Machine Learning, 39, 59–91, 2000.

A Machine Learning Approach to POS Tagging

LLUÍS MÀRQUEZ lluism@lsi.upc.es
LLUÍS PADRÓ padro@lsi.upc.es
HORACIO RODRÍGUEZ horacio@lsi.upc.es
Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, c/ Jordi Girona 1–3. Barcelona 08034, Catalonia

Editor: Raymond Mooney

Abstract. We have applied the inductive learning of statistical decision trees and relaxation labeling 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, consisting of a set of statistical decision trees expressing distribution of tags and words in some relevant contexts. The acquired decision trees have been directly used in a tagger that is both relatively simple and fast, and which has been tested and evaluated on the Wall Street Journal (WSJ) corpus with competitive accuracy. However, better results can be obtained by translating the trees into rules to feed a flexible relaxation labeling 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 limited training material is available, which is crucial in any process of constructing, from scratch, an annotated corpus. We show that high levels of accuracy can be achieved with our system in this situation, and report some results obtained when using it to develop a 5.5 million words Spanish corpus from scratch.

Keywords: part of speech tagging, corpus-based and statistical language modeling, decision trees induction, constraint satisfaction, relaxation labeling

1. Introduction

Part of Speech (POS) Tagging is a very basic and 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. It is very useful for a number of NLP applications: as a pre-processing step to syntactic parsing, in information extraction and retrieval (e.g. document classification in internet searchers), text to speech systems, corpus linguistics, etc.

The base of POS tagging is that many words being ambiguous regarding their POS, in most cases they can be completely disambiguated by 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. Furthermore, there are even 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	NN	NN	IN	NN	IN	JJ	IN
RB	VB	VBD	RB	VB	RB	VBD	JJ
		VBN	RP			VBN	NN
							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 effort has been devoted to improving 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 knowledge involved as a set of rules (or constraints) written by linguists. The linguistic models range from a few hundreds to several thousand 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, Leech, & Sampson, 1987), which used bi-gram information and was the probabilistic version of TAGGIT. It was later improved by DeRose (1988) by using dynamic programming. The tagger by Church (1988) used a tri-gram 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 by Merialdo (1994).

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

Other tasks are also approached through statistical methods. The speech recognition field is very productive in this issue—actually, *n*-gram modeling was used in speech recognition before being used in POS tagging. Recent works in this field try 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). These are some approaches that we may see incorporated to POS tagging tasks in the short term.

Although the statistical 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, 1995) automatically learns a set of transformation rules which best repair the errors committed by a most-frequent-tag tagger, Samuelsson, Tapanainen, and Voutilainen (1996) acquire Constraint Grammar rules from tagged corpora, Daelemans et al. (1996) apply instance-based learning. The work that we present here uses decision trees induced from tagged corpora (Màrquez & Rodríguez, 1997, 1998), which are exploited, together with other statistical and linguistic information, in a hybrid environment that applies relaxation techniques over a set of constraints (Padró 1996, 1998).

The accuracy reported by most statistic taggers surpasses 96–97% while linguistic Constraint Grammars surpass 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 this accuracy being perfectly acceptable for most NLP systems. So why waste time in designing yet another tagger? What does an increase 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, especially considering that the propagation of these 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 minimize the cost of transportation.

The accuracy of taggers is usually measured against a test corpora of the same characteristics as the corpus used for training. Nevertheless, no serious attempts have been made to evaluate the portability of taggers to corpora from other domains or with different characteristics and tag distributions.

Finally, some specific problems must be addressed when applying taggers to languages other than English. In addition to 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-accuracy tagger for producing annotated text that could then be used for retraining the tagger and learning a more accurate model—the usefulness of this approach relies heavily on the quality of the retraining material. So, if we want to guarantee low-noise retraining corpora, we have to provide methods able to achieve high accuracy, both on known and unknown words, using 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 around 70,000 words. For the sake of comparability we have included experiments performed over a reference corpus of English. However, we also report the results obtained applying the presented techniques to annotate the LEXESP Spanish corpus, proving that a very good accuracy can be achieved at a fairly low human labor cost.

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 labeling based tagger, respectively. Comparative results between them in the special case of using a small training corpus and the joint use of both taggers to annotate a Spanish corpus are reported in Section 5. Finally, the main conclusions and an overview of the future work can be found in Section 7.

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 they 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 for 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. 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 a heuristic for a very naive disambiguation algorithm which consists of choosing for each word its most probable tag according to the lexical probability. Note that such a tagger does not use any contextual information, but only the frequencies of isolated words. Figure 1 shows the performance of this *most-frequent-tag tagger* (MFT) 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 reasonable to think that it could be significantly raised simply by adding more training corpus in order to estimate the lexical probabilities more effectively.

