A Flexible POS Tagger Using an Automatically Acquired Language Model*

Lluís Màrquez
LSI - UPC
c/ Jordi Girona 1-3
08034 Barcelona. Catalonia
lluism@lsi.upc.es

Lluís Padró
LSI - UPC
c/ Jordi Girona 1-3
08034 Barcelona. Catalonia
padro@lsi.upc.es

Abstract

We present an algorithm that automatically learns context constraints using statistical decision trees. We then use the acquired constraints in a flexible POS tagger. The tagger is able to use information of any degree: n-grams, automatically learned context constraints, linguistically motivated manually written constraints, etc. The sources and kinds of constraints are unrestricted, and the language model can be easily extended, improving the results. The tagger has been tested and evaluated on the WSJ corpus.

1 Introduction

In NLP, it is necessary to model the language in a representation suitable for the task to be performed. The language models more commonly used are based on two main approaches: first, the linguistic approach, in which the model is written by a linguist, generally in the form of rules or constraints (Voutilainen and Järvinen, 1995). Second, the automatic approach, in which the model is automatically obtained from corpora (either raw or annotated)[1], and consists of n-grams (Garside et al., 1987; Cutting et al., 1992), rules (Hindle, 1989) or neural nets (Schmid, 1994). In the automatic approach we can distinguish two main trends: The low-level data trend collects statistics from the training corpora in the form of n-grams, probabilities, weights, etc. The high level data trend acquires more sophisticated information, such as context rules, constraints, or decision trees (Daelemans et al., 1996; Màrquez and Rodríguez, 1995; Samuelsson et al., 1996). The acquisition methods range from supervised-inductive-learning-from-example algorithms (Quinlan, 1986; Aha et al., 1991) to genetic algorithm strategies (Losee, 1994), through the transformation-based error-driven algorithm used in (Brill, 1995). Still another possibility are the hybrid models, which try to join the advantages of both approaches (Voutilainen and Padró, 1997).

We present in this paper a hybrid approach that puts together both trends in automatic approach and the linguistic approach. We describe a POS tagger based on the work described in (Padró, 1996), that is able to use bi/trigram information, automatically learned context constraints and linguistically motivated manually written constraints. The sources and kinds of constraints are unrestricted, and the language model can be easily extended. The structure of the tagger is presented in figure 1.

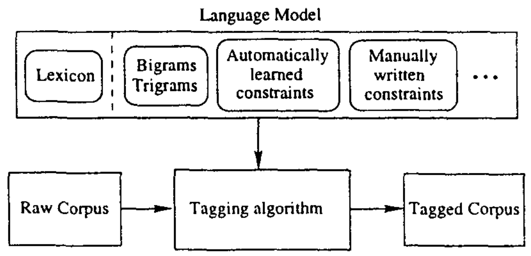

Figure 1: Tagger architecture.

We also present a constraint-acquisition algorithm that uses statistical decision trees to learn context constraints from annotated corpora and we use the acquired constraints to feed the POS tagger.

The paper is organized as follows. In section 2 we describe our language model, in section 3 we describe the constraint acquisition algorithm, and in section 4 we expose the tagging algorithm. Descriptions of the corpus used, the experiments performed and the results obtained can be found in sections 5 and 6.

2 Language Model

We will use a hybrid language model consisting of an automatically acquired part and a linguist-written part.

*This research has been partially funded by the Spanish Research Department (CICYT) and inscribed as TIC96-1243-C03-02

[1]When the model is obtained from annotated corpora we talk about supervised learning, when it is obtained from raw corpora training is considered unsupervised.

The automatically acquired part is divided in two kinds of information: on the one hand, we have bigrams and trigrams collected from the annotated training corpus (see section 5 for details). On the other hand, we have context constraints learned from the same training corpus using statistical decision trees, as described in section 3.

The linguistic part is very small —since there were no available resources to develop it further— and covers only very few cases, but it is included to illustrate the flexibility of the algorithm.

