# A Machine Learning Approach to POS Tagging

LLUíS MÀRQUEZ AND LLUíS PADRÓ AND HORACIO RODRíGUEZ

{lluism,padro,horacio}@lsi.upc.es

*Departament de Llenguatges i Sistemes Informàtics*
*Universitat Politècnica de Catalunya*
*c/ Jordi Girona 1–3. Barcelona 08034, Catalonia*
*Tel: +34 3 4015652 Fax: +34 3 4017014*

**Editor:**

**Abstract.** We have applied inductive learning of statistical decision trees and relaxation labelling to the Natural Language Processing (NLP) task of morphosyntactic disambiguation (Part Of Speech Tagging). The learning process is supervised and obtains a language model oriented to resolve POS ambiguities. This model consists of a set of statistical decision trees expressing distribution of tags and words in some relevant contexts. The acquired language models are complete enough to be directly used as sets of POS disambiguation rules, and include more complex contextual information than simple collections of $n$–grams usually used in statistical taggers. We have implemented a quite simple and fast tagger that has been tested and evaluated on the Wall Street Journal (WSJ) corpus with a remarkable accuracy. However, better results can be obtained by translating the trees into rules to feed a flexible relaxation labelling based tagger. In this direction we describe a tagger which is able to use information of any kind (n–grams, automatically acquired constraints, linguistically motivated manually written constraints, etc.), and in particular to incorporate the machine learned decision trees. Simultaneously, we address the problem of tagging when only small training material is available, which is crucial in any process of constructing, from scratch, an annotated corpus. We show that quite high accuracy can be achieved with our system in this situation.

**Keywords:** Part of speech tagging, corpus–based and statistical language modeling, decision trees induction, constraint satisfaction, relaxation labelling.

## 1. Introduction

Part of Speech (POS) Tagging is a very well known Natural Language Processing (NLP) problem which consists of assigning to each word of a text the proper morphosyntactic tag in its context of appearance. The base of POS tagging is that being many words ambiguous regarding their POS, in most cases they can be completely disambiguated taking into account an adequate context. For instance, in the sample sentence presented in table 1, the word *shot* is disambiguated as a past participle because it is preceded by the auxiliary *was*. Although in this case the word is disambiguated simply by looking at the preceding tag, it must be taken into account that the preceding word could be ambiguous, or that the necessary context could be much more complicated than merely the preceding word. Even, there are cases in which the ambiguity is non resolvable using only morphosyntactic features of the context, and require semantic and/or pragmatic knowledge.

*Table 1.* A sentence and its POS ambiguity –appearing tags, from the Penn Treebank corpus, are described in appendix A–.

The_DT first_JJ time_NN he_PRP was_VBD shot_VBN in_IN the_DT hand_NN as_IN he_PRP chased_VBD the_DT robbers_NNS outside_RB ._.

| first | time | shot | in | hand | as | chased | outside |
|---|---|---|---|---|---|---|---|
| JJ<br>RB | NN<br>VB | NN<br>VBD<br>VBN | IN<br>RB<br>RP | NN<br>VB | IN<br>RB | JJ<br>VBD<br>VBN | IN<br>JJ<br>NN<br>RB |

### 1.1. Existing Approaches to POS Tagging

Starting with the pioneer tagger TAGGIT (Greene & Rubin 1971), used for an initial tagging of the Brown Corpus (BC), a lot of efforts have been devoted to improve the quality of the tagging process in terms of accuracy and efficiency. Existing taggers can be classified into three main groups according to the kind of knowledge they use: linguistic, statistic and machine–learning family. Of course some taggers are difficult to classify into these classes and hybrid approaches must be considered.

Within the linguistic approach most systems codify the involved knowledge as a set of rules (or constraints) written by linguists. The linguistic models range from a few hundreds to several thousands of rules, and they usually require years of labor. The work of the TOSCA group (Oostdijk 1991) and more recently the development of Constraint Grammars in the Helsinki University (Karlsson et al. 1995) can be considered the most important in this direction.

The most extended approach nowadays is the statistical family (obviously due to the limited amount of human effort involved). Basically it consists of building a statistical model of the language and using this model to disambiguate a word

sequence. The language model is coded as a set of co-occurrence frequencies for different kinds of linguistic phenomena.

This statistical acquisition is usually found in the form of n-gram collection, that is, the probability of a certain sequence of length $n$ is estimated from its occurrences in the training corpus.

In the case of POS tagging, usual models consist of tag bi–grams and tri–grams (possible sequences of two or three consecutive tags, respectively). Once the n–gram probabilities have been estimated, new examples can be tagged by selecting the tag sequence with highest probability. This is roughly the technique followed by the widespread Hidden Markov Model taggers. Although the form of the model and the way of determining the sequence to be modeled can also be tackled in several ways, most systems reduce the model to unigrams, bi–grams or tri–grams. The seminal work in this direction is the CLAWS system (Garside et al. 1987), which used bi–gram information and was the probabilistic version of TAGGIT. It was later improved in (DeRose 1988) by using dynamic programming. The tagger by (Church 1988) used a trigram model. Other taggers try to reduce the amount of training data needed to estimate the model, and use the Baum-Welch re-estimation algorithm (Baum 1972) to iteratively refine an initial model obtained from a small hand–tagged corpus. This is the case of the Xerox tagger (Cutting et al. 1992) and its successors. Those interested in the subject can find an excellent overview in (Merialdo 1994).

Recent works try to not to limit the model to a fixed order n–gram by combining different order n–grams, morphological information, long–distance n–grams, or triggering pairs (Rosenfeld 1994, Ristad & Thomas 1996, Saul & Pereira 1997).

Other works that can be placed in the statistical family are those of (Schmid 1994a) which performs energy–function optimization using neural nets. Comparisons between linguistic and statistic taggers can be found in (Chanod & Tapanainen 1995, Samuelsson & Voutilainen 1997).

Although the statistic approach involves some kind of learning, supervised or unsupervised, of the parameters of the model from a training corpus, we place in the machine-learning family only those systems that include more sophisticated information than a n–gram model. Brill's tagger (Brill 1992, Brill 1995) automatically learns a set of transformation rules which best repair the errors commited by a most–likely–tag tagger, (Samuelsson et al. 1996) acquire Constraint Grammar rules from tagged corpora, (Daelemans et al. 1996) apply instance–based learning, and finally, the work that we present here –based on (Màrquez & Rodríguez 1997)– uses decision trees induced from tagged corpora, and combines the learned knowledge in the hybrid approach described in (Padró 1996) which consists of applying relaxation techniques over a set of constraints involving statistical, linguistic and machine–learning–obtained information.

The accuracy reported by most statistic taggers overcomes 96–97% while linguistic Constraint Grammars overcome 99% allowing a residual ambiguity of 1.026 tags per word. These accuracy values are usually computed on a test corpus which has not been used in the training phase. Some corpora commonly used as test benches are

the Brown Corpus, the Wall Street Journal (WSJ) corpus and the British National Corpus (BNC).

### *1.2. Motivation and Goals*

Taking the above accuracy figures into account one may think that POS tagging is a solved and closed problem being this accuracy perfectly acceptable for most NLP systems. So why wasting time in designing yet another tagger? What does an increasing of 0.3% in accuracy really mean?

There are several reasons for thinking that there is still work to do in the field of automatic POS tagging.

When processing huge running texts, and considering an average length per sentence of 25–30 words, if we admit an error rate of 3–4% then it follows that, on average, each sentence contains one error. Since POS tagging is a very basic task in most NLP understanding systems, starting with an error in each sentence could be a severe drawback, specially considering that the propagation of this errors could grow more than linearly. Other NLP tasks that are very sensitive to POS disambiguation errors can be found in the domain of Word Sense Disambiguation (Wilks & Stevenson 1997) and Information Retrieval (Krovetz 1997).

Another issue refers to the need of adapting and tuning taggers that have acquired (or learned) their parameters from a specific corpus onto another one –which may contain texts from other domains– trying to minimizing the cost of transportation.

The accuracy of taggers is usually measured against a test corpora of the same characteristics than the corpus used for training. Nevertheless, no serious attempts have been performed to evaluate the accuracy of taggers corpora with different characteristics, or even domain-specific.

Finally, some specific problems must be addressed when applying taggers to other languages than English. Beside the problems derived from the richer morphology of some particular languages, there is a more general problem consisting of the lack of large manually annotated corpora for training.

