Improving POS Tagging Using Machine–Learning Techniques

Lluís Màrquez[1], **Horacio Rodríguez**[2], **Josep Carmona**[1] and **Josep Montolio**[1]

[1] TALP Research Center. Dep. LSI - Universitat Politècnica de Catalunya
c/ Jordi Girona 1-3. 08034 Barcelona. Catalonia
lluism@lsi.upc.es

[2] Dep. IMA - Universitat de Girona
Horacio.Rodriguez@ima.udg.es

Abstract

In this paper we show how machine learning techniques for constructing and combining several classifiers can be applied to improve the accuracy of an existing English POS tagger (Màrquez and Rodríguez, 1997). Additionally, the problem of data sparseness is also addressed by applying a technique of generating *convex pseudo–data* (Breiman, 1998). Experimental results and a comparison to other state–of–the–art taggers are reported.

Keywords: POS Tagging, Corpus–based modeling, Decision Trees, Ensembles of Classifiers.

1 Introduction

The study of general methods to improve the performance in classification tasks, by the combination of different individual classifiers, is a currently very active area of research in supervised learning. In the machine learning (ML) literature this approach is known as *ensemble*, *stacked*, or *combined classifiers*. Given a classification problem, the main goal is to construct several independent classifiers, since it has been proven that when the errors committed by individual classifiers are uncorrelated to a sufficient degree, and their error rates are low enough, the resulting combined classifier performs better than all the individual systems (Ali and Pazzani, 1996; Tumer and Ghosh, 1996; Dietterich, 1997).

Several methods have been proposed in order to construct ensembles of classifiers that make uncorrelated errors. Some of them are general, and they can be applied to any learning algorithm, while other are specific to particular algorithms. From a different perspective, there exist methods for constructing *homogeneous* ensembles, in the sense that a unique learning algorithm has been used to acquire each individual classifier, and *heterogeneous* ensembles that combine different types of learning paradigms[1].

Impressive results have been obtained by applying these techniques on the so–called unstable learning algorithms (e.g. induction of decision trees, neural networks, rule–induction systems, etc.). Several applications to real tasks have been performed, and, regarding NLP, we find ensembles of classifiers in context–sensitive spelling correction (Golding and Roth, 1999), text categorization (Schapire and Singer, 1998; Blum and Mitchell, 1998), and text filtering (Schapire et al., 1998). Combination of classifiers have also been applied to POS tagging. For instance, van Halteren (1996) combined a number of similar taggers by way of a straightforward majority vote. More recently, two parallel works (van Halteren et al., 1998; Brill and Wu, 1998) combined, with a remarkable success, the output of a set of four taggers based on different principles and feature modelling. Finally, in the work by Màrquez et al. (1998) the combination of taggers is used in a bootstrapping algorithm to train a part of speech tagger from a limited amount of training material.

The aim of the present work is to improve an existing POS tagger based on decision trees (Màrquez and Rodríguez, 1997), by using ensembles of classifiers. This tagger treats separately the different types (classes) of ambiguity by considering a different decision tree for each class. This fact allows a selective construction of ensembles of decision trees focusing on the most relevant ambiguity classes, which greatly vary in size and difficulty. Another goal of the present work is to try to alleviate the problem of data sparseness by applying a method, due

[1] An excellent survey covering all these topics can be found in (Dietterich, 1997).

to Breiman (1998), for generating new pseudo–examples from existing data. As we will see in section 4.2 this technique will be combined with the construction of an ensemble of classifiers.

The paper is organized as follows: we start by presenting the two versions of the POS tagger and their evaluation on the reference corpus (sections 2 and 3). Sections 4 and 5 are, respectively, devoted to present the machine–learning improvements and to test their implementation. Finally, section 6 concludes.

2 Tree–based Taggers

Decision trees have been successfully applied to a number of different NLP problems and, in particular, in POS tagging they have proven to be an efficient and compact way of capturing the relevant information for disambiguating. See (Màrquez, 1999) for a broad survey on this issue.

In this approach to tagging, the ambiguous words in the training corpus are divided into classes corresponding to the sets of tags they can take (i.e, 'noun-adjective', 'noun-adjective-verb', etc.). These sets are called *ambiguity classes* and a decision tree is acquired for each of them. Afterwards, the tree–base is applied in a particular disambiguation algorithm.