A sample rule of the linguistic part:

```
10.0 (%vauxiliar%)
   (-[VBN IN , : JJ JJS JJR])+
   <VBN>;
```

This rule states that a tag *past participle* (**VBN**) is very compatible (10.0) with a left context consisting of a **%vauxiliar%** (previously defined macro which includes all forms of "have" and "be") provided that all the words in between don't have any of the tags in the set **[VBN IN , : JJ JJS JJR]**. That is, this rule raises the support for the tag *past participle* when there is an auxiliary verb to the left but only if there is not another candidate to be a past participle or an adjective inbetween. The tags **[IN , :]** prevent the rule from being applied when the auxiliary verb and the participle are in two different phrases (a comma, a colon or a preposition are considered to mark the beginning of another phrase).

The constraint language is able to express the same kind of patterns than the Constraint Grammar formalism (Karlsson et al., 1995), although in a different formalism. In addition, each constraint has a compatibility value that indicates its strength. In the middle run, the system will be adapted to accept CGs.

3 Constraint Acquisition

Choosing, from a set of possible tags, the proper syntactic tag for a word in a particular context can be seen as a problem of classification. Decision trees, recently used in NLP basic tasks such as tagging and parsing (McCarthy and Lehnert, 1995; Daelemans et al., 1996; Magerman, 1996), are suitable for performing this task.

A decision tree is a n–ary branching tree that represents a *classification rule* for classifying the *objects* of a certain domain into a set of mutually exclusive *classes*. The domain objects are described as a set of attribute-value pairs, where each *attribute* measures a relevant feature of an object taking a (ideally small) set of discrete, mutually incompatible *values*. Each non–terminal node of a decision tree represents a question on (usually) one attribute. For each possible value of this attribute there is a branch to follow. Leaf nodes represent concrete classes.

Classify a new object with a decision tree is simply following the convenient path through the tree until a leaf is reached.

Statistical decision trees only differs from common decision trees in that leaf nodes define a conditional probability distribution on the set of classes.

It is important to note that decision trees can be directly translated to rules considering, for each path from the root to a leaf, the conjunction of all questions involved in this path as a condition and the class assigned to the leaf as the consequence. Statistical decision trees would generate rules in the same manner but assigning a certain degree of probability to each answer.

So the learning process of contextual constraints is performed by means of learning one statistical decision tree for each class of POS ambiguity[2] and converting them to constraints (rules) expressing compatibility/incompatibility of concrete tags in certain contexts.

Learning Algorithm

The algorithm we used for constructing 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, but not on the examples order (Quinlan, 1986). Briefly, the algorithm works as a recursive process that departs from considering the whole set of examples at the root level and constructs the tree in a top–down way branching at any non–terminal node according to a certain *selected* attribute. The different values of this attribute induce a partition of the set of examples in the corresponding subsets, in which the process is applied recursively in order to generate the different subtrees. The recursion ends, in a certain node, either when all (or almost all) the remaining examples belong to the same class, or when the number of examples is too small. These nodes are the leafs of the tree and contain the conditional probability distribution, of its associated subset of examples, on the possible classes.

The heuristic function for selecting the most useful attribute at each step is of a crucial importance in order to obtain simple trees, since no backtracking is performed. There exist two main families of attribute–selecting functions: *information*–based (Quinlan, 1986; López, 1991) and *statistically*–based (Breiman et al., 1984; Mingers, 1989).

Training Set

For each class of POS ambiguity the initial example set is built by selecting from the training corpus

[2]Classes of ambiguity are determined by the groups of possible tags for the words in the corpus, i.e, *noun-adjective*, *noun-adjective-verb*, *preposition-adverb*, etc.

all the occurrences of the words belonging to this ambiguity class. More particularly, the set of attributes that describe each example consists of the part-of-speech tags of the neighbour words, and the information about the word itself (orthography and the proper tag in its context). The window considered in the experiments reported in section 6 is 3 words to the left and 2 to the right. The following are two real examples from the training set for the words that can be preposition and adverb at the same time (IN-RB conflict).

```
VB DT NN <"as",IN> DT JJ
NN IN NN <"once",RB> VBN TO
```

Approximately 90% of this set of examples is used for the construction of the tree. The remaining 10% is used as fresh test corpus for the pruning process.

Attribute Selection Function

For the experiments reported in section 6 we used a attribute selection function due to López de Mántaras (López, 1991), which belongs to the information-based family. Roughly speaking, it defines a distance measure between partitions and selects for branching the attribute that generates the closest partition to the *correct partition*, namely the one that joins together all the examples of the same class.

