A Case Study in Part-of-Speech Tagging Using the ICOPOST Toolkit

Ingo Schröder
Arbeitsbereich NATS
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
22527 Hamburg
Germany
ingo.schroeder@informatik.uni-hamburg.de

1 Introduction

Part-of-speech tagging is an important processing step for many natural language systems. It has been tackled with a number of different approaches, both machine learning algorithms and methods relying on rules that are hand crafted by human experts. This paper investigates and compares four of the more popular machine learning approaches to POS tagging that have been implemented within the ICOPOST toolkit [Schröder 2002] which is freely available under the GNU public license[1] from the author's home page at `http://nats-www.informatik.uni-hamburg.de/~ingo/icopost/`.

2 The task of tagging

Tagging is the task of classifying words in a natural language text with respect to a specific criterion. Different types of tagging can be distinguished based on the specific criterion employed:

Syntactic word class: Identify the syntactic category, i.e., the part-of-speech (POS), of a word in the context of a sentence. This helps subsequent stages of processing, e.g., parsing, because the ambiguity is reduced from the beginning. Usually, the process only takes into account the immediate neighborhood (but see subsequent sections).

> The horse raced[/VBD? VBN?] past the barn fell.

Word sense: Identify the intended meaning of a word in a given context. A successful disambiguation requires considering more global aspects of the utterance, e.g., what the topic of the discourse is about.

> Plants[/factory? vegetable?] are known to be dangerous.

Attachment: Identify the site in a sentence where a phrase attaches to. This usually requires syntactic as well as semantic information. Often (at least for the purpose of an evaluation) the problem is reduced to a binary decision.

[1] See `http://www.gnu.org/licenses/licenses.html`.

He read the article in[/←noun? ←verb?] the newspaper/train.

(Underspecified) syntactic parsing: This combines the task of POS tag resolution and attachment disambiguation. Either the exact position of attachment is left underspecified, i.e., only the direction of attachment is given, [Karlsson 1990; Karlsson et al. 1995; Joshi & Srinivas 1994] or a complete disambiguation is achieved [Heinecke et al. 1998; Järvinen & Tapanainen 1997; Brants, Skut & Krenn 1997].

Lassen Sie uns noch[/@AD-A>? @AD-A<?] einen Termin machen.

Sentence boundary detection: The (reduced) task is to distinguish interpunctuation marks that end a sentence and those that do not.

He saw Mr .[/EOS?] Jones surrounded by the crowd .[/EOS?]

Tagging based on syntactic word class is the most frequently encountered form of tagging in the field of natural language processing and this paper focusses on this type. It is interesting for a number of reasons:

- The problem is clearly defined and well understood.
- The task is relatively easy but hard enough. A "good" performance can already be achieved with very simple methods, but a perfect system ultimately requires complete understanding (It is said to be "AI-hard".).
- Evaluation methods and comparison measures are available.
- Because only a relatively simple annotation scheme is required, large corpora are available which make different machine learning approaches feasible.

3 Evaluation criteria

Two evaluation measures are widely used for evaluating POS taggers: precison & recall and accuracy & ambiguity.

Given a natural language text $w_1...w_n$ with corresponding POS tags $t_1...t_n$ which are assumed to be the "correct" tags (golden standard), a tagger can "guess" tags $G_1...G_n, G_i = \{g_i^1...g_i^{i_m}\}$ for the words, i.e., it suggests the (alternative) tags $g_i^1...g_i^{i_m}$ for word w_i. It is generally assumed that a tagger emits at least one guess per word, i.e., $\forall_{i=1}^n |G_i| \geq 1$.[2] A single tag tagger outputs exactly one guess per word, i.e., $\forall_{i=1}^n |G_i| = 1$, while a multi tag tagger potentially emits more than one tag per word.

Precision *pr* and recall *rec* are defined as follows:

$$\text{pr} = \frac{\#\text{ correct tags}}{\#\text{ emitted tags}} = \frac{\sum_{i=1}^n |G_i \cap \{t_i\}|}{\sum_{i=1}^n |G_i|}$$

[2]Note that this must not necessarily be the best choice for a tagger since by simply omitting a difficult decision it could increase its precision measure.

$$\text{rec} = \frac{\#\text{ correct tags}}{\#\text{ reference tags}} = \frac{\sum_{i=1}^{n} |G_i \cap \{t_i\}|}{n}$$

Obviously, a perfect precision can be achieved by emitting no tags at all and a perfect recall is possible by always emitting all possible tags. The goal is to achieve both a high precision and a high recall.

An alternative measure that we will use in this paper is accuracy and ambiguity. Since standard taggers are single tag tagger, i.e., they output exactly one tag per word, one often reports accuracy *acc* that is defined as recall above. For the special case of single tag taggers the definition can be simplified:

$$\text{acc}_{\text{single}} = \frac{\#\text{ correct tags}}{\#\text{ reference tags}} = \frac{\sum_{i=1}^{n} G_i = \{t_i\}}{n}$$

Ambiguity *amb* is defined as follows:

$$\text{amb} = \frac{\#\text{ emitted tags}}{\#\text{ reference tags}} = \frac{\sum_{i=1}^{n} |G_i|}{n}$$

For single tag taggers the ambiguity is always one by definition. When we report accuracies but no ambiguity values in the following sections, it is implied that we are talking about single tag taggers.

4 The corpora used

For all evaluations in this report we use one or both of two natural language corpora. This section briefly introduces both of them.

4.1 NEGRA corpus

The NEGRA corpus version 2.0 [Skut et al. 1997] consists of German sentences taken from the newspaper "Frankfurter Rundschau" annotated with POS tags, morphological information as well as discontinuous phrase structure trees. Only the tags have been used in this work.

The tagset consists of 55 tags and is largely identical to the Stuttgart-Tübingen tag set [Schiller et al. 1995, STTS].

Here is an example:

```
"/$( Sex/FM Sells/FM "$( ,/$, das/PDS wissen/VVFIN die/ART Marketing-Strategen/NN
nicht/PTKNEG erst/ADV seit/APPR Madonna/NE ./$.
```

Table 1 contains some corpus statistics.

The pseudo word '–' (two hyphens) has seven tags. However, it is not a real word but used to represent words obviously omitted in the corpus.

#tags	#types	%types	#tokens	%tokens
1	49,189	95.937%	238,545	67.178%
2	1,884	3.675%	45,586	12.838%
3	164	0.320%	46,789	13.176%
4	32	0.062%	20,090	5.658%
5	1	0.002%	2,715	0.765%
6	1	0.002%	1,363	0.384%
7	1	0.002%	8	0.002%
Σ	51,272	100.000%	355,096	100.000%

- no. of sentences: 20,602
- no. of word tokens: 355,096
- no. of word types: 51,272
- average sentence length: 17.24
- no. of part-of-speech tags: 55
- mean lexical tag ambiguity: 1.61
- tag unigram cross entropy: 4.27

Table 1: Corpus statistics for the NEGRA corpus version 2.

...	Boutros	Ghali	nimmt	**an**	den	Gesprächen	teil	.						
	NE	NE	VVFIN	APPR	ART	NN	PTKVZ	$.						
Diese	Zurückhaltung		kommt	**an**	in	Langenhain	.							
PDAT	NN		VVFIN	PTKVZ	APPR	NE	$.							
...	war	von	Beginn	**an**	die	bessere	Mannschaft	.						
	VAFIN	APPR	NN	APZR	ART	ADJA	NN	$.						
	Vom	15.	August	**an**	nimmt	...	Anmeldungen	entgegen	.					
	APPRART	ADJA	NN	APPO	VVFIN		NN	PTKVZ	$.					
	...	mußte	...	bereits	**an**	die	100	000	Mark	...	aufbringen	.		
		VMFIN		ADV	ADV	ART	CARD	CARD	NN		VVINF	$.		
Und	wie	die	Probleme	**an**	besten	formuliert	...	werden	können	.				
KON	PWAV	ART	NN	PTKA	ADJA	VVPP		VAINF	VMFIN	$.				