Regarding the learning algorithm, we use a particular implementation of a top–down induction of decision trees (TDIDT) algorithm, belonging to the supervised learning family. This algorithm is quite similar to the well–known CART (Breiman et al., 1984), and C4.5 (Quinlan, 1993), but it incorporates some particularities in order to better fit the domain at hand.

Training examples are collected from annotated corpora and they consist of the target word to be disambiguated and some information of its local context in the sentence.

All words not present in the training corpus are considered unknown. In principle, we have to assume that they can take any tag corresponding to open categories (i.e., noun, proper noun, verb, adjective, adverb, cardinal, etc.), which sum up to 20 in the Penn Treebank tagset. In this approach, an additional ambiguity class for unknown words is considered, and so, they are treated exactly in the same way as the other ambiguous words, except by the type of information used for acquiring the trees, which is enriched with a number of morphological features.

Once the tree–model has been acquired, it can be used in many ways to disambiguate a real text. In the following sections, 2.1 and 2.2, we present two alternatives.

2.1 RTT: A Reductionistic Tree–based Tagger

RTT is a *reductionistic* tagger in the sense of Constraint Grammars (Karlsson et al., 1995). In a first step a word-form frequency dictionary 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.

More particularly, at each step and for each ambiguous word (at a sentence level) the work performed in parallel is: 1) The target word is "passed" through its corresponding decision tree; 2) The resulting probability distribution is used to multiplicatively update the probability distribution of the word; and 3) The tags with very low probabilities are filtered out.

For more details, we refer the reader to (Màrquez and Rodríguez, 1997).

2.2 STT: A Statistical Tree–based Tagger

The aim of statistical or probabilistic tagging (Church, 1988; Cutting et al., 1992) is to assign the most likely sequence of tags given the observed sequence of words. For doing so, two kinds of information are used: the lexical probabilities, i.e, the probability of a particular tag conditional on the particular word, and the contextual probabilities, which describe the probability of a particular tag conditional on the surrounding tags.

Contextual (or transition) probabilities are usually reduced to the conditioning of the preceding tag (bigrams), or pair of tags (trigrams), however, the general formulation allows a broader definition of context. In this way, the set of acquired statistical decision trees can be seen as a compact representation of a rich contextual model, which can be straightforwardly incorporated inside a statistical tagger. The point here is that the context is not restricted to the $n-1$ preceding tags as in the n–gram formulation. Instead, it is extended to all the contex-

tual information used for learning the decision trees.

The Viterbi algorithm (described for instance in (DeRose, 1988)), in which n–gram probabilities are substituted by the application of the corresponding decision trees, allows the calculation of the most-likely sequence of tags with a linear cost on the sequence length. However, one problem appears when applying conditionings on the right context of the target word, since the disambiguation proceeds from left to right and, so, the right hand side words may be ambiguous. Although dynamic programming can be used to calculate the most likely sequence of tags to the right (in a forward–backward approach), we use a simpler approach which consists of calculating the contextual probabilities by a weighted average of all possible tags for the right context.

Additionally, the already presented tagger allows a straightforward incorporation of n–gram probabilities, by linear interpolation, in a back–off approach including, from most general to most specific, unigrams, bigrams, trigrams and decision trees. From now on, we will refer to STT as STT$^+$ when using n–gram information.

Due to the high ambiguity of unknown words, their direct inclusion in the statistical tagger would result in a severe decreasing of performance. To avoid this situation, we apply the tree for unknown words in a pre–process for filtering low probable tags. In this way, when entering to the tagger the average number of tags per unknown word is reduced from 20 to 3.1.

3 Evaluation of the Taggers

3.1 Domain of Application

We have used a portion of about 1,17 Mw of the Wall Street Journal (WSJ) corpus, tagged according to the Penn Treebank tag set (45 different tags). The corpus has been randomly partitioned into two subsets to train (85%) and test (15%) the system. See table 1 for some details about the used corpus.

The training corpus has been used to create a word form lexicon —of 45,469 entries— with the associated lexical probabilities for each word.

The training corpus contains 239 different ambiguity classes, with a number of examples ranging from few dozens to several thousands (with a maximum of 34,489 examples for the *preposition–adverb–particle* ambiguity). It is noticeable that only the 36 most frequent ambiguity classes concentrate up to 90% of the ambiguous occurrences of the training corpus. Table 2 contains more information about the number of ambiguity classes necessary to cover a concrete percentage of the training corpus.