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,
```

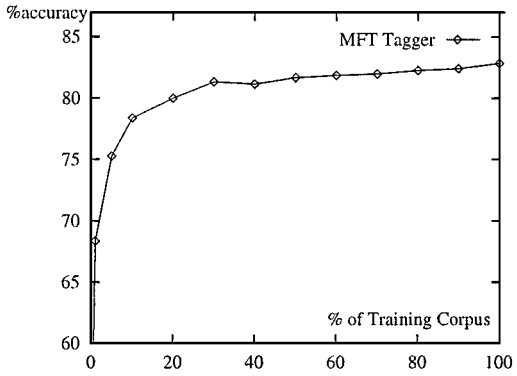

Figure 1. Performance of the “most frequent tag” heuristic related to the training set size.

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

From a set of possible tags, choosing the proper syntactic tag for a word in a particular context can be stated as a problem of classification. In this case, classes are identified with tags. Decision trees, recently used in several NLP tasks, such as speech recognition (Bahl et al., 1989), POS tagging (Schmid, 1994; Márquez & 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 these sets *ambiguity classes*. It is obvious that there is 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. Figure 2 represents part of this taxonomy together with the inclusion relation, extracted from the WSJ.

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, compared to common classification domains in machine learning. Firstly, there is a very large number of training examples: up to 60,000 examples for a single tree. Secondly, there is quite a 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 (Voutilainen, 1994), is that it is not possible

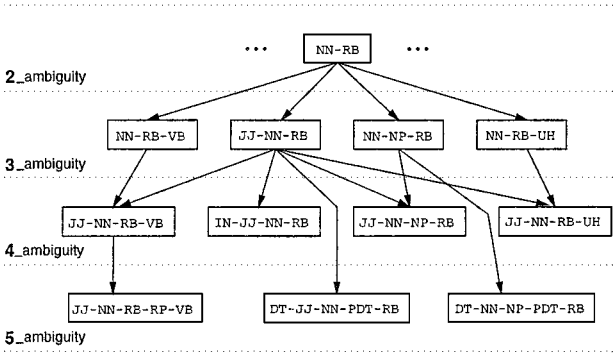

Figure 2. A part of the ambiguity-class taxonomy for the WSJ corpus.

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
...					

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 neighbor words and to the orthographic characteristics of the word to be disambiguated. All of them are discrete attributes.

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).

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.

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 at 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—including *Gini Diversity Index* (Breiman et al., 1984), *Quinlan's Gain Ratio* (Quinlan, 1986), RELIEF-F (Kononenko, Šimec, & Robnik-Šikonja, 1995), χ^2 *Test*, and *Symmetrical Tau* (Zhou & Dillon, 1991)—, with no significant differences between them, we used an attribute selection function proposed by López de Mántaras (1991), belonging to the information-theory-based family, which showed a slightly higher stability than the others and which is proved not to be biased towards attributes with many values and capable of generating smaller trees, with no loss of accuracy, compared with those of Quinlan's Gain Ratio (López de Mántaras, Cerquides, & Garcia, 1996). Roughly speaking, it defines

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?	”	”

a distance measurement between partitions and selects for branching the attribute that generates the closest partition to the *correct partition*, namely the one that perfectly classifies the training data.

Let X be a set of examples, $\mathcal{C}$ the set of classes and $P_{\mathcal{C}}(X)$ the partition of X according to the values of $\mathcal{C}$. The selected attribute will be the one that generates the closest partition of X to $P_{\mathcal{C}}(X)$. For that we need to define a distance measurement between partitions. Let $P_A(X)$ be the partition of X induced by the values of attribute A. The average information of such partition is defined as follows:

$$I(P_A(X)) = -\sum_{a \in P_A(X)} p(X,a) \log_2 p(X,a),$$

where $p(X,a)$ is the probability for an element of X belonging to the set a which is the subset of X whose examples have a certain value for the attribute A, and it is estimated by the ratio $\frac{|X \cap a|}{|X|}$. This average information measurement reflects the randomness of distribution for the elements of X between the classes of the partition induced by A. If we now consider the intersection between two different partitions induced by attributes A and B we obtain:

$$I(P_A(X) \cap P_B(X)) = -\sum_{a \in P_A(X)} \sum_{b \in P_B(X)} p(X, a \cap b) \log_2 p(X, a \cap b).$$

Conditioned information of $P_B(X)$ given $P_A(X)$ is:

$$\begin{aligned} I(P_B(X) \mid P_A(X)) &= I(P_A(X) \cap P_B(X)) - I(P_A(X)) \\ &= -\sum_{a \in P_A(X)} \sum_{b \in P_B(X)} p(X, a \cap b) \log_2 \frac{p(X, a \cap b)}{p(X,a)}. \end{aligned}$$

It is easy to show that the measurement

$$d(P_A(X), P_B(X)) = I(P_B(X) \mid P_A(X)) + I(P_A(X) \mid P_B(X))$$

is a distance. Normalizing, we obtain

$$d_N(P_A(X), P_B(X)) = \frac{d(P_A(X), P_B(X))}{I(P_A(X) \cap P_B(X))},$$

with values in [0, 1]. So, finally, the selected attribute will be that one that minimizes the normalized distance: $d_N(P_C(X), P_A(X))$.

2.2.4. Branching strategy. When dealing with discrete attributes, usual TDIDT algorithms consider a branch for each value of the selected attribute. However there are other possibilities. 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 subsequently joining the resulting subsets into groups for which we have insufficient statistical evidence for there being different distributions. This statistical evidence is tested with a χ^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 a 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 by 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.

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 to 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 certainly 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.).

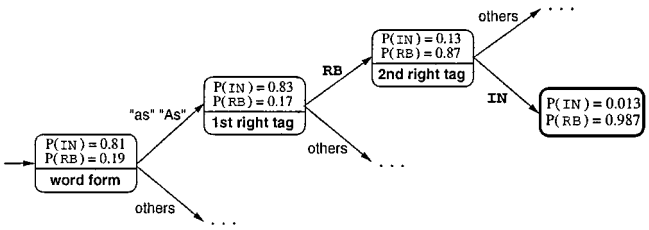

Figure 3. Example of a decision tree branch.

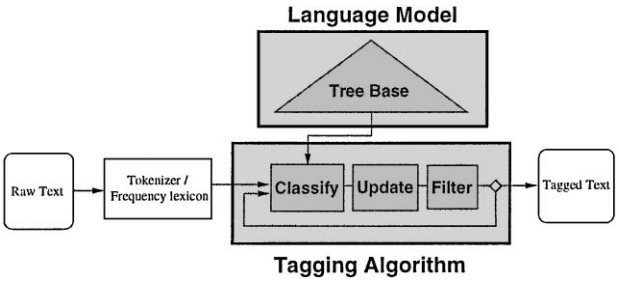

Figure 4. Architecture of TREETAGGER.

3. TREETAGGER: 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 the 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.

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. The weight for each path is actually its probability, which is calculated according to the current probabilities of the POS tags of the features involved in the nodes of the path.
2. Use the resulting probability distribution to update the probability distribution of the word. The probability updating is done by simply multiplying previous probabilities per new probabilities coming from the trees and renormalizing the results (so they sum to one again).

Table 4. Example of disambiguation.

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

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 probable tag for each still-ambiguous word to completely disambiguate the text. 2) Accept the residual ambiguity (for successive treatment).

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

Another important point is to determine an appropriate stopping criterion—since the procedure is heuristic, the convergence is not guaranteed, however this is not the case in our experiments. First experiments seem to indicate that the performance increases up to a unique maximum and then softly decreases as the number of iterations increases. This phenomenon is studied by Padró (1998) and the noise in the training and test sets is suggested to be the major cause. For the sake of simplicity, in the experiments reported in following sections, the number of iterations was experimentally fixed to three. Although it might seem an arbitrary decision, broad-ranging experiments performed seem to indicate that this value results in a good average tagging performance in terms of accuracy and efficiency.

3.1. Using TREETAGGER

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 the tags that these words have in the test corpus, but no lexical probability (i.e. assigning uniform distribution). This approach corresponds to the assumption of having a

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

Amb. class	#Exs	#Nodes	%Error	TT-errors(%)	MFT-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

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 500KB of storage. The learning algorithm (in a Common Lisp implementation) took about 12 CPU-hours running on a SUN SparcStation-10 with 64 Mb of primary memory. 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 classification error over the set of examples used for pruning. This last figure could be taken as a rough estimation of the error of the trees when used in TREETAGGER, though it is not exactly true, since in the pruning examples the neighboring tags are given their correct tags from the supervised annotation, 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. The results obtained can be seen at a different levels of granularity.

- 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 committed by a *most-frequent-tag* tagger (MFT column). TT column shows the number and percentage of errors committed 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 MFT 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

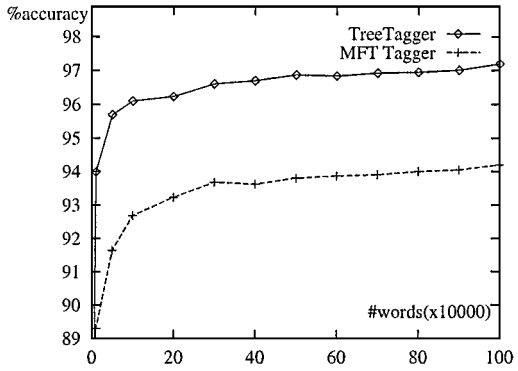

Figure 5. Performance of the tagger related to the training set size.

an over-estimation of the generalization performance of the tree—predicted error rate (6.07%) is much lower than the real (12.08%)—which 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).
  Márquez and Rodríguez (1997) show that these results are as good (and better in some cases) as the results of a number of state-of-the-art taggers based on automatic model acquisition.

In addition, we present in figure 5 the performance achieved by our tagger with increasing sizes of the training corpus. Results in accuracy are computed over all words. The same figure includes MFT results, which can be seen as a reference baseline.

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

One way to easily evaluate the quality of the class-probability estimates given by a classifier is to calculate a *rejection curve*. That is to plot a curve showing the percentage of correctly classified test cases whose *confidence level* exceeds a given value. In the case of statistical decision trees this confidence level can be straightforwardly computed from the class probabilities given by leaves of the trees. In our case we calculate the confidence level as the difference in probability between the two most probable cases (if this difference is large, then the chosen class is clearly much better than the others; if the difference is small, then the chosen class is nearly tied with another class). A rejection curve that increases smoothly, indicates that the confidence level produced by the classifier can be transformed into an accurate probability measurement.