Let X be a set of examples, C the set of classes and $P_C(X)$ the partition of X according to the values of C. The selected attribute will be the one that generates the closest partition of X to $P_C(X)$. For that we need to define a distance measure 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 measure reflects the randomness of distribution of the elements of X between the classes of the partition induced by A. If we consider now 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

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

It is easy to show that the measure

$$d(P_A(X), P_B(X)) = I(P_B(X)|P_A(X)) + I(P_A(X)|P_B(X))$$

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 the selected attribute will be that one that minimizes the measure: $d_N(P_C(X), P_A(X))$.

Branching Strategy

Usual TDIDT algorithms consider a branch for each value of the selected attribute. This strategy is not feasible when the number of values is big (or even infinite). In our case the greatest number of values for an attribute is 45 —the tag set size— which is considerably big (this means that the branching factor could be 45 at every level of the tree[3]). Some systems perform a previous recasting of the attributes in order to have only binary-valued attributes and to deal with binary trees (Magerman, 1996). This can always be done but the resulting features lose their intuition and direct interpretation, and explode in number. We have chosen a mixed approach which consist of splitting for all values and afterwards joining the resulting subsets into groups for which we have not enough statistical evidence of being different distributions. This statistical evidence is tested with a χ^2 test at a 5% level of significance. In order to avoid zero probabilities the following smoothing is performed. In a certain set of examples, the probability of a tag t_i is estimated by

$$\hat{p}(t_i) = \frac{|t_i| + \frac{1}{m}}{n+1},$$

where m is the number of possible tags and n the number of examples.

Additionally, all the subsets that don't imply a reduction in the *classification error* are joined together in order to have a bigger set of examples to be treated in the following step of the tree construction. The classification error of a certain node is simply: $1 - \max_{1 \leq i \leq m} (\hat{p}(t_i))$.

Experiments reported in (Màrquez and Rodríguez, 1995) show that in this way more compact and predictive trees are obtained.

Pruning the Tree

Decision trees that correctly classify all examples of the training set are not always the most predictive ones. This is due to the phenomenon known as *overfitting*. It occurs when the training set has a certain amount of misclassified examples, which is obviously the case of our training corpus (see section 5). If we

[3]In real cases the branching factor is much lower since not all tags appear always in all positions of the context.

force the learning algorithm to completely classify the examples then the resulting trees would fit also the noisy examples.

The usual solutions to this problem are: 1) Prune the tree, either during the construction process (Quinlan, 1993) or afterwards (Mingers, 1989); 2) Smooth the conditional probability distributions using fresh corpus[4] (Magerman, 1996).

Since another important requirement of our problem is to have small trees we have implemented a post-pruning technique. In a first step the tree is completely expanded and afterwards it is pruned following a minimal cost-complexity criterion (Breiman et al., 1984). Roughly speaking this is a process that iteratively cut those subtrees producing only marginal benefits in accuracy, obtaining smaller trees at each step. The trees of this sequence are tested using a, comparatively small, fresh part of the training set in order to decide which is the one with the highest degree of accuracy on new examples. Experimental tests (Màrquez and Rodríguez, 1995) have shown that the pruning process reduces tree sizes at about 50% and improves their accuracy in a 2–5%.

An Example

Finally, we present a real example of the simple acquired contextual constraints for the conflict IN–RB (preposition-adverb).

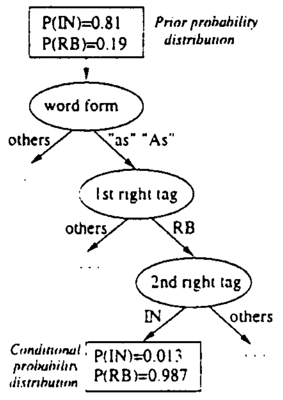

Figure 2: Example of a decision tree branch.

The tree branch in figure 2 is translated into the following constraints:

```
-5.81 <["as" "As"],IN> ([RB]) ([IN]);
2.366 <["as" "As"],RB> ([RB]) ([IN]);
```

which express the compatibility (either positive or negative) of the word–tag pair in angle brackets with the given context. The compatibility value for each constraint is the mutual information between the tag and the context (Cover and Thomas, 1991). It is directly computed from the probabilities in the tree.

