current TE systems follow either a statistical or a linguistic approach [8]. Recently, however (as discussed below) some hybrid approaches are trying to overcome the limitations of those purely one-sided approaches and have included both linguistic and statistical elements.

Linguistics based approaches are mainly based on syntactic patterns. Usually, terms are described by a regular expression based on the Part-of-Speech of the sequence of words found in the text under analysis. TERMS [7] is a prototypical system following this approach. The approach followed by LEXTER [3], a well-known TE system, is quite similar but uses knowledge of what is known not to be a term. In spite of their linguistic approach, both systems include some basic statistical information. A different linguistic approach, shown in [1], benefits from the decomposition of term candidates in their Greek and Latin forms.

Statistical based systems range from the simplicity of frequency counts to the calculation of complex statistical indicators for measuring the collocational strength of the words occurring in the term candidates. The main problems found by these approaches are that frequent words or high-scored collocational pairs are not limited to terms but may occur with general language. Other approaches include some linguistic data to overcome these limitations. This information may be used a priori, as in ACABIT [5], or a posteriori. TRUCKS, a modern hybrid system [11], provides a more balanced use of these kinds of information.

The main shortcomings encountered in most term extractors are[1]: 1) *Noise*. Many of the candidates are not real terms and have to be rejected by a manual post-process. This problem is mainly posed by linguistic systems and has to do with the inadequacy of purely syntactic patterns for isolating and detecting terms. 2) *Silence*. Some of the actual terms are not detected by the TE system. This problem usually affects systems including some kind of statistical knowledge. Often, these limitations are related to the difficulty in dealing with mono–lexeme or coordinated terms.

Term extraction provides an appropriate framework for combining some of the techniques typically used for this task. In fact, the combination of multiple classifiers is a technique that has been successfully applied in several NLP tasks, e.g., tagging, parsing, or text classification and filtering, leading to significant improvement on the results obtained by individual methods. Recently, a TE system has been proposed based on combining the results obtained from different individual term extractors (TE) and applying different kinds of voting [14]. Since those TE's are based on very different Knowledge Sources, a further attempt can be made. This would imply using a metalearning approach to learn to combine such TE's in a way that each TE would be chosen in the situation it performs best. In this paper we follow this approach using a boosting algorithm.

[1] *Noise* and *silence* are commonly used in the terminology domain as complementary of the terms *precision* and *recall*, respectively.

**Overview**. The paper is organized as follows: Section 2 is devoted to briefly explain the overall organization of the system and the individual TE's. Section 3 describes the experiments carried out to evaluate the individual TE's and the basic way of combination. In section 4 the boosting approach is presented and tested. Finally, section 5 summarizes some qualitative conclusions.

## 2 A System Proposal for Term Extraction

The aim of our TE system is to analyze a set of textual documents in the medical domain and produce a list of term candidates (tc) ordered by their *termhood*. In order to achieve this objective the following is proposed: 1) Building a number of individual TE's where each of them is based on a different kind of information; and 2) Combining the results (i.e., the set of candidates and the certainty factor of each candidate) of each TE. Once obtained, the list of tc can be processed in several ways depending on the intended use of the extracted terms, ranking from automatic acceptance of those tc's having a termhood over a certain threshold to manual checking of best scored tc by a terminologist.

The architecture of the system is depicted in figure 1. There are three main modules,which take part after a linguistic pre-processing:

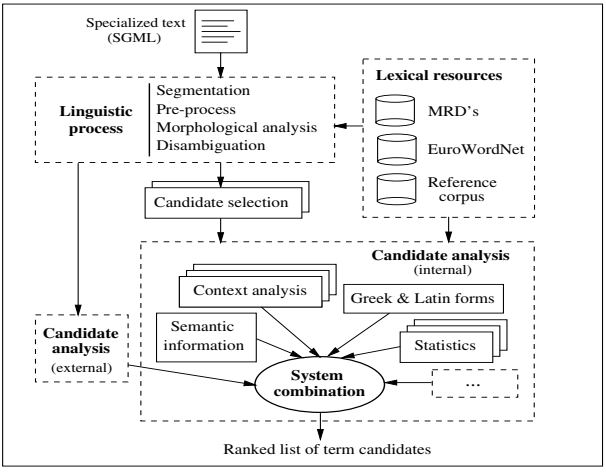


**Fig. 1.** Architecture of the system proposed

1. *Candidate Selection*. It consists of the selection of the sequences of units that can be potentially terminological.
2. *Candidate Analysis*. It consists of a number of TE's that have to score the term candidates with a termhood measure.
3. *System Combination*. The results of the different extractors are combined in order to produce the final set of candidates. This is mainly the issue that will be addressed in this paper.