The rejection curve for our classifier, included in figure 6, increases fairly smoothly, giving the idea that that the acquired statistical decision trees provide good confidence estimates. This is in close connection with the aforementioned positive results yielded by the tagger when disambiguation in the low-confidence cases is not required.

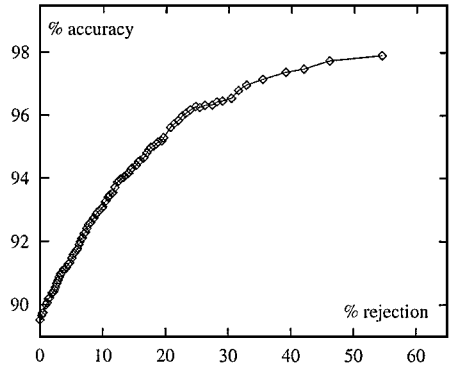

Figure 6. Rejection curve for the trees acquired with full training set.

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 very probably has a certain error rate that makes it necessary to consider the noise it introduces. So it seems clear that we have to deal with unknown words in order to obtain more realistic figures about the real performance of our tagger.

There are several approaches to dealing with 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 disambiguate between them. On the other hand, other approaches include a pre-process that tries to guess the set of candidate tags for each unknown word to feed the tagger with this information. Padró (1998) provides 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 come 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-frequent-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:

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

#Exs.	TREETAGGER 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)

1) In reference to 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) In reference to context: only the preceding and the following POS tags. 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 with a close approach we have implemented IGTREE system (Daelemans et al., 1996) and we have tested its performance exactly under the same conditions as ours.

IGTREE system is a memory-based POS tagger which stores in memory the whole set of training examples and then predicts the part of speech tags for new words in particular contexts by extrapolation from the most similar cases held in memory (*k*-nearest neighbor retrieval algorithm). The main connection point to the work presented here is that huge example bases are indexed using a tree-based formalism, and that the retrieval algorithm is performed by using the generated trees as classifiers. Additionally, these trees are constructed on the base of a previous weight assignment for attributes (contextual and orthographic attributes used for disambiguating are very similar to ours) using Quinlan's Gain Ratio (Quinlan, 1986).

Note that the final pruning step applied by IGTREE to increase the compression factor even more has also been implemented in our version. The results of IGTREE are also included in Table 6. Figures 7 and 8 contain the plots corresponding to the same results.

Observe 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 7 that our generalization performance is better. On the other hand, figure 8 seems to indicate that the growing factor in the number of nodes is linear in both cases, but clearly lower in ours.

Important aspects contributing to the lower size are the merging of attribute values and the post pruning process applied in our algorithm. Experimental results showed that the tree size is reduced by up to 50% on average without loss in accuracy (Màrquez, 1999).

The better performance is probably due to the fact that IGTREEs are not actually decision trees (in the sense of trees acquired by a supervised algorithm of top-down induction, that use a certain *attribute selection function* to decide at *each step* which is the attribute that

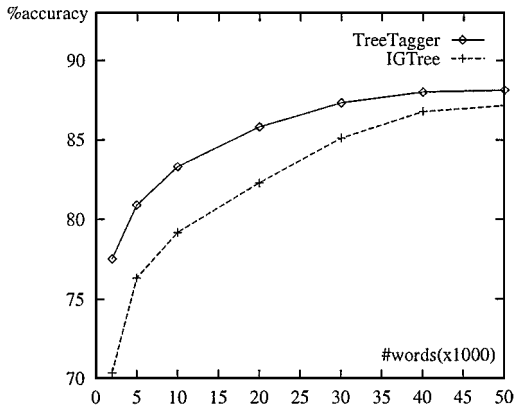

Figure 7. Accuracy vs. training set size for *unknown* words.

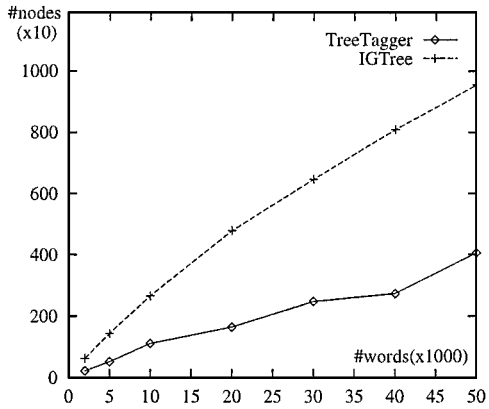

Figure 8. Number of nodes of the trees for *unknown* words.

best contributes to discriminate between the current set of examples), but only a tree-based compression of a base of examples inside a kind of weighted nearest-neighbor retrieval algorithm. The representation and the weight assignment for attributes allows us to think of IGTREEs as the decision trees that would be obtained by applying the usual top-down induction algorithm with a very naive attribute selection function consisting of making a previous unique ranking of attributes using Quinlan's Gain Ratio over all examples and later selecting the attributes according to this ordering. Again, experimental results show that it is better to reconsider the selection of attributes at each step than to decide on an a priori fixed order (Màrquez, 1999).

Of course, these conclusions have to be taken in the domain of small training sets, since the same plot in figure 7 suggests that the difference between the two methods decreases as the training set size increases. Using bigger 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). So our results can be considered better than theirs in the sense that our system needs less resources for achieving the same performance. Note that the same result holds when using the whole training set: They report a tagging accuracy of 96.4%, training with a 2Mwords training set, while our results, slightly over 97%, were achieved using only 1.2 Mwords.

4. RELAX: A relaxation labeling based tagger

Up to now we have described 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 labeling methods, which enables the use of models coming from different sources, as well as their combination and cooperation.

The tagger we present has the architecture described in figure 9: 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 descent (Larrosa & Meseguer, 1995a).

Although relaxation operations had long been used in engineering fields to solve systems of equations (Southwell, 1940), they did not achieve their breakthrough success until relaxation labeling—their extension to the symbolic domain—was applied to the field of constraint propagation, especially in low-level vision problems (Waltz, 1975; Rosenfeld, Hummel, & Zucker, 1976).

Relaxation labeling is a technique that can be used to solve consistent labeling problems (CLPs), as described by Larrosa and Meseguer (1995b). A consistent labeling 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.

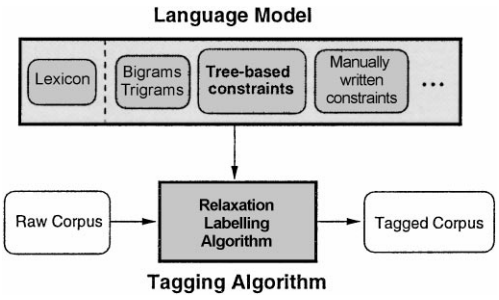

Figure 9. Architecture of RELAX tagger.

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, Landgrebe, & Swain, 1981; Lloyd, 1983). Nevertheless, many traditional AI problems can be stated as a labeling problem: the traveling salesman problem, *n*-queens, or any other combinatorial problem (Aarts & Korst, 1987).

The application of constraint satisfaction to perform NLP tasks is not a novel idea. The relaxation labeling algorithm in particular was first pointed out as useful for such tasks by Pelillo and Refice (1994), Pelillo and Maffione (1994), who used POS tagging as a toy problem to test some methods to improve the computation of constraint compatibility coefficients for relaxation processes. Applications to real NLP problems, dealing with large unrestricted texts, are presented in the work by Padró (1996), Voutilainen and Padró (1997), Màrquez and Padró (1997), Padró (1998).

From our point of view, the most remarkable feature of the algorithm is that, since it deals with context constraints, the model it uses can be improved by writing into the constraint formalism any available knowledge. The constraints used may come from different sources: statistical acquisition, machine-learned models or hand coding. An additional advantage is that the tagging algorithm is independent of the complexity of the model.

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.

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

A *weighted labeling* 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 step 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 assignment of label j to variable i with the labels of neighboring 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).

$Inf(r) = C_r \times p_{k_1}^{r_1}(m) \times \cdots \times p_{k_d}^{r_d}(m)$, 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).

Although the C_r compatibility values for each constraint may be computed in different ways, recent experiments (Padró, 1996, 1998) point out that the best results in our case are obtained when computing compatibilities as the *mutual information* between the tag and the context. Mutual information measures how informative is a discrete random variable with respect to another, and is computed as the expectation of the expression in (1) for every possible pair of values (Cover & Thomas, 1991). Since we are interested on events rather than on distributions, we will use the corresponding expression for the outcomes A and B rather than its expectation (Krenn & Samuelsson, 1997).

$$MI(A, B) = \log \frac{P(A, B)}{P(A) \cdot P(B)} \tag{1}$$

If A and B are independent events, the conditional probability of A given B will be equal to the marginal probability of A and the measurement will be zero. If the conditional probability is larger, it means than the two events tend to appear together more often than they would by chance, and the measurement yields a positive number. Inversely, if the conditional occurrence is scarcer than chance, the measurement is negative. Although Mutual information is a simple and useful way to assign *compatibility* values to our constraints, a promising possibility still to be explored is assigning them by Maximum Entropy Estimation (Rosenfeld, 1994; Ratnaparkhi, 1997; Ristad, 1997).

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

Table 7. Pseudo code of the relaxation labeling algorithm.