Although a *bootstrapping* approach can be carried out, using a low-accurate tagger for producing annotated text that could be used then for learning a more accurate model, the results of such approach are dubious and methods are needed which achieve high accuracy, both on known and unknown words, with a small high-quality training corpus.

In this direction, we are involved in a project for tagging Spanish and Catalan corpora (over 5M words) with limited linguistic resources, that is, departing from a manually tagged core of a size not greater than 50,000 words.

For the sake of comparability the experiments reported here are performed over a reference corpus of English. However we fairly believe that qualitative results could be extrapolated to Spanish or Catalan.

The paper is organized as follows: In section 2 we describe the application domain, the language model learning algorithm and the model evaluation. In sections 3 and 4 we describe the language model application through two taggers: A decision tree

based tagger and a relaxation labelling based tagger, respectively. Comparative results between them in the special case of using a small training corpus are reported in section 5. Finally, the main conclusions and an overview of the future work can be found in section 6.

## 2. Language Model Acquisition

To enable a computer system to process natural language, it is required that language is *modeled* in some way, that is, that the phenomena occurring in language are characterized and captured, in such a way that it can be used to predict or recognize future uses of language: (Rosenfeld 1994) defines language modeling as *the attempt to characterize, capture and exploit regularities in natural language*, and states that the need of language modeling arises from the great deal of variability and uncertainty present in natural language.

As described in section 1, language models can be hand written, statistically derived, or machine learned. In this paper we present the use of a machine learned model combined with statistically acquired models. A testimonial use of hand written models is also included.

### *2.1. Description of the Training Corpus and the Word Form Lexicon*

We have used a portion of $1,170,000$ words of the WSJ, tagged according to the Penn Treebank tag set, to train and test the system. Its most relevant features are the following.

The tag set contains 45 different tags[1]. About 36.5% of the words in the corpus are ambiguous, with an ambiguity ratio of 2.44 tags/word over the ambiguous words, 1.52 overall.

The corpus contains 243 different ambiguity classes, but they are not all equally important. In fact, only the 40 most frequent ambiguity classes cover 83.95% of the occurrences in the corpus, while the 194 most frequent cover almost all of them (>99.50%).

The training corpus has also been used to create a word form lexicon — of 49,206 entries — with the associated lexical probabilities for each word. These probabilities are estimated simply by counting the number of times each word appears in the corpus with each different tag. This simple information provides an heuristic for a very naive disambiguation algorithm which consists of choosing for each word its most probable tag according to the lexical probability. Figure 1 shows the performance of a *most–likely–tag tagger* on the WSJ domain for different sizes of the training corpus.

The reported figures refer to ambiguous words and they can be taken as a lower bound for any tagger. More particularly, it is clear that for a training corpus bigger than 400,000 words, the accuracy obtained is around 81–83%. However it is not sensible to think that it could raise significantly by just adding more training corpus for better estimating the lexical probabilities.

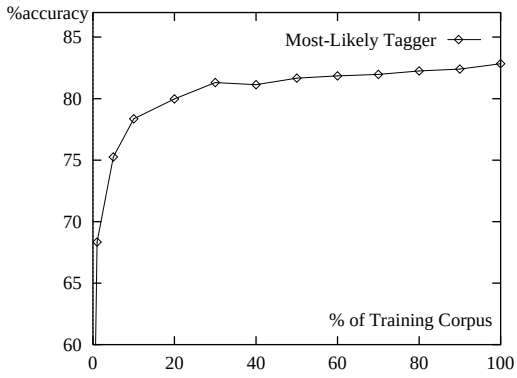


*Figure 1.* Performance of the most likely heuristic related to the training set size

Due to errors in corpus annotation, the resulting lexicon has a certain amount of noise. In order to partially reduce this noise, the lexicon has been filtered by manually checking the entries for the most frequent 200 words in the corpus —note that the 200 most frequent words in the corpus represent over half of it—. For instance the original lexicon entry (numbers indicate frequencies in the training corpus) for the very common word *the* was:

```
the CD 1 DT 47715 JJ 7 NN 1 NNP 6 VBP 1,
```

since it appears in the corpus with the six different tags: `CD` (cardinal), `DT` (determiner), `JJ` (adjective), `NN` (noun), `NNP` (proper noun) and `VBP` (verb–personal form). It is obvious that the only correct reading for *the* is determiner.

### *2.2. Learning Algorithm*

Choosing, from a set of possible tags, the proper syntactic tag for a word in a particular context can be seen as a problem of classification. In this case, classes are identified with tags. Decision trees, recently used in several NLP tasks, such as tagging (Schmid 1994b, Màrquez & Rodríguez 1995, Daelemans et al. 1996), parsing (McCarthy & Lehnert 1995, Magerman 1996), sense disambiguation (Mooney 1996) and information extraction (Cardie 1994), are suitable for performing this task.

*2.2.1. Ambiguity Classes and Statistical Decision Trees* It is possible to group all the words appearing in the corpus according to the set of their possible tags (i.e. *adjective–noun*, *adjective–noun–verb*, *adverb–preposition*, etc.). We will call this sets *ambiguity classes*. It is obvious that there exists an inclusion relation between these classes (i.e. all the words that can be *adjective*, *noun* and *verb*, can be, in particular, *adjective* and *noun*), so the whole set of ambiguity classes is viewed as a taxonomy with a DAG structure. In figure 2 there is represented a part of this taxonomy together with the inclusion relation, extracted from the WSJ.

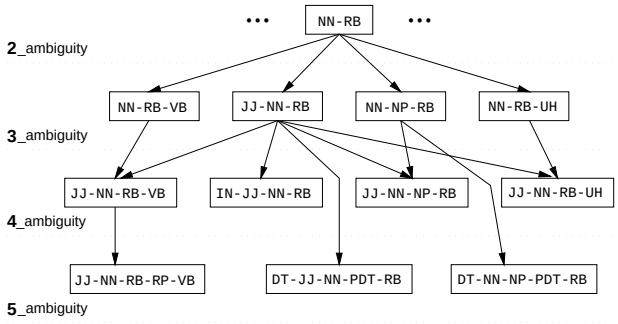


*Figure 2.* A part of the ambiguity-class taxonomy for the WSJ corpus

In this way we split the general POS tagging problem into one classification problem for each ambiguity class.

We identify some remarkable features of our domain, comparing with common classification domains in Machine Learning field. Firstly, there is a very large number of training examples: up to 60,000 examples for a single tree. Secondly, there is a quite significant noise in both the training and test data: WSJ corpus contains about 2–3% of mistagged words.

The main consequence of the above characteristics, together with the fact that simple context conditions cannot explain all ambiguities —see (Voutilainen 1994)— , is that it is not possible to obtain trees to completely classify the training examples. Instead, we aspire to obtain more adjusted probability distributions of the words over their possible tags, conditioned to the particular contexts of appearance. So we will use *Statistical* decision trees, instead of common decision trees, for representing this information.

The algorithm we used to construct the statistical decision trees is a non–incremental supervised learning–from–examples algorithm of the TDIDT (Top Down Induction of Decision Trees) family. It constructs the trees in a top–down way, guided by the distributional information of the examples (Quinlan 1993).

*2.2.2. Training Set and Attributes* For each ambiguity class a set of examples is built by selecting from the training corpus all the occurrences of the words belonging to this ambiguity class. The set of attributes that describe each example refer to the part–of–speech tags of the neighbour words and to the orthography characteristics of the word to be disambiguated.

For the common ambiguity classes the set of attributes consists of a window covering 3 tags to the left and 2 tags to the right —this size as well as the final set of attributes has been determined on an empirical basis— and the *word–form*. Table 2 shows real examples from the training set for the words that can be preposition and adverb (`IN-RB` ambiguity class).

*Table 2.* Training examples of the *preposition–adverb* ambiguity class

| $tag_{-3}$ | $tag_{-2}$ | $tag_{-1}$ | <word,tag> | $tag_{+1}$ | $tag_{+2}$ |
|---|---|---|---|---|---|
| RB | VBD | IN | <"after",IN> | DT | NNS |
| VB | DT | NN | <"as",IN> | DT | JJ |
| DT | JJ | NNS | <"as",RB> | RB | IN |
| JJ | NN | NNS | <"below",RB> | VBP | DT |
| ... | | | | | |

A new set of orthographic features is incorporated in order to deal with a particular ambiguity class, namely *unknown words*, that will be introduced in following sections. See table 3 for a description of the whole set of attributes.