### 2.1 Individual Term Extraction Systems

Four families of TE's, corresponding to four different kinds of sources, have been implemented. We include below a short description of each TE.

**Semantic content extractor**. This module is based on the idea that the more likely it is that the components of the tc belong to the domain the more likely it is that the tc will be a real term (i.e. medicalhood is a good indicator of termhood). The module uses EuroWordNet (EWN)[2] to determine whether a given word belongs to the medical domain or not. EWN presents some important limitations: It is a general purpose (not restricted to medical domain) ontology and it lacks domain information. It was chosen, however, because it has a relatively high coverage of medical domain entries and a good coverage of Spanish language.

The lack of domain information was tackled by identifying and marking about 30 medical borders (mb) and assuming that all hyponyms that fall under these borders belong to the medical domain. Thus, we may say that "disease" constitutes a mb as all diseases registered in EWN are hyponyms of this synset.

To cope with polysemy a "medical coefficient" (mc) was defined to measure the termhood of each tc. This coefficient can be computed straightforwardly as the ratio between the number of medical senses and the total number of senses registered in EWN. Despite its simplicity, this way of computing mc's works fairly well. In [15] some improvements have been proposed and evaluated. This coefficient is used to define a threshold of medical sense. Any noun with an mc higher than such value is considered to have a medical sense. This coefficient can be considered a measure of the specialization of a word. Due to EWN particular features, this method is prone to detect monosemic non-highly specialized words.

**Greek and Latin forms**. The medical vocabulary of many languages includes words that can be split up into their Greek and Latin (G&L) word forms. This feature is important because such words are highly specialized and do not occur in standard lexica. It is relatively easy to split such words and obtain the meaning of their components. This method has high accuracy although very limited coverage. Therefore, it is a good candidate to be combined with other TE's.

**Context analyzer**. Our approach is mainly based on Maynard's TE system [11] with minor improvements. The basis for context analysis are: a) Words surrounding "prime" tc's, (i.e, already known terms or tc's having a high score) can become useful clues for other terms, and b) Among context words, those that are "prime" tc's and semantically similar to the tc under evaluation provide additional information. Thus, our context factor (cf) has two components: a lexical context factor and a semantic context factor. The "lexical cf" is based on words that surround "prime" candidates while the "semantic cf" depends on the conceptual distance between the tc and the "prime" terms that appear in their context. Three choices have to be made: 1) The selection of the "prime" tc,

[2] EWN [16] is a general-purpose multilingual lexical DB based on Princeton WordNet and covering Spanish and other european languages. WordNet's are structured in lexical-semantic units linked by basic semantic relations.

2) The selection of the context window, and 3) The definition of an appropriate similarity measure. We have considered "prime" term candidates those having a maximal mc (a bootstrapping approach could have been followed instead, starting with an initial list of true tc's). We have experimented with several context window sizes and different relative weighting of lexical and semantic factors. The semantic distance was computed on the EWN hierarchy (see [15]).

**Collocational analysis**. We have also used some traditional statistical methods to rank candidates. Several techniques involving different association measures between the words included in poly-lexeme terms, e.g. noun–adjective (NJ) and noun–preposition–noun (NPN), have been tried. Our intuition is that two components having a high association in a medical domain are more likely to form a term. Three association criteria (loglike, mutual information, and cubed mutual information, $MI^3$ [5]) have been considered. Most of these methods have been previously used for the related task of detecting collocations.