Table 2: Example concordances for 'an'.

The most ambiguous real word is 'an'. It is used as a preposition (APPR 77.5%), a separable verb prefix (PTKVZ 18.1%), the second part of a circumposition (APZR 3.8%), a postposition (APPO 5 times), an adverb (ADV once) and a particle for the comparative (PTKA once). The readings as postposition are probably annotation mistakes. The particle for the comparative is a typo in the corpus ('an besten' instead of 'am besten') (cf. Table 2).

Figure 1 shows the percentage of unknown words and the dictionary size depending on the number of tokens already seen. The statistics has been gathered from the German data of the 'European Corpus Initiative Multilingual Corpus I' of which the NEGRA corpus is a subcorpus. It can be observed that the unknown word rate converges to about two percent. Occasional down peaks are due to the local homogenity of subcorpora; up peaks occur between subcorpora. The rate of growth for the number of encountered words is almost constant after about two million words. This rejects the hypothesis that a large enough dictionary is sufficient to overcome the problem of unknown words.

4.2 Wall Street Journal corpus

The Wall Street Journal (WSJ) corpus [Marcus, Santorini & Marcinkiewicz 1993] consists of sentences taken from English newspaper text.

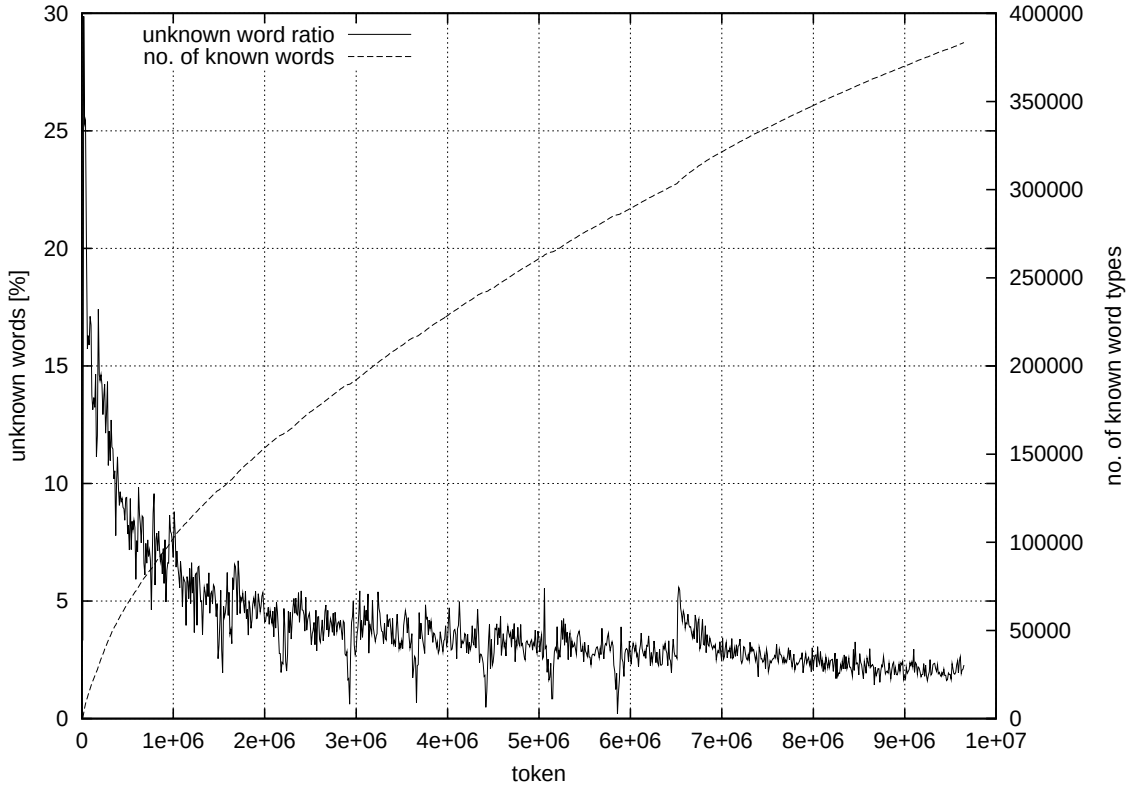


Figure 1: Unknown word rate and dictionary size depending on no. of word tokens seen: Only words consisting entirely of Latin characters plus German umlauts are considered.

Here is an example. Note that the period is used both as a word and as a part-of-speech tag.

There/EX were/VBD many/JJ pioneer/NN PC/NN contributors/NNS ./.

#tags	#type	%type	#token	%token
1	44,431	86.348%	577,419	44.789%
2	5,634	10.949%	268,422	20.821%
3	1,203	2.338%	236,128	18.316%
4	136	0.264%	34,773	2.697%
5	37	0.072%	32,214	2.499%
6	13	0.025%	113,344	8.792%
7	2	0.004%	26,901	2.087%
Σ	51,456	100.000%	1,289,201	100.000%

- no. of sentences: 56,762
- no. of word tokens: 1,289,201
- no. of word types: 51,456
- average sentence length: 22.66
- no. of part-of-speech tags: 45
- mean lexical tag ambiguity: 2.32
- tag unigram cross entropy: 4.33

Table 3: Corpus statistics for the WSJ corpus.

Table 3 contains some corpus statistics. This corpus if compared to the NEGRA corpus has a higher mean lexical tag ambiguity which seems logical because German has a richer morphology leading to more word forms per base form.

The two most ambiguous words are 'a' and 'down'. 'a' is most often used as an article DT (99.89%) but also occurs as a symbol SYM, foreign word FW, singular proper noun NNP, adjective JJ, list item marker LS and preposition IN. The tags for 'down' are more evenly distributed. It is tagged as adverb RB (49.19%), preposition IN (29.15%), particle RP (19.85%), adjective JJ, singular or mass noun NN, non-3rd-singular present verb VBP and comparative adverb RBR (cf. Table 4).

On the week , UAL was IN DT NN , NNP VBD	**down**	nearly 40 % . RB CD NN .
By 10:30 a.m. the Dow was IN CD NN DT NNP VBD	**down**	62.70 . IN CD .
He will keep the ball PRP MD VB DT NN	**down**	, move it around . RP , VB PRP RB .
... the first of five or six ... DT JJ IN CD CC CD	**down**	quarters . JJ NNS .
buyers could n't put their ... NNS ... MD RB VB PRP$	**down**	payment on a charge card ... NN IN DT NN NN ...
the company shuts ... DT NN ... NNS	**down**	operations VBP NNS ...
Texas Instruments went NNP NNP VBD	**down**	2 1|8 to RBR CD CD TO ...

Table 4: Example concordances for 'down'.

5 Tagging Paradigms

Quite a few different approaches to tagging have been developed:

Manual finite state rules are converted to finite-state machines [Voutilainen 1995] that consume the sequence of words and emit part-of-speech symbols. They can be implemented to run efficiently but require human expertise to write the rules. Samuelsson & Voutilainen [1997] compared such an approach to a statistical HMM tagger and reported that the manual tagger performs at least one magnitude better than the statistical one.

Neural networks are potential candidates for the classification task since they learn abstractions from examples [Schmid 1994a].