*Table 3.* List of considered attributes

| Attribute | Values | Number of values |
|---|---|---|
| $tag_{-3}$ | any tag in the Penn Treebank | 45 |
| $tag_{-2}$ | " | " |
| $tag_{-1}$ | " | " |
| $tag_{+1}$ | " | " |
| $tag_{+2}$ | " | " |
| word form | any word of the ambiguity class | <847 |
| first character | any printable ASCII character | <190 |
| last character | " | " |
| last-1 character | " | " |
| last-2 character | " | " |
| capitalized? | {yes,no} | 2 |
| other capital letters? | " | " |
| multi–word? | " | " |
| has numeric character? | " | " |

Attributes with many values (i.e. the word–form and pre/suffix attributes used when dealing with unknown words) are treated by dynamically adjusting the number of values to the $N$ most frequent and joining the rest in a new *otherwise* value. The maximum number of values is fixed to 45 (the number of different tags) in order to have more homogeneous attributes.

*2.2.3. Attribute Selection Function* After testing several attribute selection functions, with no significant differences between them, we used an attribute selection function due to (López de Mántaras 1991), belonging to the information–based family, which showed a slightly higher stability than the others. Roughly speaking, it defines a distance measure between partitions and selects for branching the attribute that generates the closest partition to the *correct partition*, namely the one that joins together all the examples of the same class.

*2.2.4. Branching Strategy* Usual TDIDT algorithms consider a branch for each value of the selected attribute. However other solutions are possible. For instance, some systems perform a previous recasting of the attributes in order to have binary–valued attributes (Magerman 1996). The motivation could be efficiency (dealing only with binary trees has certain advantages), and avoiding excessive data fragmentation (when there is a large number of values). Although this transformation of attributes is always possible, the resulting attributes lose their intuition and direct interpretation, and explode in number. We have chosen a mixed approach which consists of splitting for all values and afterwards joining the resulting subsets into groups for which we have not enough statistical evidence of being different distributions. This statistical evidence is tested with a $\chi^2$ test at a 95% confidence level, with a previous smoothing of data in order to avoid zero probabilities.

*2.2.5. Pruning the Tree* In order to decrease the effect of *over-fitting*, we have implemented a post pruning technique. In a first step the tree is completely expanded and afterwards is pruned following a minimal cost–complexity criterion (Breiman et al. 1984), using a comparatively small fresh part of the training set. The alternative of smoothing the conditional probability distributions of the leaves using fresh corpus (Magerman 1996) has been left out because we also wanted to reduce the size of the trees. Experimental tests have shown that in our domain the pruning process reduces tree sizes up to 50% and improves their accuracy in a 2–5%.

*2.2.6. An Example* Finally, we present a real example of a decision tree branch learned for the class `IN-RB` which has a clear linguistic interpretation.

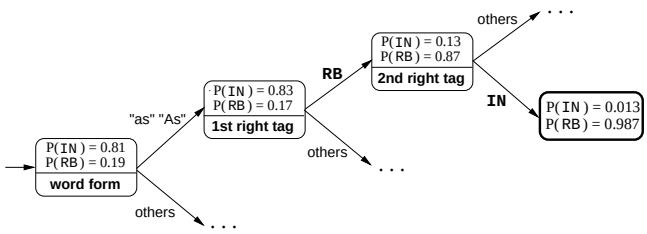


*Figure 3.* Example of a decision tree branch

We can observe in figure 3 that each node in the path from the root to the leaf contains a question on a concrete attribute and a probability distribution. In the root it is the prior probability distribution of the class. In the other nodes it represents the probability distribution conditioned to the answers of the questions preceding the node. For example the second node says that the word *as* is more commonly a preposition than an adverb, but the leaf says that the word *as* is almost sure an adverb when it occurs immediately before another adverb and a preposition (this is the case of *as much as*, *as well as*, *as soon as*, etc.).

## 3. A Tree–Based Tagger

Using the model described in the previous section, we have implemented a *reductionistic* tagger in the sense of Constraint Grammars (Karlsson et al. 1995). In a initial step a word-form frequency dictionary constructed from the training corpus provides each input word with all possible tags with their associated lexical probability. After that, an iterative process reduces the ambiguity (discarding low probable tags) at each step until a certain stopping criterion is satisfied. The whole process is represented in figure 4. See also table 4 for the real process of disambiguation of a part of the sentence presented in table 1.

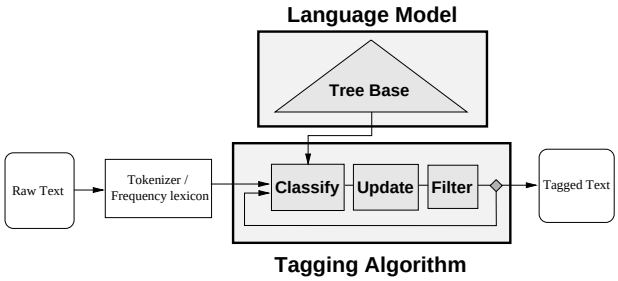


*Figure 4.* Architecture of the Tree–based tagger

More particularly, at each step and for each ambiguous word the work to be done in parallel is:

1. Classify the word using the corresponding decision tree. The ambiguity of the context (either left or right) during classification may generate multiple answers for the questions of the nodes. In this case, all the paths are followed and the result is taken as a weighted average of the results of all possible paths.

2. Use the resulting probability distribution to update the probability distribution of the word (the updating of the probabilities is done by simply multiplying previous probabilities per new probabilities coming from the tree).

3. Discard the tags with *almost* zero probability, that is, those with probabilities lower than a certain *discard boundary* parameter.

After the stopping criterion is satisfied some words could still remain ambiguous. Then there are two possibilities: 1) Choose the most-likely tag for each still ambiguous word to completely disambiguate the text. 2) Accept the residual ambiguity (for treating it in successive stages).

Note that a unique iteration forcing the complete disambiguation is equivalent to use directly the trees as classifiers and results in a very efficient tagger, while performing several steps reduces progressively the efficiency but takes advantage of the statistical nature of the trees to get more accurate results.

*Table 4.* Example of disambiguation

| ... | as | he | chased | the | robbers | outside | . |
|---|---|---|---|---|---|---|---|
| it.0 | IN:0.83<br>RB:0.17 | PRP:1 | JJ:0.25<br>VBD:0.28<br>VBN:0.57 | DT:1 | NNS:1 | IN:0.54<br>JJ:0.36<br>NN:0.06<br>RB:0.04 | .:1 |
| it.1 | IN:0.96<br>RB:0.04 | PRP:1 | VBD:0.97<br>VBN:0.03 | DT:1 | NNS:1 | IN:0.01<br>JJ:0.01<br>RB:0.98 | .:1 |
| it.2 | IN:1 | PRP:1 | VBD:1 | DT:1 | NNS:1 | RB:1 | .:1 |

stop

Another important point is to determine an appropriate stopping criterion. First experiments seem to indicate than the performance increases up to a unique maximum and then softly decreases as the number of iterations increases. For the experiments reported in the following sections, the number of iterations was simply fixed to three.

### *3.1. Using the Tagger*

We divided the WSJ corpus in two parts: $1,120,000$ words were used as a training/pruning set, and $50,000$ words as a fresh test set. We used a lexicon —described in section 2.1— derived from training corpus, containing all possible tags for each word, as well as their lexical probabilities. For the words in the test corpus not appearing in the training set, we stored all possible tags, but no lexical probability (i.e. assigning uniform distribution). This approach corresponds to the assumption of having a morphological analyzer that provides all possible tags for unknown words. In following experiments we will treat unknown words in a less informed way.

From the 243 ambiguity classes the acquisition algorithm learned a base of 194 trees covering 99.5% of the ambiguous words and requiring about 500 Kb of storage. The first four columns of table 5 contain information about the trees learned for the ten most representative ambiguity classes. They present figures about the number of examples used for learning each tree, their number of nodes and the estimation of their error rate when tested on a sample of new examples. This last figure could be taken as a rough estimation of the error of the trees when used in the Tree–based tagger, though it is not exactly true, since here learning examples are fully disambiguated in their context, while during tagging both contexts –left and right– can be ambiguous.

The tagging algorithm, running on a SUN UltraSparc2, processed the test set at a speed of >300 words/sec. Obtained results can be seen at a different levels of granularity.