Training examples for the `unknown-word` ambiguity class were collected from the training corpus in the following way: First, the training corpus is randomly divided into twenty parts of equal size. Then, the first part is used to extract the examples which do not occur in the remaining nineteen parts, that is, taking the 95% of the corpus as known and the remaining 5% to extract the examples. This procedure is repeated with each of the twenty parts, obtaining approximately 22,500 examples from the whole corpus. The choice of dividing by twenty is not arbitrary. 95%–5% is the proportion that results in a percentage of unknown words very similar to the test set (i.e., 2.25%)[2].

Finally, the test set has been used as completely fresh material to test the taggers. All results on tagging accuracy reported in this paper have been obtained against this test set.

3.2 Results

In this experiment we used six basic discrete–valued features to disambiguate known ambiguous words, which are: the part–of–speech tags of the three preceding and two following words, and the orthography of the word to be disambiguated.

For tagging unknown words, we used 20 attributes that can be classified into three groups:

- *Contextual information*: part–of–speech tags of the two preceding and following words.
- *Orthographic and Morphological information* (about the target word): prefixes (first two symbols) and suffixes (last three symbols); Length; Multi–word?; Capitalized?; Other capital letters?; Numerical characters?; Contain dots?
- *Dictionary–related information*: Does the target word contains any known word as a prefix (or a suffix)?; Is the target word

[2] See (Màrquez, 1999) for a discussion on the appropriateness of this procedure.

	S	W	W/S	AW	T/W	T/AW	T/DW	U
Training	40,977	998,354	24.36	339,916 (34.05%)	1.48	2.40	—	—
Test	7,167	175,412	24.47	59,440 (33.89%)	1.45	2.40	3.49	3,941 (2.25%)
Total	48,144	1,173,766	24.38	399,356 (34.02%)	1.47	2.40	—	—

Table 1: Information about the WSJ training and test corpora. S: number of sentences; W: number of words; W/S: average number of words per sentence; AW: number and percentage of ambiguous words; T/W: average number of tags per word; T/AW: average number of tags per ambiguous *known* word; T/DW: average number of tags per ambiguous word (including unknown words); and U: number and percentage of unknown words

	50%	60%	70%	80%	90%	95%	99%	100%
# Classes	8	11	14	18	36	57	111	239

Table 2: Number of ambiguity classes that cover the $x\%$ of the ambiguous words of the training corpus

the prefix (or the suffix) of any word in the lexicon?

The last group of features are inspired in those applied by Brill (1995) when addressing unknown words.

The learning algorithm[3] acquired, in about thirty minutes, a base of 191 trees (the other ambiguity classes had not enough examples) which required about 0,68 Mb of storage.

The results of the taggers working with this tree–base is presented in table 3. MFT stands for a baseline most–frequent–tag tagger. RTT, STT, and STT$^+$ stand for the basic versions of the taggers presented in section 2. The overall accuracy is reported in the first column. Columns 2, 3, and 4 contain the tagging accuracy on some specific groups of words: unknown words, ambiguous words (excluding unknown words) and *known* words which is the complementary of the set of unknown words. Column 5 shows the speed of each tagger[4] and, finally, the 'Memory' column reflects the size of the used language model (the lexicon is not considered).

Three main conclusions can be extracted:

- RTT and STT approaches obtain almost the same results in accuracy, however RTT is faster.
- STT obtains better results when it incorporates bigrams and trigrams, with a slight time–space penalty.
- The accuracy of all taggers is comparable to the best state–of–the art taggers under the open vocabulary assumption (see section 5.2).

4 Machine–Learning–based Improvements

Our purpose is to improve the performance on two types of ambiguity classes, namely:

- *Most frequent ambiguity classes.* We focused on the 26 most representative classes, which concentrate the 86% of the ambiguous occurrences. From these, eight (24.1%) were already resolved at almost 100% of accuracy, while the remaining eighteen (61.9%) left some room for improvement. Section 4.1 explain which methods have been applied to construct ensembles for these eighteen classes plus the `unknown-word` ambiguity class.
- *Ambiguity classes with few examples.* We considered the set of 82 ambiguity classes with a number of examples between 50 and 3,000 and an accuracy rate lower than 95%. They agglutinate 48,322 examples (14.24% of the total ambiguous occurrences). Section 4.2 explains the applied method to increase the number of examples of these classes.