Decison trees are classification devices based on hierarchical clusters of questions. They have been used for natural language processing, e.g., for POS tagging [Magerman 1995; Schmid 1994b].

(Hidden) Markov models are the dominant model for speech recognition [Jelinek 1990], and also lend themselves to POS tagging where states correspond to (tuples of) tags and words are output symbols (cf. Section 5.1).

Maximum entropy models [Ratnaparkhi 1997b] avoid certain problems of statistical interdependence and have proven successful for tasks such as parsing and POS tagging (cf. Section 5.2).

Error-driven transformation-based learning derives transformation rules from intermediate classifications of the training set. During tagging, the rules are successively applied (cf. Section 5.3).

Example-based techniques find the training instance that is most similar to the current problem instance and assumes the same class for the new problem instance as for the similar one (cf. Section 5.4).

In this paper, we will present implementations for the last four paradigms and suggest different ways for combination to overcome certain shortcomings of the individual approaches.

5.1 Markov Models

(Hidden) Markov models (HMM) are stochastic finite-state automata with probabilities for the transitions between states and for the emission of symbols from the states [Rabiner 1990].

The forward-backward algorithm [Cutting et al. 1992; Charniak 1993; DeRose 1988] is used for unsupervised learning and relative frequencies can be used for supervised learning [Brants 2000]. The Viterbi algorithm is often used to find the most likely sequence of states for a given sequence of output symbols.

5.1.1 Background

Our presentation and implementation follow Brants [2000].

For presentation purposes, we assume a trigram model here, i.e., a Markov state is represented by a pair of tags.

Given a sequence of words $w_1...w_n$, we are looking for the sequence of tags T^* that maximizes the probability that the words are emitted by the model, i.e., we select the single most probable path (the Viterbi path) through the model. Merialdo [1994] shows that – in theory – the Viterbi criterion optimizes sentence accuracy while the maximum likelihood criterion[3] optimizes word accuracy. Although the evaluation of taggers is usually done at word level, the Viterbi algorithm is most often employed because (a) it leads to consistent tag sequences, (b) it is easier to implement, (c) it is more efficient and (d) the difference in accuracy at word level is negligible.

$$\begin{aligned} T^* &= \arg\max_{t_1...t_n} \prod_{i=1}^{n} \underbrace{P(t_i|t_{i-1}...t_1)}_{\text{transition prob.}} \cdot \underbrace{P(w_i|t_i...t_1)}_{\text{output prob.}} \\ &\approx \arg\max_{t_1...t_n} \prod_{i=1}^{n} P(t_i|t_{i-1}t_{i-2}) \cdot P(w_i|t_i) \end{aligned} \quad (1)$$

[3]The maximum likelihood criterion selects the most probable tag for each word individually by summing over all paths through the model.

$$\begin{aligned} T^* &= t_1^*, ..., t_n^* \\ t_i^* &= \arg\max_{t} \sum_{t_1...t_{i-1}tt_{i+1}...t_n} \prod_{j=1}^{n} P(t_j|t_{j-1}t_{j-2}) \cdot P(w_j|t_j) \end{aligned}$$

The *forward-backward algorithm* [Manning & Schütze 2000; Krenn & Samuelsson 1996] can be used to efficiently compute the most probable tag for each word.

Equation 1 uses the Markov assumptions that the transition and output probabilities only depend on the current state but not on earlier states.

The transition probabilities are estimated from the relative frequencies (Equations 2 to 4) and smoothed using deleted interpolation (Equation 5).

$$\hat{P}(t_k) = \frac{f(t_k)}{N} \tag{2}$$

$$\hat{P}(t_k|t_j) = \frac{f(t_j, t_k)}{f(t_j)} \tag{3}$$

$$\hat{P}(t_k|t_i, t_j) = \frac{f(t_i, t_j, t_k)}{f(t_i, t_j)} \tag{4}$$

$$P(t_i|t_{i-1}t_{i-2}) = \lambda_1\hat{P}(t_i) + \lambda_2\hat{P}(t_i|t_{i-1}) + \lambda_3\hat{P}(t_i|t_{i-2}t_{i-1}) \tag{5}$$

Lexical (output) probabilities are estimated using relative frequencies for known words (Equation 6). For unknown words, a successive abstraction scheme is employed which looks at successively shorter suffixes (Equations 7 to 9). We (again) borrow the calculation of the parameter θ from Brants [2000] and use a single context-independent value.

$$\hat{P}(w|t) = \frac{f(w, t)}{f(t)} \tag{6}$$

$$\hat{P}(t|c_{n-i+1}...c_n) = \frac{f(t, c_{n-i+1}...c_n)}{f(c_{n-i+1}...c_n)} \tag{7}$$

$$P(t) = \hat{P}(t) \tag{8}$$

$$P(t|c_{n-i+1}...c_n) = \frac{\hat{P}(t|c_{n-i+1}...c_n) + \theta P(t|c_{n-i+2}...c_n)}{1+\theta} \tag{9}$$

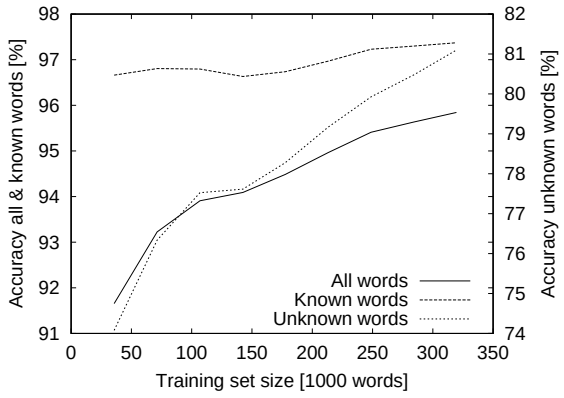
5.1.2 Implementation

Our implementation T3 [Schröder 2002] consists of approximately 1350 lines of C code and processes typically 5000 words/second.

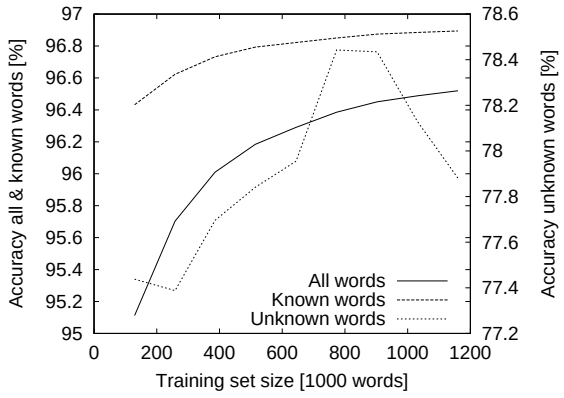
TnT is a similar, probably more sophisticated implementation by Thorsten Brants. It is available free of charge for research purposes from `http://www.coli.uni-sb.de/~thorsten/tnt/`.

5.1.3 Evaluation

Training set size Figure 2 graphically illustrates the run of the accuracy curves for the achieved accuracies for all words, known words only and unknown words only on the NEGRA and the WSJ corpora depending on the size of the training set. All numbers have been ten-fold cross-validated.



(a) NEGRA



(b) WSJ

Figure 2: T3: Accuracy depending on training set size.

Summary Table 5 summarizes the achieved tagging accuracies (ten-fold cross-validated) on the NEGRA and the WSJ corpora. The performance of 96.83% (NEGRA) and 96.52% (WSJ) word accuracy are comparable to the best results reported by other researchers, for instance those reported by Brants [2000].