Note that neither the context analyzer extractor nor the G&L extractor impose constraints on the length and composition of the tc. Therefore, both methods can be applied to any pattern accepted by the Candidate Selection Module. The association measures involved in the collocational analysis extractor have been calculated only between the two main words appearing in the pattern (i.e., noun–adjective in the NJ pattern and $noun_1$–$noun_2$ in the NPN pattern). Finally, the semantic content extractor can be applied to any pattern having at least one component with a mc above the threshold. In practice, however, the results were satisfactory only for noun (N) and NJ patterns.

## 3 System Evaluation

The system was trained and tuned using a Spanish corpus (henceforth corpus#1) taken from the IULA LSP corpus[3]. It consists of a number of abstracts of medical reports on asthma that amount to 100,000 words, manually annotated. A shorter (10,000 words) corpus (henceforth corpus#2), was used for testing.

As previously said, the system can be applied to any pattern accepted by the Candidate Selection Module (214 different patterns were detected in corpus#1, see [15]) although not all TE's work on all patterns. However, the distribution of real terms is highly biased and only three patterns actually occur on our corpora: 696 N terms, 664 NJ terms, and 86 NPN terms were manually detected in corpus#1. All along the paper, we will present the average results over the three patterns (ALL). Additionally, the results on the most frequent pattern (N) will be presented separately.

**Results**. Individual methods present different recall/precision behaviours, reflecting the differences of the involved knowledge sources. The results for N and

[3] This corpus has been collected at the Institute for Applied Linguistics of the Universitat Pompeu Fabra. See `http://www.iula.upf.es/corpus/corpubca.htm`.

the other patterns present high differences as well. We summarize next the results of individual methods (see [14] or [15] for details).

The Semantic Content TE has a limited coverage (due to its dependence on EWN). For the N pattern the precision achieved was 96.3% with a recall of 30%. For the whole problem, the precision and recall values are 97.2% and 23.6%, respectively. Figures obtained for the G&L forms were 90% in precision and 8% in recall. Best results for the Context analyzer (cf) were 81.6% and 19.2% in recall for the N pattern (64%–21% on the whole problem) although in this case still accurate results can be obtained at higher recall levels, as shown in figure 2. Best results for collocational methods were obtained using $MI^3$: 50.3%–15.1%.

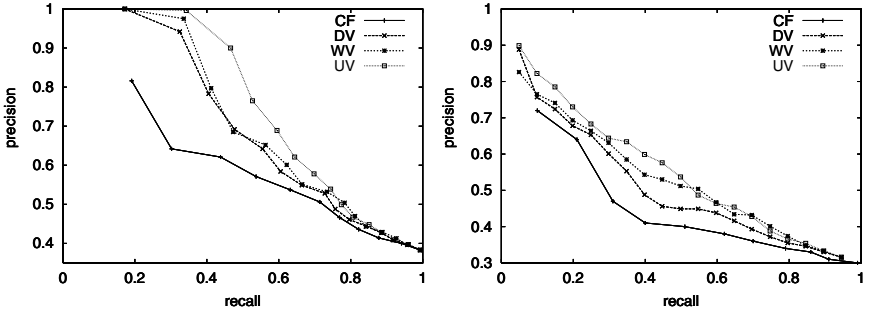


**Fig. 2.** Precision–recall curves obtained by the individual cf extractor and the combination schemes DV, UV, WV on the N (left) and ALL (right) patterns

Regarding the combination of methods, several voting schemes were tried, following [9]: a) Simple voting, i.e., each method reports a term/no term status from each single term extractor. It allows two variations: democratic (DV) and non–democratic voting (UV) according to whether all the TE's have the same weight or not. b) Numeric voting, i.e, each tc is considered according to its termhood value provided by each TE. We experimented with several forms of combination: maximum, minimum, median and weighted (WV). In the UV voting scheme, several TE's are considered with priority in some cases, e.g., G&L forms, Semantic Content with mc=1, etc. In WV, the results of each extractor are normalized according to the maximum score of the corresponding TE.

All combination methods were systematically better at all recall levels, as shown in figure 2, where the best individual method, context factor (cf), is compared with three forms of combination, i.e. DV, UV, WV. For instance, in the case of nouns, the best recall for a simple method is 19.1% (precision 81,6%); for combined methods recall increases up to 32% (precision 99.1%). The results are clearly better for nouns than for the poly–lexeme patterns. This may be due to the fact that NJ and NPN patterns are much more sensitive to the EWN missing data and so the treatment of some kind of adjectives should be improved. We also found that about 15% of terms were not detected due to tagging mistakes, term variation and missing dictionary entries.