[3]The programs were implemented using PERL-5.0 and they were run on a SUN UltraSparc2 machine with 194Mb of RAM.

[4]More than absolute figures what is important here is the performance of each tagger relative to the others.

Tagger	Overall	Known	Ambiguous	Unknown	Speed	Memory
MFT	92.75%	94.25%	83.40%	27.43%	2818 w/s	0 Mb
RTT	96.61%	97.01%	91.36%	79.22%	426 w/s	0.68 Mb
STT	96.63%	97.02%	91.40%	79.60%	321 w/s	0.68 Mb
STT$^+$	96.84%	97.21%	91.95%	80.70%	302 w/s	0.90 Mb

Table 3: Tagging accuracy, speed, and storage requirement of RTT and STT taggers

4.1 Ensembles of Decision Trees

The general methods for constructing ensembles of classifiers are based on four techniques: 1) Resampling the training data, e.g. *Boosting* (Freund and Schapire, 1995), *Bagging* (Breiman, 1996), and *Cross-validated Committees* (Parmanto et al., 1996); 2) Combining different input features (Cherkauer, 1996; Tumer and Ghosh, 1996); 3) Changing output representation, e.g. ECOC (Dietterich and Bakiri, 1995) and PWC-CC (Moreira and Mayoraz, 1998); and 4) Injecting randomness (Dietterich, 1998).

We tested several of the preceding methods on our domain. Below, we briefly describe those that reported major benefits.

4.1.1 Bagging (BAG)

From a training set of n examples, several samples of the same size are extracted by randomly drawing, with replacement, n times. Such new training sets are called *bootstrap replicates*. In each replicate, some examples appear multiple times, while others do not appear. A classifier is induced from each bootstrap replicate and then they are combined in a voting approach. The technique is called *bootstrap aggregation*, from which the acronym *bagging* is derived. In our case, the bagging approach was performed following the description of Breiman (1996), constructing 10 replicates for each data set[5].

4.1.2 Combining Feature Selection Criteria (FSC)

In this case, the idea is to obtain different classifiers by applying several different functions for feature selection inside the tree induction algorithm. In particular, we have selected a set of seven functions that achieve a similar accuracy, namely: *Gini Impurity Index*, *Information Gain* and *Gain Ratio*, *Chi-square statistic* (χ^2), *Symmetrical Tau criterion*, RLM (a distance-based method), and a version of RELIEF-F which uses the Information Gain function to assign weights to the features. The first five are described, for instance, in (Sestito and Dillon, 1994), RLM is due to López de Mántaras (1991), and, finally, RELIEF-F is described in (Kononenko et al., 1995). Since the applied feature selection functions are based on different principles, we expect to obtain biased classifiers with complementary information.

4.1.3 Combining Features (FCOMB)

We have extended the basic set of six features with lexical information about words appearing in the local context of the target word, and with the ambiguity classes of the same words. In this way, we consider information about the surrounding words at three different levels of specificity: word form, POS tag, and ambiguity class.

Very similar to Brill's lexical patterns (Brill, 1995), we also have included features to capture collocational information. Such features are obtained by composition of the already described single attributes and they are sequences of contiguous words and/or POS tags (up to three items).

The resulting features were grouped according to their specificity to generate ensembles of eight trees[6]. The idea here is that specific information (lexical attributes and collocational patterns) would produce classifiers that cover concrete cases (hopefully, with a high precision), while more general information (POS tags) would produce more general (but probably less precise) trees. The combination of both type of trees should perform better because of the complementarity of the information.

[5] Several authors indicate that most of the potential improvement provided by bagging is obtained within the first ten replicates.

[6] The features for dealing with unknown words were combined in a similar way to create ensembles of 10 trees. For details, see (Màrquez, 1999).

4.2 Generating Pseudo–Examples (CPD)

Breiman (1998), describes a simple and effective method for generating new pseudo–examples from existing data and incorporating them into a tree–based learning algorithm to increase prediction accuracy in domains with few training examples. We call this method CPD (standing for *generation of Convex Pseudo–Data*).

The method for obtaining new data from the old is similar to the process of *gene combination* to create new generations in genetic algorithms. First, two examples of the same class are selected at random from the training set. Then, a new example is generated from them by selecting attributes from one or another parent according to a certain probability. This probability depends on a single *generation parameter* (a real number between 0 and 1), which regulates the amount of change allowed in the combination step.