	Sentences		All words		Known words		Unknown words	
Corpus	Acc.	SD	Acc.	SD	Acc.	SD	Acc.	SD
NEGRA	63.41	0.876	96.83	0.097	97.95	0.062	85.96	0.790
WSJ	50.56	0.573	96.52	0.044	96.89	0.034	77.88	0.839

Table 5: T3: Ten-fold cross-validated tagging results.

5.2 Maximum Entropy Models

Maximum entropy (ME) models are a probabilistic framework for machine learning [Ratnaparkhi 1997b].

Ratnaparkhi [1998a] is the major researcher who applied ME models to natural language processing problems: POS tagging [Ratnaparkhi 1996], prepositional phrase attachment [Ratnaparkhi, Reynar & Roukos 1994; Ratnaparkhi 1998b] and parsing [Ratnaparkhi, Roukos & Ward 1994; Ratnaparkhi 1997a].

5.2.1 Background

Intuitively but also from a theoretical point of view, it can be argued that if we have certain information (e.g., example outcomes) about a probability distribution, we should choose the "simplest" distribution that fits that information.

> "We shall not make any additional asumptions besides the explicit ones. From all admissible probability distributions, the one that maximizes the entropy (or "uncertainty") is the most appropriate choice." —— Ratnaparkhi [1998a]

Throwing a dice is a simple example. If we observe the 10 outcomes 1, 5, 1, 5, 5, 1, 1, 3, 1 and 5, then the "best estimation" of the probability distribution is $p(1) = 0.5, p(3) = 0.1$ and $p(5) = 0.4$ because this distribution does not assume anything beyond the observation, e.g., that the dice might be fair or that the probability of an even number is non-zero. Of course, this example simplifies matters; with only 10 outcomes no statistical generalization should be made.

Equation 10 formalizes this *Principle of Maximum Entropy*.

$$p^* = \arg\max_{p \in \mathcal{P}} H(p) \quad \text{with} \quad H(p) = -\sum_{a,b} \tilde{p}(b) p(a|b) \log p(a|b) \tag{10}$$

We model our expectations using contextual predicates and features. Given a set of possible contexts $\mathcal{B}$ and a set a outcomes $\mathcal{A}$ (or classes), our training set looks like this: $\mathcal{T} = \{(a_1, b_1), ..., (a_N, b_N)\}$ with $a_i \in \mathcal{A}$ and $b_i \in \mathcal{B}$. Contextual predicates have the form $cp : \mathcal{B} \mapsto \{0, 1\}$ and features map pairs of outcome and context to true and false: $f : \mathcal{A} \times \mathcal{B} \mapsto \{0, 1\}$. For our purpose, a form as in Equation 11 suggests itself.

$$f_{cp,a'}(a, b) = \begin{cases} 1 & : \text{ if } a = a' \text{ and } cp(b) = 1 \\ 0 & : \text{ otherwise} \end{cases} \tag{11}$$

As a set $\mathcal{P}$ of base distributions, we use an exponential model with parameters α_j for each feature f_j.

$$p(a|b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)} \quad \text{with normalization} \quad Z(b) = \sum_a \prod_{j=1}^{k} \alpha_j^{f_j(a,b)} \tag{12}$$

The subset $P \subset \mathcal{P}$ of all exponential distributions $\mathcal{P}$ satisfies our expectations, i.e., the empirical feature expectations $E_{\tilde{p}} f_j$ equal the model expectations $E_p f_j$.

$$P = \{p | E_p f_j = E_{\tilde{p}} f_j, 1 \leq j \leq k\} \quad \text{with}$$

$$E_{\tilde{p}} f_j = \sum_{a,b} \tilde{p}(a, b) f_j(a, b) = \frac{1}{N} \sum_{i=1}^{N} f_j(a_i, b_i) \quad \text{and} \quad E_p f_j = \sum_{a,b} \tilde{p}(b) p(a|b) f_j(a, b)$$

Generalized Iterative Scaling is an iterative algorithm that adjusts the parameters α_j and guarantees a non-negative improvement at each iteration.

$$\alpha_j^{(0)} = 1.0 \quad \text{and} \quad \alpha_j^{(n+1)} = \alpha_j^{(n)} \left(\frac{E_{\tilde{p}} f_j}{E_{p^{(n)}} f_j}\right)^{1/C} \quad \text{with}$$

$$E_{p^{(n)}} f_j = \sum_{a,b} \tilde{p}(b) p^{(n)}(a|b) f_j(a, b) \quad \text{and} \quad p^{(n)}(a|b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{(n) f_j(a,b)}$$

C is a correction constant and defined as the maximum number of active features in the training set $\mathcal{T}$: $C = \max_{(a,b) \in \mathcal{T}} \sum_{j=1}^{k} f_j(a, b)$.

5.2.2 Implementation

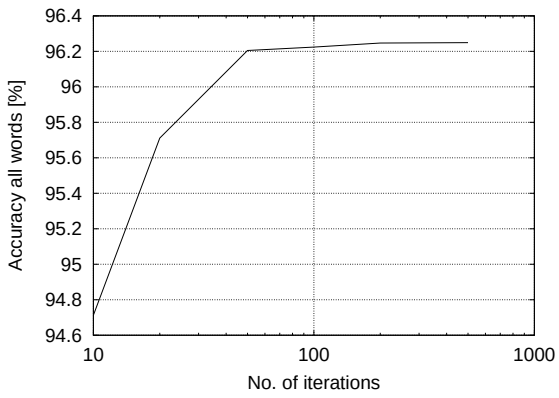
Our implementation MET [Schröder 2002] consists of approximately 2000 lines of C code.

Ratnaparkhi [1996] offers Java classes of his implementation MXPOST. There also exists a free Java implementation that is available from `http://maxent.sourceforge.net`.

5.2.3 Evaluation

Case sensitivity Switching from a case-insensitive lexicon to a case-sensitive lexicon results in an increase in tagging accuracy from 96.19% to 96.76% (ten-fold cross-validated) on the NEGRA corpus.

Number of iterations Figure 3 gives the achieved accuracy (not cross-validated) on the NEGRA corpus depending on the number of iterations the training was run. Although in this case a small improvement was achieved even after 100 iterations this must not always be the case. On a different test set the accuracy actually decreased when more than 100 iterations were used. Therefore we generally limit the training to 100 iterations.



No. of iterations	Accuracy [%]
10	94.713
20	95.711
50	96.205
100	96.224
200	96.247
500	96.249

Figure 3: MET: Accuracy on NEGRA test set depending on number of iterations.

Size of training set Table 6 summarizes how the accuracy depends on the training set size (not cross-validated).

Training [%]	10.0	19.9	29.8	39.9	50.0	60.0	70.0	79.9	90.0
Training [words]	129351	257005	384627	514564	644077	773394	902385	1030694	1159999
Sentence	41.25	45.39	48.09	49.52	50.50	51.12	52.02	52.81	53.73
All words	95.36	95.93	96.25	96.43	96.52	96.58	96.65	96.73	96.82
Known words	96.30	96.55	96.74	96.85	96.88	96.91	96.95	97.02	97.08
Unknown words	83.06	83.57	84.13	84.11	84.54	84.17	84.14	83.90	83.89

Table 6: MET: Accuracy on WSJ depending on training set size.

Summary Table 7 summarizes the achieved tagging accuracies (ten-fold cross-validated) on the NEGRA and the WSJ corpora.

	Sentences		All words		Known words		Unknown words	
Corpus	Acc.	SD	Acc.	SD	Acc.	SD	Acc.	SD
NEGRA	62.72	1.033	96.76	0.117	97.54	0.084	89.23	0.805
WSJ	53.49	0.385	96.82	0.038	97.08	0.040	83.62	0.866

Table 7: MET: Ten-fold cross-validated tagging results.