[4]Of course, this can be done only in the case of statistical decision trees.

4 Tagging Algorithm

Usual tagging algorithms are either n-gram oriented –such as Viterbi algorithm (Viterbi, 1967)– or ad-hoc for every case when they must deal with more complex information.

We use relaxation labelling as a tagging algorithm. Relaxation labelling is a generic name for a family of iterative algorithms which perform function optimization, based on local information. See (Torras, 1989) for a summary. Its most remarkable feature is that it can deal with any kind of constraints, thus the model can be improved by adding any constraints available and it makes the tagging algorithm independent of the complexity of the model.

The algorithm has been applied to part-of-speech tagging (Padró, 1996), and to shallow parsing (Voutilainen and Padró, 1997).

The algorithm is described as follows:

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

Let $t_i = \{t^i_1, t^i_2, \ldots, t^i_{m_i}\}$ be the set of possible labels (POS tags) for variable v_i.

Let CS be a set of constraints between the labels of the variables. Each constraint $C \in CS$ states a "compatibility value" C_r for a combination of pairs variable–label. Any number of variables may be involved in a constraint.

The aim of the algorithm is to find a weighted labelling[5] such that "global consistency" is maximized. Maximizing "global consistency" is defined as maximizing for all v_i, $\sum_j p^i_j \times S_{ij}$, where p^i_j is the weight for label j in variable v_i and S_{ij} the support received by the same combination. The support for the pair variable–label expresses *how compatible* that pair is with the labels of neighbouring variables, according to the constraint set. It is a vector optimization and doesn't maximize *only* the sum of the supports of all variables. It finds a weighted labelling such that any other choice wouldn't increase the support for *any* variable.

The support is defined as the sum of the influence of every constraint on a label.

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

where:
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^i_j).
$Inf(r) = C_r \times p^{r_1}_{k_1}(m) \times \ldots \times p^{r_d}_{k_d}(m)$, is the product of the current weights[6] for the labels appearing

[5]A weighted labelling is a weight assignment for each label of each variable such that the weights for the labels of the same variable add up to one.

[6]$p^r_k(m)$ is the weight assigned to label k for variable r at time m.

in the constraint except (v_i, t^i_j) (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).

Briefly, what the algorithm does is:

1. Start with a random weight assignment[7].
2. Compute the support value for each label of each variable.
3. Increase the weights of the labels more compatible with the context (support greater than 0) and decrease those of the less compatible labels (support less than 0)[8], using the updating function:

$$p^i_j(m+1) = \frac{p^i_j(m) \times (1 + S_{ij})}{\sum_{k=1}^{k_i} p^i_k(m) \times (1 + S_{ik})}$$

$$\text{where} \quad -1 \leq S_{ij} \leq +1$$

4. If a stopping/convergence criterion[9] is satisfied, stop, otherwise go to step 2.

The cost of the algorithm is proportional to the product of the number of words by the number of constraints.

5 Description of the corpus

We used the Wall Street Journal corpus to train and test the system. We divided it in three parts: 1,100 Kw were used as a training set, 20 Kw as a model-tuning set, and 50 Kw as a test set.

The tag set size is 45 tags. 36.4% of the words in the corpus are ambiguous, and the ambiguity ratio is 2.44 tags/word over the ambiguous words, 1.52 overall.

We used a lexicon derived from training corpora, that contains all possible tags for a word, as well as their lexical probabilities. For the words in test corpora not appearing in the train set, we stored all possible tags, but no lexical probability (i.e. we assume uniform distribution)[10].

The noise in the lexicon was filtered by manually checking the lexicon entries for the most frequent 200 words in the corpus[11] to eliminate the tags due to errors in the training set. For instance the original lexicon entry (numbers indicate frequencies in the training corpus) for the very common word *the* was

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

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

The training set was used to estimate bi/trigram statistics and to perform the constraint learning.

The model-tuning set was used to tune the algorithm parameterizations, and to write the linguistic part of the model.

The resulting models were tested in the fresh test set.