```
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.           for each t_j possible label for v_i
8.               p_j^i(m+1) := p_j^i(m) x (1 + S_ij) / sum_{k=1}^{k_i} p_k^i(m) x (1 + S_ik)
9.           end for
10.      end for
11.  until no more changes
```

1. Start in a random labeling P_0. In our 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 from other variable labels (i.e. see how compatible is the current weight assignment with the current weight assignments 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—less 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 to wait for no more changes from one iteration to the next.

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 by 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.

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

Note that this *global consistency* idea makes the algorithm robust: The problem of having mutually incompatible constraints (there is no combination of label assignment which

satisfies all the constraints) is solved because relaxation does not necessarily find an exclusive combination of labels—i.e. a unique label for each variable—, but a weight for each possible label such that constraints are satisfied to the maximum possible degree. This is especially useful in our case, since constraints will be automatically acquired, and different knowledge sources will be combined, so constraints might not be fully consistent.

The 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. In our case, this enables us to use binary (bi-gram) or ternary (tri-gram) constraints, as well as more sophisticated constraints (decision tree branches or hand-written constraints).
- Its flexibility, we do not 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).

The drawbacks of the algorithm are:

- Its cost. N being 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. In our application to POS tagging, the bottleneck is the number of constraints, which may be several thousand. The average number of tags per ambiguous word is about 2.5, and an average sentence contains about 10 ambiguous words.
- Since it acts as an approximation of gradient descent 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 relaxation labeling applications, constraints would 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, constraints are too many and too complicated to be easily written by hand.
- The difficulty of stating by hand what the *compatibility value* is for each constraint. If we deal with combinatorial problems with an exact solution (e.g. traveling salesman), the constraints will be either fully compatible (e.g. stating that it is possible to go to any city from any other), fully incompatible (e.g. stating that it is not possible to be twice in the same city), or will have a value straightforwardly derived from the distance between cities. But if we try to model more sophisticated or less exact problems (such as POS

tagging), we will have to establish a way of assigning graded compatibility values to constraints.
- The difficulty of choosing the most suitable support and updating functions for each particular problem.

4.2. Using machine-learned constraints

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

The Constraint Grammars formalism (Karlsson et al., 1995) was used to code the tree branches. CG is a widespread formalism used to write context constraints. Since it is able to represent any regular context pattern, we will use it to represent all our constraints, *n*-gram patterns, hand-written constraints, or decision-tree branches.

Since the CG formalism is intended for linguistic uses, the statistical contribution has no place in it: Constraints can state only full compatibility (constraints that SELECT a particular reading) or full incompatibility (constraints that REMOVE a particular reading). 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, which—as described in Section 4.1—was computed as the mutual information between the focus tag and the context.

The translation of bi/tri-grams to context constraints is straightforward. A left prediction bi-gram and its right prediction counterpart would be:

```
4.21  (NN)          3.82  (DT)
      (-1 DT);            (1 NN);
```

The training corpus contains 1404 different bi-grams. Since they are used both for left and right prediction, they are converted in 2808 binary constraints.

A tri-gram may be used in three possible ways (i.e. the ABC tri–gram pattern generates the constraints: C, given it is preceded by AB; A, given it is followed by BC; and B, given it is preceded by A and followed by C):

```
2.16  (VB)          1.54  (NN)          1.82  (DT)
      (-2 DT)             (-1 DT)             (1 NN)
      (-1 NN);            (1 VB);             (2 VB);
```

The 17387 tri-gram patterns in the training corpus produce 52161 ternary constraints.

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 decision trees acquired for the 44 most frequent ambiguity classes result in a set of 8473 constraints.

The main advantage of RELAX 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 no other participle, adjective nor any phrase change in between.

The obtained results for the different knowledge combination are shown in Table 8. The results produced by two baseline taggers—MFT: most-frequent-tag tagger, HMM: bi-gram Hidden Markov Model tagger by Elworthy (1993)—are also reported. B stands for bi-grams, T for tri-grams, and C for the constraints acquired by the decision tree learning algorithm. Results using a sample of 20 linguistically-motivated constraints (H) can be found in Table 9. These constraints deal with the participle-past tense ambiguity (1 constraint), with the noun-adjective ambiguity (2 constraints) and with special constructions using particles such as "about" (4 constraints), "as" (7), "up" (3), "out" (2) and "more" (1).

Table 8. Results of baseline tagger and of the RELAX tagger using every combination of constraint kinds.

	MFT	HMM	B	T	BT	C	BC	TC	BTC
Ambig.	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%

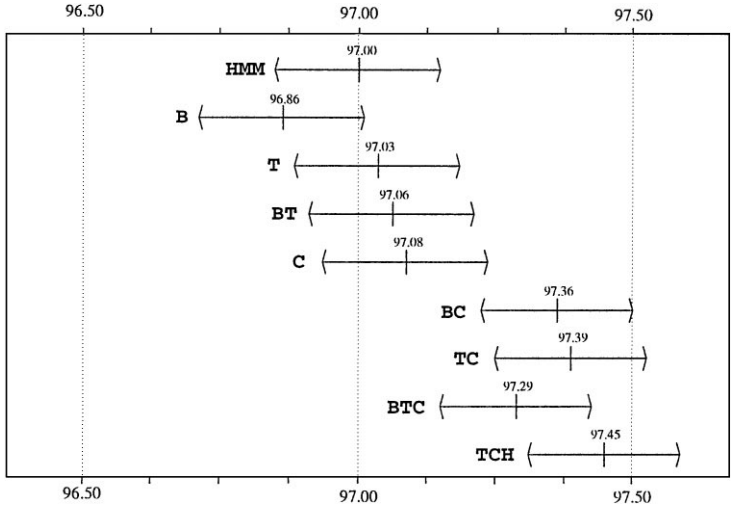

Figure 10. 95% confidence intervals for the RELAX tagger results.

Since the cost of the algorithm depends linearly on the number of constraints, the use of the tri-gram constraints (either alone or combined with the others) makes the disambiguation about six times slower than when using BC, and about 20 times slower than when using only B.

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. Màrquez and Padró (1997) provide more details on the experiments and comparisons with other current taggers.

Figure 10 shows the 95% confidence intervals for the results in Table 8. The main conclusions that can be drawn from those data are described below.

- RELAX 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.
- 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 indicates that each model contains information which was not included in the other, and that relaxation labeling combines them properly.

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 for determining the behavior of our taggers when used with language models coming from scarce training data, in order to best exploit them for the development of Spanish and Catalan tagged corpora starting from scratch.

5.1. Testing performance on WSJ

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 for the model. Note that a 45-tag tag set produces a tri-gram 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 labeling algorithm.

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 11.

The presented figures point out the following conclusions:

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

	MFT	TREETAGGER	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%

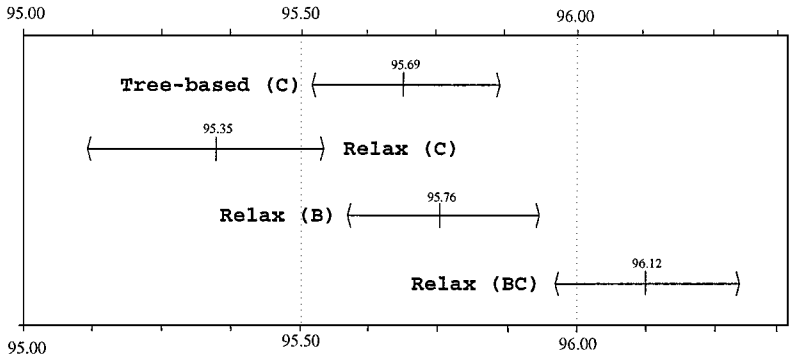

Figure 11. 95% confidence intervals for both tagger results.

- We think this result is quite accurate. In order to corroborate this statement we can compare our accuracy of 96.12% with the 96.0% reported by Daelemans et al. (1996) for the IGTREE Tagger trained with a double size corpus (100 Kw).
- TREETAGGER yields a higher performance than the RELAX 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 TREETAGGER uses some intermediate node probabilities. Since only complete branches are translated to constrains—partial branches were not used to avoid excessive growth in the number of constraints—, the RELAX tagger does not use intermediate node information and produces lower results. A more exhaustive translation of tree information into constraints is an issue that should be studied in the short run.
- The RELAX 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 RELAX 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 RELAX.

5.2. Tagging the LEXESP *Spanish corpus*

The LEXESP Project is a multi-disciplinary effort headed by the Psychology Department at the University of Barcelona. It aims to create a large database of language usage in order to enable and encourage research activities in a wide range of fields, from linguistics to medicine, through psychology and artificial intelligence, among others. One of the main issues of this database of linguistic resources is the LEXESP corpus, which contains 5.5 Mw of written material, including general news, sports news, literature, scientific articles, etc.

The corpus has been morphologically analyzed with the MACO+ system, a fast, broad-coverage analyzer (Carmona et al., 1998). The tag set contains 62 tags. The percentage of ambiguous words is 39.26% and the average ambiguity ratio is 2.63 tags/word for the ambiguous words (1.64 overall).

From this material, 95 Kw were hand-disambiguated to get an initial training set of 70 Kw and a test set of 25 Kw. It has to be noted that the size of the training set is much lower than the usual one million word training corpus derived from the WSJ. This is a common problem when dealing with languages with a reduced amount of available linguistic resources, since the manual tagging of a big enough training corpus is very expensive, both in time and human labor.

Some methods have been developed to avoid the need of fully supervised training. In POS tagging we find the Baum–Welch re-estimation algorithm which has been successfully used to improve tagger accuracies when limited disambiguated material is available (Cutting et al., 1992; Elworthy, 1994; Merialdo, 1994). Brill (1995) presented a weak-supervised

version of the transformation-based learning algorithm for tagging. Recently, similar techniques that use unsupervised data to aid supervised training have been applied in the domain of text categorization (Blum & Mitchell, 1998; Nigam et al., 1998).

In our case, we applied a bootstrapping method taking advantage of the use of both taggers (TREETAGGER and RELAX) in order to automatically disambiguate the LEXESP corpus. The procedure applied starts by using the small hand-tagged portion of the corpus as an initial training set. After training both taggers, they are used to disambiguate further fresh material. The cases in which both taggers assign the same tag are used to enlarge the language model, incorporating it to the training set and retraining both taggers. This procedure could be iterated to obtain progressively better language models.