## 4 Using Boosting to Combine Basic Term Extractors

In the previous section we have shown how the voting combination of several TE's leads to a significant performance improvement, comparing to the individual results. In this section, we will see how this combination step can be further improved by considering the new learning problem of "how to find the best combination of individual term extractors on the training examples", which will be addressed by means of a boosting–based learning algorithm.

In machine learning approaches, the combination of a set of heterogeneous classifiers is usually performed by defining a metalearning step in which a meta–level classifier is trained to characterize the situations in which each of the base classifiers is able to make correct predictions. *Stacked generalization* [17] is probably the most popular exemplar among the existing approaches for metalearning.

In our situation, the approach can be simpler since the base TE's perform their predictions based on some external linguistic knowledge sources, and do not perform a real training process. Thus, the problem of learning how to combine them can be directly posed as a learning problem in which the predictions of the base TE's are codified as regular features in order to complete the descriptions of training examples. See section 4.2 for details about the features used.

### 4.1 AdaBoost Algorithm

In this section the generalized AdaBoost algorithm with confidence–rated predictions is briefly sketched. We assume that the reader is familiar with the related concepts (see [12] otherwise). It has to be noted that this algorithm has been applied, with significant success, to a number of NLP disambiguation tasks, such as: Part–of–speech tagging and PP–attachment [2], text categorization [13], and word sense disambiguation [6].

The purpose of boosting is to find a highly accurate classification rule by combining many *weak classifiers* (or weak hypotheses), each of which may be only moderately accurate. The weak hypotheses are learned sequentially, one at a time, and, conceptually, at each iteration the weak hypothesis is biased to classify the examples which were most difficult to classify by the preceding weak hypotheses. The final weak hypotheses are linearly combined into a single rule called the *combined hypothesis*.

Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be the set of $m$ training examples, where each instance $x_i$ belongs to an instance space $\mathcal{X}$ and $y_i \in \{-1, +1\}$ is the class or label associated to $x_i$ (which, in this case, stand for *non-term* and *term*). The AdaBoost algorithm maintains a vector of weights as a distribution $D_t$ over examples. At round $t$, the goal of the weak learner algorithm is to find a weak hypothesis $h_t : \mathcal{X} \to \mathbb{R}$ with moderately low error with respect to the weights $D_t$. In this setting, weak hypotheses $h_t(x)$ make real–valued confidence–rated predictions. Initially, the distribution $D_1$ is uniform, but after each iteration, the boosting algorithm increases (or decreases) the weights $D_t(i)$ for which $h_t(x_i)$ makes a bad (or good) prediction, with a variation proportional to the confidence $|h_t(x_i)|$. The final hypothesis, $f : \mathcal{X} \to \mathbb{R}$, computes its predictions using a

weighted vote of the weak hypotheses $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. For each example $x$, the sign of $f(x)$ is interpreted as the predicted class ($-1$ or $+1$), and the magnitude $|f(x)|$ is interpreted as a measure of confidence in the prediction. Such a function can be used either for classifying new unseen examples or for ranking them according to the confidence degree. The latter is the goal in the problem of term extraction.

**Weak Rules**. In this work we have used weak hypotheses which are simple rules with real–valued predictions. Such simple rules test the value of a boolean predicate and make a prediction based on that value. The predicates used refer to the attributes that describe the training examples (e.g. "the word *health* appears in the context of the term candidate", or to the predictions given by the individual term extractors (e.g. "cf's confidence is *high*"). Formally, based on a given predicate $p$, weak hypotheses $h$ are considered that make predictions of the form: $h(x) = c_0$ if $p$ holds in $x$, and $c_1$ otherwise. Where the $c_0$ and $c_1$ are real numbers. See [12] for the details about how to calculate the $c_i$ values given a certain predicate $p$ in the AdaBoost framework.