5.3 Error-driven Transformation-based Learning

Error-driven transformation-based learning (EDTBL) is a machine learning framework where a sequence of transformation rules is derived that successively improves an initially naïve classification [Brill 1993].

The approach is appealing for a number of reasons. First, it is simple so that it is easily understood and implemented (at least in its simplest form). Second, a model consists of a relatively small set of rules (instead of hundreds or thousands of numerical parameters) that can be interpreted by human experts. And third, the classification is simple and efficient since one only has to apply the learned rules to the new problem instance.

It has (at least) been applied to NLP tasks such as POS tagging [Brill 1995; Brill 1997], prepositional phrase attachment disambiguation [Brill & Resnik 1994] and parsing [Brill 1996].

5.3.1 Background

One of the advantages of EDTBL is that it is a greedy algorithm that is not based on probability models, i.e., it is easier to understand and less error-prone in implementation.

Given a training set $T = \{\langle w_1, t_1\rangle ... \langle w_n, t_n\rangle\}$ each sample $\langle w_i, t_i\rangle$ is initially tagged with an arbitrary class g_i: $\langle w_i, g_i, t_i\rangle$. This initial classification can be completely uninformed, e.g., all samples receive the first tag in the tagset, but better results can be achieved if, for instance, known words are tagged with their most likely tag and unknown words with the most likely tag.

A set of rule templates $\{R_1 ... R_r\}$ determines which transformation rules are considered. Of course, simple templates lead to simple rules that can efficiently be tested but lack context sensitivity; the opposite holds for complex templates.

Let r_i^j be the rule that results from template R_j when instantiated at the position i, i.e., at sample $\langle w_i, g_i, t_i\rangle$. Let $g(r_i^j)$ be the number of samples where rule r_i^j changes the guess g_k to the correct tag t_k; similarly, $b(r_i^j)$ is the number of samples where the rule changes the guess from the correct tag to a wrong one.

At each iteration, the learning algorithm selects the rule r^* with maximum score, outputs it and applies it to the training corpus.

$$r^* = \arg\max_{r_i^j} g(r_i^j) - b(r_i^j)$$

The learning stops when the best rules' score drops below a certain threshold.

Classification simply involves the same kind of initial naïve tagging and successive rule application.

5.3.2 Implementation

Our implementation TBT [Schröder 2002] consists of approximately 2000 lines of C code and is relatively straight-forward but not terribly efficient.

Brill [1995] offers an implementation that is available at `ftp://ftp.cs.jhu.edu/pub/brill/Programs/`.

A recent faster implementation has been published by Ngai & Florian [2001] at `http://nlp.cs.jhu.edu/~rflorian/fntbl/index.html`.

5.3.3 Evaluation

Splitting of the training set During training the training set has to be split yet another time into the portion from which the lexicon is generated and the portion from which the rules are learned. Table 8 gives results on a separate test set with different relations between the sizes of the two portions. Although the trend is not monotonic, it seems as if the best results can be achieved when leaving the major part of the training set for learning the actual rules.

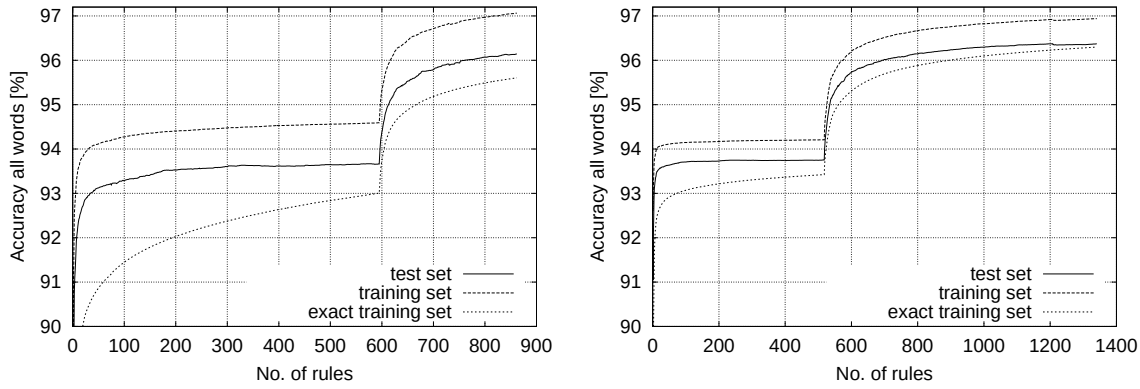
Portion [%]			Accuracy [%]			
test	lexicon	rules	Sentences	All words	Known words	Unknown words
10	70	20	51.72	95.12	96.57	81.04
10	50	40	53.13	95.00	96.59	79.47
10	30	60	54.25	95.30	96.80	80.64
10	10	80	55.65	95.64	96.95	82.87

Table 8: TBT: Accuracy on NEGRA depending on split of training set.

Number of rule applications Figure 4 shows how the accuracy on a test set and the training test (not cross-validated) evolves when more and more rules are applied. It can be observed that the quality is still improving when the last rules are applied. This can be a hint that the training has been aborted too early since typical quality curves on a test set show a slight degradation for the final rules.[4]

It can be observed the during training the accuracy strictly increases when a rule is applied. This is not strictly true when the test set is tagged or when a deviating lexicon or parameter set is chosen for the training set. Nevertheless it is obvious that the degradation is minimal. It can also clearly be seen that rules 595 (NEGRA) and 519 (WSJ) are the last rules for unknown words; after that the context rules have been learned which give the accuracy curve a second boost.

[4] Training has been stopped after the best improvement was less than five counts.



(a) NEGRA (b) WSJ

Figure 4: TBT: Accuracy depending on number of rules applied. 'exact training set' means that the same lexicon and the same parameters as during the training are used; 'training set' means full training lexicon and test set parameters; 'test set' means full training lexicon and test set parameters.

Summary Table 9 summarizes the achieved tagging accuracies (ten-fold cross-validated) on the NEGRA and the WSJ corpora.

	Sentences		All words		Known words		Unknown words	
Corpus	Acc.	SD	Acc.	SD	Acc.	SD	Acc.	SD
NEGRA	57.91	1.115	96.08	0.117	97.08	0.120	86.42	0.648
WSJ	48.58	1.647	96.27	0.140	96.57	0.136	81.30	0.931

Table 9: TBT: Ten-fold cross-validated tagging results.

5.4 Example-based Models

Example-based models (also called memory-based, instance-based or distance-based) are based on the assumption that cognitive behavior can be achieved by looking at past experiences that resemble the current problem rather than learning and applying abstract rules.

Daelemans et al. [1996] applied the framework to the problem of POS tagging and later compared the performance to that of other taggers [Zavrel & Daelemans 1999].

5.4.1 Background

Problem instances are characterized by a set of features $f_1, ..., f_n$. Each feature f_i can take a value v_i^j from the set of admissible values for that feature $V_i = \{v_i^1, ..., v_i^m\}$.

The idea behind memory-based models is to simply store all problem instances including the correct classification (or a compact representation) from the training set. During classification one or more

instances that are similar to the new problem instance are retrieved from the instance base and used to predict the classification (k-nearest neighbor). The distance $\Delta(X, Y)$ of two problem instances is often defined as the weighted sum of the distances of the corresponding feature values.

$$\Delta(X, Y) = \sum_{i=1}^{n} \phi_i \cdot \delta_i(X, Y)$$

The distance between feature values is simply one if the values are equal and zero otherwise since in our domain the values are discrete.

$$\delta_i(X, Y) = \begin{cases} 1 & : \text{ if } f_i(X) = f_i(Y) \\ 0 & : \text{ otherwise} \end{cases} \tag{13}$$

The feature weights ϕ_i denote the importance of the feature for the classification task.