In the original paper, Breiman does not propose any optimization of the generation parameter, instead, he performs a limited amount of trials with different values and simply reports the best result. In our domain, we observed a big variance on the results depending on the concrete values of the generation parameter. Instead of trying to tune it, we generate several training sets using different values of the generation parameter and we construct an ensemble of decision trees. In this way, we make the global classifier independent of the particular choice, and we generally obtain a combined result which is more accurate than any of the individuals.

5 Experiments and Results

5.1 Constructing and Evaluating Ensembles

First, the three types of ensembles were applied to the 19 selected ambiguity classes in order to decide which is the best in each case. The evaluation was performed by means of a 10–fold cross–validation using the training corpus. The obtained results confirm that all methods contribute to improve accuracy in almost all domains. The absolute improvement is not very impressive but the variance is generally very low and, so, the gain is statistically significant in the majority of cases. Summarizing, BAG wins in 8 cases, FCOMB in 9, and FSC in 2 (including the `unknown-word` class).

These results are reported in table 4, in which the error rate of a single basic tree is compared to the results of the ensembles for each ambiguity class[7]. The last column presents the percentage of error reduction for the best method in each row.

Second, CPD was applied to the 82 selected ambiguity classes, with positive results in 59 cases, from which 25 were statistically significant (again in a 10–fold cross–validation experiment). These 25 classes agglutinate 20,937 examples and the error rate was diminished, on average, from 20.16% to 18.17%.

5.2 Tagging with the Enriched Model

Ensembles of classifiers were learned for the ambiguity classes explained in the previous sections using the best technique in each case. These ensembles were included in the tree–base, used by the basic taggers of section 3, substituting the corresponding individual trees, and both taggers were tested again using the enriched model.

At runtime, the combination of classifiers was done by averaging the results of each individual decision tree.

In order to test the relative improvement of each component, the inclusion of the ensembles is performed in three steps: 'CPD' stands for the ensembles for infrequent ambiguity classes, 'ENS' stands for the ensembles for frequent ambiguity classes and unknown words, and 'CPD+ENS' stands for the inclusion of both. Results are described in table 5.

Some important conclusions are:

- The best result of each tagger is significantly better than each corresponding basic version, and the accuracy consistently grows as more components are added.
- The relative improvement of STT^+ is lower than those of RTT and STT, suggesting than the better the tree–based model is, the less relevant is the inclusion of n–gram information.
- The special treatment of low frequent ambiguity classes results in a very small contribution, indicating that there is no much

[7] These figures are calculated by averaging the results of the ten folds.

	A-class	#exs	%exs	Basic	BAG	FSC	FCOMB	BestER
1	IN-RB-RP	34,489	10.16%	8.30%	7.31%	7.79%	**7.23%**	12.89%
2	VBD-VBN	25,882	7.63%	7.44%	**5.93%**	6.64%	6.28%	20.30%
3	NN-VB-VBP	24,522	7.23%	4.10%	3.70%	3.84%	**3.58%**	12.68%
4	VB-VBP	17,788	5.24%	4.13%	**3.62%**	3.94%	3.76%	12.35%
5	JJ-NN	17,077	5.03%	14.71%	**13.30%**	13.50%	13.55%	9.59%
6	NNS-VBZ	15,295	4.51%	5.14%	4.37%	4.59%	**4.34%**	15.56%
7	NN-NNP	13,824	4.07%	9.67%	9.10%	8.37%	**6.83%**	29.37%
8	JJ-VBD-VBN	11,403	3.36%	19.18%	17.91%	18.05%	**17.27%**	9.96%
9	NN-VBG	9,597	2.83%	14.11%	**12.53%**	12.93%	12.99%	11.20%
10	JJ-NNP	8,724	2.57%	5.10%	4.50%	4.56%	**4.35%**	14.71%
11	JJ-RB	8,722	2.57%	10.45%	**8.86%**	9.75%	9.68%	15.22%
12	DT-IN-RB-WDT	8,419	2.48%	7.01%	**6.49%**	6.84%	6.53%	7.42%
13	JJR-RBR	2,868	0.85%	16.40%	15.84%	15.28%	**14.72%**	10.24%
14	NNP-NNPS-NNS	2,808	0.83%	36.50%	36.50%	35.14%	**35.00%**	4.11%
15	JJ-NN-RB	2,625	0.77%	15.31%	13.32%	**11.83%**	12.44%	22.73%
16	JJ-NN-VB	2,145	0.63%	13.32%	13.87%	12.99%	**12.75%**	4.28%
17	JJ-NN-VBG	1,986	0.59%	20.30%	**17.98%**	18.79%	18.23%	11.43%
18	JJ-VBG	1,980	0.58%	21.11%	**18.89%**	19.39%	19.60%	10.52%
	Total	210,154	61.93%	9.35%	8.38%	8.61%	8.25%	**13.40%**
19	unknown-word	22,594	—	20.87%	17.47%	**16.86%**	17.21%	**19.26%**