*Table 5.* Tree information and number and percentages of error for the most difficult ambiguity classes

| Amb. class | #exs | #nodes | %error | TT–errors(%) | ML–errors(%) |
|---|---|---|---|---|---|
| VBD-VBN | 25,902 | 844 | 7.53% | 91 (6.47%) | 267 (18.98%) |
| NN-VB-VBP | 24,465 | 576 | 3.32% | 51 (4.02%) | 255 (20.10%) |
| VB-VBP | 17,690 | 313 | 3.67% | 46 (4.97%) | 226 (24.42%) |
| IN-RB-RP | 26,964 | 99 | 7.13% | 164 (9.14%) | 210 (11.70%) |
| DT-IN-RB-WDT | 8,312 | 271 | 6.07% | 56 (12.08%) | 187 (40.34%) |
| JJ-VBD-VBN | 11,346 | 761 | 18.75% | 95 (16.70%) | 180 (31.64%) |
| JJ-NN | 16,922 | 680 | 16.30% | 122 (14.01%) | 144 (16.54%) |
| NN-VBG | 9,503 | 564 | 16.54% | 58 (10.84%) | 116 (21.68%) |
| NNS-VBZ | 15,233 | 688 | 4.37% | 44 (6.19%) | 81 (11.40%) |
| JJ-RB | 8,650 | 854 | 11.20% | 48 (10.84%) | 73 (16.49%) |
| NN-VB | 14,614 | 221 | 1.11% | 12 (1.63%) | 67 (9.10%) |
| Total | 179,601 | 5,871 | | 787 | 1,806 |

- The performance of some of the learned trees is shown in last two columns of table 5. The corresponding ambiguity classes concentrate the 62.5% of the errors commited by a *most–likely* tagger (ML column). **TT** column shows the number and percentage of errors commited by our tagger. On the one hand we can observe a remarkable reduction in the number of errors (56.4%). On the other hand it is useful to identify some problematic cases. For instance, JJ-NN seems to be the most difficult ambiguity class, since the associated tree obtains only a slight error reduction from the most-likely baseline tagger (15.3%) —this is not surprising since semantic knowledge is necessary to fully disambiguate between noun and adjective—. Results for the DT-IN-RB-WDT ambiguity reflect an over estimation of the generalization performance of the tree —predicted error rate (6.07%) is much lower than the real (12.08%)—. This may be indicating a problem of over pruning.

- Global results are the following: when forcing a complete disambiguation the resulting accuracy was 97.29%, while accepting residual ambiguity the accuracy rate increased up to 98.22%, with an ambiguity ratio of 1.08 tags/word over the ambiguous words and 1.026 tags/word overall. In other words, 2.75% of the words remained ambiguous (over 96% of them retaining only 2 tags). In (Màrquez & Rodríguez 1997) it is shown that these results are, at least, as good as the results of a number of the non linguistically motivated state–of–the–art taggers.

In addition, we present in figure 5 the performance achieved by our tagger with increasing sizes of the training corpus. Results in accuracy are taken over all words. The same figure includes most-likely results, which can be seen as a lower bound.

Following the intuition, we see that performance grows as the training set size grows. The maximum is at 97.29%, as previously said.

A particular case of our interest is the performance achieved when using only 50,000 training examples. In this case the accuracy rate is 95.69%. This figure

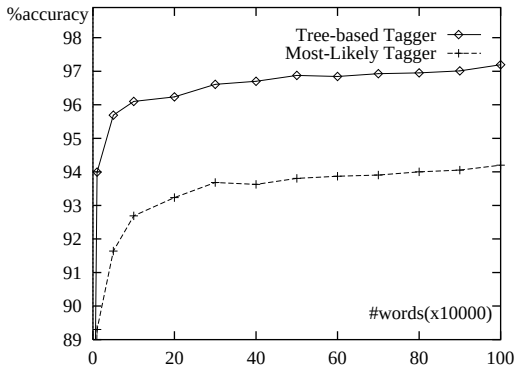


*Figure 5.* Performance of the tagger related to the training set size

has been calculated as the average of the results obtained by repeating the experiment ten times on completely different training material. This mean value has a confidence interval of ±0.17%, at 95% confidence rate.

We think this result is quite accurate. In order to corroborate this statement we can compare our result on a training set of 100K examples (an accuracy of 96.16%) with the figures reported by (Daelemans et al. 1996) for the IGTree Tagger (96.0%) —IGTree system is a Memory–Based POS tagger which uses a tree representation of the training set of examples—. So our results can be considered, at least, as good as theirs.

Section 5 is devoted to further experiments on this issue.

### *3.2. Unknown words*

*Unknown* words are those words not present in the lexicon (i.e. in our case, the words not present in the training corpus). In the previous experiments we have not considered the possibility of unknown words. Instead we have assumed a morphological analyzer providing the set of possible tags with a uniform probability distribution. However, this is not the most realistic scenario. Firstly, a morphological analyzer is not always present (due to the morphological simplicity of the treated language, the existence of some efficiency requirements, or simply the lack of resources). Secondly, if it is available, it surely has a certain error rate that makes necessary to considered the noise it introduces. So it seems clear that we have deal with unknown words in order to obtain more realistic figures about the real performance of our tagger.

There exist several approaches to deal with real unknown words. On the one hand one can assume that unknown words may potentially take any tag, excluding those tags corresponding to closed categories (preposition, determiner, etc.), and try to the disambiguate between them. On the other hand, other approaches includes a pre–process that tries to guess the set of candidate tags for each unknown word to

feed the tagger with this information. See (Padró 1997) for a detailed explanation of the methods.

In our case we consider unknown words as words belonging to the ambiguity class containing all possible tags corresponding to open categories (i.e. noun, proper noun, verb, adjective, adverb, cardinal, etc.). The number of candidate tags sum to 20, so we state a classification problem with 20 different classes. We have estimated the proportion of each of these tags appearing naturally in the WSJ as unknown words and we have collected the examples from the training corpus according to these proportions. The most frequent tag, `NNP` (proper noun), represents almost 30% of the sample. This fact establishes a lower bound for accuracy of 30% in this domain (i.e. the performance that a *most–likely–tag* tagger would obtain).

We have used very simple information about the orthography and the context of unknown words in order to improve these results. In particular, from an initial set of 17 potential attributes, we have empirically decided the most relevant, which turned out to be the following: 1) On the word form: the first letter, the last three letters, and other four binary–valued attributes accounting for capitalization, whether the word is a multi–word or not, and for the existence of some numeric characters in the word. 2) On the context: just the preceding and the following POS tags. Note that this set of attributes is fully described in table 3.

Table 6 shows the generalization performance of the trees learned from training sets of increasing sizes up to 50,000 words. In order to compare these figures again with the results of IGTree, we have implemented IGTree algorithms (including the pruning step) and we have tested its performance exactly under the same condition as ours. These results are also shown in table 6. Figures 6 and 7 contain the plots corresponding to the same results.

*Table 6.* Generalization performance of the trees for unknown words

| #exs. | Tree–based Tagger accuracy(#nodes) | IGTree accuracy(#nodes) |
|---|---|---|
| 2,000 | 77.53% (224) | 70.36% (627) |
| 5,000 | 80.90% (520) | 76.33% (1438) |
| 10,000 | 83.30% (1112) | 79.18% (2664) |
| 20,000 | 85.82% (1644) | 82.30% (4783) |
| 30,000 | 87.32% (2476) | 85.11% (6477) |
| 40,000 | 88.00% (2735) | 86.78% (8086) |
| 50,000 | 88.12% (4056) | 87.14% (9554) |

Note that our system produces better quality trees than those of IGTree — we measure this quality in terms of generalization performance (how well these trees fit new examples) and size (number of nodes)—. On the one hand, we see in figure 6 that our generalization performance is better. On the other hand, figure 7 seems to indicate that the growing factor in the number of nodes is linear in both cases, but clearly lower in ours.

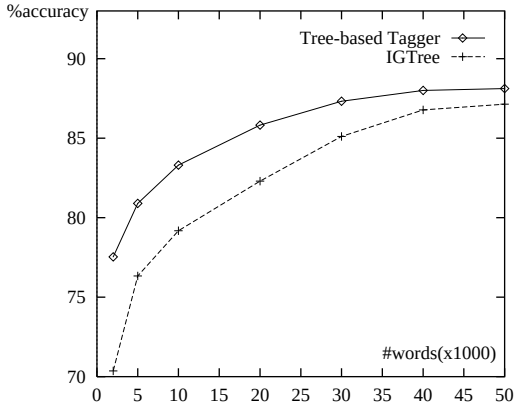


*Figure 6.* Accuracy vs. training set size for *unknown* words

Of course, these conclusions have to be taken in the domain of small training sets — the same plot in figure 7 suggests that the difference between the two methods decreases as the training set size increases—. Using big corpora for training might improve performance significantly. For instance, (Daelemans et al. 1996) report an accuracy rate of 90.6% on unknown words when training with the whole WSJ (2 million words).