The point here is that the cases in which both taggers coincide present a higher accuracy, and thus can be used as new retraining set with a lower error rate than that obtained using a single tagger. For instance, using a single tagger trained with the hand-disambiguated training set, we can tag 200,000 fresh words and use them to retrain the tagger. In our case, the best tagger would tag this new set with 97.4% accuracy. Merging this result with the previous hand-disambiguated set, we would obtain a 270 Kw corpus with an error rate of 1.9%. On the other hand, given that both taggers agree in 97.5% of the cases in the same 200 Kw set, and that 98.4% of those cases are correctly tagged, we get a new corpus of 195 Kw with an error rate of 1.6%. If we add the manually tagged 70 Kw we get a 265 Kw corpus with an 1.2% error rate, which is significantly lower than 1.9%.

The main results obtained with this approach are summarized below: Starting with the manually tagged training corpus, the best tagger combination achieved an accuracy of 93.1% on ambiguous words and 97.4% overall. After one bootstrapping iteration, using the coincidence cases in a fresh set of 800 Kw, the accuracy was increased up to 94.2% for ambiguous words and 97.8% overall. It is important to note that this improvement is statistically significant and that it has been achieved in a completely automatic re-estimation process. In our domain, further iterations did not result in new significant improvements.

For a more detailed description we refer the reader to the work by Màrquez & Rodriguez (1998), where experiments using different sizes for the retraining corpus are reported, as well as different combination techniques, such as weighted interpolation and/or previous hand checking of the tagger disagreement cases.

From the aforementioned results we emphasize the following conclusions:

- A 70 Kw manually-disambiguated training set provides enough evidence to allow our taggers to get fairly good results. In absolute terms, results obtained with the LEXESP Spanish corpus are better than those obtained for WSJ English corpus. One of the reasons contributing to this fact may be the less noisy training corpus. However it should be investigated if the part of speech ambiguity cases for Spanish are simpler on average.
- The combination of two (or more) taggers seems to be useful to:
  - Obtain larger training corpora with a reduced error rate, which enable the learning procedures to build more accurate taggers.
  - Building a tagger which proposes a single tag when both taggers coincide and two tags when they disagree. Depending on user needs, it might be worthwhile to accept a higher remaining ambiguity in favor of a higher recall. With the models acquired

from the best training corpus, we get a tagger with a recall of 98.3% and a remaining ambiguity of 1.009 tags/word, that is, 99.1% of the words are fully disambiguated and the remaining 0.9% keep only two tags.

6. Discussion

The comparison of POS taggers is a delicate issue since many factors can affect the final accuracy figures: different sizes for training and test sets, evaluation over different corpora—and thus, different tag distributions—, different tag sets, etc. Even when comparing systems under similar conditions, results might be non-conclusive if noisy material is used for the test (see Padró & Màrquez, 1998, for details).

Nevertheless, an orienting comparison between our system and other current taggers—using the same corpus (WSJ) and training and test sets with similar sizes—can be sketched. To do so, we selected a representative sample of the most accurate statistical and machine-learning taggers that can be found in the literature: Daelemans et al. (1996) report an accuracy of 96.4% (which, taking into account that 5.5% of the test set are unknown words, with an accuracy rate of 90.6%, yields a figure of about 96.7% over known words). Similarly, Brill (1994) obtains a 97.2% over known words, Weischedel et al. (1993) get 96.7% and Ratnaparkhi (1996) reports 96.7%.

Although our figures are higher than the ones reported by the previous systems—we obtained 97.36% on known words over WSJ with similar training sizes—, we believe that quantitative comparison is not conclusive in this case, since conditions are not similar enough to enable us to state that the reported difference in accuracy is significant.

Nevertheless, we do believe that a qualitative comparison can be performed, and that our system presents the following advantages:

- *Flexibility*: It enables a rich feature representation, which can easily incorporate different sources of available information (morhpological, syntax, semantics, etc.). It also provides a numerical weight or probability estimation for each tag.
- *Decision Trees Acquisition*: A possible way of acquiring statistical knowledge is the presented algorithm for DT acquisition, which has been proved (Màrquez & Rodríguez, 1998) to be more accurate and compact than other representations—such that of Daelemans et al. (1996)—for the POS tagging task.
- *Linguistic Knowledge*: Our system is able to deal with linguistic knowledge when expressed in the form of regular context constraints. Thus, it is able to take advantage of the robustness of statistical systems and the coverage of infrequent cases that linguistic rules can provide.
- *Simultaneous disambiguation*: It also enables the simultaneous disambiguation of different features (i.e. POS tags, senses, etc.) as described by Padró (1998).

The main drawback of our hybrid system is the computational cost of applying the context rules. This causes a speed *vs.* accuracy trade-off which may make the system unsuitable for cases in which high tagging speed and high accuracy are required.

Note that the above comparisons only mention taggers based on automatic model acquisition. We have intentionally avoided the comparison with linguistic taggers because we think

that both approaches such be considered in a complementary—rather than exclusive—way. It is a fact that years of work on machine learning for tagging have resulted in taggers that significantly underperform the linguistic taggers based on hand-written rules in the style of Constraint Grammars (Karlsson et al., 1995). We think that automatic taggers will never be able to fully compete with manual linguistic taggers. This may be explained by the coverage of infrequent cases and exceptions which—due to their low frequency—are much difficult to acquire in a statistical approach. However, it is also accepted that linguistic taggers have a much higher acquisition cost and that they are less portable. The hybrid approach takes advantage of both models, achieving a more adequate balance of acquisition cost and accuracy.

7. Summary and further work

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 acquired models in a simple and fast tree-based tagger obtaining fairly good results. We also have combined the models with *n*-gram statistics in a flexible relaxation-labeling based tagger. Reported figures show that both models properly collaborate in order to improve the results.