6 Experiments and results

The whole WSJ corpus contains 241 different classes of ambiguity. The 40 most representative classes[12] were selected for acquiring the corresponding decision trees. That produced 40 trees totaling up to 2995 leaf nodes, and covering 83.95% of the ambiguous words. Given that each tree branch produces as many constraints as tags its leaf involves, these trees were translated into 8473 context constraints.

We also extracted the 1404 bigram restrictions and the 17387 trigram restrictions appearing in the training corpus.

Finally, the model-tuning set was tagged using a bigram model. The most common errors commited by the bigram tagger were selected for manually writing the sample linguistic part of the model, consisting of a set of 20 hand-written constraints.

From now on C will stands for the set of acquired context constraints. **B** for the bigram model, **T** for the trigram model, and **H** for the hand-written constraints. Any combination of these letters will indicate the joining of the corresponding models (**BT**, **BC**, **BTC**, etc.).

In addition, **ML** indicates a baseline model containing no constraints (this will result in a most-likely tagger) and **HMM** stands for a hidden Markov model bigram tagger (Elworthy, 1992).

We tested the tagger on the 50 Kw test set using all the combinations of the language models. Results are reported below.

The effect of the acquired rules on the number of errors for some of the most common cases is shown in table 1. XX/YY stands for an error consisting of a word tagged YY when it should have been XX. Table 2 contains the meaning of all the involved tags.

Figures in table 1 show that in all cases the learned constraints led to an improvement.

It is remarkable that when using C alone, the number of errors is lower than with any bigram

[7] We use lexical probabilities as a starting point.

[8] Negative values for support indicate *incompatibility*.

[9] We use the criterion of stopping when there are no more changes, although more sophisticated heuristic procedures are also used to stop relaxation processes (Eklundh and Rosenfeld, 1978; Richards et al. , 1981).

[10] That is, we assumed a morphological analyzer that provides all possible tags for unknown words.

[11] The 200 most frequent words in the corpus cover over half of it.

[12] In terms of number of examples.

	ML	C	B	BC	T	TC	BT	BTC
JJ/NN+NN/JJ	73+137	70+94	73+112	69+102	57+103	61+95	67+101	62+93
VBD/VBN+VBN/VBD	176+190	71+66	88+69	63+56	56+57	55+57	65+60	59+61
IN/RB+RB/IN	31+132	40+69	66+107	43+17	77+68	47+67	65+98	46+83
VB/VBP+VBP/VB	128+147	30+26	49+43	32+27	31+32	32+18	28+32	28+32
NN/NNP+NNP/NN	70+11	44+12	72+17	45+16	69+27	50+18	71+20	62+15
NNP/NNPS+NNPS/NNP	45+14	37+19	45+13	46+15	54+12	51+12	53+14	51+14
"that"	187	53	66	45	60	40	57	45
Total	1341	631	820	630	703	603	731	651

Table 1: Number of some common errors commited by each model

NN	Noun
JJ	Adjective
VBD	Verb - past tense
VBN	Verb - past participle
RB	Adverb
IN	Preposition
VB	Verb - base form
VBP	Verb - personal form
NNP	Proper noun
NNPS	Plural proper noun

Table 2: Tag meanings

and/or trigram model, that is, the acquired model performs better than the others estimated from the same training corpus.

We also find that the cooperation of a bigram or trigram model with the acquired one, produces even better results. This is not true in the cooperation of bigrams and trigrams with acquired constraints (**BTC**), in this case the synergy is not enough to get a better joint result. This might be due to the fact that the noise in **B** and **T** adds up and overwhelms the context constraints.

The results obtained by the baseline taggers can be found in table 3 and the results obtained using all the learned constraints together with the bi/trigram models in table 4.

	ambiguous	overall
ML	85.31%	94.66%
HMM	91.75%	97.00%

Table 3: Results of the baseline taggers

On the one hand, the results in tables 3 and 4 show that our tagger performs slightly worse than a HMM tagger in the same conditions[13], that is, when using only bigram information.