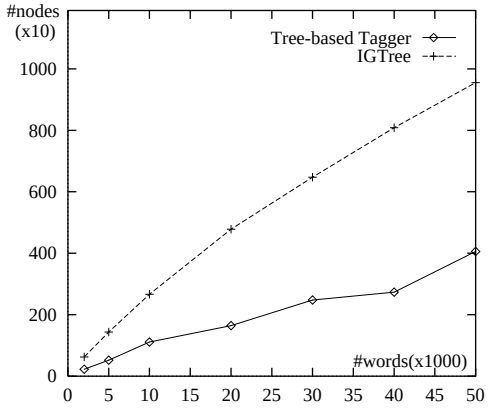


*Figure 7.* Number of nodes of the trees for *unknown* words

## 4. A Relaxation Labelling Based Tagger

We have described up to now a decision tree acquisition algorithm used to automatically obtain a language model for POS tagging, and a classification algorithm which uses the obtained model to disambiguate fresh texts.

Once the language model has been acquired, it would be useful that it could be used by different systems and extended with new knowledge. In this section we will describe a flexible tagger based on relaxation labelling methods, which enables the use of models coming from different sources, as well as their combination and cooperation.

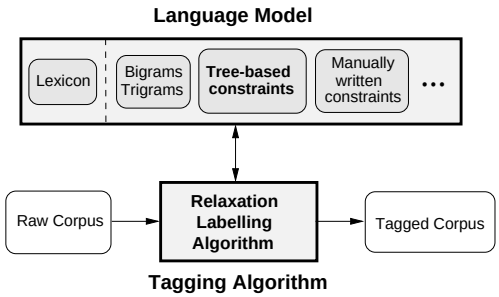


*Figure 8.* Architecture of the relaxation labelling based tagger

The tagger we present has the architecture described in figure 8: A unique algorithm uses a language model consisting of constraints obtained from different knowledge sources.

Relaxation is a generic name for a family of iterative algorithms which perform function optimization, based on local information. They are closely related to neural nets (Torras 1989) and gradient step (Larrosa & Meseguer 1995b).

Although relaxation operations had been long used in engineering fields to solve systems of equations (Southwell 1940), they didn't got their biggest success until (Waltz 1975, Rosenfeld et al. 1976) applied their extension to symbolic domain –relaxation labelling– to constraint propagation field, specially in low-level vision problems.

Relaxation labelling is a technique that can be used to solve consistent labelling problems (CLPs) –see (Larrosa & Meseguer 1995a)–. A consistent labelling problem consists of, given a set of variables, assigning to each variable a value compatible with the values of the other ones, satisfying –to the maximum possible extent– a set of compatibility constraints.

In the Artificial Intelligence field, relaxation has been mainly used in computer vision –since it is where it was first used– to address problems such as corner and edge recognition or line and image smoothing (Richards et al. 1981, Lloyd 1983). Nevertheless, many traditional AI problems can be stated as a labelling prob-

lem: the traveling salesman problem, n-queens, or any other combinatorial problem (Aarts & Korst 1987).

The utility of the algorithm to perform NLP tasks was pointed out in the work by (Pelillo & Refice 1994, Pelillo & Maffione 1994), where POS tagging was used as a toy problem to test some methods to improve the computation of constraint compatibility coefficients for relaxation processes. Nevertheless, the first application to a real NLP problems, on unrestricted text is the work presented in (Padró 1996, Màrquez & Padró 1997, Padró 1997, Voutilainen & Padró 1997).

From out point of view, the most remarkable feature of the algorithm is that it can deal with any kind of constraints, thus the model can be improved by adding any constraints available and it makes the tagging algorithm independent of the complexity of the model. The used constraints may come from different sources: statistical acquisition, machine learned models or hand coding.

### *4.1. The Algorithm*

Although in this section the relaxation algorithm is described from a general point of view, its application to POS tagging is straightforwardly performed considering each word as a variable and each of its possible POS tags as a label.

Let $V = \{v_1, v_2, \ldots, v_N\}$ be a set of variables (words).

Let $T_i = \{t_1^i, t_2^i, \ldots, t_{m_i}^i\}$ be the set of possible labels (POS tags) for variable $v_i$ (where $m_i$ is the number of different labels that are possible for $v_i$).

Let $C$ be a set of constraints between the labels of the variables. Each constraint is a *compatibility value* for a combination of pairs variable–label.

| | | |
|---|---|---|
| 0.53 | $[(v_1, A)(v_3, B)]$ | binary constraint (e.g. bi–gram) |
| 0.29 | $[(v_1, A)(v_3, B)(v_6, C)]$ | ternary constraint (e.g. tri–gram) |

The first constraint states that the combination of variable $v_1$ having label $A$, and variable $v_3$ having label $B$ has a compatibility value of 0.53. Similarly, the second constraint states the compatibility value for the three pairs variable–value it contains.

Constraints can be of any order, so we can define the compatibility value for combinations of any number of variables (obviously we can have combinations of at most $N$ variables).

The aim of the algorithm is to find a *weighted labelling* such that *global consistency* is maximized.

A *weighted labelling* is a weight assignment for each possible label of each variable:

> $P = (p^1, p^2, \ldots, p^N)$ where each $p^i$ is a vector containing a weight for each possible label of $v_i$, that is: $p^i = (p_1^i, p_2^i, \ldots, p_{m_i}^i)$

Since relaxation is an iterative process, the weights vary in time. We will note the weight for label $j$ of variable $i$ at time $n$ as $p_j^i(n)$, or simply $p_j^i$ when the time step is not relevant.

Maximizing *global consistency* is defined as maximizing for each variable $v_i$, $(1 \leq i \leq N)$, the average support for that variable, which is defined as the weighted sum of the support received by each of its possible labels, that is:

$$\sum_{j=1}^{m_i} p_j^i \times S_{ij}$$

where $S_{ij}$ is the support received by that pair from the context.

The support for a pair variable–label ($S_{ij}$) expresses *how compatible* is the assignation of label $j$ to variable $i$ with the labels of neighbouring variables, according to the constraint set.

Although several support functions may be used, we chose the following one, which defines the support as the sum of the influence of every constraint on a label.

$$S_{ij} = \sum_{r \in R_{ij}} Inf(r)$$

where $R_{ij}$ and $Inf(r)$ are defined as follows:

$R_{ij}$ is the set of constraints on label $j$ for variable $i$, i.e. the constraints formed by any combination of variable–label pairs that includes the pair $(v_i, t_j^i)$.

$Inf(r) = C_r \times p_{k_1}^{r_1}(m) \times \ldots \times p_{k_d}^{r_d}(m)$, is the product of the current weights for the labels appearing in the constraint except $(v_i, t_j^i)$ (representing *how applicable* the constraint is in the current context) multiplied by $C_r$ which is the constraint compatibility value (stating *how compatible* the pair is with the context).

The performed *global consistency* maximization is a vector optimization. It doesn't maximize –as one might think– the sum of the supports of all variables. It finds a weighted labelling such that any other choice wouldn't increase the support for *any* variable given –of course– that such a labelling exists. If such a labelling does not exist, the algorithm will end in a local maximum.

The pseudo–code for the relaxation algorithm can be found in table 7. It consists of the following steps:

1. Start in a random labelling $P_0$. In out case, we select a better informed starting point, which are the lexical probabilities for each word tag.

2. For each variable, compute the support that each label receives from the current weights for the labels of the other variables (i.e. see how compatible is the current weighting with the current weightings of the other variables, given the set of constraints).

3. Update the weight of each variable label according to the support obtained by each of them (that is, increase weight for labels with high support –greater than zero–, and decrease weight for those with low support –lesser than zero–). The

chosen updating function in our case was:

$$p_j^i(m+1) = \frac{p_j^i(m) \times (1 + S_{ij})}{\sum_{k=1}^{k_i} p_k^i(m) \times (1 + S_{ik})}$$

4. iterate the process until a convergence criterion is met. The usual criterion is waiting for no significant changes.

1. $P := P_0$
2. *repeat*
3. *for* each variable $v_i$
4. *for* each $t_j$ possible label for $v_i$
5. $S_{ij} := \sum_{r \in R_{ij}} Inf(r)$
6. *end for*
7. $p_j^i(m+1) := \frac{p_j^i(m) \times (1 + S_{ij})}{\sum_{k=1}^{k_i} p_k^i(m) \times (1 + S_{ik})}$
8. *end for*
9. *until* no more changes

*Table 7.* Pseudo code of the relaxation labelling algorithm.