Table 4: Comparative results (error rates) of different ensembles on the most significant ambiguity classes

Tagger	Overall	Known	Ambig.	Unknown	Speed	Memory
RTT	96.61%	97.00%	91.36%	79.21%	426 w/s	0.68Mb
RTT(CPD)	96.66%	97.06%	91.51%	79.25%	366 w/s	0.93Mb
RTT(ENS)	96.99%	97.30%	92.23%	83.25%	97 w/s	3.53Mb
RTT(CPD+ENS)	97.05%	97.37%	92.48%	83.30%	89 w/s	3.78Mb
STT	96.63%	97.02%	91.40%	79.60%	321 w/s	0.68Mb
STT(CPD)	96.69%	97.07%	91.56%	79.69%	261 w/s	0.93Mb
STT(ENS)	97.05%	97.36%	92.38%	83.78%	70 w/s	3.53Mb
STT(CPD+ENS)	97.10%	97.40%	92.51%	83.68%	64 w/s	3.78Mb
STT+	96.84%	97.21%	91.95%	80.70%	302 w/s	0.90Mb
STT+(CPD)	96.88%	97.25%	92.09%	80.77%	235 w/s	1.15Mb
STT+(ENS)	97.19%	97.48%	92.73%	84.47%	65 w/s	3.75Mb
STT+(CPD+ENS)	97.22%	97.51%	92.81%	84.54%	60 w/s	3.97Mb

Table 5: Tagging accuracy, speed, and storage requirements of enriched RTT and STT taggers

to win from these classes, unless we were able to fix their errors in a much greater proportion than we really did.

- The price to pay for the enriched models is a substantial overhead in storage requirement and speed decreasing, which in the worst case is divided by 5.

In order to compare our results to others, we list in table 6 the results reported by several state-of-the-art POS taggers, tested on the WSJ corpus with the open vocabulary assumption. In that table, TBL stands for Brill's transformation-based error-driven tagger (Brill, 1995), ME stands for a tagger based on the maximum entropy modelling (Ratnaparkhi, 1996), SPATTER stands for a statistical parser based on decision trees (Magerman, 1996), IGTREE stands for the memory-based tagger by Daelemans et al. (1996), and, finally, TComb stands for a tagger that works by combination of a statistical trigram-based tagger,

Tagger	Train	Test	Overall	Known	Unknown	Ambig
TBL	950 Kw	150 Kw	96.6%	—	82.2%	—
ME	963 Kw	193 Kw	96.5%	—	86.2%	—
SPATTER	~975 Kw	47 Kw	96.5%	—	—	—
IGTREE	2,000 kw	200 Kw	96.4%	96.7%	90.6%	—
TComb	1,100 Kw	265 Kw	97.2%	—	—	—
STT^{+}(CPD+ENS)	998 Kw	175 Kw	97.2%	97.5%	84.5%	92.8%

Table 6: Comparison of different taggers on the WSJ corpus

TBL and ME (Brill and Wu, 1998).

Comparing to all the individual taggers we observe that our approach reports the highest accuracy, and that it is comparable to that of TComb obtained by the combination of three taggers. This is encouraging, since we have improved an individual POS tagger which could be further introduced as a better component in an ensemble of taggers.

Unfortunately, the performance on unknown words is difficult to compare, since it strongly depends on the used lexicon. For instance, IGTREE does not include in the lexicon the numbers appearing in the training set, and, so, any number in the test set is considered unknown (they report an unusually high percentage of unknown words: 5.5% compared to our 2.25%). The fact that numbers are very easy to recognize could explain their outstanding results on tagging unknown words. ME also reports a higher percentage of unknown words, 3.2%, while TBL says nothing about this issue.