Comparison between the results obtained using large training corpora (see Section 4.2) with those obtained with fairly small 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.

Deeper application of the techniques, together with the collaboration of both taggers in a voting approach was used to develop from scratch a 5.5 Mw annotated Spanish corpus (LEXESP) with an estimated accuracy of 97.8%. This result confirms the portability of the the proposed method and shows that a very high accuracy is possible for Spanish tagging with a relatively low manual effort. More details about this issue are described by Màrquez et al. (1998).

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).

In reference to the information that this algorithm uses, we would like 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, Tapanainen, and Voutilainen (1996).

We are also specially interested in extending the experiments involving combinations of more than two taggers in a double direction: first, to obtain less noisy corpora for the retraining steps in bootstrapping processes; and second, to construct ensembles of classifiers to increase global tagging accuracy (Halteren, Zavrel, & Daelemans, 1998; Brill & Wu, 1998). We plan to apply these techniques to develop taggers and annotated corpora for the Catalan language in the near future.

More detailed research should be done in order to establish quantitative conclusions to compare tagger performances. The cross evaluation of the main state-of-the-arst taggers in a range of operating conditions is a work we plan to start in the short run. It is also necessary to establish a standard benchmark for the evaluation of POS taggers, to reliably evaluate the results of future research in this field.

The development of hybrid models such as the one presented here is a promising trend of research we think should be further explored. Statistical approaches are fast and accurate techniques to construct language models that explain language phenomena frequently appearing. Linguistic approaches are suitable to model low frequency cases, and could be used to enrich the automatically acquired models. A system able to incorporate both kinds of knowledge sources will provide an accurate tagger at a low human labor cost.

We conclude by saying that we have carried out first attempts (Padró, 1998) in using the same techniques to tackle another classification problem in the NLP area, namely Word Sense Disambiguation (WSD). We believe, as other authors do, that we can take advantage of treating both problems jointly.

Acknowledgments

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 consolidated research group 1997SGR 00051).

Appendix A

We list below a description of the Penn Treebank tag set, used for tagging the WSJ corpus. For a complete description of the corpus see the work by Marcus, Marcinkiewicz, & Santorini (1993).

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.	‘‘	Left open double quote
NNPS	Proper noun, plural		sing. present	’	Right close single quote
PDT	Predeterminer	VBZ	Verb, 3rd ps.	’’	Right close double quote
POS	Possessive ending		sing. present		

References

Aarts, E.H. & Korst, J.H. (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, Vol. 258.

Bahl, L.R., Brown, P.F., DeSouza, P.V., & Mercer, R.L. (1989). A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing, 37*(7), 1001–1008.

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

Blum, A. & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory, COLT-98* (pp. 92–100). Madison, Wisconsin.

Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984). *Classification and regression trees*. The Wadsworth Statistics/Probability Series. Belmont, CA: Wadsworth International Group.

Brill, E. (1992). A simple rule-based part-of-speech tagger. *Proceedings of the 3rd Conference on Applied Natural Language Processing, ANLP* (pp. 152–155). ACL.

Brill, E. (1994). Some advances in rule-based part-of-speech tagging. *Proceedings of the 12th National Conference on Artificial Intelligence, AAAI* (pp. 722–727).

Brill, E. (1995). Unsupervised learning of disambiguation rules for part-of-speech tagging. *Proceedings of the 3rd Workshop on Very Large Corpora* (pp. 1–13). Massachusetts.

Brill, E. & Wu, J. (1998). Classifier combination for improved lexical disambiguation. *Proceedings of the Joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, COLING-ACL* (pp. 191–195). Montréal, Canada.

Cardie, C. (1994). *Domain specific knowledge acquisition for conceptual sentence analysis*. Ph.d. Thesis, University of Massachusets. Available as CMPSCI Technical Report 94–74, University of Massachusetts.

Carmona, J., Cervell, S., Màrquez, L., Martí, M., Padró, L., Placer, R., Rodríguez, H., Taulé, M., & Turmo, J. (1998). An environment for morphosyntactic processing of unrestricted spanish text. *Proceedings of the 1st International Conference on Language Resources and Evaluation, LREC* (pp. 915–922). Spain: Granada.

Chanod, J.-P. & Tapanainen, P. (1995). Tagging french—Comparing a statistical and a constraint-based method. *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics, EACL* (pp. 149–156). Dublin, Ireland.

Church, K.W. (1988). A stochastic parts program and noun phrase parser for unrestricted text. *Proceedings of the 1st Conference on Applied Natural Language Processing, ANLP* (pp. 136–143). ACL.

Cover, T.M. & Thomas, J.A. (Eds.). (1991). *Elements of information theory*. John Wiley & Sons.

Cutting, D., Kupiec, J., Pedersen, J., & Sibun, P. (1992). A practical part-of-speech tagger. *Proceedings of the 3rd Conference on Applied Natural Language Processing, ANLP* (pp. 133–140). ACL.

Daelemans, W., Zavrel, J., Berck, P., & Gillis, S. (1996). MBT: A memory-based part-of-speech tagger generator. *Proceedings of the 4th Workshop on Very Large Corpora* (pp. 14–27). Copenhagen, Denmark.

DeRose, S.J. (1988). Grammatical category disambiguation by statistical optimization. *Computational Linguistics, 14*, 31–39.

Elworthy, D. (1993). Part-of-speech and phrasal tagging. Working Paper #10, ESPRIT BRA-7315 Acquilex II.

Elworthy, D. (1994). Does Baum-Welch re-estimation help taggers? *Proceedings of the 4th Conference on Applied Natural Language Processing, ANLP* (pp. 53–58). ACL.

Garside, R., Leech, G., & Sampson, G. (Eds.) (1987). *The computational analysis of English: A corpus-based approach*. London: Longman.

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

Halteren, H.v., Zavrel, J., & Daelemans, W. (1998). Improving data driven wordclass tagging by system combination. *Proceedings of the Joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, COLING-ACL* (pp. 491–497). Montréal, Canada.

Karlsson, F., Voutilainen, A., Heikkilä, J., & Anttila, A. (Eds.). (1995). *Constraint grammar: A language-independent system for parsing unrestricted text*. Berlin: Mouton de Gruyter.

Kononenko, I., Šimec, E., & Robnik-Šikonja, M. (1995). Overcoming the myopia of inductive learning algorithms with RELIEFF. *Applied Intelligence, 10*, 39–55.

Krenn, B. & Samuelsson, C. (1997). *The linguists' guide to statistics: Don't panic*. Technical Report Universität des Saarlandes. Postscript version of December 19, 1997 at URL: http://coli.uni-sb.de/∼ christer.