The support computing and weight changing must be performed in parallel, to avoid that changing a weight for a label would affect the support computation of the others.

We could summarize this algorithm saying that at each time step, a variable changes its label weights depending on how compatible is that label with the labels of the other variables at that time step. If the constraints are consistent, this process converges to a state where each variable has weight 1 for one of its labels and weight 0 for all the others.

Note that the *global consistency* idea –defined as the maximization of the average support received by each variable from the context– makes the algorithm robust: The problem of having mutually incompatible constraints (there is no combination of label assignations which satisfies all the constraints) is solved because relaxation does not necessarily find an exclusive combination of labels –i.e. an unique label for each variable–, but a weight for each possible label such that constraints are satisfied to the maximum possible degree. This is specially useful in our case, since

constraints will be automatically acquired, and different knowledge sources will be combined, so constraints are likely not to be fully consistent.

Advantages of the algorithm are:

- Its highly local character (each variable can compute its new label weights given only the state at previous time step). This makes the algorithm highly parallelizable (we could have a processor to compute the new label weights for each variable, or even a processor to compute the weight for each label of each variable).
- Its expressiveness, since we state the problem in terms of constraints between variable labels.
- Its flexibility, we don't have to check absolute consistency of constraints.
- Its robustness, since it can give an answer to problems without an exact solution (incompatible constraints, insufficient data, ...)
- Its ability to find local-optima solutions to NP problems in a non-exponential time (only if we have an upper bound for the number of iterations, i.e. convergence is fast or the algorithm is stopped after a fixed number of iterations).

Drawbacks of the algorithm are:

- Its cost. Being $N$ the number of variables, $v$ the average number of possible labels per variable, $c$ the average number of constraints per label, and $I$ the average number of iterations until convergence, the average cost is $N \times v \times c \times I$, that is, it depends linearly on $N$, but for a problem with many labels and constraints, or if convergence is not quickly achieved, the multiplying terms might be much bigger than $N$.
- Since it acts as an approximation of gradient step algorithms, it has their typical convergence problems: Found optima are local, and convergence is not guaranteed, since the chosen step might be too large for the function to optimize.
- In general, constraints must be written manually, since they are the modeling of the problem. This is good for easy-to-model domains or reduced constraint-set problems, but in the case of POS tagging or WSD constraints are too many and too complicated to be easily written by hand.
- The difficulty to state which is the *compatibility value* for each constraint. If we deal with combinatorial problems with an exact solution (e.g. traveling salesman), the constraints will be all fully compatible (e.g. stating that it is possible to go to any city from any other) or fully incompatible (e.g. stating that it is not possible to be twice in the same city). But if we try to model more sophisticated or less exact problems (such as POS tagging) things will not be black or white. We will have to establish a way of assigning a compatibility value to each constraint.

- The difficulty to choose the support and updating functions more suitable for each particular problem.

### *4.2. Using Machine Learned Constraints*

In order to feed the relaxation labelling algorithm with the language model acquired by the decision tree learning algorithm, the group of the 44 most representative trees (covering 83.95% of the examples) were translated into a set of weighted context constraints. The relaxation labelling based tagger was fed not only with that constraints, but also with bi/tri–gram information.

The constraint formalism used to code the tree branches was Constraint Grammars (Karlsson et al. 1995), a widespread formalism used to write context constraints. Since the CG formalism is intended for linguistic uses, the statistical contribution has no place in it: Constraints can state only full compatibility (*SELECT* constraints) or full incompatibility (*REMOVE* constraints). Thus, we slightly extended the formalism to enable the use of real-valued compatibilities, in such a way that constraints are not assigned a *REMOVE/SELECT* command, but a real number indicating the constraint compatibility value.

The usual way of expressing trees as a set of rules was used to construct the context constraints. For instance the tree branch represented in figure 3 was translated into the two following constraints:

```
−5.81 (IN)                2.366 (RB)
      (0 "as" "As")             (0 "as" "As")
      (1 RB)                    (1 RB)
      (2 IN);                   (2 IN);
```

which express the compatibility (either positive or negative) of the tag in the first line with the given context (i.e. the focus word is `as`, the first word to the right has tag `RB` and the second has tag `IN`).

The compatibility value for each constraint is computed as the *mutual information* (Cover & Thomas 1991) between the tag and the context. Mutual information measures how informative is an event with respect to another, and is computed as

$$MI(A,B) = log\frac{P(A \cap B)}{P(A)P(B)}$$

If $A$ and $B$ are independent events, the joint probability will be equal to the product of the separate probabilities and the measure will be zero. If the joint probability is larger, it means than the two events tend to appear together more often than they would by chance, and the measure yields a positive number. Inversely, if the joint occurrence is scarcer than chance the measure is negative. Mutual information is a simple and useful way to assign *compatibility* values to our constraints.

The main advantage of the relaxation labelling tagger is its ability to deal with constraints of any kind. This enables us to combine statistical n–grams (written in the form of constraints) with the learned decision tree models, and even with linguistically motivated hand written constraints, such as the following,

```
10.0  (VBN)
      (*-1 VAUX BARRIER (VBN) OR (IN) OR (<,>) OR
                        (<:>) OR (JJ) OR (JJS) OR (JJR));
```

which states a high compatibility value for a `VBN` (participle) tag when preceded by an auxiliary verb, provided that there is not any other participle, adjective nor a phrase change in between.

The obtained results for the different knowledge combination are shown in table 8. The results produced by two baseline taggers (**ML**: most likely, **HMM**: bi–gram Hidden Markov Model tagger by (Elworthy 1993)) are also reported. **B** stands for bi-grams, **T** for trigrams, and **C** for the constraints acquired by the decision tree learning algorithm. Results using a sample of 20 hand–written constraints (**H**) can be found in table 9.

Those results show that the addition of the automatically acquired context constraints led to an improvement in the accuracy of the tagger, overcoming the bi/tri–gram models and properly cooperating with them. See (Màrquez & Padró 1997) for more details on the experiments and comparisons with other current taggers.

*Table 8.* Results of baseline tagger and of the relaxation labelling tagger using every combination of constraint kinds

| | ML | HMM | B | T | BT | C | BC | TC | BTC |
|---|---|---|---|---|---|---|---|---|---|
| ambiguous | 85.31% | 91.75% | 91.35% | 91.82% | 91.92% | 91.96% | 92.72% | 92.82% | 92.55% |
| overall | 94.66% | 97.00% | 96.86% | 97.03% | 97.06% | 97.08% | 97.36% | 97.39% | 97.29% |

*Table 9.* Results of our tagger using every combination of constraint kinds plus hand written constraints

| | H | BH | TH | BTH | CH | BCH | TCH | BTCH |
|---|---|---|---|---|---|---|---|---|
| ambiguous | 86.41% | 91.88% | 92.04% | 92.32% | 91.97% | 92.76% | 92.98% | 92.71% |
| overall | 95.06% | 97.05% | 97.11% | 97.21% | 97.08% | 97.37% | 97.45% | 97.35% |

Figure 9 shows the 95% confidence intervals for the results in table 8. The main conclusions that can be drawn from those data are described below.

- The relaxation labelling algorithms is slightly worse than the HMM tagger when using the same information (bi–grams). This may be due to a higher sensitivity to noise in the training corpus.

- There are two significantly distinct groups: Those using only statistical information, and those using statistical information plus the decision trees model. The n–gram models and the learned model belong to the first group, and any combination of a statistical model with the acquired constraint belongs to the second group.

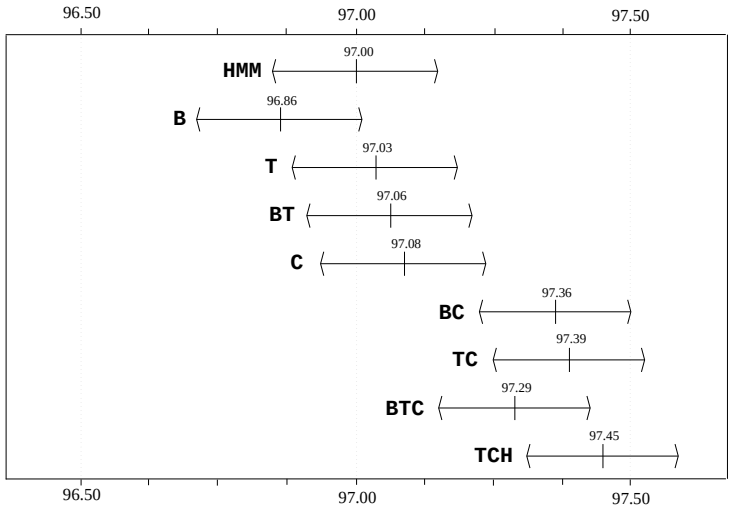


*Figure 9.* 95% confidence intervals for the relaxation tagger results

- Although the hand written constraints improve the accuracy of any model, the size of the linguistic constraint set is too small to make this improvement statistically significant.