6 Conclusions and Further Work

In this paper, we have applied several ML techniques for constructing ensembles of classifiers to address the most representative and/or difficult cases of ambiguity within a decision–tree–based English POS tagger. As a result, the overall accuracy has been significantly improved. Comparing to other approaches, we see that our tagger performs better on the WSJ corpus and under the open vocabulary assumption, than a number of state–of–the–art POS taggers, and similar to another approach based on the combination of several taggers[8].

[8]However, it has to be said that the pure statistical or machine–learning based approaches to POS tagging still significantly underperform some sophisticated manually constructed systems, such as the English shallow parser based on Constraint Grammars developed at the Helsinki University (Samuelsson and Voutilainen, 1997).

The cost of this improvement has been quantified in terms of storage requirement and speed of the resulting enriched taggers. Of course, there exists a clear tradeoff between accuracy and efficiency which should be resolved on the basis of the user needs. Although all proposed techniques are fully automatic, it has to be said that the construction of appropriate ensembles requires a significant human and computational effort.

There are several features that should be further studied with respect to the used methods for constructing the ensembles of decision trees, the way they are combined and included in the taggers, etc. However, we are now more interested on experimenting with the inclusion of our tagger as a component in an ensemble of pre-existing taggers, in the style of (Brill and Wu, 1998; van Halteren et al., 1998).

More generally, one may think that, after all the involved effort, the achieved improvement seems small. On this particular, we think that we are moving very close to the best achievable results using fully statistically–based techniques, and that some kind of specific human knowledge should be jointly considered in order to achieve the next qualitative step. We also think that other issues than simply 'accuracy rates' are becoming more important in order to test and evaluate the real utility of different approaches for tagging. Such aspects, that should be studied in the near future, refer to the ability of adapting to new domains (tuning), the types of errors committed and their influence on the task at hand, the language independence assumption, etc.

Acknowledgments

This research has been partially funded by the Spanish Research Department (CICYT's ITEM project TIC96–1243–C03–02), by the EU Com-

mission (EuroWordNet LE4003) and by the Catalan Research Department (CIRIT's consolidated research group 1997SGR 00051, and CREL project).

References

K. M. Ali and M. J. Pazzani. 1996. Error Reduction through Learning Multiple Descriptions. *Machine Learning*, 24(3):173–202.

A. Blum and T. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co–Training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory, COLT-98*, pages 92–100, Madison, Wisconsin.

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

L. Breiman. 1996. Bagging Predictors. *Machine Learning*, 24(2):123–140.

L. Breiman. 1998. Using Convex Pseudo–Data to Increase Prediction Accuracy. Technical Report, Statistics Department. University of California, Berkeley, CA.

E. Brill and J. Wu. 1998. Classifier Combination for Improved Lexical Disambiguation. In *Proceedings of the joint COLING-ACL'98*, pages 191–195, Montréal, Canada.

E. Brill. 1995. Transformation–based Error–driven Learning and Natural Language Processing: A Case Study in Part–of–speech Tagging. *Computational Linguistics*, 21(4):543–565.

E. Charniak. 1993. *Statistical Language Learning.* The MIT Press, Cambridge, Massachusetts.

K.J. Cherkauer. 1996. Human Expert–level Performance on a Scientific Image Analysis Task by a System Using Combined Artificial Neural Networks. In P. Chan, editor, *Working Notes of the AAAI Workshop on Integrating Multiple Learned Models*, pages 15–21.

K. W. Church. 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceedings of the 1st Conference on Applied Natural Language Processing, ANLP*, pages 136–143. ACL.

D. Cutting, J. Kupiec, J. Pederson, and P. Sibun. 1992. A Practical Part–of–speech Tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing, ANLP*, pages 133–140. ACL.

W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A Memory–Based Part–of–speech Tagger Generator. In *Proceedings of the 4th Workshop on Very Large Corpora*, pages 14–27, Copenhagen, Denmark.

S. J. DeRose. 1988. Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics*, 14:31–39.

T. G. Dietterich and G. Bakiri. 1995. Solving Multiclass Learning Problems via Error–Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286.