This type of weak rules can be seen as extremely simple decision trees with one internal node and two leaves, which are sometimes called *decision stumps*. Furthermore, the criterion for finding the best weak rule can be seen as a natural splitting criterion and used for performing decision–tree induction [12]. In this way, we can consider an additional parameter in the learning algorithm that accounts for the depth of the decision trees induced at each iteration.

### 4.2 Feature Representation

All tc's have been considered training examples, and, thus, all three patterns have been treated simultaneously. Each example consists of the set of occurrences of the corresponding tc in corpus#1 (although, for practical reasons, we have limited the number of occurrences to 10). The number of examples in the corpus is 4,693, from which 1,446 are N, NJ, or NPN terms (the exact proportions are presented in section 3). The set of features used for training purposes includes the results of several variants of the individual TE's together with some of the data used by such TE's. They are the following:

1. The medical coefficient (mc).
2. "G&L form prediction", which is 1 when the tc has been recognized by the corresponding TE, and 0 otherwise.
3. Context factors, which include the output of the Context Analysis method in its basic form and some variations. It also includes the lexical and semantic components separately.
4. Medical borders (mb) of the tc.
5. Number of occurrences of the tc in the training set.

Since the version of AdaBoost used works only with binary features, a discretizing process has been performed on all non–binary attributes. In particular, we have considered equal width intervals of 3 and 10 parts in the domain range

of these attributes. The redundancy of this representation is not a problem for the AdaBoost learning algorithm. As we will see later, no overfitting is produced, and it seems that it is able to select the appropriate granularity level.

It has to be said that a number of experiments have been carried out in order to select an appropriate feature set for the task. In particular, we have considered the context words, collected from a window of 10 words to the left and to the right of the different occurrences of each tc. The results obtained using these new features were slightly worse, probably due to the small size of the learning corpus, which is unable to provide reliable estimates of word frequency counts. In the same way, the Collocational analysis extractor was finally excluded from the feature set since it was very irregular in its predictions and contributed negatively to the final performance. See [15] for a detailed description of the whole experimental setting developed for the task. Finally, observe that the final set of predictors used does not perform any learning process on the training corpus, and, therefore, they can be considered directly as regular features, contributing to ease the meta–learning process.

### 4.3 Evaluation

We have tested the approach proposed using corpus#1, with the standard 10-fold cross validation technique. Consequently, the results presented in this section are the average figures among the ten folds. AdaBoost was run using decision stumps as default weak learners.

Figure 3 shows the precision–recall curves obtained by the algorithm on the N and ALL patterns at different number of learning rounds ($T = 10, 50, 100, 150, 200$). It can be observed that above a minimum number of rounds ($T = 50$), the system performance remains stable and that no overfitting is produced.

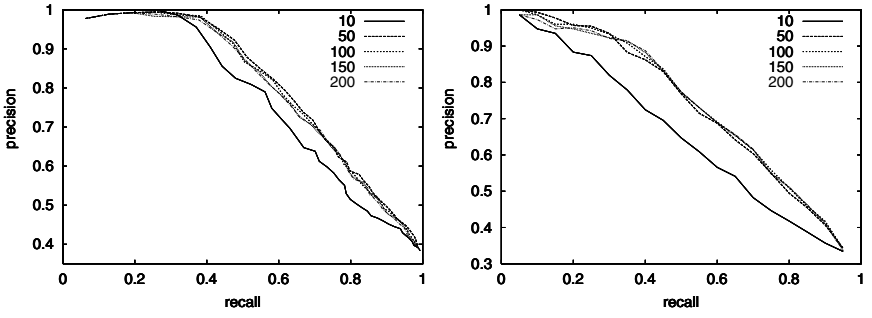


**Fig. 3.** AdaBoost precision–recall curves for N (left) and ALL (right) patterns varying the number of rounds

In order to determine whether the already presented curves are good or not, we compare these results with those obtained using the best individual term

extractors (cf and mc) and the best voting method (UV). Figure 4 contains such comparison (AdaBoost results correspond to a training process of 50 rounds).