[13]Hand analysis of the errors commited by the algorithm suggest that the worse results may be due to noise in the training and test corpora, i.e., relaxation algorithm seems to be more noise-sensitive than a Markov model. Further research is required on this point.

	ambiguous	overall
B	91.35%	96.86%
T	91.82%	97.03%
BT	91.92%	97.06%
C	91.96%	97.08%
BC	92.72%	97.36%
TC	92.82%	97.39%
BTC	92.55%	97.29%

Table 4: Results of our tagger using every combination of constraint kinds

On the other hand, those results also show that since our tagger is more flexible than a HMM, it can easily accept more complex information to improve its results up to 97.39% without modifying the algorithm.

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

Table 5: Results of our tagger using every combination of constraint kinds and hand written constraints

Table 5 shows the results adding the hand written constraints. The hand written set is very small and only covers a few common error cases. That produces poor results when using them alone (H), but they are good enough to raise the results given by the automatically acquired models up to 97.45%.

Although the improvement obtained might seem small, it must be taken into account that we are moving very close to the best achievable result with these techniques.

First, some ambiguities can only be solved with semantic information, such as the Noun–Adjective ambiguity for word *principal* in the phrase *the principal office*. It could be an adjective, meaning *the* *main office*, or a noun, meaning *the school head office*.

Second, the WSJ corpus contains noise (mistagged words) that affects both the training and the test sets. The noise in the training set produces noisy -and so less precise- models. In the test set, it produces a wrong estimation of accuracy, since correct answers are computed as wrong and vice-versa.

For instance, verb participle forms are sometimes tagged as such (*VBN*) and also as adjectives (*JJ*) in other sentences with no structural differences:

- ... failing_VBG to_TO voluntarily_RB submit_VB the_DT *requested_VBN* information_NN ...
- ... a_DT large_JJ sample_NN of_IN *married_JJ* women_NNS with_IN at_IN least_JJS one_CD child_NN ...

Another structure not coherently tagged are noun chains when the nouns are ambiguous and can be also adjectives:

- ... Mr._NNP Hahn_NNP ,_, the_DT 62-year-old_JJ chairman_NN and_CC *chief_NN executive_JJ officer_NN* of_IN Georgia-Pacific_NNP Corp._NNP ...
- ... Burger_NNP King_NNP 's_POS *chief_JJ executive_NN officer_NN* ,_, Barry_NNP Gibbons_NNP ,_, stars_VBZ in_IN ads_NNS saying_VBG ...
- ... and_CC Barrett_NNP B._NNP Weekes_NNP ,_, chairman_NN ,_, president_NN and_CC *chief_JJ executive_JJ officer_NN* ._.
- ... the_DT company_NN includes_VBZ Neil_NNP Davenport_NNP ,_, 47_CD ,_, president_NN and_CC *chief_NN executive_NN officer_NN* ;_:

All this makes that the performance cannot reach 100%, and that an accurate analysis of the noise in WSJ corpus should be performed to estimate the actual upper bound that a tagger can achieve on these data. This issue will be addressed in further work.

7 Conclusions

We have presented an automatic constraint learning algorithm based on statistical decision trees.

We have used the acquired constraints in a part-of-speech tagger that allows combining any kind of constraints in the language model.

The results obtained show a clear improvement in the performance when the automatically acquired constraints are added to the model. That indicates that relaxation labelling is a flexible algorithm able to combine properly different information kinds, and that the constraints acquired by the learning algorithm capture relevant context information that was not included in the n-gram models.

It is difficult to compare the results to other works, since the accuracy varies greatly depending on the corpus, the tag set, and the lexicon or morphological analyzer used. The more similar conditions reported in previous work are those experiments performed on the WSJ corpus: (Brill, 1992) reports 3-4% error rate, and (Daelemans et al., 1996) report 96.7% accuracy. We obtained a 97.39% accuracy with trigrams plus automatically acquired constraints, and 97.45% when hand written constraints were added.

8 Further Work

Further work is still to be done in the following directions:

- Perform a thorough analysis of the noise in the WSJ corpus to determine a realistic upper bound for the performance that can be expected from a POS tagger.

On the constraint learning algorithm:

- Consider more complex context features, such as non-limited distance or barrier rules in the style of (Samuelsson et al., 1996).
- Take into account morphological, semantic and other kinds of information.
- Perform a global smoothing to deal with low-frequency ambiguity classes.