- The combination of the two kinds of model produces significantly better results than any separate use. This points that each model contains information which was not included in the other, and that relaxation labelling properly combines them.

## 5. Using Small Training Sets

In this section we will discuss the results obtained when using the two taggers described above to apply the language models learned from small training corpus.

The motivation for this analysis is the need of determining the behavior of our taggers when used with language models coming from scarce training data in order to best exploit them to develop a Spanish and Catalan tagged corpora starting from scratch.

In particular we used 50,000 words of the WSJ corpus to automatically derive a set of decision trees and collect bi–gram statistics. Tri–gram statistics were not considered since the size of the training corpus was not large enough to reasonably estimate the big number of parameters of the model —note that a 45–tag tag set produces a trigram model of over 90,000 parameters, which obviously cannot be estimated from a set of 50,000 occurrences—.

Using this training set the learning algorithm was able to reliably acquire over 80 trees representing the most frequent ambiguity classes —note that the training

data was insufficient for learning sensible trees for about 150 ambiguity classes—. Following the formalism described in the previous section we translated these trees into a set of about 4,000 constraints to feed the relaxation labelling algorithm.

*Table 10.* Comparative results using different models acquired from small training corpus

| | ML | Tree–based | Relax(C) | Relax(B) | Relax(BC) |
|---|---|---|---|---|---|
| ambiguous | 75.35% | 87.29% | 86.29% | 87.50% | 88.56% |
| overall | 91.64% | 95.69% | 95.35% | 95.76% | 96.12% |

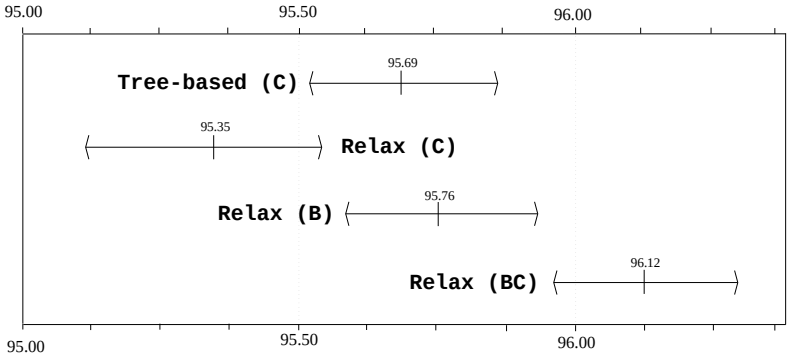


*Figure 10.* 95% confidence intervals for both tagger results

The results in table 10 are computed as the average of ten experiments using randomly chosen training sets of 50,000 words each. **B** stands for the bi–gram model and **C** for the learned decision tree (either in the form of trees or translated to constraints). The corresponding confidence intervals can be found in figure 10.

The presented figures point out the following conclusions:

- The tree–based tagger yields a higher performance than the relaxation labelling based tagger when both use only the **C** model. This is caused by the fact that due to the scarceness of the data, a significant amount of test cases do not match any complete tree branch, and thus the tree–based tagger uses some intermediate node probabilities. Since only complete branches are translated to constrains, the relaxation labelling tagger does not use intermediate node information and produces lower results —restriction rules corresponding to internal nodes are not considered because the total number of rules would increase drastically and because there also would be problems of multiple application for all the restrictions coming from the same branch—.

- The relaxation tagger using the **B** model produces better results than any of the taggers when using the **C** model alone. The cause of this is related with the

aforementioned problem of estimating a big number of parameters with a small sample. Since the model consists of six features, the number of parameters to be learned is still larger than in the case of tri–grams, thus the estimation is not as complete as it could be.

- The relaxation tagger using the **BC** model produces better results (statistically significant at a 95% confidence level) than any other combination. This suggests that, although the tree model is not complete enough on its own, it contains different information than the bi–gram model. Moreover this information is proved to be very useful when combined with the **B** model by the relaxation–based tagger.

## 6. Conclusions

In this work we have presented and evaluated a machine learning based algorithm for obtaining statistical language models oriented to POS tagging.

We have directly applied the learned models in a simple and fast tree–based tagger obtaining quite good results. We also have combined the models with n–gram statistics in a flexible relaxation labelling based tagger. Reported figures show that both models properly collaborate in order to improve the results.

Both model learning and testing have been performed on the WSJ corpus.

Comparison between the results obtained using large training corpora (see section 4.2) with those obtained with 50,000 words training sets (see section 5) points out that the best policy in both cases is the combination of the learned tree–based model with the best n–gram model. In the first case, the reported accuracy (97.36%) is, if not better, at least as good as that of a number of current non linguistic based taggers (see (Màrquez & Padró 1997) for further details). In the second case we think that an accuracy of 96.12% is a very good starting point for our planned bootstrapping project of annotating over 5 million words of Spanish and Catalan corpora —similar experiments, also on the WSJ corpus, by (Daelemans et al. 1996) yield an accuracy of 96.0% when training with 100,000 words—.

However, further work is still to be done in several directions. Referring to the language model learning algorithm, we are interested in testing more informed attribute selection functions, considering more complex questions in the nodes and finding a good smoothing procedure for dealing with very small ambiguity classes. See (Màrquez & Rodríguez 1997) for a first approach.

About the information that this algorithm uses, we want to explore the inclusion of more morphological and semantic information, as well as more complex context features, such as non–limited distance or barrier rules in the style of (Samuelsson et al. 1996).

Regarding the current work, we are beginning to apply our taggers to Spanish and Catalan languages. In this direction we are interested in verifying whether the so far obtained results also hold for those languages.

We conclude saying that we have done first attempts (Padró 1997) in using the same techniques to tackle another classification problem in NLP area, namely Word

Sense Disambiguation (WSD). We believe, as other authors do, that we can take profit of treating both problems jointly.

## Acknowledgments

We specially thank Horacio Rodríguez for his encouraging support and his insightful advice in the development of this work.

This research has been partially funded by the Spanish Research Department (CICYT's ITEM project TIC96–1243–C03–02), by the EU Commission (EuroWordNet LE4003) and by the Catalan Research Department (CIRIT's quality research group 1995SGR 00566).

## Appendix A

Here follows a description of the Penn Treebank tag set, used for tagging the WSJ corpus. For a complete description of the corpus see (Marcus et al.93).

| | | | | | |
|---|---|---|---|---|---|
| `CC` | Coordinating conjunction | `PRP` | Personal pronoun | `WDT` | *wh*-determiner |
| `CD` | Cardinal number | `PP$` | Possessive pronoun | `WP` | *wh*-pronoun |
| `DT` | Determiner | `RB` | Adverb | `WP$` | Possessive *wh*-pronoun |
| `EX` | Existential *there* | `RBR` | Adverb, comparative | `WRB` | *wh*-adverb |
| `FW` | Foreign word | `RBS` | Adverb, superlative | `#` | Pound sign |
| `IN` | Preposition | `RP` | Particle | `$` | Dollar sign |
| `JJ` | Adjective | `SYM` | Symbol | `.` | End of sentence |
| `JJR` | Adjective, comparative | `TO` | *to* | `,` | Comma |
| `JJS` | Adjective, superlative | `UH` | Interjection | `:` | Colon, semi-colon |
| `LS` | List item marker | `VB` | Verb, base form | `(` | Left bracket character |
| `MD` | Modal | `VBD` | Verb, past tense | `)` | Right bracket character |
| `NN` | Noun, singular | `VBG` | Verb, gerund | `"` | Straight double quote |
| `NNP` | Proper noun, singular | `VBN` | Verb, past participle | ‘ | Left open single quote |
| `NNS` | Noun, plural | `VBP` | Verb, non-3rd ps. sing. present | ‘‘ | Left open double quote |
| `NNPS` | Proper noun, plural | | | ’ | Right close single quote |
| `PDT` | Predeterminer | `VBZ` | Verb, 3rd ps. sing. present | ’’ | Right close double quote |
| `POS` | Possessive ending | | | | |

## Notes

1. The size of tag sets differ greatly from one domain to another. Depending on the contents, complexity and level of annotation they are moving from 30–40 to several hundreds of different tags. Of course, these differences have important effects in the performance rates reported by different systems and imply difficulties when comparing them. See (Krenn & Samuelsson 1996) for a more detailed discussion on this issue.