Entropy $H(T)$ (or uncertainty) of a set of problem instances is the average number of bits required to encode the classification.

$$H(T) = -\sum_{t \in T} p(t) \cdot \log p(t)$$

Information gain IG is the difference in entropy before and after the value of a feature is known and can serve as a feature weight since features that contribute substantially to the disambiguation process have a huge information gain.

Let T be the training set and $T_{f_i = v_i^j} \subset T$ be the subset where the feature f_i has a value v_i^j.

$$\mathrm{IG}(T, f_i) = H(T) - \sum_{v_i^j \in V_i} H(T_{f_i = v_i^j}) \cdot \frac{|T_{f_i = v_i^j}|}{|T|}$$

If the number of values varies greatly among the features the measure of information gain prefers those features with a lot of values. Therefore, we use the *gain ratio GR* measure as a feature weight in these cases which is the ratio of information gain and *split information SI*.

$$\begin{aligned} \mathrm{SI}(T, f_i) &= \sum_{v_i^j \in V_i} \frac{|T_{f_i = v_i^j}|}{|T|} \cdot \log \frac{|T_{f_i = v_i^j}|}{|T|} \\ \phi_i = \mathrm{GR}(T, f_i) &= \mathrm{IG}(T, f_i) / \mathrm{SI}(T, f_i) \end{aligned}$$

During classification of a new problem instance i the training instance $t \in T$ with the smallest distance $\Delta(t, i)$ is retrieved from the case base and the classification of t is predicted as a class for i.

The main problem for a memory-based classifier is that for each new task it has to search the whole instance base to find the closest match. To overcome this efficiency problem the instance base can be converted to a tree where each level corresponds to a feature and each branch on one level to a feature value [Daelemans, van den Bosch & Weijters 1997]. The features with higher weights are at the top of the tree. Standard tree search techniques, e.g., branch & bound, can be employed to find a minimal distance instance more quickly. [Zavrel & Daelemans 1999] showed that even the relative rough approximation where only a single path is followed in the tree (without backtracking) gives good results as long as the weights of the features are different enough.

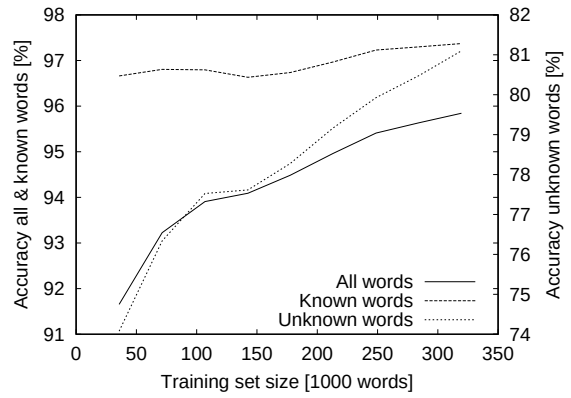
5.4.2 Implementation

Our implementation ET [Schröder 2002] consists of less than 1000 lines of C code.

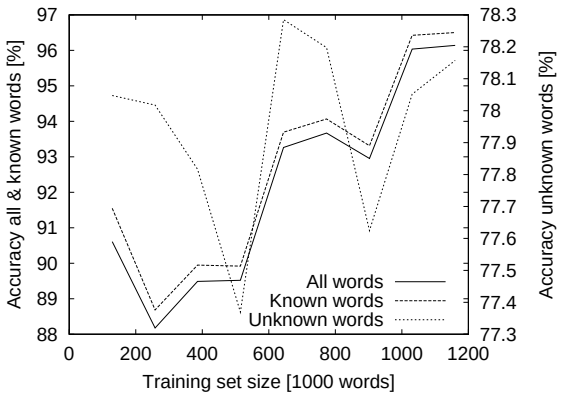
Zavrel & Daelemans [1999] provide a general toolkit for memory-based learning at `http://ilk.kub.nl/`.

5.4.3 Evaluation

Size of training set Figure 5 shows how the accuracy depends on the size of the training set. Note that the results for unknown words on the WSJ corpus are within a one percent range (77.3% to 78.3%). A further difference is that closed class categories have been excluded for unknown word on the NEGRA corpus but not the WSJ corpus.



(a) NEGRA



(b) WSJ

Figure 5: ET: Accuracy depending on training set size.

Summary Table 10 summarizes the achieved accuracies on the NEGRA and the WSJ corpus.

Corpus	Sentences Acc.	SD	All words Acc.	SD	Known words Acc.	SD	Unknown words Acc.	SD
NEGRA	55.81	2.585	95.84	0.365	97.37	0.316	81.10	1.034
WSJ	47.31	0.698	96.14	0.044	96.50	0.044	78.16	0.898

Table 10: ET: Ten-fold cross-validated tagging results.

6 Comparison of tagging paradigms

Different POS tagging paradigm have already been compared several times, for instance for Swedish [Megyesi 2001] and Slovene [Džeroski, Erjavec & Zavrel 2000]. However, for the first time several approaches have been completely re-implemented in a uniform framework and made freely available.

Table 11 summarizes the achieved accuracy results for the two corpora and four taggers plus a baseline which outputs the most probable lexical tag for known words and the unique most probable tag for unknown words.

Corpus	Tagger	Accurary Sent.	All	Known	Unknown
NEGRA	Baseline	31.80	90.98	94.99	52.36
	T3	63.41	96.83	97.95	85.06
	MET	62.72	96.76	97.54	89.23
	TBT	57.91	96.08	97.08	86.42
	ET	55.82	95.84	97.37	81.10
WSJ	Baseline	25.05	92.57	94.11	15.87
	T3	50.56	96.52	96.89	77.88
	MET	53.49	96.82	97.08	83.62
	TBT	48.58	96.27	96.57	81.30
	ET	47.20	96.14	96.50	78.18

Table 11: Comparison of tagging accuracy.

Table 12 summarizes advantages and disadvantages for the taggers.

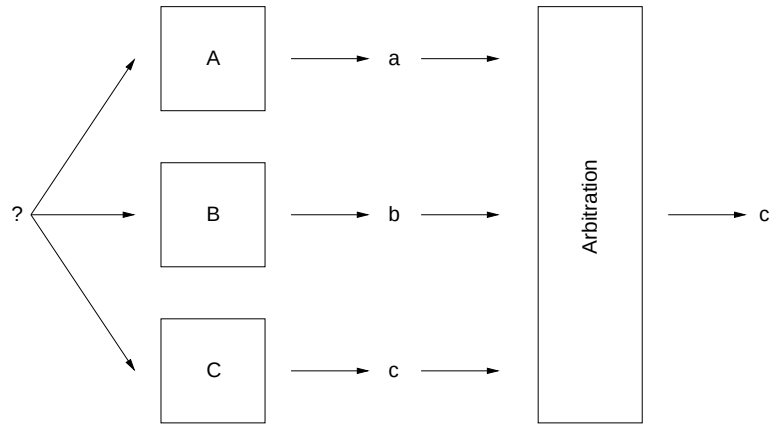
Tagger	Efficiency training	Efficiency tagging	Perspicuous model
Baseline	++	++	++
T3	++	++	-
MET	-	+	-
TBT	--	+	++
ET	-	--/+	-

Table 12: Characterization of taggers.