On the tagging algorithms

- Study the convergence properties of the algorithm to decide whether the lower results at convergence are produced by the noise in the corpus.
- Use back-off techniques to minimize interferences between statistical and learned constraints.
- Use the algorithm to perform simultaneously POS tagging and word sense disambiguation, to take advantage of cross influences between both kinds of information.

References

D.W. Aha, D. Kibler and M. Albert. 1991 Instance-based learning algorithms. In *Machine Learning*. 7:37-66. Belmont, California.

L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone. 1984 Classification and Regression Trees. The Wadsworth Statistics/Probability Series. Wadsworth International Group, Belmont, California.

E. Brill. 1992 A Simple Rule-Based Part-of-Speech. In *Proceedings of the Third Conference on Applied Natural Language Processing.* ACL.

E. Brill. 1995 Unsupervised Learning of Disambiguation Rules for Part-of-speech Tagging. In *Proceedings of 3rd Workshop on Very Large Corpora.* Massachusetts.

T.M. Cover and J.A. Thomas (Editors) 1991 Elements of information theory. John Wiley & Sons.

D. Cutting, J. Kupiec, J. Pederson and P. Sibun. 1992 A Practical Part-of-Speech Tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing.*, ACL.

J. Eklundh and A. Rosenfeld. 1978 Convergence Properties of Relaxation Labelling. Technical Report no. 701. Computer Science Center. University of Maryland.

D. Elworthy. 1993 Part-of-Speech and Phrasal Tagging. Technical report, SPRIT BRA-7315 Acquilex II, Working Paper WP #10.

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

R. Garside, G. Leech and G. Sampson (Editors) 1987 *The Computational Analysis of English.* London and New York: Longman.

D. Hindle. 1989 Acquiring disambiguation rules from text. In *Proceedings ACL'89.*

F. Karlsson 1990 Constraint Grammar as a Framework for Parsing Running Text. In H. Karlgren (ed.), *Papers presented to the 13th International Conference on Computational Linguistics, Vol. 3.* Helsinki. 168–173.

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.

R. López. 1991 A Distance-Based Attribute Selection Measure for Decision Tree Induction. Machine Learning. Kluwer Academic.

R.M. Losee. 1994 Learning Syntactic Rules and Tags with Genetic Algorithms for Information Retrieval and Filtering: An Empirical Basis for Grammatical Rules. Information Processing & Management, May.

M. Magerman. 1996 Learning Grammatical Structure Using Statistical Decision-Trees. In *Lecture Notes in Artificial Intelligence 1147. Grammatical Inference: Learning Syntax from Sentences.* Proceedings ICGI-96. Springer.

L. Màrquez and H. Rodríguez. 1995 Towards Learning a Constraint Grammar from Annotated Corpora Using Decision Trees. ESPRIT BRA-7315 Acquilex II, Working Paper.

J.F. McCarthy and W.G. Lehnert. 1995 Using Decision Trees for Coreference Resolution. In *Proceedings of 14th International Joint Conference on Artificial Intelligence (IJCAI'95).*

J. Mingers. 1989 An Empirical Comparison of Selection Measures for Decision-Tree Induction. In *Machine Learning.* 3:319–342.

J. Mingers. 1989 An Empirical Comparison of Pruning Methods for Decision-Tree Induction. In *Machine Learning.* 4:227–243.

L. Padró. 1996 POS Tagging Using Relaxation Labelling. In *Proceedings of 16th International Conference on Computational Linguistics.* Copenhagen, Denmark.

J.R. Quinlan. 1986 Induction of Decision Trees. In *Machine Learning.* 1:81–106.

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

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

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

H. Schmid 1994 Part-of-speech tagging with neural networks. In *Proceedings of 15th International Conference on Computational Linguistics.* Kyoto, Japan.

C. Torras. 1989 Relaxation and Neural Learning: Points of Convergence and Divergence. *Journal of Parallel and Distributed Computing.* 6:217–244

A.J. Viterbi. 1967 Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. In *IEEE Transactions on Information Theory.* pg 260–269, April.

A. Voutilainen and T. Järvinen. 1995 Specifying a shallow grammatical representation for parsing purposes. In *Proceedings of the 7th meeting of the European Association for Computational Linguistics.* 210–214.

A. Voutilainen and L. Padró. 1997 Developing a Hybrid NP Parser. In *Proceedings of ANLP'97.*