Krovetz, R. (1997). Homonymy and polysemy in information retrieval. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics. Joint ACL/EACL* (pp. 72–79). Madrid, Spain.

Larrosa, J. & Meseguer, P. (1995a). An optimization-based heuristic for maximal constraint satisfaction. *Proceedings of International Conference on Principles and Practice of Constraint Programming* (pp. 103–120).

Larrosa, J. & Meseguer, P. (1995b). Constraint satisfaction as global optimization. *Proceedings of 14th International Joint Conference on Artificial Intelligence, IJCAI '95* (pp. 579–584).

Lloyd, S.A. (1983). An optimization approach to relaxation labelling algorithms. *Image and Vision Computing, 1*(2), 85–91.

López de Mántaras, R. (1991). A distance-based attribute selection measure for decision tree induction. *Machine Learning*, *6*(1), 81–92.

López de Mántaras, R., Cerquides, J., & Garcia, P. (1996). Comparing information-theoretic attribute selection measures: A statistical approach research report 96-16, IIIA. To appear in Artificial Intelligence Communications.

Magerman, D.M. (1996). Learning grammatical structure using statistical decision-trees. *Proceedings of the 3rd International Colloquium on Grammatical Inference, ICGI* (pp. 1–21). Springer-Verlag Lecture Notes Series 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*, *19*(2), pp. 313–330.

Màrquez, L. (1999). *Part-of-speech tagging: A machine-learning approach based on decision trees*. Ph.d. Thesis, Dep. Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya.

Màrquez, L. & Padró, L. (1997). A flexible POS tagger using an automatically acquired language model. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics. Joint ACL/EACL* (pp. 238–245). Madrid, Spain.

Màrquez, L., Padró, L., & Rodríguez, H. (1998). Improving tagging accuracy by voting taggers. *Proceedings of the 2nd Conference on Natural Language Processing & Industrial Applications, NLP+IA/TAL+AI* (pp. 149–155). New Brunswick, Canada.

Màrquez, L. & Rodríguez, H. (1995). Towards learning a constraint grammar from annotated corpora using decision trees. Working Paper #21, ESPRIT BRA–7315 Acquilex II.

Màrquez, L. & Rodríguez, H. (1997). Automatically acquiring a language model for POS tagging using decision trees. *Proceedings of the Second Conference on Recent Advances in Natural Language Processing, RANLP* (pp. 27–34). Tzigov Chark, Bulgaria.

Màrquez, L. & Rodríguez, H. (1998). Part-of-speech tagging using decision trees. *Proceedings of the 10th European Conference on Machine Learning, ECML* (pp. 25–36). Chemnitz, Germany. (Lecture Notes in Artificial Intelligence, Vol. 1398. Claire Nédellec and Céline Rouveirol Eds., Springer).

McCarthy, J.F. & Lehnert, W.G. (1995). Using decision trees for coreference resolution. *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI* (pp. 1050–1055).

Merialdo, B. (1994). Tagging english text with a probabilistic model. *Computational Linguistics*, *20*(2), 155–171.

Mooney, R.J. (1996). Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing, EMNLP* (pp. 82–91).

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. *Proceedings of the 15th National Conference on Artificial Intelligence, AAAI-98* Madison, Wisconsin.

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

Padró, L. (1996). POS tagging using relaxation labelling. *Proceedings of the 16th International Conference on Computational Linguistics, COLING* (pp. 877–882). Copenhagen, Denmark.

Padró, L. (1998). A hybrid environment for syntax-semantic tagging. Ph.d. Thesis, Dep. Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya.

Padró, L. & Màrquez, L. (1998). On the evaluation and comparison of taggers: The effect of noise in testing corpora. *Proceedings of the Joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, COLING-ACL* (pp. 997–1002). Montréal, Canada.

Pelillo, M. & Maffione, A. (1994). Using simulated annealing to train relaxation labelling processes. *Proceedings of ICANN '94* (pp. 250–253).

Pelillo, M. & Refice, M. (1994). Learning compatibility coefficients for relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 16*(9), 933–945.

Quinlan, J.R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann Publishers Inc.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.

Ratnaparkhi, A. (1996). A maximum entropy part-of-speech tagger. *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing, EMNLP*.

Ratnaparkhi, A. (1997). *A simple introduction to maximum entropy models for natural language processing*. Technical Report 97-08, Institute for Research in Cognitive Science, University of Pennsylvania.

Richards, J., Landgrebe, D., & Swain, P. (1981). On the accuracy of pixel relaxation labelling. *IEEE Transactions on Systems, Man and Cybernetics*, *11*(4), 303–309.

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

Ristad, E.S. (1997). Maximum entropy modeling for natural language. *Joint ACL/EACL Tutorial Program*, Madrid, Spain.

Rosenfeld, R. (1994). *Adaptive statistical language modelling: A maximum entropy approach*. Ph.d. Thesis, School of Computer Science, Carnegie Mellon University.

Rosenfeld, R., Hummel, R., & Zucker, S. (1976). Scene labelling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, *6*(6), 420–433.

Samuelsson, C., Tapanainen, P., & Voutilainen, A. (1996). Inducing constraint grammars. *Proceedings of the 3rd International Colloquium on Grammatical Inference, ICGI* (pp. 146–155). Montpellier, France.

Samuelsson, C. & Voutilainen, A. (1997). Comparing a linguistic and a stochastic tagger. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics* (pp. 246–253). Madrid, Spain.

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

Schmid, H. (1994a). Part-of-speech tagging with neural networks. *Proceedings of the 15th International Conference on Computational Linguistics, COLING* (pp. 172–176). Kyoto, Japan.

Schmid, H. (1994b). Probabilistic part-of-speech tagging using decision trees. *Proceedings of the Conference on New Methods in Language Processing* (pp. 44–49). 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*, 217–244.

Voutilainen, A. (1994). *Three studies of grammar-based surface parsing on unrestricted English text*. Ph.d. Thesis, Department of General Linguistics. University of Helsinki.

Voutilainen, A. & Padró, L. (1997). Developing a hybrid NP parser. *Proceedings of the 5th Conference on Applied Natural Language Processing, ANLP* (pp. 80–97). Washington DC: ACL.

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

Weischedel, R., Schwartz, R., Palmucci, J., Meteer, M., & Ramshaw, L. (1993). Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, *19*(2), 359–382.

Wilks, Y. & Stevenson, M. (1997). Combining independent knowledge sources for word sense disambiguation. *Proceedings of the Second Conference on Recent Advances in Natural Language Processing, RANLP* (pp. 1–7), Tzigov Chark, Bulgaria.

Zhou, X. & Dillon, T.S. (1991). A statistical–heuristic feature selection criterion for decision tree induction. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 13*(8), 834–841.

Received December 2, 1997
Accepted December 1, 1998
Final manuscript December 1, 1998