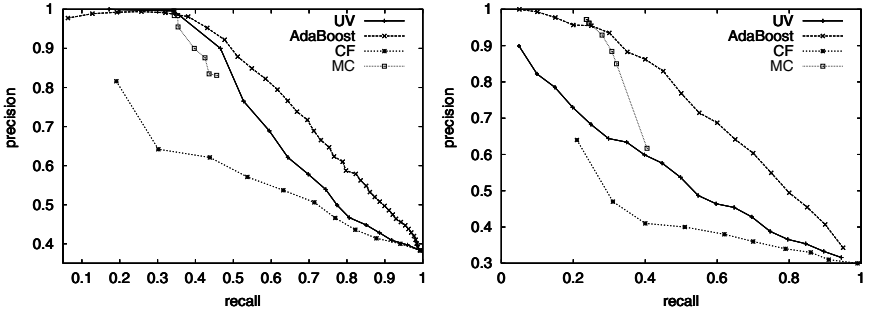


**Fig. 4.** Precision–recall curves obtained by AdaBoost, the individual cf and mc term extractors, and the best voting approach UV, for N (left) and ALL (right) patterns

It can be observed that in the whole problem, AdaBoost performs systematically better than the other methods at all levels of recall, achieving significantly higher precision values (the maximum difference against the UV voting scheme is almost 30 points). In the particular case of N pattern, AdaBoost achieves again the best curve, with significantly better precision results for recall values over 45–50%. At lower recall levels (specially for values lower than 30%), the differences in precision are not significant due to the extremely high precision obtained by the individual mc extractor. In order to complete the information of figure 4 with some numerical results, we present in figure 5 the differences in precision between all methods, achieved at two fixed recall levels of 30% and 50% (which are the usual lower and upper bounds for the proportion of terms that naturally appear in specialized texts).

| | UV | cf | mc | AdaBoost |
|---|---|---|---|---|
| UV | | 19.5 | 0.0 | **0.0** |
| cf | -29.6 | | -19.6 | **-19.2** |
| mc | -7.9 | 21.7 | | **0.0** |
| AdaBoost | **2.9** | **32.5** | **10.8** | |

| | UV | cf | mc | AdaBoost |
|---|---|---|---|---|
| UV | | 15.0 | -25.2 | **-28.0** |
| cf | -13.5 | | -40.0 | **-44.5** |
| mc | — | — | | **-4.0** |
| AdaBoost | **23.3** | **36.5** | — | |

**Fig. 5.** Differences in precision (in percentage points) between methods at fixed points of recall. Patterns: N (left table) and ALL (right table). Recall points: 30% (above diagonal) and 45% (below diagonal)

We also performed some experiments by boosting deeper decision trees (in the same way as it is done [4] for constructing a high precision clause identifier), but the results obtained were systematically slightly worse comparing to the basic

model of decision stumps. In this case, we guess that maybe we are working with a learning problem that can be (almost) explained by a linear combination of functions of a single variable (feature). But another explanation could be that the richer learning representations are more prone to overfit the training corpus (specially when it is not very large).

Assessment of TE systems is a very difficult task. On the one hand, there is very limited agreement between human experts on deciding whether a tc is a real term or not (in [15], an evaluation of this agreement rate using the Kappa statistic is presented). On the other hand, direct comparisons between TE systems are not possible due to the language and domain dependencies of most systems and the lack of benchmark collections. Therefore, the only way of performing a comparison of the TE presented in this paper is to apply a state-of-the-art TE system to our corpus. The only well known system publicly available is FASTR. We have used it in a series of limited experiments but a complete comparison is not fair because the "english" FASTR uses a meta–grammar not available for Spanish. Another possibility is using Maynard's TRUCKS system as a referent for comparison. As explained before, our Context Analyzer extractor is heavily based on TRUCKS. The main differences between our adaptation and the original are: 1) The replacement of UMLS by EWN as the lexical source; and 2) The way of selecting the "prime" tc's. These issues are discussed in detail in [15]. Briefly, the first change is due to the problems of coverage and presentation of Spanish terms in UMLS, while the second tries to perform a more accurate selection of "almost sure" tc's. As a result of such improvements, our Context Analyzer extractor outperforms Maynard's (for Spanish and in our domain) and it could be considered a valid baseline when compared to AdaBoost.