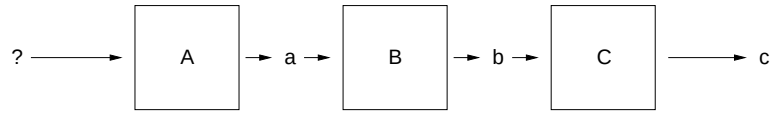
7 Parallel combination

The principle of structural redundancy states that a combination of n functional equivalent (but ideally internally different) components can fulfill a task more robustly and more accurately. Having available different machine learning methods for the same task, i.e., POS tagging in this case, it is only a small step to exploit this principle.

Two principal architectures (cf. Figure 6) lend themselves to combining classifiers: parallel and sequential. In a parallel configuration all methods classify the problem at hand and an additional arbitration mechanism is used to compute the overall result from the intermediate separate results. Actually, the parallel combination of different POS taggers has been studied quite extensively, for instance by van Halteren, Zavrel & Daelemans [2001]. The sequential configuration will be discussed in the next section.



(a) parallel



(b) sequential

Figure 6: Parallel and sequential combination of POS taggers.

Parallel architectures have been studied both theorectically and practically under the keyword *ensembles of classifiers* [Hansen & Salamon 1990; Dietterich 1997]. Two prerequisites for the success of a (weighted) majority voter in a parallel architecture have been identified:

- The individual components must make somewhat different errors, i.e., there exists a certain potential for mutual compensation of errors.
- The individual classifiers must have an error rate smaller than 50%.

The latter condition is generally satisfied for the problem of POS tagging. The former can be verified using the *complementary error rate* as suggested by Brill & Wu [1998].

$$\text{comp(A, B)} = 1 - \frac{\#\text{ common errors of A and B}}{\#\text{ errors in A}}$$

The (asymmetrical) complementary error rate comp(A, B) of two classifiers A and B is the percentage of time when classifier A is wrong that classifier B is correct. Two identical classifiers have a complementary error rate of zero. If B gets all error cases of A correct the complementary error rate is one. Table 13 lists the complementary error rate for the taggers discussed and evaluated in this paper. Obviously, the different taggers make sufficiently different errors.

comp(A, B) [%]	T3	MET	TBT	ET
T3		39.20	42.98	35.97
MET	40.48		42.51	39.61
TBT	53.82	52.43		44.13
ET	50.86	52.62	47.03	

(a) NEGRA

comp(A, B) [%]	T3	MET	TBT	ET
T3		37.69	39.03	37.75
MET	31.84		37.48	35.67
TBT	43.03	46.58		37.82
ET	43.93	47.02	40.07	

(b) WSJ

Table 13: Complementary error rate of taggers.

A number of advantages have been identified for a parallel architecture which are discussed in the following sections.

No. of taggers	4	3	2	1	0
Correct	92.79	3.67	1.34	0.99	1.22
Accumulated correct	92.79	96.46	97.80	98.78	100.00
Tag ambiguity	0.937	0.983	1.012	1.068	

(a) NEGRA

No. of taggers	4	3	2	1	0
Correct	93.47	2.75	1.31	1.04	1.43
Accumulated correct	93.47	96.22	97.53	98.57	100.00
Tag ambiguity	0.946	0.984	1.012	1.058	

(b) WSJ

Table 14: Combining the individual suggestions of different taggers.

Higher accuracy The potential of voting schemes to improve the POS tagging accuracy has already been studied, for example by Brill & Wu [1998; Brill [2000]. Table 14 lists the number of times when a certain number of our taggers agree on the correct tag. It can be observed that the percentage of times when at least two taggers found the correct tag is significant greater than the accuracy of the best individual tagger. On the NEGRA corpus test set, for instance, the individual taggers score 96.50% (T3), 96.82% (MET), 96.37% (TBT) and 96.10% (ET) while in 97.80% of the cases at least two taggers find the correct tag. We could not find a real advantage for a pure majority voter which outputs a tag only if an unambiguous majority of taggers voted for the tag since only 96.65% accuracy was achieved in this case. However, when breaking ties randomly, i.e., adding half of the 1.15% cases of 2:2 ties, and one quarter of the 1:1:1:1 ties, we gain another 0.58% accuracy. Thus a majority voter can be expected to achieve 97.23% accuracy which is better than the best individual tagger.

Higher recall Even more tags can be recovered when allowing more than a single tag per word. Table 14 shows that a considerable gain in recall can be achieved in this case. Emitting every tag leads to an accuracy of 98.78% on the NEGRA corpus while increasing the tag ambiguity to 1.068 tags per words. On the WSJ 98.57% accuracy with an ambiguity 1.058 tags per words is possible.

Bootstrapping Màrquez, Padró & Rodríguez [1998] suggest bootstrapping of taggers as a further advantage of parallel combination of taggers.

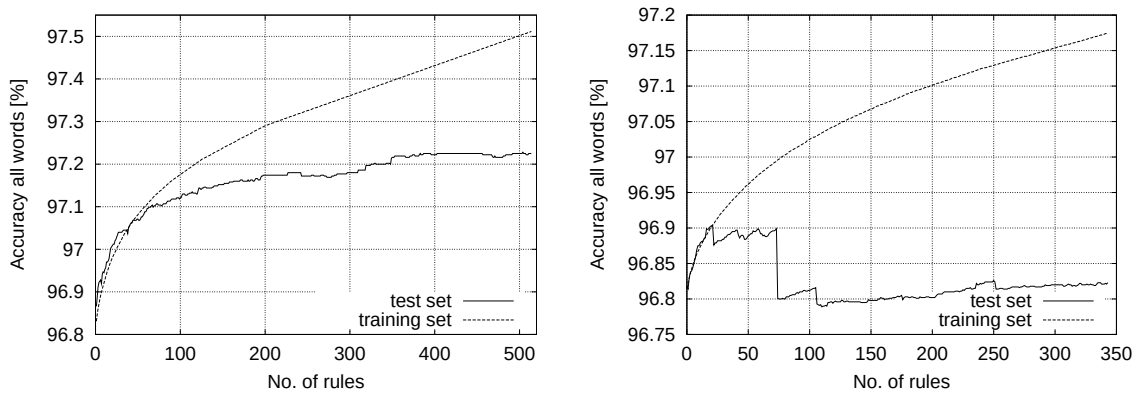
8 Sequential Combination

A sequential architecture for POS taggers as shown in Figure 6(b) has – as far as the author knows – not been systematically investigated.

Error-driven transformation-based learning (EDTBL) is particularly well-suited for a sequential architecture for a number of reasons:

- EDTBL already assumes an initial annotation of the input. Usually a trivial one is chosen such as the most probable tag of each word, but more sophisticated ones are also possible. Actually, Brill [1995] suggests using the output of another tagger but does not further investigate this issue.
- Starting with the output of a sophisticated tagger has the additional advantage that the number of errors in the training corpus is relatively small (compared to a trivial annotation) so that learning rules is more efficient.

A different view on the distinction between parallel and sequential combination refers to the information that a later component (i.e., the arbitration mechanism or a second stage tagger) has access to. A parallel arbitrater has only access to the output of the taggers while for a sequential tagger linguistic context information (as all taggers) and the guess of the preceding component is available. Hence, if a EDTBL component uses only the decisions of the preceding taggers it would play the role of the arbitration component in Figure 6(a) and the whole architecture would resemble the parallel configuration. Here we are investigating a sequential tagger combination.



(a) NEGRA (b) WSJ

Figure 7: TBT:

For the NEGRA corpus, we started from the output of the T3 tagger. After 514 rules the accuracy on the training corpus improved from 96.81% to 97.51%. Training was stopped after less than two errors were corrected. On the test set, the accuracy increased from 96.85% to 97.23%.