T. G. Dietterich. 1997. Machine Learning Research: Four Current Directions. *AI Magazine*, 18(4):97–136.

T. G. Dietterich. 1998. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, pages 1–22.

Y. Freund and R. E. Schapire. 1995. A Decision–Theoretic Generalization of On–line Learning and an Application to Boosting. In *Proceedings of the 2nd European Conference on Computational Learning Theory, EuroCOLT'95*, Barcelona, Spain.

A. R. Golding and D. Roth. 1999. A Winnow-based Approach to Spelling Correction. *Machine Learning, Special issue on Machine Learning and Natural Language Processing.*

H. van Halteren, J. Zavrel, and W. Daelemans. 1998. Improving Data Driven Wordclass Tagging by System Combination. In *Proceedings of the joint COLING-ACL'98*, pages 491–497, Montréal, Canada.

H. van Halteren. 1996. *Comparison of Tagging Strategies, a Prelude to Democratic Tagging.* S. Hockney and N. Ide (eds.), Clarendon Press. Research in Humanities Computing 4. Selected papers for the ALLC/ACH Conference, Christ Church, Oxford.

F. Karlsson, A. Voutilainen, J. Heikkilä, and A. Anttila, editors. 1995. *Constraint Grammar: A Language–Independent System for Parsing Unrestricted Text.* Mouton de Gruyter, Berlin and New York.

I. Kononenko, E. Šimec, and M. Robnik-Šikonja. 1995. Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF. *Applied Intelligence*, 10:39–55.

R. López de Mántaras. 1991. A Distance–Based Attribute Selection Measure for Decision Tree Induction. *Machine Learning, Kluwer Academic*, 6(1):81–92.

D. M. Magerman. 1996. Learning Grammatical Structure Using Statistical Decision–Trees. In *Proceedings of the 3rd International Colloquium on Grammatical Inference, ICGI.* Springer-Verlag Lecture Notes Series in Artificial Intelligence 1147.

L. Màrquez and H. Rodríguez. 1997. Automatically Acquiring a Language Model for POS Tagging Using Decision Trees. In *Proceedings of the Second Conference on Recent Advances in Natural Lan-*

guage Processing, RANLP, pages 27–34, Tzigov Chark, Bulgaria.

L. Màrquez, L. Padró, and H. Rodríguez. 1998. Improving Tagging Accuracy by Voting Taggers. In *Proceedings of the 2nd Conference on Natural Language Processing & Industrial Applications, NLP+IA/TAL+AI*, pages 149–155, New Brunswick, Canada.

L. Màrquez. 1999. *Part-of-Speech Tagging: A Machine Learning Approach based on Decision Trees.* Phd. Thesis, Dep. Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya. (Forthcoming)

M. Moreira and E. Mayoraz. 1998. Improved Pairwise Coupling Classification with Correcting Classifiers. In *Proceedings of the 10th European Conference on Machine Learning, ECML*, pages 160–171, Chemnitz, Germany.

B. Parmanto, P.W. Munro, and H.R. Doyle. 1996. Improving Committee Diagnosis with Resampling Techniques. In M.C. Mozer D.S. Touretzky and M.E. Hesselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 882–888. MIT Press., Cambridge, MA.

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

L. R. Rabiner. 1990. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Readings in Speech Recognition (eds. A. Waibel, K. F. Lee). Morgan Kaufmann Publishers, Inc., San Mateo, CA.

A. Ratnaparkhi. 1996. A Maximum Entropy Part–of–speech Tagger. In *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing, EMNLP'96.*

C. Samuelsson and A. Voutilainen. 1997. Comparing a Linguistic and a Stochastic Tagger. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 246–253, Madrid, Spain.

R. E. Schapire and Y. Singer. 1998. BoosTexter: A system for multiclass multi-label text categorization. Unpublished. Postscript version available at AT&T Labs.

R. E. Schapire, Y. Singer, and A. Singhal. 1998. Boosting and Rocchio applied to text filtering. In *In Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval, SIGIR '98.*

S. Sestito and T. S. Dillon. 1994. *Automated Knowledge Acquisition.* T. S. Dillon (ed.), Series in Computer Systems Science and Engineering. Prentice Hall, New York/London.

K. Tumer and J. Ghosh. 1996. Error Correlation and Error Reduction in Ensemble Classifiers. *Connection Science. Special issue on combining artificial neural networks: ensemble approaches*, 8(3 and 4):385–404.