## 5 Conclusions

In this paper it has been presented a general system for term extraction in the medical domain, which is based on the combination of several simple and independent TE's. It has been shown that in this domain even a simple combination scheme based on voting is able to consistently improve the results of the individual TE's, and that using a more sophisticated learning algorithm for performing an additional metalearning step leads to further improvements. We have empirically shown that the improvements achieved by AdaBoost in this second approach are quite significant and very relevant to the task.

Moving to different domains or including additional individual TE's does not imply important problems. The only resource used by the system that needs some tuning is EWN (corpora and dictionaries need no changes). Furthermore, EWN is a general language resource and the customization task is reduced to identify the concepts equivalent to our *medical borders*. This task has to be carried out carefully and some knowledge about the domain and the organization of EWN is required, but this is not a labour intensive task. For instance, only 30 mb were detected in the medical domain. An alternative and not already explored possibility could be the use of existing domain codes (as those proposed by [10]).

**Acknowledgments.** This research has been partially funded by the European Commission (IST-1999-12392 project) and the Spanish Research Department (PB-96-0293 and TIC2000-0335-C03-02 projects).

## References

1. Ananiadou, S.: A Methodology for Automatic Term Recognition. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING*, pages 1034–1038, Kyoto, Japan, 1994.
2. Abney, S., Schapire, R.E. and Singer, Y.: Boosting Applied to Tagging and PP–attachment. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, EMNLP-VLC*, pages 38–45, College Park, MD, 1999.
3. Bourigault, D.: *LEXTER, un Logiciel d'EXtraction de TERminologie. Application à l'acquisition des connaissances à partir de textes.* Phd. Thesis, École des Hautes Études en Sciences Sociales, Paris, 1994.
4. Carreras, X. and Màrquez, L.: Boosting Trees for Clause Splitting. To appear in *Proceedings of the 5th Conference on Computational Natural Language Learning, CoNLL'01*, Tolouse, France, 2001.
5. Daille, B.: *Approche mixte pour l'extraction de terminologie: statistique lexicale et filtres linguistiques.* Phd. Thesis, Université Paris VII, 1994.
6. Escudero, G; Màrquez, L. and Rigau, G.: Boosting Applied to Word Sense Disambiguation. In *Proceedings of the 12th European Conference on Machine Learning, ECML*, Barcelona, Spain, 2000.
7. Justeson, J. and Katz, S.: Technical Terminology: Some Linguistic Properties and an Algorithm for Identification in Text. *Natural Language Engineering*,1(1),1994.
8. Kageura, K. and Umino, B.: Methods for Automatic Term Recognition: A Review. *Terminology*, 3(2):259–289, 1996.
9. Kittler, J.; Hatef, M.; Duin, R. and Matas, J.: On Combining Classifiers. *IEEE Transations on Pattern Analysis and Machine Intelligence*, 20(3):226–238, 1998.
10. Magnini, B. and Cavaglia, G.: Integrating Subject Field Codes into WordNet. *Proceedings of the 2nd International Conference on Language resources and Evaluation, LREC2000*, Atenas .
11. Maynard, D.: *Term Recognition Using Combined Knowledge Sources.* Phd. Thesis, Manchester Metropolitan Univ., Faculty of Science and Engineering, 1999.
12. Schapire, R.E. and Singer, Y.: Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3):297–336, 1999.
13. Schapire, R.E. and Singer, Y.: BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2/3):135–168, 2000.
14. Vivaldi, J. and Rodríguez, H.: Improving Term Extraction by Combining Different Techniques. In *Proceedings of the Workshop on Computational Terminology for Medical and Biological Applications*, pages 61–68, Patras, Greece, 2000.
15. Vivaldi, J.: *A Multistrategy Approach to Term Candidate Extraction.* Phd. Thesis (forthcoming). Dep. LSI, Technical University of Catalonia, Barcelona, 2001
16. Vossen, P. (ed.): *EuroWordNet: A Multilingual Database with Lexical Semantic Networks.* Kluwer Academic Publishers, Dordrecht, 1998.
17. Wolpert, D. H.: Stacked Generalization. *Neural Networks, Pergamon Press*, 5:241-259, 1992.