## References


Aarts, E.H.L. & Korst, J.H.M. (1987). Boltzmann machines and their applications. In J.W. de Bakker, A.J. Nijman & P.C. Treleaven (Eds.), *Proceedings PARLE (Parallel Architectures and Languages Europe).* Lecture Notes in Computer Science 258.

Baum, L.E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process. *Inequalities 3:1-8.*

Breiman, L., Friedman, J.H., Olshen, R.A. & Stone, C.J. (1984). Classification and Regression Trees. *Wadsworth International Group, Belmont, California.*

Brill, E. (1992). A Simple Rule–Based Part–of–Speech Tagger. In *Proceedings of the 3rd ACL Conference on Applied Natural Language Processing.*

Brill, E. (1995). Unsupervised Learning of Disambiguation Rules for Part–of–speech Tagging. In *Proceedings of 3rd Workshop on Very Large Corpora.*

Cardie, C. (1994). Domain Specific Knowledge Acquisition for Conceptual Sentence Analysis. *PhD Thesis, University of Massachusets, Amherst, MA.*

Chanod, J.P. & Tapanainen, P. (1995). Tagging French - comparing a statistical and a constraint-based method. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95).*

Church, K.W. (1988). A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceeding of the 2nd ACL Conference on Applied Natural Language Processing.*

Cover, T.M. & Thomas, J.A. (1991). *Elements of Information Theory.* John Wiley & Sons.

Cutting, D., Kupiec, J., Pederson, J. & Sibun, P. (1992). A Practical Part–of–Speech Tagger. In *Proceedings of the 3rd ACL Conference on Applied Natural Language Processing.*

DeRose, S.J. (1988). Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics 14(1), pp. 31–39.*

Elworthy, D. (1993). Part–of–Speech and Phrasal Tagging. Technical Report, ESPRIT BRA–7315 Acquilex II, WP #10.

Daelemans, W., Zavrel, J., Berck, P. & Gillis, S. (1996). MTB: A Memory–Based Part–of–Speech Tagger Generator. In *Proceedings of 4th Workshop on Very Large Corpora, Copenhagen.*

Garside, R., Leech, G. & Sampson, G. (1987). *The Computational Analysis of English.* Longman.

Greene, B.B., & Rubin, G.M. (1971). Automatic Grammatical Tagging of English. Technical Report, Department of Linguistics, Brown University.

Karlsson, F., Voutilainen, A., Heikkilä, J. & Anttila, A. (1995). *Constraint Grammar. A Language-Independent System for Parsing Unrestricted Text.* Mouton de Gruyter.

Krenn, B. & Samuelsson, C. (1996). *The Linguist's Guide to Statistics. Don't Panic.* Universität des Saarlandes. Saarbrücken. Germany. http://coli.uni-sb.de

Krovetz, R. (1997). Homonymy and Polysemy in Information Retrieval. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, ACL '97.*

Larrosa, J. & Meseguer, P. (1995). Constraint Satisfaction as Global Optimization. In *Proceedings of 14th International Joint Conference on Artificial Intelligence, IJCAI '95.*

Larrosa, J. & Meseguer, P. (1995). An Optimization-based Heuristic for Maximal Constraint Satisfaction. In *Proceedings of International Conference on Priciples and Practice of Constraint Programming.*

Lloyd, S.A. (1983). An optimization approach to relaxation labelling algorithms. *Image and Vision Computing, Vol.1, n.2.*

López de Mántaras, R. (1991). A Distance–Based Attribute Selection Measure for Decision Tree Induction. *Machine Learning*, Kluwer Academic.

Magerman, M. (1996). Learning Grammatical Structure Using Statistical Decision–Trees. In *Proceedings of the 3rd International Colloquium on Grammatical Inference, ICGI '96.* Lecture Notes in Artificial Intelligence 1147.

Marcus, M.P., Marcinkiewicz, M.A. & Santorini, B. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics, Vol.19, n.2.*

Màrquez, L. & Rodríguez, H. (1995). Towards Learning a Constraint Grammar from Annotated Corpora Using Decision Trees. ESPRIT BRA–7315 Acquilex II, Working Paper n.15,1995.

Màrquez, L. & Padró, L. (1997). A Flexible POS Tagger Using an Automatically Acquired Language Model. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, ACL '97.*

Màrquez, L. & Rodríguez, H. (1997). *Automatically Acquiring a Language Model for POS Tagging Using Decision Trees.* In Proceedings of the Second Conference on Recent Advances in Natural Language Processing, RANLP '97.

McCarthy, J.F. & Lehnert, W.G. (1997). Using Decision Trees for Coreference Resolution. In *Proceedings of 14th International Joint Conference on Artificial Intelligence IJCAI '95.*

Merialdo, B. (1994). Tagging English Text with a Probabilistic Model. *Computational Linguistics 20(2), pp. 155-171.*

Mooney, R.J. (1996). Comparative Experiments on Disambiguating Word Senses: An Illustration of the Role of Bias in Machine Learning. In *Proceedings of Conference on Empirical Methods in NLP, EMNLP '96.*

Oostdijk, N. (1991). *Corpus Linguistic and the automatic analysis of English.* Rodopi, Amsterdam.

Padró, L. (1996). POS Tagging Using Relaxation Labelling. *In Proceedings of 16th International Conference on Computational Linguistics, COLING '96.*

Padró, L. (1997). A Hybrid Environment for Syntax–Semantic Tagging. *PhD Thesis, Dep. Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 1997.* Forthcoming.

Pelillo, M. & Refice, M. (1994). Learning Compatibility Coefficients for Relaxation Labeling Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.16, n.9.*

Pelillo, M. & Maffione, A. (1994). Using Simulated Annealing to Train Relaxation Labelling Processes. In *Proceedings of ICANN '94.*

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann. San Mateo, CA.

Rosenfeld, R., Hummel, R., Zucker, S. (1976). Scene labelling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics. Vol.6, n.6.*

Rosenfeld, R. (1994). Adaptive Statistical Language Modeling: A Maximum Entropy Approach. *PhD Thesis. School of Computer Science, Carnegie Mellon University.*

Richards, J., Landgrebe, D., Swain, P. (1981). On the accuracy of pixel relaxation labelling. *IEEE Transactions on System, Man and Cybernetics Vol.11.*

Ristad, E.S. & Thomas, R.G. (1996). Nonuniform Markov Models. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing.* Munich, Germany.

Samuelsson, C., Tapanainen, P. & Voutilainen, A. (1996). Inducing Constraint Grammars. In *Proceedings of the 3rd International Colloquium on Grammatical Inference.*

Samuelsson, C. & Voutilainen, A. (1997). Comparing a Linguistic and a Stochastic Tagger. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, ACL '97.*

Saul, L. & Pereira, F. (1997). Aggregate and mixed-order Markov models for statistical language processing. In *Proceedings of 2nd Conference on Empirical Methods for Natural Language Processing, EMNLP 97.*

Schmid, H. (1994). Part–of–speech tagging with neural networks. In *Proceedings of 15th International Conference on Computational Linguistics, COLING '94.*

Schmid, H. (1994). Probabilistic Part–of–Speech Tagging Using Decision Trees. In *Proceedings of the Conference on New Methods in Language Processing.* Manchester, UK.

Southwell, R. (1940). *Relaxation Methods in Engineering Science.* Clarendon.

Torras, C. (1989). Relaxation and Neural Learning: Points of Convergence and Divergence. *Journal of Parallel and Distributed Computing 6, pp.217-244.*

Voutilainen, A. (1994). Three Studies of Grammar–Based Surface Parsing on Unrestricted English Text. *PhD Thesis. Department of General Linguistics. University of Helsinki.*

A. Voutilainen & L. Padró. (1997). Developing a Hybrid NP Parser. In *Proceedings of the 5th ACL Conference on Applied Natural Language Processing.*

Waltz, D. (1975). *Understanding line drawings of scenes with shadows: Psychology of Computer Vision.* P. Winston, New York: McGraw-Hill.

Wilks, Y. & Stevenson, M. (1997). Combining Independent Knowledge Sources for Word Sense Disambiguation. In *Proceedings of the 2nd Conference on Recent Advances in Natural Language Processing, RANLP '97.*