A similar experiment for the WSJ corpus started from the output of the MET tagger. The accuracy increased from 96.82% to 97.18% on the training set. 343 rules were learned and training was stopped after less than five errors were corrected. On the test set, however, the accuracy practically did not increase (only from 96.817% to 96.823%). Looking at the results more closely, it is apparent that only two to four very bad rules are responsible for the disappointing result. It is unclear why such massively destructive rules are among the learned rules. Obviously there exist general differences between the training set and the test set. Further experiments will show whether different splits into training and test set will lead to more favorable results.

Figure 7 graphically shows the run of the curves.

References

Thorsten Brants. 2000. TnT - a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, Seattle, WA, USA.

Thorsten Brants, Wojciech Skut & Brigitte Krenn. 1997. Tagging grammatical functions. In *Proceedings of EMNLP-2*, Providence, RI.

Eric Brill. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the the 31st Annual Meeting of the ACL.*

Eric 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.

Eric Brill. 1996. Transformation-based error-driven parsing. In Harry Bunt & Masara Tomita, ed., *Recent Advances in Parsing Technology.* Kluwer Academic Publishers, Dordrecht.

Eric Brill. 1997. Unsupervised learning of disambiguation rules for part of speech tagging. In *Natural Language Processing Using Very Large Corpora.* Kluwer Academic Publishers.

Eric Brill. 2000. Automatic grammar induction: Combining, reducing and doing nothing. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT-2000)*, pages 1–5, Trento, Italy.

Eric Brill & Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th International Conference on Computational Linguistics.*

Eric Brill & Jun Wu. 1998. Classifier combination for improved lexical disambiguation. In *Proc. Joint Conference COLING/ACL '98*, pages 191–195, Montréal, Canada.

Eugene Charniak. 1993. *Statistical Language Learning.* Cambridge, MA: The MIT Press.

Doug Cutting, Julian Kupiec, Jan Pedersen & Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Trento, Italy.

Walter Daelemans, Antal van den Bosch & Ton Weijters. 1997. Igtree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

Walter Daelemans, Jakub Zavrel, Peter Berck & Steven Gillis. 1996. MBT: A memory-based part of speech tagger-generator. In Eva Ejerhed & Ido Dagan, ed., *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 14–27.

Steven J. DeRose. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39.

Tom Dietterich. 1997. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136.

Sašo Džeroski, Tomaž Erjavec & Jakub Zavrel. 2000. Morphosyntactic tagging of Slovene: Evaluating taggers and tagsets. In *Proceedings of the 2nd International Conference on Language Resources & Evaluation (LREC-2000)*, Athens.

L. K. Hansen & P. Salamon. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001.

Johannes Heinecke, Jürgen Kunze, Wolfgang Menzel & Ingo Schröder. 1998. Eliminative parsing with graded constraints. In *Proceedings of the Joint Conference COLING/ACL-98*, pages 526–530, Montréal, Canada.

Fred Jelinek. 1990. Self-organizing language models for speech recognition. In Alex Waibel & Kai-Fu Lee, ed., *Readings in Speech Recognition*. Morgan Kaufmann, San Mateo, CA, pages 450–506.

Aravind Joshi & Bangalore Srinivas. 1994. Disambiguation of super parts of speech (or supertags): Almost parsing. In *Proceedings of the International Conference on Computational Linguistics (COLING-94)*, Kyoto, Japan.

Timo Järvinen & Pasi Tapanainen. 1997. A dependency parser for English. Technical Report TR-1, Department of General Linguistics, University of Helsinki, Finland.

Fred Karlsson. 1990. Constraint grammar as a framework for parsing running text. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING-90)*, pages 168–173, Helsinki, Finland.

Fred Karlsson, Atro Voutilainen, Juha Heikkilä & Arto Anttila, ed.. 1995. *Constraint Grammar – A Language-Independent System for Parsing Unrestricted Text*. Berlin, Germany: Mouton de Gruyter.

Brigitte Krenn & Christer Samuelsson. 1996. The linguist's guide to statistics. ftp://coli.uni-sb.de/pub/coli/doc/stat/stat_cl.ps.gz.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics (ACL-95)*.

Christopher D. Manning & Hinrich Schütze. 2000. *Foundations of Statistical Natural Language Processing*. 2nd edition. Cambridge, MA, USA: MIT Press.

M. Marcus, B. Santorini & M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2).

Beáta Megyesi. 2001. Comparing data-driven learning algorithms for PoS tagging of swedish. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2001)*, pages 151–158, Carnegie Mellon University, Pittsburgh, PA, USA.

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

Lluís Màrquez, Lluís Padró & Horacio Rodríguez. 1998. Improving tagging accuracy by using voting taggers. In *Proceedings 2nd International Conference on Natural Language Processing and Industrial Applications*, Moncton, Canada.

Grace Ngai & Radu Florian. 2001. Transformation based learning in the fast lane. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*.

Lawrence R. Rabiner. 1990. A tutorial on hidden markov models and selected applications in speech recognition. In Alex Waibel & Kai-Fu Lee, ed., *Readings in Speech Recognition*. Morgan Kaufmann, San Mateo, CA, pages 267–290. See also Errata http://www.media.mit.edu/ rahimi/rabiner/rabiner-errata/.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the First Conference on Empirical Methods in Natural Language Processing*, University of Pennsylvania, PA.

Adwait Ratnaparkhi. 1997a. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Brown University, Providence, USA.

Adwait Ratnaparkhi. 1997b. A simple introduction to maximum entropy models for natural language processing. Technical Report 97-08, Institute for Research in Cognitive Science.

Adwait Ratnaparkhi. 1998a. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Adwait Ratnaparkhi. 1998b. Unsupervised statistical models for prepositional phrase attachment. In *Proc. of COLING/ACL '98*, pages 1079–1085, Montréal, CA.

Adwait Ratnaparkhi, Jeffrey C. Reynar & Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the ARPA Human Language Technology Workshop*, pages 250–255.

Adwait Ratnaparkhi, Salim Roukos & R. Todd Ward. 1994. A maximum entropy model for parsing. In *Proceedings of the International Conference on Spoken Language Processing*, pages 803–806, Yokohama, Japan.

Christer Samuelsson & Atro Voutilainen. 1997. Comparing a linguistic and a stochastic tagger. In *Proceedings of the 35th Annual Meeting of the ACL and 8th Conference of the European Chapter of the ACL*, Madrid, Spain.

Anne Schiller, Simone Teufel, Christine Stöckert & Christine Thielen. 1995. Vorläufige Guidelines für das Tagging deutscher Textcorpora mit STTS. Draft of 14th November 1995.

Helmut Schmid. 1994a. Part-of-speech tagging with neural networks. In *Proceedings of the International Conference on Computational Linguistics (COLING-94)*, pages 172–176, Kyoto, Japan.

Helmut Schmid. 1994b. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Ingo Schröder. 2002. ICOPOST: User's manual. `http://nats-www.informatik.uni-hamburg.de/~ingo/icopost/`.

Wojciech Skut, Brigitte Krenn, Thorsten Brants & Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97)*, Washington, D. C., USA.

Hans van Halteren, Jakub Zavrel & Walter Daelemans. 2001. Improving accuracy in wordclass tagging through combination of machine learning systems. *Computational Linguistics*, 27(2):199–230.

A. Voutilainen. 1995. A syntax-based part of speech analyser. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL-95)*, pages 157–164.

Jakub Zavrel & Walter Daelemans. 1999. Recent advances in memory-based part-of-speech tagging. In *Actas del VI Simposio Internacional de Comunicacion Social*, Santiago de Cuba, Cuba.